SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 1–18
© 2005 Society for Industrial and Applied Mathematics

CACHE ISSUES OF ALGEBRAIC MULTIGRID METHODS FOR LINEAR SYSTEMS WITH MULTIPLE RIGHT-HAND SIDES*

GUNDOLF HAASE† AND STEFAN REITZINGER‡

Abstract. This paper is concerned with the solution of a linear system with multiple right-hand sides. Such problems arise from nonlinear, time-dependent, inverse, or optimization problems. In order to solve these problems efficiently we use block variants of the conjugate gradient method and combine them with cache aware algorithms. Further, we describe the acceleration of algebraic multigrid (AMG) methods by using such cache aware algorithms. Finally, numerical studies are presented that show the high efficiency of the proposed methods.

Key words. algebraic multigrid, block conjugate gradient methods, cache algorithms, nonuniform memory access

AMS subject classifications. 65Y05, 65N55, 68W10, 65N30

DOI. 10.1137/S1064827502405112

1. Introduction.

The simulation of stationary processes is described by elliptic, (self-adjoint) partial differential equations (PDEs). Solving such a direct problem, e.g., determining the state variables from given source and boundary data, via a finite element (FE) discretization is standard in numerical software today. The situation changes for time-dependent, nonlinear, optimization, or inverse problems. Here, the numerical solution strategy yields a sequence of linear equations, and it is often necessary to solve a huge number of linear systems with different right-hand sides in order to solve the overall problem. In this paper we discuss related solution methods for multiple right-hand sides and the constant system matrix resulting from a self-adjoint elliptic PDE, i.e.,

$$K_h \underline{u}_h^i = \underline{f}_h^i, \qquad i = 1, \ldots, N_r, \tag{1.1}$$

where $K_h \in \mathbb{R}^{N_h \times N_h}$ denotes a symmetric positive definite (SPD) system matrix, $\underline{f}_h^i \in \mathbb{R}^{N_h}$ denotes a given ith right-hand side, $\underline{u}_h^i \in \mathbb{R}^{N_h}$ denotes the corresponding solution vector, and N_r expresses the overall number of right-hand sides. We are especially interested in the case where the system matrix stems from a second order self-adjoint elliptic PDE, e.g., the potential equation, Lamé equations, or Maxwell equations. In addition we distinguish two different settings of the given right-hand sides.

1. The right-hand sides can be computed in advance, e.g., inverse or optimization problems.
2. The right-hand sides are computed successively, e.g., time-dependent problems.

*Received by the editors April 4, 2002; accepted for publication (in revised form) October 22, 2004; published electronically August 17, 2005. This research has been partially supported by the Austrian Science Fund FWF within the SFB "Numerical and Symbolic Scientific Computing" under grant SFB F013 and under project P14953 "Robust Algebraic Multigrid Methods and their Parallelization."
http://www.siam.org/journals/sisc/27-1/40511.html

†Institute for Mathematics and Scientific Computing, Karl-Franzens University of Graz, Heinrichstr. 36, A-8010 Graz, Austria (gundolf.haase@uni-graz.at).

‡Institute for Analysis and Computational Mathematics, Johannes Kepler University Linz, Altenberger Str. 69, A-4040 Linz, Austria (reitz@numa.uni-linz.ac.at).

In many applications the right-hand sides $\{\underline{f}_h^i\}_{i=1}^{N_r}$ are not arbitrary, but the differences are small in a certain sense for $i = 1, \ldots, N_r$ or at least for a subsequence $i = s_1, \ldots, s_2$, $1 \leq s_1 < s_2 \leq N_r$. In [2, 20, 21, 26, 29] the properties of such a sequence of right-hand sides were exploited by using variants of the conjugate gradient (CG) (or, equivalently, the Lanczos) method. The first ideas concerning the CG method with multiple right-hand sides are due to [20, 21], and later an analysis of the method was given in [2, 26, 29]. An application for the sensitivity analysis in shape optimization can be found in [4]. These methods have to store all Krylov subspace vectors, which requires additional memory in comparison to simpler solution methods. Memory was very expensive at the time of the earlier papers and therefore the methods were not very popular. Additionally, the known preconditioners for the CG method were not optimal and therefore the Krylov subspace became very large. However, variants were suggested (see [26, 29]) that do not need to store the Krylov subspace if the right-hand sides can be calculated in advance.

At the present time there are optimal preconditioners (with respect to memory and arithmetic costs) such as multigrid (MG) methods [1, 28, 10] available for nontrivial practical problems so that the Krylov subspace can be kept small. For some theory on MG preconditioners, see [14]. Since geometric MG methods have several drawbacks when they are applied to engineering applications, we use algebraic multigrid (AMG) methods which essentially require the system matrix and the right-hand side of the original discretization; see [25, 23, 9, 28] for details.

The second part of the paper is concerned with cache aware techniques for MG methods and reduction of arithmetic costs in an implementation. In contrast to cache optimization techniques for tensor product grids [7, 15, 16], we focus our attention on cache aware programming techniques for unstructured finite element method (FEM) grids [12, 5, 6]. Although most of the techniques used in our paper have been well known for decades [11], whether it pays to use these techniques also depends on the specialties of the problem and the hardware. Furthermore, we apply such cache aware algorithms for simultaneous right-hand sides even if they do not correlate.

The remainder of the paper is organized as follows: Section 2 describes the algorithmic tuning of MG codes in general, which is especially applied to the AMG software package PEBBLES [22]. Section 3 is concerned with simultaneous right-hand sides and an appropriate memory management. Additionally, we describe the preconditioned CG version for multiple right-hand sides. Section 4 presents numerical studies with academic and real life applications, followed by conclusions and further remarks in section 5.

2. Algorithmic improvements of multigrid. We have to solve (1.1) N_r times (e.g., $N_r \approx 10000$) for different right-hand sides by using an MG preconditioned CG method. Therefore, even a minor saving of solution time for one solve accumulates to a substantially faster response to the overall problem.

All algorithms have the problem that their response time is much more determined by data access than by floating point operations. This is caused by the characteristics of recent processors where the floating point unit can operate much faster than the data can be transferred from main memory into its registers. The problem increases with systems of equations resulting from unstructured FE-meshes.

2.1. Improved data structures. All CPUs possess up to three levels of cache, and having all data available in the fastest cache would be the perfect data management for all sorts of algorithms. Unfortunately, our problems are much too big to fit in the caches and so the CPU has to load the data needed next into the faster caches (also taking into account the cache hierarchy). Each time the CPU does not

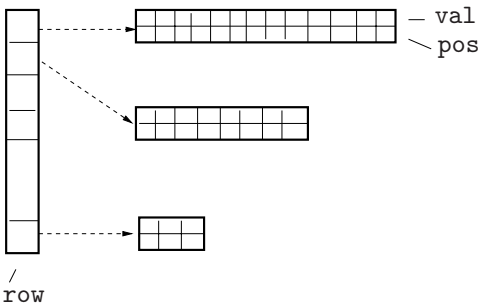


FIG. 2.1. *Principle structure of the original matrix memory management.*

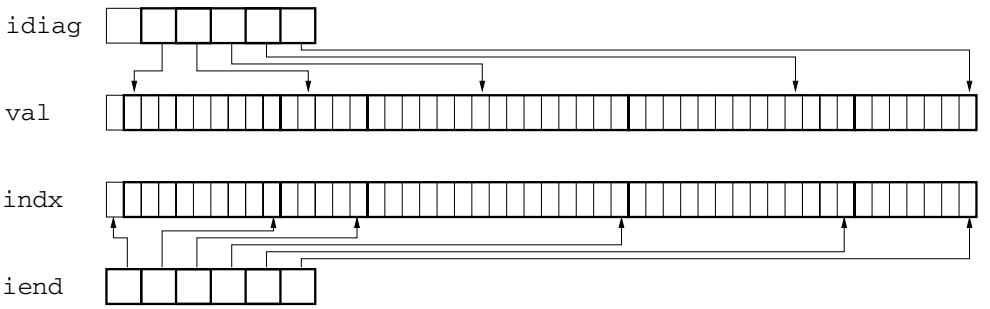


FIG. 2.2. *Matrix in CRS storage with additional diagonal pointer.*

find its data in the cache, i.e., a cache miss occurs, the processor has to stop until the data is transferred from main memory or lower level caches, and the whole execution of that algorithm is significantly delayed. Recent processor and compiler technology tries to decrease the number of cache misses by data prefetching; i.e., more data than is actually needed is loaded into the cache. This technique speculates that subsequent data in memory will be needed soon for calculations [5]. Data is always transferred as full cache line, e.g., 128 bytes long, from main memory into the caches instead of loading only a single real number. Therefore, our data structures should support the cache line technique by storing and accessing data linearly in the memory, and this data should be reused as often as possible without being removed from the cache. The prefetching of data can be additionally supported by a very careful use of pointers and other indirect addressing techniques. Figure 2.1 presents a closer look at the very flexible pointer-based storage scheme for the sparse stiffness matrix K_h resulting from an unstructured FE-mesh used in the original AMG-code PEBBLES [22] (written in C++). However, the memory access pattern of these data structures is not well suited for cache efficient implementations since data of adjacent matrix rows may be stored in nonadjacent memory areas. Consequently, we replaced that storage scheme by the classical compressed row storage (CRS) format for sparse matrices containing vectors `val`, `indx`, `iend` for storing matrix values, column indices, and indices of last row entry; see Figure 2.2. We added an additional vector storing the diagonal pointers `idiag` to accelerate smoothing procedures. The symmetry of the matrix is not exploited in the storage scheme; i.e., we store all nonzero entries, but this allows faster algorithms for symmetric matrices (see Algorithm 4). The interpolation weights should be stored in the original CRS format. These changes in the matrix and interpolation weight storage accelerated the code by 10–15%.

2.2. Avoiding data traffic. Another way to accelerate MG-codes consists of avoiding data traffic and arithmetic operations that are not necessary. Let us have a closer look at the classical MG algorithm (see, e.g., [10, 28, 1]), presented in Algorithm 1, where $q = 1$ denotes the finest level and $q = \ell$ the coarsest level. $\{K_j, P_j\}_{j=1}^{\ell}$ includes system and interpolation matrices for all levels. In the following we discuss the MG algorithm for one right-hand side only and denote two consecutive fine and coarse levels by q and $q+1$, respectively.

Algorithm 1 Multigrid $\underline{u}_q \longleftarrow \mathrm{MG}(\{K_j, P_j\}_{j=1}^{\ell}, \underline{u}_q, \underline{f}_q, q)$.

if $q == \ell$ **then**
  $\underline{u}_q \longleftarrow$ SOLVE$(\, K_q \cdot \underline{u}_q \;=\; \underline{f}_q \,)$
else
  $\widetilde{\underline{u}}_q \longleftarrow$ SMOOTHF$(K_q, \underline{u}_q, \underline{f}_q)$
  $\underline{d}_q \longleftarrow \underline{f}_q - K_q \cdot \widetilde{\underline{u}}_q$
  $\underline{d}_{q+1} \longleftarrow (P_q)^T \cdot \underline{d}_q$
  $\underline{w}_{q+1} \longleftarrow 0$
  $\underline{w}_{q+1} \longleftarrow \mathrm{MG}(\{K_j, P_j\}_{j=1}^{\ell}, \underline{w}_{q+1}, \underline{d}_{q+1}, q+1)$
  $\underline{w}_q \longleftarrow P_q \cdot \underline{w}_{q+1}$
  $\widetilde{\underline{u}}_q \longleftarrow \widetilde{\underline{u}}_q + \underline{w}_q$
  $\underline{u}_q \longleftarrow$ SMOOTHB$(K_q, \widehat{\underline{u}}_q, \underline{f}_q)$
end if

A modular concept of the original MG algorithm realizes the calculation of the defect $\underline{d}_q \longleftarrow \underline{f}_q - K_q \cdot \widetilde{\underline{u}}_q$ in a way that the result of matrix-vector multiplication is stored in a temporary vector which is subtracted from $\underline{f}_q$ afterwards. Both steps can be combined in a routine DEFECT which saves two accesses to the temporary vector.

The interpolation of the correction from the coarse grid and the correction of the existing solution vector with it, i.e.,

$$\underline{u}_q = \underline{u}_q + P_q \cdot \underline{w}_{q+1}, \tag{2.1}$$

is usually realized in three steps ($\underline{w}_q \longleftarrow 0$, $\underline{w}_q \longleftarrow \underline{w}_q + P_q \cdot \underline{w}_{q+1}$, $\underline{u}_q \longleftarrow \underline{u}_q + \underline{w}_q$). The direct implementation of (2.1) by loop fusion in a routine INTCORR requires only minor changes in the given interpolation routine but saves initialization and access to the auxiliary vector $\underline{w}_q$. Algorithm 2 presents the modified MG algorithm.

Algorithm 2 Modified multigrid $\underline{u}_q \longleftarrow \mathrm{MG}(\{K_j, P_j\}_{j=1}^{\ell}, \underline{u}_q, \underline{f}_q, q)$.

if $q == \ell$ **then**
  $\underline{u}_q \longleftarrow$ SOLVE$(\, K_q \cdot \underline{u}_q \;=\; \underline{f}_q \,)$
else
  $\widetilde{\underline{u}}_q \longleftarrow$ SMOOTHF$(K_q, \underline{u}_q, \underline{f}_q)$
  $\underline{d}_q \longleftarrow$ DEFECT$(K_q, \widetilde{\underline{u}}_q, \underline{f}_q)$
  $\underline{d}_{q+1} \longleftarrow (P_q)^T \cdot \underline{d}_q$
  $\underline{w}_{q+1} \longleftarrow 0$
  $\underline{w}_{q+1} \longleftarrow \mathrm{MG}(\{K_j, P_j\}_{j=1}^{\ell}, \underline{w}_{q+1}, \underline{d}_{q+1}, q+1)$
  $\widehat{\underline{u}}_q \longleftarrow$ INTCORR$(P_q, \widetilde{\underline{u}}_q, \underline{w}_{q+1})$
  $\underline{u}_q \longleftarrow$ SMOOTHB$(K_q, \widehat{\underline{u}}_q, \underline{f}_q)$
end if

2.3. Merging of defect calculation and smoother. A careful investigation of Algorithm 2 leads to the suspicion that some redundant calculations will be performed in SmoothF and Defect since the smoother as well as the defect calculation requires matrix-vector operations. And indeed, some results of the last Gauss–Seidel forward (analogously, backward) iteration (2.2) can be reused in the following defect calculation (2.3), $\forall i = 1, \ldots, n := N_q$ (= number of unknowns of level q):

$$\widetilde{u}_i = K_{ii}^{-1} \underbrace{\left(\underbrace{f_i - \sum_{j=1}^{i-1} K_{ij}\widetilde{u}_j}_{=:r} - \sum_{j=i+1}^{n} K_{ij}u_j \right)}_{=:v} = K_{ii}^{-1} v, \tag{2.2}$$

$$d_i = f_i - \sum_{j=1}^{n} K_{ij}\widetilde{u}_j = \underbrace{f_i - \sum_{j=1}^{i-1} K_{ij}\widetilde{u}_j}_{r} - \underbrace{K_{ii}\widetilde{u}_i}_{=v} - \sum_{j=i+1}^{n} K_{ij}\widetilde{u}_j$$

$$= r - v - \sum_{j=i+1}^{n} K_{ij}\widetilde{u}_j. \tag{2.3}$$

Note that we have access only to the nonzero elements of a matrix row although we write $j = 1, \ldots, n$ in order to simplify the notation. We combine the last smoothing sweep (2.2) and defect correction (2.3) for general matrices to Algorithm 3. The old vector $\underline{u}$ is overwritten by the new vector $\underline{\widetilde{u}}$, and thus no additional memory is required.

Algorithm 3 Merging of last smoothing sweep and defect calculation for general matrices in CRS.

for all $i = 1, \ldots, n$ **do**
  $r \longleftarrow f_i - \sum_{j=1}^{i-1} K_{ij}\widetilde{u}_j$
  $inv \longleftarrow 1.0/K_{ii}$
  $v \longleftarrow r - \sum_{j=i+1}^{n} K_{ij}u_j$
  $\widetilde{u}_i \longleftarrow v * inv$
  $d_i \longleftarrow r - v$
end for
for all $i = 1, \ldots, n$ **do**
  $d_i \longleftarrow d_i - \sum_{j=i+1}^{n} K_{ij}\widetilde{u}_j$
end for

The matrix is usually sparse and contains in sum NZE nonzero matrix elements in its n rows (NZE $= n^2$ in case of a fully occupied matrix). Algorithm 3 requires $\frac{3}{2} \cdot$ NZE arithmetic operations and accesses with matrix elements instead of $2 \cdot$ NZE when smoothing and defect calculation are separated. Unfortunately, we have to start again with the first row when accessing some matrix elements in the second loop; i.e., the upper triangular block of the matrix has to be transferred into the cache again. The two loops in the update of the defect, i.e., the last i-loop and the sum therein,

$$d_i \longleftarrow d_i - \sum_{j=i+1}^{n} K_{ij}\widetilde{u}_j \qquad \forall i = 1, \ldots, n,$$

can be interchanged and lead together with a swapping of indices i and j to

$$\{d_j \longleftarrow d_j - K_{ji}\widetilde{u}_i\}_{j=1}^{i-1} \qquad \forall i = 2, \ldots, n. \tag{2.4}$$

Now we could merge the two i-loops in Algorithm 3, but accessing the transposed matrix pattern would be much worse than before. In the case of a symmetric matrix, i.e., $K_{ij} = K_{ji}$, we can use K_{ij} instead of K_{ji} in (2.4) and suddenly we have to load each matrix row only once into cache. The combined routine SmoothFDefect for symmetric matrices is represented in Algorithm 4.

Algorithm 4 SmoothFDefect—Last smoothing sweep and defect calculation for symmetric matrices in CRS.

```
for all i = 1,...,n do
  r ⟵ f_i − Σ_{j=1}^{i−1} K_ij ũ_j
  inv ⟵ 1.0/K_ii
  v ⟵ r − Σ_{j=i+1}^{n} K_ij u_j
  ũ_i ⟵ v ∗ inv
  d_i ⟵ r − v
  for all j = 1,...,i − 1 do
    d_j ⟵ d_j − K_ij ũ_i
  end for
end for
```

2.4. Special initial solution for the first smoother sweep. The correction on the coarse grid $\underline{w}_{q+1}$ is set to zero before the MG-procedure for the next coarser level is called in Algorithm 2. This means that the recursive procedure $\mathrm{MG}(\{K_j, P_j\}_{j=1}^{\ell}, \underline{u}_q, \underline{f}_q, q)$ will get $\underline{u}_q = 0$ as a parameter and also as the initial solution for the first presmoothing sweep on that coarser level. This allows us to simplify iteration (2.2) to

$$\widetilde{u}_i \;=\; K_{ii}^{-1}\left(f_i - \sum_{j=1}^{i-1} K_{ij}\widetilde{u}_j\right), \tag{2.5}$$

which saves half of the arithmetic and memory operations and can be implemented in a routine SmoothF0.

Investigations on smoothers for two-dimensional (2D) tensor product grids with 5-point stencil discretization [31, 15] showed that merging three to five red-black Gauss–Seidel smoothing sweeps into one subroutine can accelerate the overall smoothing time by a factor of 2 to 3 on one of our target architectures (SGI OCTANE). But the latter requires combining five or more smoothing sweeps. Similar work on unstructured FE-meshes [6] achieved speedups in the range of 1.6 to 2. From our experiences with three-dimensional (3D) unstructured FE-meshes, it does not pay off to use more than two Gauss–Seidel iterations in the pre- as well as in the postsmoothing for simple potential problems. Therefore in our experiments we used a multigrid V-cycle with only one presmoothing and one postsmoothing sweep. This allows us to combine (2.5)

with (2.3) on coarser levels to obtain

$$\widetilde{u}_i = K_{ii}^{-1} \underbrace{\left(f_i - \sum_{j=1}^{i-1} K_{ij} \widetilde{u}_j \right)}_{=:v=:r},$$

$$d_i = - \sum_{j=i+1}^{n} K_{ij} \widetilde{u}_i,$$

resulting in NZE arithmetic operations and memory accesses when the only smoothing sweep is combined with the defect calculation in a routine SmoothF0Defect. The realization of SmoothF0 and SmoothF0Defect can be easily derived from Algorithm 4; the latter profits again from the matrix symmetry.

In order to replace SmoothF and Defect in Algorithm 2, the general routine SmoothDef$(K_q, \underline{u}_q, \underline{f}_q, \underline{d}_q, \nu, q)$ for merging smoothing and defect calculation on all levels has to take into account the implementation differences regarding the number ν of smoothing sweeps and the actual level q. The initial correction $\underline{u}_q$ on the coarser grids ($q > 1$) is always zero, but we have to assume an arbitrary nonzero initial guess for the solution $\underline{u}_1$ on the fine grid. Replacing SmoothF and Defect in Algorithm 2 with SmoothDef results in a fast MG-procedure.

Remark 2.1.

K_q NZE$_q$ q
NZE$_q$ N_q^2
$2 \cdot$ NZE$_q$ NZE$_q$

2.5. Coarse grid solver. Another important ingredient of AMG methods is the coarse grid solver. Since the average number of nonzero entries is growing on coarser levels in general, the application of a sparse solver becomes more and more expensive, and in addition the overall convergence rate of the AMG method worsens. Thus we like to construct a minimal number of levels ℓ. On the other side, the factorization of the coarse matrix using a sparse direct solver is relatively slow compared to the setup of AMG for many unknowns N_ℓ. Consequently, we have to find a compromise for when to stop the AMG coarsening process. In two dimensions we apply a sparse direct solver up to 50,000 unknowns, and the 3D limit is 10,000 unknowns. The package SuperLU [3] is used as a sparse direct solver in our applications.

3. Multiple right-hand sides. The second part of the paper concerns the efficient treatment of multiple right-hand sides. We discuss two different methods. The first is related to cache aware algorithms, and in principle a linear system with several right-hand sides is solved simultaneously. Therefore we need a special memory treatment of the right-hand sides. The other technique is related to the preconditioned CG method, where the previous technique is also applied.

3.1. Simultaneous right-hand sides. The idea behind this method simply consists of solving several equations at the same time independently of potential right-hand side correlations. Thus this method is well suited if the right-hand sides can be calculated in advance. The system matrix K_h remains the same, and so we change the code such that BLOCKSIZE systems of equations can be solved at the same time; see Algorithm 5.

Algorithm 5 Blocked loop for BLOCKSIZE right-hand sides.

```
for all s = 1, 1 + BLOCKSIZE, 1 + 2 * BLOCKSIZE, ..., N_r do
  ⋮
  for all k = s, ..., s + BLOCKSIZE − 1 do
    Solve K_h u_h^k = f_h^k
  end for
  ⋮
end for
```

The innermost loop in Algorithm 5 can be expressed as

$$\text{blockdiag}\{K_h\}\left(\underline{u}_h^k\right)_{k=s}^{s+\text{BLOCKSIZE}-1} \;=\; \left(\underline{f}_h^k\right)_{k=s}^{s+\text{BLOCKSIZE}-1} \quad ;$$

i.e., there is a chance to handle all BLOCKSIZE right-hand sides and solution vectors at the same time accessing the (unstructured) matrix K_h only once per matrix-vector operation inside the preconditioned CG (PCG) solution method used for solving the systems of equations. Let us have a look at two possible implementations of the matrix-vector operation where the matrix is stored in CRS format pointing to the end of the row; see Figure 2.2. First, we store BLOCKSIZE vectors separately as represented in Algorithm 6.

Algorithm 6 $\left(\underline{v}^k\right) \longleftarrow K \cdot \left(\underline{u}^k\right)$ with BLOCKSIZE separate vectors.

```
for all i = 1, 2, ..., size do                          {# rows in matrix}
  j1 = iend[i-1]+1 , j2 = iend[i]                       {begin, end of row}
  for all k = 1, 2, ..., BLOCKSIZE do
    v[i][k] = 0.0
  end for
  for all j = j1, j1 + 1, ..., j2 do
    jj = indx[j]                                        {column pointer}
    aij = val[j]                                        {matrix entry}
    for all k = 1, 2, ..., BLOCKSIZE do
      v[i][k] += aij * u[jj][k]
    end for
  end for
end for
```

Second, we store the first entries of all BLOCKSIZE vectors followed by the second entries; i.e., we merge the arrays. Together with a loop interchange, this should result in a better cache performance according to [11, section 5.5]. The implementation with block vectors is realized in Algorithm 7.

Both algorithms require the same number of memory accesses, and each matrix element is used BLOCKSIZE times in the innermost loop. But the block vector version in Algorithm 7 is much better suited for state-of-the-art processors since more useful data is found in one cache line; see also Figure 3.1 for illustration.

Algorithm 7 $\left(\underline{v}^k\right) \longleftarrow K \cdot \left(\underline{u}^k\right)$ with block vectors.

```
for all i = 1,2,...,size do                          {# rows in matrix}
  j1 = iend[i-1]+1 , j2 = iend[i]                    {begin, end of row}
  ii = (i-1)*BLOCKSIZE
  for all k = 1,2,...,BLOCKSIZE do
    v[ii+k] = 0.0
  end for
  for all j = j1, j1+1, ..., j2 do
    jj = (indx[j]-1)*BLOCKSIZE                       {column pointer}
    aij = val[j]                                     {matrix entry}
    for all k = 1,2,...,BLOCKSIZE do
      v[ii+k] += aij * u[jj+k]
    end for
  end for
end for
```

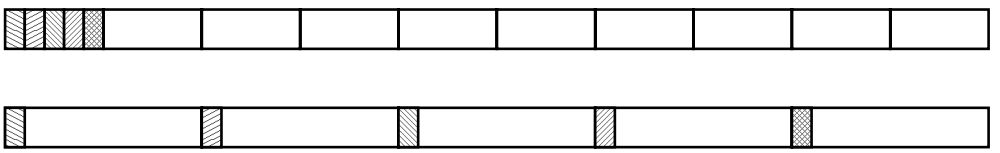


FIG. 3.1. *Storage scheme for separate vectors (bottom) and block vectors (top).*

The indirect addressing of vectors $\underline{u}_m$ via the column pointer causes misprefetching of data for the cache so that the cache lines have to be reloaded. In that case Algorithm 6 has to reload BLOCKSIZE-1 times more cache lines than Algorithm 7. Therefore, Algorithm 7 has a higher cache hit rate than Algorithm 6 and is faster at the end.

The changes in the CG algorithm are not dramatic. One should only be aware that the scalars of the CG algorithm will be represented by small vectors containing BLOCKSIZE scalars. The original stopping criterion for the CG tests the relative error with respect to the given tolerance. Now, we have to guarantee that each of the BLOCKSIZE errors is less than the tolerance. The simultaneous calculation of the inner product can be seen in Algorithm 8.

Algorithm 8 Inner products $(\alpha^k)_{k=1}^{\text{BLOCKSIZE}} \longleftarrow \left\langle \left(\underline{u}^k\right), \left(\underline{v}^k\right) \right\rangle$.

```
for all k = 1,2,...,BLOCKSIZE do
  α[k] = 0.0
end for
for all i = 0,1,...,size-1 do
  ii = i*BLOCKSIZE
  for all k = 1,2,...,BLOCKSIZE do
    α[k] += v[ii+k]*u[ii+k]
  end for
end for
```

Similar changes as in Algorithm 7 have to be made in the MG-components smoothing, defect calculation, interpolation, and prolongation.

REMARK 3.1.

```
for (k=1; k<=BLOCKSIZE; k++) d[jj+k] -= aij * u[ii+k];
```

SMOOTHFDEFECT 4 aij← K_{ij} BLOCKSIZE

```
#define
```

2 ∼3.5.

BLOCKSIZE= 5 BLOCKSIZE MULT SMOOTHB SMOOTHF SMOOTHF0 SMOOTHFDEFECT SMOOTHF0DEFECT DEFECT h_1, h_2, h_3, h_4, h_5 $h[1, \ldots, 5]$ 15–25%

3.2. Preconditioned Lanczos–Galerkin projection methods. While the multiple right-hand side method does not assume any information on the right-hand side properties, i.e., they are completely independent from each other, the second strategy profits from the speculation or the potential knowledge that sequencing right-hand sides are close to each other in a certain sense. Typically such problem setting arises from time-dependent equations where the right-hand sides most often vary slowly in time. We study the method of [26, 29], where a Lanczos method was used to derive a technique for the solution of a linear system with multiple right-hand sides. The condition number $\kappa(K_h) = \frac{\lambda_{\max}(K_h)}{\lambda_{\min}(K_h)}$ of the linear systems we are interested in behaves like $\mathcal{O}(h^{-2})$ as h tends to zero, where h denotes the typical mesh length of an FE-discretization. Thus the matrices are ill-conditioned, and therefore a CG method would require many iterations in order to solve the linear system up to a relative accuracy ϵ.

In order to circumvent this problem we use a preconditioner C_h which is optimal with respect to memory and arithmetic costs. Consequently, a constant number of iterations is required in order to solve a linear equation up to a fixed relative accuracy ϵ. Let us assume that the linear system (1.1) is preconditioned by C_h and we have $N_r = 2$, i.e.,

$$C_h^{-1/2} K_h C_h^{-1/2} \underline{v}_h^i = C_h^{-1/2} \underline{f}_h^i, \qquad \underline{u}_h^i = C_h^{-1/2} \underline{v}_h^i, \qquad i = 1, 2.$$

Using the abbreviations $B_h = C_h^{-1/2} K_h C_h^{-1/2}$ and $\underline{g}_h^i = C_h^{-1/2} \underline{f}_h^i$, we can apply the original technique of [26, 29] to the equations

$$B_h \underline{v}_h^i = \underline{g}_h^i, \qquad i = 1, 2\,. \tag{3.1}$$

The CG method applied to (3.1) and $i = 1$ produces after m steps a Krylov subspace

$$\mathcal{K}_m(B_h, \underline{z}_h^{1,0}) = \operatorname{span}\{B_h^0 \underline{z}_h^{1,0}, B_h \underline{z}_h^{1,0}, \ldots, B_h^{m-1} \underline{z}_h^{1,0}\}$$

with $\underline{z}_h^{1,0} = \underline{g}_h^1 - B_h \underline{v}_h^{1,0}$, $\underline{v}_h^{1,0} \equiv 0$. Furthermore, the quantities of the Lanczos method are written down for the original linear system (1.1). The approximative solution $\underline{u}_h^{1,m}$ of the original system (1.1) can be written in the form

$$\underline{u}_h^{1,m} = \sum_{i=0}^{m-1} \beta_i^1 \cdot C_h^{-1} \underline{r}_h^{1,i}$$

with $\underline{r}_h^{1,i} = K_h^i \underline{r}_h^{1,0}$, $\underline{r}_h^{1,0} = \underline{f}_h^1$. The coefficients β_i^j are calculated via the formula

$$\beta_i^j = (T_m^{-1} \alpha^j)_i, \qquad j = 1, 2\,, \tag{3.2}$$

where $T_m \in \mathbb{R}^{m \times m}$ is the tridiagonal matrix arising from the Lanczos process; see, e.g., [26, 29]. The vector $\alpha^j \in \mathbb{R}^m$ is calculated by

$$\alpha_i^j = \frac{\langle C_h^{-1} \underline{r}_h^{1,i}, \underline{f}_h^j \rangle}{\langle C_h^{-1} \underline{r}_h^{1,i}, \underline{r}_h^{1,i} \rangle}, \qquad j = 1, 2\,. \tag{3.3}$$

For further considerations let $\underline{f}_h^2$ be another right-hand side for (1.1). Since $\underline{f}_h^1$ and $\underline{f}_h^2$ are assumed to be close to each other, we perform the Galerkin projection onto $\mathcal{K}_m(B_h, \underline{z}_h^{1,0})$ of $\underline{f}_h^2$, i.e.,

$$\tilde{\underline{f}}_h^2 = \sum_{i=0}^{m-1} \alpha_i^2 \cdot \underline{r}_h^{1,i}\,,$$

with α_i^2 defined in (3.3). Consequently we get an approximate solution $\tilde{\underline{u}}_h^2$ for the solution of (1.1) in the form

$$\tilde{\underline{u}}_h^2 = \sum_{i=1}^{m-1} \beta_i^2 \cdot C_h^{-1} \underline{r}_h^{1,i}$$

with right-hand side $\underline{f}_h^2$ with β_i^2 defined in (3.2). From a heuristical point of view, $\tilde{\underline{u}}_h^2$ is a good approximation to $\underline{u}_h^2$ if the right-hand sides $\underline{f}_h^1$ and $\underline{f}_h^2$ are close. A rigorous analysis and algorithms are given in [26, 29, 2].

Without an optimal preconditioner C_h, the required basis might be very large, and therefore the memory requirements are unacceptable. But with an optimal preconditioner as, e.g., AMG, the required basis is very small. However we have to store $2 \cdot m$ vectors in $\mathbb{R}^{N_h}$, namely, $\{C_h^{-1} \underline{r}_h^{1,i}\}_{i=0}^{m-1}$ and $\{\underline{r}_h^{1,i}\}_{i=0}^{m-1}$. For the efficient calculation of the Galerkin projection we use the presented memory management for multiple vectors. In this way cache can be used efficiently. In order to perform a stable Galerkin projection we use the modified Gram–Schmidt algorithm (see [29]).

If the right-hand sides can be calculated in advance, then no storage of the Krylov basis is required [26, 29, 2].

4. Numerical studies. The presented methods are implemented in the AMG software package PEBBLES [22]. We always applied one symmetric V-cycle with one pre- and one postsmoothing step as preconditioner C_h. The CG iteration was stopped when a relative error reduction of ϵ had been achieved in the $\| \cdot \|_{KC^{-1}K}$-norm.

We used the C++ MIPSpro Compiler, Version 7.30, with the compiler options
`-64 -mips4 -r10000 -Ofast=ip27 -LNO:pf2=on:prefetch=2`
`-OPT:IEEE_arithmetic=3:roundoff=3:alias=RESTRICT -IPA`.
on OCTANE and ORIGIN running under IRIX 6.5. Adapting the options to the 400 MHz R14000 MIPS-processor on the SGI ORIGIN 3800 did not influence the run time too much. The processor possesses a 32 KB L1 data cache with 64 byte cache lines and a 4 MB L2 cache with 128 byte cache lines. Both caches are two-way set associative, and they use the least recently used (LRU) replacement policy. The SGI Octane-workstation (195MHz R10000) has a 1MB L2 cache with 128 byte cache lines and a 32 KB L1 data cache using 32 byte cache lines. The LINUX computer consisted of a 1.5 GHz AMD Athlon processor with a 256 kB L2 cache and a 64 kB L1 data cache both using 64 byte cache lines. We used the distribution by S.u.S.e. (version 7.1) as operating system and the g++ (gcc) compiler, Version 2.95, with the compiler options `-O3 -m486 -ffast-math`. Again, we tried several fancy options without any further gain in CPU-time.

The Lanczos–Galerkin approach from section 3.2 has been tested only on the LINUX system because its speedup is caused only by the reduction of the iteration numbers and not by hardware characteristics.

4.1. A 2D case study: Simultaneous right-hand sides. The first test example is concerned with the potential equation on the unit square and zero Dirichlet boundary conditions on one side of the square. On the remaining boundary we assume homogeneous Neumann boundary conditions. For an FE-discretization we use bilinear FE-functions on the equidistant grid. The right-hand sides are assumed to be of the form

$$(\underline{f}_h^i)_j = \sum_{k=0}^{n} \delta_{i,j+k} \cdot \zeta_k, \quad \zeta_k \neq 0\,, \qquad i = 1, \ldots, N_r, \quad j = 1, \ldots, N_h,$$

which are linear dependent if $n < N_r$. δ_{ij} denotes the Kronecker symbol.

First we study the method of simultaneous right-hand sides (see subsection 3.1) and fix the number of unknowns by $N_h = 90,000$. The memory required for storing all matrices and vectors is approximately 15MB + BLOCKSIZE*10MB. Since we do not need assumptions on the right-hand sides, we fix $n = N_r = \text{BLOCKSIZE}$ and $\zeta_k = 1$, $k = 0, \ldots, n$. The linear equation is solved up to a relative accuracy $\epsilon = 10^{-8}$. The performance gain with respect to simultaneous right-hand sides can be seen in Figure 4.1.

The first observation is that we get a rapid speedup for BLOCKSIZE = 5 and that the saturation effect starts at BLOCKSIZE = 20; i.e., it makes no sense to use more than 20 simultaneous right-hand sides. The gap between the graphs for OCTANE and ORIGIN is caused by the larger secondary cache on the SGI ORIGIN machine so that the cache hit rate was already better for the original code on it. The smaller caches on the LINUX computer moved the saturation point there to BLOCKSIZE = 5.

The speedups for the manually unrolled loops are significantly better than the normal speedups. Therefore, it is worthwhile to take some time implementing it for special applications. The large speedup difference for the LINUX computer may be caused by some weakness in the optimization strategy of the gcc compiler.

The speedup is only a measure of improving a code on a certain computer. The most interesting question is the CPU-time for solving a problem. Figure 4.2 presents the time for solving the systems of equations (1.1) resulting from our test example.

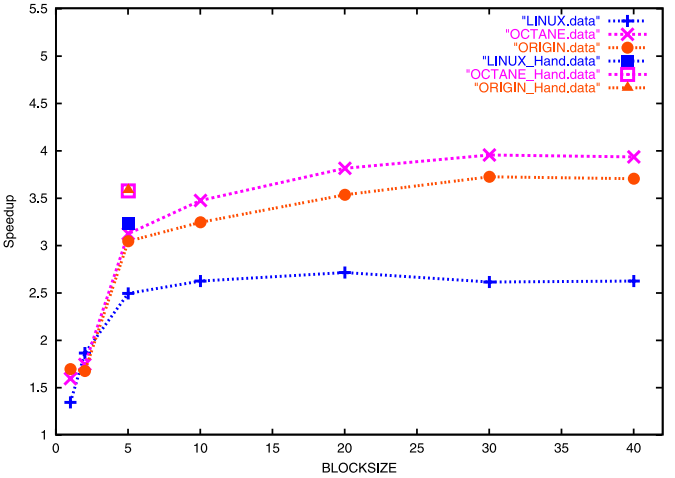


FIG. 4.1. *Overall performance gain with respect to simultaneous right-hand sides.*

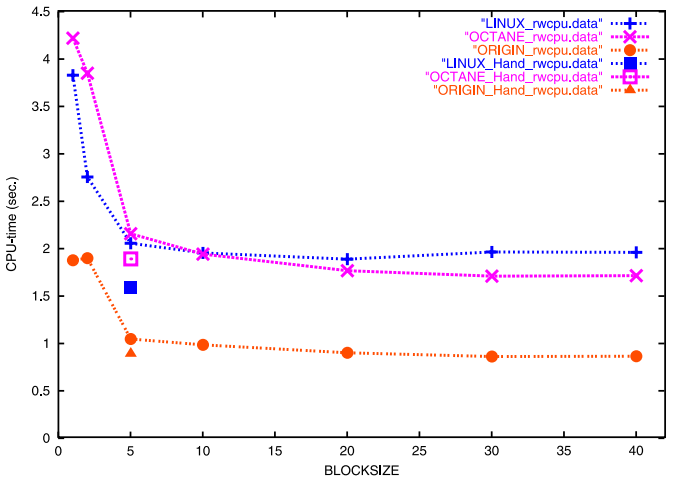


FIG. 4.2. *CPU-time per system of equations with respect to simultaneous right-hand sides.*

The performance gain due to the improved memory management and algorithmic improvements from section 2 can be also seen in Figure 4.1 as performance gain for one right-hand side. This sequential improvement is 1.7 on the ORIGIN, at little less than 1.6 on the OCTANE workstation, and it reduces to 1.4 on the LINUX computer. We did the tuning on the OCTANE computer for 90,000 unknowns, i.e., approximately 25 MB of data, due to the improved memory management and algorithmic improvements from section 2. These acceleration factors were mainly achieved by reducing the cache misses which improved the ratio of arithmetics to data transfer.

4.2. A 2D case study: Multiple right-hand sides. In contrast to the previous section we now consider a time-dependent problem but solve the sequence of arising equations with the technique of subsection 3.2. Let us assume the discretization of a parabolic equation on the unit square with implicit time-stepping and appropriate

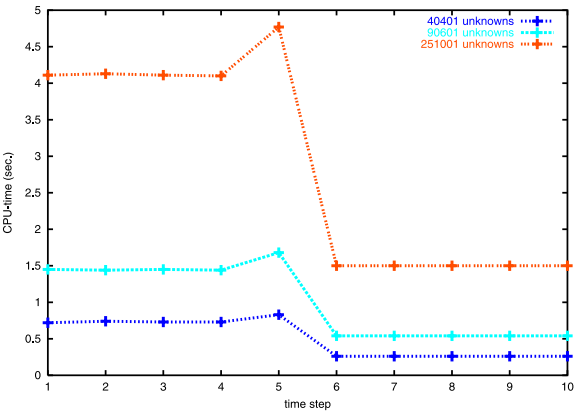


FIG. 4.3. *CPU-time for* 10 *time steps.*

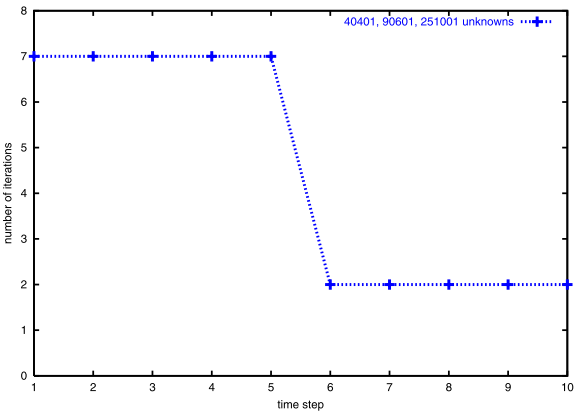


FIG. 4.4. *Number of iterations for* 10 *time steps.*

boundary and initial conditions. Thus we get

$$(\Delta t \cdot K_h + M_h)\underline{u}_h^{i+1} = \underline{f}_h^i$$

with K_h the stiffness matrix, M_h the mass matrix, and Δt the time increment. The right-hand sides have the form

$$(\underline{f}_h^i)_j = \Delta t / N_h - (M_h \underline{u}_h^i)_j,$$

where i denotes the time step. This problem is solved up to a relative accuracy $\epsilon = 10^{-6}$, and $\Delta t = 10^{-1}$.

The required CPU-time and the number of iterations for the time-dependent problem using the Lanczos–Galerkin technique are depicted in Figures 4.3 and 4.4. The overall CPU-time for the solution of the problem with 40,401, 90,601, and 251,001 unknowns is 5.05 seconds, 10.14 seconds, and 28.73 seconds on the LINUX system, respectively. The OCTANE system would result in a similar run time behavior. If we solved the problem without the projection technique, the required CPU-time would be significantly larger, especially for growing N_h. We observed acceleration factors up to 10–20 depending on the problem. Again the optimal AMG preconditioner is required to be h-independent.

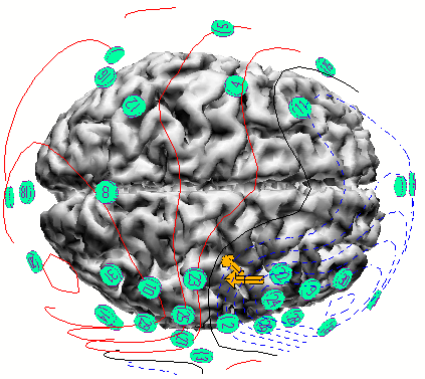


FIG. 4.5. *Dipole fit in the human brain.*

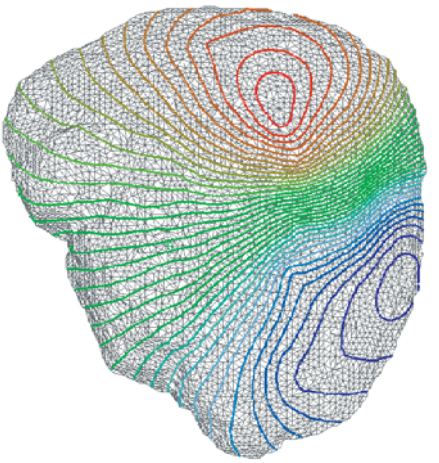


FIG. 4.6. *Solution of one dipole from an FE-discretization.*

4.3. Inverse source localization in the human brain. Finally we present a real life problem for which the technique of simultaneous right-hand sides (see subsection 3.1) is applied. The usage of the techniques is reported for the inverse source localization in the human brain[1] [36]. First, numerical studies of the behavior of our AMG solver applied to this problem are presented in [35, 34].

The problem setting is as follows. In order to determine sources in the human brain from given measurements, a system of linear equations has to be solved for approximately 10,000 right-hand sides within a certain inverse toolbox. The underlying anisotropic potential equation is discretized by the FE-method. The human head is discretized by tetrahedral finite elements, and its mesh contains 118,299 nodes. The resulting stiffness matrix has 1,815,991 nonzero elements. This sums to approximately 40MB + BLOCKSIZE*15MB for storing all matrices and vectors. The rather large number of right-hand sides that must be investigated in the inverse problem is caused by data measurements via magnetoencephalography (MEG) [36, 8, 32, 27, 34]. There exist very good techniques for data from electroencephalography (EEG) that significantly reduce the number of direct problems under consideration [18, 17, 13, 19, 30], and recent research shows that similar techniques can also be applied to MEG [33].

In Figure 4.5 the principle structure of the problem and the computational domain are presented. In Figure 4.6 a typical solution is given for one specific right-hand side. The system of equations with 118,299 unknowns with one right-hand side is solved in 10 seconds on the LINUX and the OCTANE systems, and in 5 seconds on the ORIGIN. The same system of equations with five right-hand sides and manual loop unrolling is solved in 22–23 seconds on LINUX/OCTANE and in 10.5 seconds on the ORIGIN.

Let us assume we could solve a bigger problem on a single processor computer in 50 seconds. From Figure 4.8 (and Figure 4.7) we see that we get a speedup of 2.3 for this real life problem. Equivalently, one right-hand side can be solved in about 21 seconds. Both figures concentrate on the performance gain by simultaneous right-hand sides without the additional acceleration by algorithmic improvements from section 2.

[1] Courtesy of C. Wolters, MPI Leipzig, Germany, and the SimBio project.

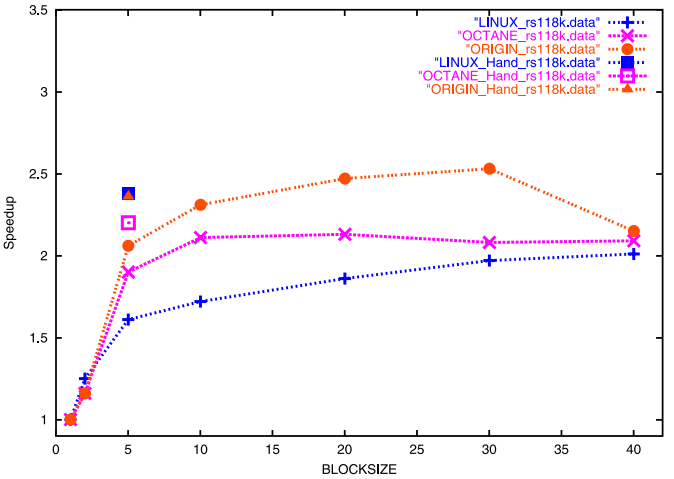


FIG. 4.7. *Performance gain with respect to simultaneous right-hand sides.*

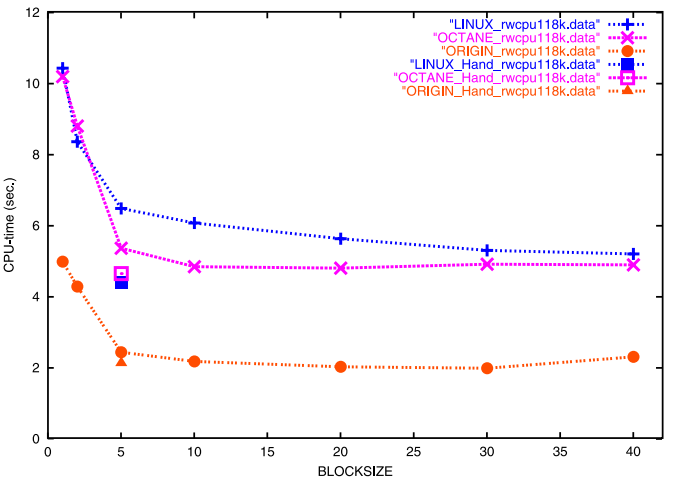


FIG. 4.8. *CPU-time per system of equations with respect to simultaneous right-hand sides.*

REMARK 4.1. 6 [34], 10 12 2.1 10,000 6

5. Conclusion and further remarks. In this paper we give an overview on cache aware techniques for the solution of linear systems with multiple right-hand sides. We have shown that a good AMG preconditioner for the PCG method improves the projection techniques in a straightforward way. Moreover, we can use cache effects from section 3.1 in the case of simultaneous right-hand sides if the vectors are known in advance. Here, the acceleration by these cache effects reaches a factor of 4. The multiple right-hand side approach from section 3.2 is much better suited for an a priori unknown sequence of right-hand sides as can be seen in the experiments in section 4.2.

A straightforward step is the parallelization of the proposed techniques in order to gain the required overall speedup for real life applications [24].

REFERENCES

[1] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[2] T. F. Chan and W. L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[3] J. W. Demmel, J. R. Gilbert, and X. S. Lie, *SuperLU User's Guide*, www.nersc.gov/∼xiaoye/SuperLU/ (October 2003).

[4] Z. Dostal, V. Vondrak, and J. Rasmussen, *Implementation of iterative solvers in shape optimization*, in Structural and Multidisciplinary Optimization, Vol. 1, W. Gutkowski and Z. Mroz, eds., IFTR Warsaw, Warsaw, Poland, 1997, pp. 443–448.

[5] C. C. Douglas, G. Haase, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss, *Portable memory hierarchy techniques for PDE solvers: Part* I, SIAM News, 33, no. 5 (2000), pp. 1, 8–9.

[6] C. C. Douglas, G. Haase, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss, *Portable memory hierarchy techniques for PDE solvers: Part* II, SIAM News, 33, no. 6 (2000), pp. 1, 10–11, 16.

[7] C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss, *Cache optimization for structured and unstructured grid multigrid*, Electron. Trans. Numer. Anal., 10 (2000), pp. 21–40.

[8] M. Dümpelmann, T. Knösche, A. Anwander, and C. Wolters, *Sensitivity of inverse source reconstruction results towards forward model inaccuracies: Application of a universal sensitivity analysis software framework in different tissue conductivity ratios*, in Proceedings of the 13th International Conference on Biomagnetism (BIOMAG 2002), Jena, Germany, 2002, pp. 694–696.

[9] G. Haase, U. Langer, S. Reitzinger, and J. Schöberl, *Algebraic multigrid methods based on element preconditioning*, Int. J. Comput. Math., 78 (2001), pp. 575–598.

[10] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 3rd ed., Kaufmann, San Mateo, CA, 2003.

[12] J. Hu, *Cache Based Multigrid on Unstructured Grids in Two and Three Dimensions*, Ph.D. thesis, Department of Mathematics, University of Kentucky, Lexington, KY, 2000.

[13] C. R. Johnson, M. Mohr, U. Rüde, A. Samsonov, and K. Zyp, *Multilevel Methods for Inverse Bioelectric Field Problems*, in Multiscale and Multiresolution Methods, T.J Barth, T. Chan, and R. Haimes, eds., Lect. Notes Comput. Sci. Eng. 20, Springer-Verlag, 2001, pp. 331–346.

[14] M. Jung and U. Langer, *Applications of multilevel methods to practical problems*, Surveys Math. Indust., 1 (1991), pp. 217–257.

[15] M. Kowarschik, U. Rüde, N. Thürey, and C. Weiss, *Performance optimization of* 3*D multigrid on hierarchical memory architectures*, in Proceedings of the 2002 Conference on Applied Parallel Computing (PARA'02), Espoo, Finland, Lecture Notes in Comput. Sci. 2367, Springer-Verlag, New York, 2002, pp. 307–316.

[16] M. Kowarschik and C. Weiss, *An overview of cache optimization techniques and cache-aware numerical algorithms*, in Proceedings of the GI-Dagstuhl Forschungseminar: Algorithms for Memory Hierarchies, Lecture Notes in Comput. Sci. 2625, Springer-Verlag, New York, 2003, pp. 213–232.

[17] M. Mohr, *Comparison of Solvers for a Bioelectric Field Problem*, Tech. report 01-2, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, 2001.

[18] M. Mohr, C. Popa, and U. Rüde, *An Experimental Analysis of a Differential Inverse Problem*, Tech. report 99–1, Lehrstuhl für Informatik 10 (Systemsimulation), Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany, 1999.

[19] M. Mohr and B. Vanrumste, *Comparing the performance of solvers for a bioelectric field problem*, in Computational Science - ICCS 2002, Part II, Lecture Notes Comput. Sci. 2330, P. Sloot, C. Tan, J. Dongarra, and A. Hoekstra, eds., Springer-Verlag, New York, 2002, pp. 528–537.

[20] D. P. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 292–322.

[21] B. N. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.
[22] PEBBLES, *User's Guide*, Johannes Kepler University Linz, SFB "Numerical and Symbolic Scientific Computing," Linz, Austria, 2002; available online at http://www.numa.uni-linz.ac.at/Research/Projects/pebbles.html.
[23] S. Reitzinger, *Algebraic Multigrid Methods for Large Scale Finite Element Equations*, Schriften der Johannes Kepler Universität Linz, Reihe C–Technik und Naturwissenschaften 36, Universitätsverlag Rudolf Trauner, Linz, Austria, 2001.
[24] S. Reitzinger, U. Schreiber, and U. van Rienen, *Algebraic multigrid for complex symmetric matrices: Numerical studies*, J. Comput. Appl. Math., 155 (2003), pp. 405–421.
[25] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, Frontiers Appl. Math. 5, S. McCormick, ed., SIAM, Philadelphia, 1986, pp. 73–130.
[26] Y. Saad, *On the Lanczos method for solving symmetric linear systems with several right-hand sides*, Math. Comp., 48 (1987), pp. 651–662.
[27] U. Schmidt, A. Louis, C. Wolters, and M. Vauhkonen, *Efficient algorithms for the regularization of dynamic inverse problems.* II. *Applications*, Inverse Problems, 18 (2002), pp. 659–676.
[28] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.
[29] H. A. van der Vorst, *An iterative solution method for solving $f(A)x = b$, using Krylow subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.
[30] B. Vanrumste, M. Mohr, and P. J. Bones, *A multigrid solver in EEG source analysis applying the finite difference method*, International J. of Bioelectromagnetism, 4 (2002), pp. 255–256.
[31] C. Weiss, M. Kowarschik, U. Rüde, and W. Karl, *Cache-aware multigrid methods for solving Poisson's equation in two dimensions*, Computing, 64 (2000), pp. 381–399.
[32] C. Wolters, R. Beckmann, A. Rienäcker, and H. Buchner, *Comparing regularized and non-regularized nonlinear dipole fit methods: A study in a simulated sulcus structure*, Brain Topogr., 12 (1999), pp. 3–18.
[33] C. Wolters, L. Grasedyck, and W. Hackbusch, *Efficient computation of lead field bases and influence matrix for the FEM-based EEG and MEG inverse problem*, Inverse Problems, 20 (2004), pp. 1099–1116.
[34] C. Wolters, M. Kuhn, A. Anwander, and S. Reitzinger, *A parallel algebraic multigrid solver for finite element based source localization in the human brain*, Comput. Vis. Sci., 5 (2002), pp. 165–177.
[35] C. Wolters, S. Reitzinger, A. Basermann, S. Burkhardt, U. Hartmann, F. Kruggel, and A. Anwander, *Improved tissue modeling and fast solver methods for high resolution FE-modeling in EEG/MEG-source localization*, in Proceedings of the 12th International Conference of Biomagnetism, J. Nenonen, R. J. Ilmoniemi, and T. Katila, eds., 2000, pp. 655–658.
[36] C. H. Wolters, *Influence of Tissue Conductivity Inhomogeneity and Anisotropy on EEG/MEG Based Source Localization in the Human Brain*, MPI Series in Cognitive Neuroscience 39, Max-Planck-Institute of Cognitive Neuroscience, Leipzig, Germany, 2003.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 19–42
© 2005 Society for Industrial and Applied Mathematics

AUGMENTED IMPLICITLY RESTARTED LANCZOS BIDIAGONALIZATION METHODS*

JAMES BAGLAMA† AND LOTHAR REICHEL‡

Dedicated to Henk van der Vorst on the occasion of his 60th birthday

Abstract. New restarted Lanczos bidiagonalization methods for the computation of a few of the largest or smallest singular values of a large matrix are presented. Restarting is carried out by augmentation of Krylov subspaces that arise naturally in the standard Lanczos bidiagonalization method. The augmenting vectors are associated with certain Ritz or harmonic Ritz vectors. Computed examples show the new methods to be competitive with available schemes.

Key words. singular value computation, partial singular value decomposition, iterative method, large-scale computation

AMS subject classifications. 65F15, 65F50, 15A18

DOI. 10.1137/04060593X

1. Introduction.

Many problems in scientific computation require knowledge of a few of the largest or smallest singular values of a matrix and associated left and right singular vectors. These problems include the approximation of a large matrix by a matrix of low rank, the computation of the null space of a matrix, total least-squares problems (see, e.g., Björck [4, sections 4.6 and 7.6.5]), as well as the tracking of signals; see, e.g., Comon and Golub [8] for a discussion of the latter.

Let $A \in \mathbb{R}^{\ell\times n}$ be a large sparse matrix. We may assume that $\ell \geq n$, because otherwise we replace the matrix by its transpose. Let

$$\sigma_1^{(A)} \geq \sigma_2^{(A)} \geq \cdots \geq \sigma_n^{(A)} \geq 0 \tag{1.1}$$

denote the singular values of A, and let $u_j^{(A)} \in \mathbb{R}^\ell$ and $v_j^{(A)} \in \mathbb{R}^n$, $1 \leq j \leq n$, be associated left and right singular vectors, respectively. Hence,

$$Av_j^{(A)} = \sigma_j^{(A)} u_j^{(A)}, \qquad A^T u_j^{(A)} = \sigma_j^{(A)} v_j^{(A)}, \qquad 1 \leq j \leq n, \tag{1.2}$$

and

$$A = \sum_{j=1}^{n} \sigma_j^{(A)} u_j^{(A)} (v_j^{(A)})^T.$$

The matrices $U_n^{(A)} = [u_1^{(A)}, u_2^{(A)}, \ldots, u_n^{(A)}]$ and $V_n^{(A)} = [v_1^{(A)}, v_2^{(A)}, \ldots, v_n^{(A)}]$ have orthonormal columns. We refer to $\{\sigma_j^{(A)}, u_j^{(A)}, v_j^{(A)}\}$ as a singular triplet of A. Singular

*Received by the editors March 29, 2004; accepted for publication (in revised form) October 26, 2004; published electronically August 17, 2005.
http://www.siam.org/journals/sisc/27-1/60593.html

†Department of Mathematics, University of Rhode Island, Kingston, RI 02881 (jbaglama@math.uri.edu, http://hypatia.math.uri.edu/∼jbaglama). This author's research was supported in part by NSF grant DMS-0311786.

‡Department of Mathematical Sciences, Kent State University, Kent, OH 44242 (reichel@math.kent.edu, http://www.math.kent.edu/∼reichel). This author's research was supported in part by NSF grant DMS-0107858.

triplets associated with large (small) singular values are referred to as large (small) singular triplets.

This paper presents new restarted Lanczos bidiagonalization methods for computing a few of the largest or smallest singular triplets. The methods compute sequences of projections of A onto judiciously chosen low-dimensional subspaces. Restarting is implemented by augmentation of Krylov subspaces that are determined similarly as in the standard Lanczos bidiagonalization method.

Application of m steps of partial Lanczos bidiagonalization to the matrix A with initial unit vector $p_1 \in \mathbb{R}^n$ yields the decompositions

$$AP_m = Q_m B_m, \tag{1.3}$$

$$A^T Q_m = P_m B_m^T + r_m e_m^T, \tag{1.4}$$

where $P_m \in \mathbb{R}^{n\times m}$, $Q_m \in \mathbb{R}^{\ell\times m}$, $P_m^T P_m = I_m$, $P_m e_1 = p_1$, $Q_m^T Q_m = I_m$, $r_m \in \mathbb{R}^n$, and

$$P_m^T r_m = 0. \tag{1.5}$$

Further, the matrix

$$B_m := \begin{bmatrix} \alpha_1 & \beta_1 & & & & 0 \\ & \alpha_2 & \beta_2 & & & \\ & & \alpha_3 & \beta_3 & & \\ & & & & \ddots & \\ & & & & \ddots & \beta_{m-1} \\ 0 & & & & & \alpha_m \end{bmatrix} \in \mathbb{R}^{m\times m} \tag{1.6}$$

is upper bidiagonal, $I_m \in \mathbb{R}^{m\times m}$ denotes the identity matrix, and e_j denotes the jth axis vector. We refer to the decompositions (1.3)–(1.4) as a partial Lanczos bidiagonalization of A and to the vector r_m in (1.4) as the residual vector; see Björck [4, section 7.6] for a recent discussion on partial Lanczos bidiagonalization. The number of bidiagonalization steps, m, is assumed to be small enough so that the decompositions (1.3)–(1.4) with the stated properties exist.

In applications of interest to us, m is not very large, and the singular triplets $\{\sigma_j^{(B_m)}, u_j^{(B_m)}, v_j^{(B_m)}\}_{j=1}^m$ of B_m can be computed inexpensively by the Golub–Kahan algorithm [10]. Approximate singular triplets of A, denoted by $\{\tilde\sigma_j^{(A)}, \tilde u_j^{(A)}, \tilde v_j^{(A)}\}$, can be determined from the singular triplets of B_m and the matrices P_m and Q_m in the decompositions (1.3)–(1.4); see section 2 for details.

When the matrix A is large, i.e., when ℓ and possibly n are large, the storage requirement of the partial Lanczos bidiagonalization (1.3)–(1.4) is large, unless the number of Lanczos bidiagonalization steps, m, is small. However, for a small value of m, the desired singular triplets of A may be approximated poorly by computed approximate singular triplets $\{\tilde\sigma_j^{(A)}, \tilde u_j^{(A)}, \tilde v_j^{(A)}\}$. In order to circumvent this difficulty, several methods have been proposed that are based on the computation of partial Lanczos bidiagonalizations (1.3)–(1.4) with m small for a sequence of initial vectors p_1; see, e.g., [5, 13, 14, 15, 16, 17, 28]. These methods are commonly referred to as restarted partial Lanczos bidiagonalization methods. We will comment on some of these methods below. They differ in their choice of initial vector p_1 used for the

restarts and in their implementation of the restarting procedure. Ideally, we would like p_1 to be a linear combination of the right singular vectors of A associated with the desired singular values.

Sorensen [29] proposed efficient approaches for restarting the Arnoldi and the Lanczos tridiagonalization procedures. These approaches can be thought of as curtailed QR-algorithms, and, similarly to the QR-algorithms, their performance depends critically on the selection of shifts; see also [1, 7, 21, 30] for discussions and extensions. Björck, Grimme, and Van Dooren [5] derived analogous recursion formulas for a restarted Lanczos bidiagonalization method and presented an application to the solution of ill-posed problems. Recently, Kokiopoulou, Bekas, and Gallopoulos [17] applied these recursion formulas to compute a few desired singular triplets of a large sparse matrix. The shifts are applied by "chasing the bulge" in a curtailed QR-algorithm. Other implementations of restarted Lanczos bidiagonalization are discussed in [13, 14, 15, 16, 28]. The present paper describes mathematically equivalent, but numerically more robust, implementations of the methods discussed by Kokiopoulou, Bekas, and Gallopoulos [17].

This paper is organized as follows. Section 2 reviews Lanczos bidiagonalization and introduces notation used in the remainder of the paper. Section 3 describes our implementations of restarted Lanczos bidiagonalization. Restarting is carried out by augmenting the Krylov subspaces that arise naturally in the standard Lanczos bidiagonalization method by Ritz vectors or harmonic Ritz vectors associated with desired singular triplets. A few computed examples, which compare the performance of several methods and implementations, are presented in section 4.

2. Lanczos bidiagonalization. The following algorithm determines the partial Lanczos bidiagonalization (1.3)–(1.4) of the large sparse matrix $A \in \mathbb{R}^{\ell\times n}$. The number of Lanczos bidiagonalization steps, m, is typically much smaller than either one of the matrix dimensions ℓ and n. Throughout this paper $\|\cdot\|$ denotes the Euclidean vector norm or the associated induced matrix norm. We remark that the methods described can easily be modified to apply to matrices A with complex-valued entries; the main modification required is that the matrices P_m, Q_m, and B_m in (1.3)–(1.4) may have complex-valued entries, and transposition has to be replaced by transposition and complex conjugation. The MATLAB code used for the computed examples of section 4 can be applied to matrices A with complex-valued entries.

ALGORITHM 2.1. LANCZOS BIDIAGONALIZATION.

$A \in \mathbb{R}^{\ell\times n}$
A A^T
$p_1 \in \mathbb{R}^n$:
m :

$P_m = [p_1, p_2, \ldots, p_m] \in \mathbb{R}^{n\times m}$:
$Q_m = [q_1, q_2, \ldots, q_m] \in \mathbb{R}^{\ell\times m}$:
$B_m \in \mathbb{R}^{m\times m}$: (1.6) α_j β_j
$r_m \in \mathbb{R}^n$:

1 $P_1 := p_1$. $q_1 := Ap_1$.
2 $\alpha_1 := \|q_1\|$. $q_1 := q_1/\alpha_1$. $Q_1 := q_1$.
3 $j = 1 : m$
4 $r_j := A^T q_j - \alpha_j p_j$.

5 [illegible] $r_j := r_j - P_j(P_j^T r_j)$.
6 [illegible] $j < m$ [illegible]
7 $\beta_j := \|r_j\|$. $p_{j+1} := r_j/\beta_j$. $P_{j+1} := [P_j, p_{j+1}]$.
8 $q_{j+1} := Ap_{j+1} - \beta_j q_j$.
9 [illegible] $q_{j+1} := q_{j+1} - Q_j(Q_j^T q_{j+1})$.
10 $\alpha_{j+1} := \|q_{j+1}\|$. $q_{j+1} := q_{j+1}/\alpha_{j+1}$. $Q_{j+1} := [Q_j, q_{j+1}]$.
11 [illegible]
12 [illegible]

When the computations with Algorithm 2.1 are carried out in finite precision arithmetic and the columns of P_m and Q_m are not reorthogonalized, the computed columns might be far from orthogonal. We therefore reorthogonalize the columns of these matrices in lines 5 and 9 of the algorithm.

Several reorthogonalization strategies for the columns of the matrices P_m and Q_m are discussed in the literature. Larsen [18] found that when m is fairly large and one is interested in computing a few of the largest singular triplets of A, partial reorthogonalization gives comparable accuracy and requires less computational work than full reorthogonalization, but when a few of the smallest singular triplets are desired, often (essentially) full reorthogonalization is required to achieve high accuracy. Wu and Simon [33] report that when m is not large, full reorthogonalization should be carried out, because due to the overhead associated with partial reorthogonalization, the latter is not competitive. Moreover, Simon and Zha [28] show that when the matrix A is not very ill-conditioned, only the columns of one of the matrices P_m or Q_m need to be reorthogonalized. Reorthogonalization of the columns of P_m only can reduce the computational effort required to compute the partial Lanczos bidiagonalization (1.3)–(1.4) considerably when $\ell \gg n$. Algorithm 2.1 can easily be modified to implement the latter approach.

The Lanczos bidiagonalization algorithm (Algorithm 2.1) requires the diagonal entries $\alpha_j > 0$, $1 \leq j \leq m$, as well as the superdiagonal entries β_j, $1 \leq j < m$, of B_m to be nonvanishing. Assume for the moment that $\alpha_j > 0$ for $1 \leq j < m$, and $\alpha_m = 0$. Then the vector q_m determined in line 8 of the algorithm vanishes, and the decompositions (1.3)–(1.4) can be expressed as

$$AP_m = Q_{m-1}B_{m-1,m}, \qquad A^T Q_{m-1} = P_m B_{m-1,m}^T,$$

where $B_{m-1,m}$ denotes the leading $(m-1) \times m$ submatrix of B_m. It follows that A is singular and that the singular values of $B_{m-1,m}$ are singular values of A. The associated singular triplets of A can be determined from P_m, Q_{m-1}, and the singular triplets of $B_{m-1,m}$. The singular values of $B_{m-1,m}$ are nonvanishing. Moreover, the right singular vector of A associated with the zero singular value can be expressed as a linear combination of the columns of P_m, e.g., by using the singular value decomposition of B_m. However, the corresponding left singular vector of A is not readily available. For simplicity, we will in the remainder of this paper assume that all α_j are positive.

It follows from Algorithm 2.1 that β_m, the last superdiagonal element of the upper bidiagonal matrix $B_{m+1} \in \mathbb{R}^{(m+1)\times(m+1)}$ obtained by $m+1$ steps of partial Lanczos bidiagonalization, is given by

$$\beta_m := \|r_m\| \tag{2.1}$$

and therefore can be computed when the decompositions (1.3)–(1.4) are available. We may assume that $\beta_m > 0$, because otherwise the singular values of B_m are singular

values of A, and the associated singular triplets of A can be determined from the singular value decomposition of B_m and the matrices P_m and Q_m. If $\beta_m = 0$ and the determined singular triplets of A include all the desired ones, then we are done; otherwise we proceed to compute additional singular triplets by the methods described in this paper.

When $\beta_m > 0$, the last column of $P_{m+1} := [P_m, p_{m+1}]$ is given by

$$p_{m+1} := r_m/\beta_m, \tag{2.2}$$

and the relation (1.4) can be expressed as

$$A^T Q_m = P_{m+1} B_{m,m+1}^T, \tag{2.3}$$

where the matrix $B_{m,m+1} \in \mathbb{R}^{m\times(m+1)}$ is obtained by appending the column $\beta_m e_m$ to B_m. In particular, the matrices in the decomposition (2.3) are available after m Lanczos bidiagonalization steps. We also note that $B_{m,m+1}$ is the leading $m \times (m+1)$ submatrix of the matrix B_{m+1} obtained after $m+1$ steps of Lanczos bidiagonalization.

We will use the connection between partial Lanczos bidiagonalization (1.3)–(1.4) of A and partial Lanczos tridiagonalization of the matrix $A^T A$. Multiplying (1.3) by A^T from the left-hand side yields

$$A^T A P_m = P_m B_m^T B_m + r_m e_m^T B_m = P_m B_m^T B_m + \alpha_m r_m e_m^T, \tag{2.4}$$

where the last equality follows from the fact that B_m is upper bidiagonal. The matrix

$$T_m := B_m^T B_m \in \mathbb{R}^{m\times m} \tag{2.5}$$

is symmetric and tridiagonal, and the expression (2.4) is a partial Lanczos tridiagonalization of $A^T A$ with initial vector $p_1 = P_m e_1$; see, e.g., [11, section 9.1.2] for a discussion on Lanczos tridiagonalization. Since T_m is tridiagonal, (2.4) shows that the columns of P_m satisfy a three-term recurrence relation; the columns form an orthonormal basis of the Krylov subspace

$$\mathbb{K}_m(A^T A, p_1) = \operatorname{span}\{p_1, A^T A p_1, (A^T A)^2 p_1, \ldots, (A^T A)^{m-1} p_1\}. \tag{2.6}$$

Similarly, multiplying (1.4) by A from the left-hand side yields

$$A A^T Q_m = Q_m B_m B_m^T + A r_m e_m^T. \tag{2.7}$$

The columns of Q_m form an orthonormal basis of the Krylov subspace

$$\mathbb{K}_m(A A^T, q_1) = \operatorname{span}\{q_1, A A^T q_1, (A A^T)^2 q_1, \ldots, (A A^T)^{m-1} q_1\} \tag{2.8}$$

with $q_1 := A p_1$. We remark that since the columns of Q_m are not, in general, orthogonal to $A r_m$, the decomposition (2.7) typically is not a Lanczos tridiagonalization of $A A^T$.

Let $\{\sigma_j^{(B_m)}, u_j^{(B_m)}, v_j^{(B_m)}\}$, $1 \le j \le m$, be the singular triplets of B_m enumerated so that

$$\sigma_1^{(B_m)} \ge \sigma_2^{(B_m)} \ge \cdots \ge \sigma_m^{(B_m)} \ge 0. \tag{2.9}$$

Then, analogously to (1.2),

$$B_m v_j^{(B_m)} = \sigma_j^{(B_m)} u_j^{(B_m)}, \qquad B_m^T u_j^{(B_m)} = \sigma_j^{(B_m)} v_j^{(B_m)}, \qquad 1 \le j \le m, \tag{2.10}$$

and the $m \times m$ matrices of left and right singular vectors

$$U_m^{(B_m)} := [u_1^{(B_m)}, u_2^{(B_m)}, \ldots, u_m^{(B_m)}], \qquad V_m^{(B_m)} := [v_1^{(B_m)}, v_2^{(B_m)}, \ldots, v_m^{(B_m)}]$$

are orthogonal.

We determine approximate singular triplets $\{\tilde{\sigma}_j^{(A)}, \tilde{u}_j^{(A)}, \tilde{v}_j^{(A)}\}$, $1 \le j \le m$, of A from the singular triplets of B_m by

$$\tilde{\sigma}_j^{(A)} := \sigma_j^{(B_m)}, \qquad \tilde{u}_j^{(A)} := Q_m u_j^{(B_m)}, \qquad \tilde{v}_j^{(A)} := P_m v_j^{(B_m)}. \tag{2.11}$$

Combining (2.10) with (1.3)–(1.4) shows that

$$A\tilde{v}_j^{(A)} = \tilde{\sigma}_j^{(A)} \tilde{u}_j^{(A)}, \quad A^T \tilde{u}_j^{(A)} = \tilde{\sigma}_j^{(A)} \tilde{v}_j^{(A)} + r_m e_m^T u_j^{(B_m)}, \quad 1 \le j \le m. \tag{2.12}$$

The equations (2.12) suggest that an approximate singular triplet $\{\tilde{\sigma}_j^{(A)}, \tilde{u}_j^{(A)}, \tilde{v}_j^{(A)}\}$ be accepted as a singular triplet of A if $r_m e_m^T u_j^{(B_m)}$ is sufficiently small. Specifically, our numerical method accepts $\{\tilde{\sigma}_j^{(A)}, \tilde{u}_j^{(A)}, \tilde{v}_j^{(A)}\}$ as a singular triplet of A if

$$\beta_m |e_m^T u_j^{(B_m)}| \le \delta \|A\| \tag{2.13}$$

for a user-specified value of δ, where we have used (2.1). The quantity $\|A\|$ in (2.13) is easily approximated by the singular value $\sigma_1^{(B_m)}$ of largest magnitude of the bidiagonal matrix B_m. The computation of $\sigma_1^{(B_m)}$ is inexpensive because the matrix B_m is small. During the computations of the desired singular triplets of A, typically, several matrices B_m and their singular value decompositions are computed. We approximate $\|A\|$ by the largest of the singular values of all the matrices B_m generated. This generally gives a good estimate of $\|A\|$.

3. Augmented Lanczos bidiagonalization methods. It is well known that the implicitly restarted Arnoldi and Lanczos tridiagonalization methods described by Sorensen [29] can suffer from numerical instability due to propagated round-off errors. The instability can delay or prevent convergence of desired eigenvalues and eigenvectors; see Lehoucq and Sorensen [21] for a discussion and remedies. Morgan [25] showed that the implicitly restarted Arnoldi method by Sorensen [29] can be implemented by augmenting the available Krylov subspace basis by certain Ritz vectors. Such an implementation can be less sensitive to propagated round-off errors than the implementation in [29]. Recently, Wu and Simon [33] described a so-called thick-restarted Lanczos tridiagonalization method for the symmetric eigenvalue problem. The method is based on augmenting Krylov subspaces by certain Ritz vectors; it is simple to implement and is mathematically equivalent to the implicitly restarted Lanczos tridiagonalization method of Sorensen [29]. This paper presents thick-restarted Lanczos bidiagonalization methods. We remark that thick-restarting techniques have also been used in the context of the Jacobi–Davidson and Arnoldi methods for eigenvalue computations; see Stathopoulos and Saad [31] and Stathopoulos, Saad, and Wu [32] for discussions and analyses.

3.1. Augmentation by Ritz vectors. Let the partial Lanczos bidiagonalization (1.3)–(1.4) be available, and assume that we are interested in determining the k largest singular triplets of A, where $k < m$. Note that the approximate right singular

vector $\tilde{v}_j^{(A)}$ of A, defined by (2.11), is a Ritz vector of A^TA associated with the Ritz value $(\tilde{\sigma}_j^{(A)})^2$. We have

$$A^TA\tilde{v}_j^{(A)} - (\tilde{\sigma}_j^{(A)})^2\tilde{v}_j^{(A)} = \alpha_m r_m e_m^T v_j^{(B_m)}, \qquad 1 \le j \le m. \tag{3.1}$$

The observation that the right-hand sides are parallel to r_m for all j forms the basis of the restarted Lanczos tridiagonalization method by Wu and Simon [33], as well as of the restarted Lanczos bidiagonalization method of this subsection.

We derive decompositions of the form (1.3)–(1.4) which allow us to choose the first k columns of the matrix P_m as Ritz vectors. The accuracy of these approximate left singular vectors and available approximate right singular vectors is then improved by restarting the computations. It follows from the orthonormality of the columns of the matrices P_m and $V_m^{(B_m)}$ that the Ritz vectors $\tilde{v}_j^{(A)}$ defined by (2.11) are orthonormal. Moreover, (1.5) shows that the $\tilde{v}_j^{(A)}$ are orthogonal to r_m.

Let the Ritz vectors $\tilde{v}_j^{(A)}$, $1 \le j \le k$, associated with the k largest Ritz values be available, assume that $r_m \neq 0$, and introduce the matrix

$$\tilde{P}_{k+1} := [\tilde{v}_1^{(A)}, \tilde{v}_2^{(A)}, \ldots, \tilde{v}_k^{(A)}, p_{m+1}]. \tag{3.2}$$

In view of (2.2), the last column of $\tilde{P}_{k+1}$ is parallel to the residual error (3.1). It follows from (2.11) that

$$A\tilde{P}_{k+1} = [\tilde{\sigma}_1^{(A)}\tilde{u}_1^{(A)}, \tilde{\sigma}_2^{(A)}\tilde{u}_2^{(A)}, \ldots, \tilde{\sigma}_k^{(A)}\tilde{u}_k^{(A)}, Ap_{m+1}]. \tag{3.3}$$

Orthogonalization of Ap_{m+1} against the vectors $\tilde{u}_j^{(A)}$ yields

$$Ap_{m+1} = \sum_{j=1}^{k} \tilde{\rho}_j \tilde{u}_j^{(A)} + \tilde{r}_k, \tag{3.4}$$

where the remainder $\tilde{r}_k$ is orthogonal to the vectors $\tilde{u}_j^{(A)}$, $1 \le j \le k$. The coefficients $\tilde{\rho}_j := (\tilde{u}_j^{(A)})^T Ap_{m+1}$ can be evaluated inexpensively by using the right-hand side of

$$(\tilde{u}_j^{(A)})^T Ap_{m+1} = p_{m+1}^T A^T \tilde{u}_j^{(A)} = p_{m+1}^T(\tilde{\sigma}_j^{(A)}\tilde{v}_j^{(A)} + r_m e_m^T u_j^{(B_m)}) = \beta_m e_m^T \tilde{u}_j^{(B_m)}.$$

We may assume that the vector $\tilde{r}_k$ is nonvanishing, because otherwise we can terminate the iterations; see below. Introduce the matrices

$$\tilde{Q}_{k+1} := \left[\tilde{u}_1^{(A)}, \tilde{u}_2^{(A)}, \ldots, \tilde{u}_k^{(A)}, \frac{\tilde{r}_k}{\|\tilde{r}_k\|}\right] \tag{3.5}$$

and

$$\tilde{B}_{k+1} := \begin{bmatrix} \tilde{\sigma}_1^{(A)} & & 0 & \tilde{\rho}_1 \\ & \ddots & & \vdots \\ & & \tilde{\sigma}_k^{(A)} & \tilde{\rho}_k \\ 0 & & & \tilde{\alpha}_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1)\times(k+1)}. \tag{3.6}$$

Thus, $\tilde{B}_{k+1}$ may have nonvanishing entries only on the diagonal and in the last column. Substituting (3.4) into (3.3) now yields the decomposition

$$A\tilde{P}_{k+1} = \tilde{Q}_{k+1}\tilde{B}_{k+1}, \tag{3.7}$$

which is analogous to (1.3).

We turn to the matrix

$$A^T\tilde{Q}_{k+1} = \left[A^T\tilde{u}_1^{(A)}, A^T\tilde{u}_1^{(A)}, \ldots, A^T\tilde{u}_k^{(A)}, A^T\frac{\tilde{r}_k}{\|\tilde{r}_k\|}\right], \tag{3.8}$$

which we would like to express in terms of $\tilde{P}_{k+1}$ and $\tilde{B}_{k+1}^T$. This will give an analogue of the decomposition (1.4). The first k columns of (3.8) are linear combinations of the vectors $\tilde{v}_j^{(A)}$ and p_{m+1}; specifically,

$$A^T\tilde{u}_j^{(A)} = \tilde{\sigma}_j^{(A)}\tilde{v}_j^{(A)} + r_m e_m^T u_j^{(B_m)} = \tilde{\sigma}_j^{(A)}\tilde{v}_j^{(A)} + p_{m+1}\tilde{\rho}_j, \qquad 1 \le j \le k, \tag{3.9}$$

where we have used (2.2). The last column of (3.8) is orthogonal to the Ritz vectors $\tilde{v}_j^{(A)}$,

$$(\tilde{v}_j^{(A)})^T A^T\frac{\tilde{r}_k}{\|\tilde{r}_k\|} = \tilde{\sigma}_j^{(A)}\frac{(\tilde{u}_j^{(A)})^T\tilde{r}_k}{\|\tilde{r}_k\|} = 0, \qquad 1 \le j \le k,$$

and therefore it can be expressed as

$$A^T\frac{\tilde{r}_k}{\|\tilde{r}_k\|} = \tilde{\gamma}_1 p_{m+1} + \tilde{f}_{k+1}, \tag{3.10}$$

where $\tilde{f}_{k+1} \in \mathbb{R}^n$ is orthogonal to the vectors $\tilde{v}_j^{(A)}$, $1 \le j \le k$, as well as to p_{m+1}. Since

$$p_{m+1}^T A^T\frac{\tilde{r}_k}{\|\tilde{r}_k\|} = \frac{\tilde{r}_k^T}{\|\tilde{r}_k\|}Ap_{m+1} = \frac{\tilde{r}_k^T}{\|\tilde{r}_k\|}\left(\sum_{j=1}^{k}\tilde{\rho}_j\tilde{u}_j^{(A)} + \tilde{r}_k\right) = \|\tilde{r}_k\|,$$

it follows from (3.10) that $\tilde{\gamma}_1 = \|\tilde{r}_k\|$. This observation, together with (3.9) and (3.10), gives the expression

$$A^T\tilde{Q}_{k+1} = \tilde{P}_{k+1}\tilde{B}_{k+1}^T + \tilde{f}_{k+1}e_{k+1}^T, \tag{3.11}$$

which is the desired analogue of the decomposition (1.4). We remark that $\tilde{f}_{k+1}$ can be computed from (3.10).

The similarity of the decompositions (3.7) and (3.11), and (1.3)–(1.4), suggests that it may be possible to append new columns to the matrices $\tilde{P}_{k+1}$ and $\tilde{Q}_{k+1}$ in a way similar to Lanczos bidiagonalization. We will show that this is indeed the case. For notational simplicity, denote the columns of $\tilde{P}_{k+1}$ and $\tilde{Q}_{k+1}$ by $\tilde{p}_j$ and $\tilde{q}_j$, respectively, i.e.,

$$\tilde{P}_{k+1} = [\tilde{p}_1, \tilde{p}_2, \ldots, \tilde{p}_{k+1}] \in \mathbb{R}^{n\times(k+1)}, \qquad \tilde{Q}_{k+1} = [\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_{k+1}] \in \mathbb{R}^{\ell\times(k+1)}.$$

We may assume that $\tilde{f}_{k+1} \neq 0$, because otherwise it follows from (3.7) and (3.11) that the singular values of $\tilde{B}_{k+1}$ are singular values of A, and we are done. Thus, let $\tilde{\beta}_{k+1} := \|\tilde{f}_{k+1}\|$ and $\tilde{p}_{k+2} := \tilde{f}_{k+1}/\tilde{\beta}_{k+1}$. Then the matrix $\tilde{P}_{k+2} := [\tilde{P}_{k+1}, \tilde{p}_{k+2}]$ has orthonormal columns.

Let

$$\tilde{\alpha}_{k+2}\tilde{q}_{k+2} := (I - \tilde{Q}_{k+1}\tilde{Q}_{k+1}^T)A\tilde{p}_{k+2}, \tag{3.12}$$

where $\tilde{\alpha}_{k+2} > 0$ is a scaling factor, such that $\tilde{q}_{k+2}$ is of unit length. We comment on the possibility of $\tilde{\alpha}_{k+2}$ vanishing below. Equation (3.11) yields

$$A^T \tilde{Q}_{k+1} = \tilde{P}_{k+1} \tilde{B}_{k+1}^T + \tilde{\beta}_{k+1} \tilde{p}_{k+2} e_{k+1}^T, \tag{3.13}$$

and substituting (3.13) into (3.12) shows that

$$\begin{aligned} \tilde{\alpha}_{k+2} \tilde{q}_{k+2} &= A\tilde{p}_{k+2} - \tilde{Q}_{k+1}(A^T \tilde{Q}_{k+1})^T \tilde{p}_{k+2} \\ &= A\tilde{p}_{k+2} - \tilde{Q}_{k+1}(\tilde{B}_{k+1} \tilde{P}_{k+1}^T + \tilde{\beta}_{k+1} e_{k+1} \tilde{p}_{k+2}^T)\tilde{p}_{k+2} \\ &= A\tilde{p}_{k+2} - \tilde{\beta}_{k+1} \tilde{Q}_{k+1} e_{k+1} = A\tilde{p}_{k+2} - \tilde{\beta}_{k+1} \tilde{q}_{k+1}. \end{aligned} \tag{3.14}$$

Let $\tilde{Q}_{k+2} := [\tilde{Q}_{k+1}, \tilde{q}_{k+2}] \in \mathbb{R}^{\ell \times (k+2)}$, and define $\tilde{B}_{k+2} \in \mathbb{R}^{(k+2)\times(k+2)}$ by first appending the column $\tilde{\beta}_{k+1} e_{k+1}$ and then the row $\tilde{\alpha}_{k+2} e_{k+2}^T$ to $\tilde{B}_{k+1}$, i.e.,

$$\tilde{B}_{k+2} := \begin{bmatrix} \tilde{\sigma}_1^{(A)} & & 0 & \tilde{\rho}_1 & 0 \\ & \ddots & & \vdots & \vdots \\ & & \tilde{\sigma}_k^{(A)} & \tilde{\rho}_k & 0 \\ & & & \tilde{\alpha}_{k+1} & \tilde{\beta}_{k+1} \\ 0 & & & & \tilde{\alpha}_{k+2} \end{bmatrix}.$$

It now follows from (3.7) and (3.14) that

$$A\tilde{P}_{k+2} = \tilde{Q}_{k+2} \tilde{B}_{k+2}. \tag{3.15}$$

We turn to the derivation of a decomposition of the form (3.13) with $k+1$ replaced by $k+2$. Let

$$\tilde{\beta}_{k+2} \tilde{p}_{k+3} := (I - \tilde{P}_{k+2} \tilde{P}_{k+2}^T) A^T \tilde{q}_{k+2}, \tag{3.16}$$

where $\tilde{\beta}_{k+2} > 0$ is a scaling factor, such that $\tilde{p}_{k+3}$ is of unit length. We may assume that a positive coefficient $\tilde{\beta}_{k+2}$ exists; otherwise we are done; see below. Substituting (3.15) into (3.16) yields

$$\tilde{\beta}_{k+2} \tilde{p}_{k+3} = A^T \tilde{q}_{k+2} - \tilde{P}_{k+2} \tilde{B}_{k+2}^T e_{k+2} = A^T \tilde{q}_{k+2} - \tilde{\alpha}_{k+2} \tilde{p}_{k+2},$$

which, together with (3.13), shows that

$$A^T \tilde{Q}_{k+2} = \tilde{P}_{k+2} \tilde{B}_{k+2}^T + \tilde{\beta}_{k+2} \tilde{p}_{k+3} e_{k+2}^T. \tag{3.17}$$

The decompositions (3.15) and (3.17) are analogous to (3.7) and (3.13). We therefore can continue in the same fashion by appending new columns to the matrices $\tilde{P}_j$ and $\tilde{Q}_j$, and new rows and columns to the matrices $\tilde{B}_j$, for $j = k+2, k+3, \ldots$. After a total of $m-k$ steps, we obtain the decompositions

$$A\tilde{P}_m = \tilde{Q}_m \tilde{B}_m, \qquad A^T \tilde{Q}_m = \tilde{P}_m \tilde{B}_m^T + \tilde{\beta}_m \tilde{p}_{m+1} e_m^T, \tag{3.18}$$

where the matrices $\tilde{P}_m$ and $\tilde{Q}_m$ have orthonormal columns and

$$\tilde{B}_m = \begin{bmatrix} \tilde{\sigma}_1^{(A)} & & 0 & \tilde{\rho}_1 & & 0 \\ & \ddots & & \vdots & & \\ & & \tilde{\sigma}_k^{(A)} & \tilde{\rho}_k & & \\ & & & \tilde{\alpha}_{k+1} & \tilde{\beta}_{k+1} & \\ & & & & \ddots & \\ & & & & \ddots & \tilde{\beta}_{m-1} \\ 0 & & & & & \tilde{\alpha}_m \end{bmatrix}.$$

The method now proceeds by first computing the singular value decomposition of $\tilde{B}_m$ and then determining the k largest approximate singular triplets of A from the k largest singular triplets of $\tilde{B}_m$; cf. (2.11). These triplets define new decompositions of the form (3.7) and (3.11), from which we compute new decompositions of the form (3.18). The computations proceed in this manner until sufficiently accurate approximations of the k largest singular triplets of A have been determined. An algorithm is presented in subsection 3.3 below.

We remark that the computations are analogous for determining the k smallest singular triplets of A. The vectors $\tilde{v}_j^{(A)}$, $1 \leq j \leq k$, in (3.2) should then be replaced by the right approximate singular vectors of the k smallest available approximate singular triplets of A. These approximate singular triplets are used to define decompositions (3.7) and (3.11), from which we compute decompositions (3.18) in the same manner as described above. We are interested in the k smallest singular triplets of the matrix $\tilde{B}_m$ in the decompositions (3.18). These triplets yield approximations of the k smallest triplets of A. The computations are continued until sufficiently accurate approximations of the k smallest singular triplets of A have been found. However, we note that when the k smallest singular triplets of A are desired, it can be advantageous to augment by harmonic Ritz vectors instead. This is discussed in subsection 3.2.

Finally, we comment on the cases when $\tilde{r}_k$ vanishes in (3.4) and when the left-hand sides of (3.12) and (3.16) vanish. In all these cases, one can show that the singular values of the matrix $\tilde{B}_j$ also are singular values of A and that the singular triplets of $\tilde{B}_j$ yield singular triplets of A.

3.2. Augmentation by harmonic Ritz vectors. Augmenting by Ritz vectors, as described in the previous subsection, or, equivalently, shifting by Ritz values, often gives good approximations to the largest singular triplets of A. However, Kokiopoulou, Bekas, and Gallopoulos [17] observed that when seeking to compute the smallest singular triplets of A, shifting by harmonic Ritz values can give faster convergence than shifting by Ritz values. This section describes how shifting by harmonic Ritz values can be implemented via augmentation by harmonic Ritz vectors. Harmonic Ritz vectors are approximate eigenvectors of A^TA associated with harmonic Ritz values of A^TA.

Let the partial Lanczos bidiagonalization (1.3)–(1.4) of A be available, and assume that all the diagonal and superdiagonal entries of B_m, as well as β_m given by (2.1), are nonvanishing. Then, in particular, B_m is nonsingular. The harmonic Ritz values $\hat{\theta}_j$ of A^TA associated with the partial Lanczos tridiagonalization (2.4) are the eigenvalues

of the generalized eigenvalue problem

$$((B_m^T B_m)^2 + \alpha_m^2 \beta_m^2 e_m e_m^T)\hat{w}_j = \hat{\theta}_j B_m^T B_m \hat{w}_j, \qquad 1 \le j \le m, \tag{3.19}$$

with $\hat{w}_j \in \mathbb{R}^m \backslash \{0\}$; see, e.g., Morgan [24] or Paige, Parlett, and van der Vorst [27] for properties of harmonic Ritz values.

The eigenpairs $\{\hat{\theta}_j, \hat{w}_j\}$ of (3.19) can be computed without forming the matrix $B_m^T B_m$ as follows. Let

$$w_j := B_m \hat{w}_j. \tag{3.20}$$

Then (3.19) can be expressed as

$$(B_m B_m^T + \beta_m^2 e_m e_m^T) w_j = \hat{\theta}_j w_j, \tag{3.21}$$

where we may choose the eigenvectors w_j to be orthonormal. Let $B_{m,m+1}$ be the matrix in (2.3), and note that

$$B_{m,m+1} B_{m,m+1}^T = B_m B_m^T + \beta_m^2 e_m e_m^T. \tag{3.22}$$

Introduce the singular triplets $\{\sigma_j^{(B_{m,m+1})}, u_j^{(B_{m,m+1})}, v_j^{(B_{m,m+1})}\}$, $1 \le j \le m$, of the matrix $B_{m,m+1}$, and let them be enumerated so that

$$0 < \sigma_1^{(B_{m,m+1})} \le \sigma_2^{(B_{m,m+1})} \le \cdots \le \sigma_m^{(B_{m,m+1})}. \tag{3.23}$$

This enumeration differs from the one for the singular values of B_m (cf. (2.9)), because in the present subsection we are concerned with the computation of the $k < m$ smallest singular triplets of A. Throughout this subsection, we use the following simplified notation:

$$\sigma_j' = \sigma_j^{(B_{m,m+1})}, \qquad u_j' = u_j^{(B_{m,m+1})}, \qquad v_j' = v_j^{(B_{m,m+1})}.$$

The k smallest singular triplets of $B_{m,m+1}$ determine the matrices

$$\begin{aligned} U_k' &:= [u_1', u_2', \ldots, u_k'] \in \mathbb{R}^{m \times k}, \\ V_k' &:= [v_1', v_2', \ldots, v_k'] \in \mathbb{R}^{(m+1) \times k}, \\ \Sigma_k' &:= \mathrm{diag}[\sigma_1', \sigma_2', \ldots, \sigma_k'] \in \mathbb{R}^{k \times k}, \end{aligned} \tag{3.24}$$

where U_k' and V_k' have orthonormal columns, and

$$B_{m,m+1} V_k' = U_k' \Sigma_k', \qquad B_{m,m+1}^T U_k' = V_k' \Sigma_k'. \tag{3.25}$$

We refer to (3.25) as a partial singular value decomposition of $B_{m,m+1}$. It follows from (3.22) and (3.25) that $\{(\sigma_j')^2, u_j'\}$ is an eigenpair of (3.21), and (3.20) shows that $\{(\sigma_j')^2, B_m^{-1} u_j'\}$ is an eigenpair of (3.19). Thus, the eigenpairs of (3.19) and (3.21) associated with the k smallest eigenvalues can be determined from the partial singular value decomposition (3.25). Gu and Eisenstat [12] describe how the singular value decomposition of $B_{m,m+1}$ can be computed by updating the singular value decomposition of B_m; see also Bunch and Nielsen [6]. However, when A is large and m is small, the computational effort required for determining the singular value decompositions of B_m and $B_{m,m+1}$ is negligible. We will therefore not dwell on the computation of (3.25).

The harmonic Ritz vector of A^TA associated with the harmonic Ritz value $\hat{\theta}_j$ is given by

$$\hat{v}_j := P_m \hat{w}_j; \tag{3.26}$$

see, e.g., [24, 27]. Morgan and Zeng [26] recently pointed out that the residual errors associated with different harmonic Ritz pairs $\{\hat{\theta}_j, \hat{v}_j\}$ are parallel. We show this result for the problem at hand, because this property is central for our augmentation method. Thus, using (2.4), (3.20), (3.21), and (3.26), we obtain

$$\begin{aligned} A^TA\hat{v}_j - \hat{\theta}_j\hat{v}_j &= A^TAP_m\hat{w}_j - \hat{\theta}_jP_m\hat{w}_j \\ &= (P_mB_m^TB_m + \alpha_m r_m e_m^T)\hat{w}_j - \hat{\theta}_jP_m\hat{w}_j \\ &= P_m(B_m^TB_m - \hat{\theta}_jI_m)\hat{w}_j + \alpha_m r_m e_m^T\hat{w}_j \\ &= P_mB_m^{-1}(B_mB_m^T - \hat{\theta}_jI_m)w_j + r_m e_m^Tw_j \\ &= P_mB_m^{-1}(-\beta_m^2 e_m e_m^Tw_j) + r_m e_m^Tw_j \\ &= e_m^Tw_j(r_m - \beta_m^2 P_mB_m^{-1}e_m). \end{aligned}$$

It is convenient to define the scaled residual vector

$$\hat{r}_m := p_{m+1} - \beta_m P_m B_m^{-1} e_m, \tag{3.27}$$

where we have used (2.2).

We are in a position to derive relations analogous to (3.7) and (3.11) for harmonic Ritz vectors. Equations (3.20), (3.26), and (3.27) yield

$$[\hat{v}_1\sigma_1', \hat{v}_2\sigma_2', \ldots, \hat{v}_k\sigma_k', \hat{r}_m] = P_{m+1}\begin{bmatrix} B_m^{-1}U_k'\Sigma_k' & -\beta_m B_m^{-1}e_m \\ 0 & 1 \end{bmatrix},$$

where the matrix P_{m+1} is the same as in (2.3). Introduce the QR-factorization

$$\begin{bmatrix} B_m^{-1}U_k'\Sigma_k' & -\beta_m B_m^{-1}e_m \\ 0 & 1 \end{bmatrix} = Q_{k+1}'R_{k+1}', \tag{3.28}$$

where $Q_{k+1}' \in \mathbb{R}^{(m+1)\times(k+1)}$ has orthonormal columns and $R_{k+1}' \in \mathbb{R}^{(k+1)\times(k+1)}$ is upper triangular.

Consider the matrix

$$\hat{P}_{k+1} = [\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_{k+1}] := P_{m+1}Q_{k+1}'. \tag{3.29}$$

Its columns $\hat{p}_j$ are orthonormal since both P_{m+1} and Q_{k+1}' have orthonormal columns. Equations (1.3) and (3.28) yield

$$\begin{aligned} A\hat{P}_{k+1} &= [AP_m, Ap_{m+1}]Q_{k+1}' \\ &= [Q_mB_m, Ap_{m+1}]\begin{bmatrix} B_m^{-1}U_k'\Sigma_k' & -\beta_m B_m^{-1}e_m \\ 0 & 1 \end{bmatrix}(R_{k+1}')^{-1} \\ &= [Q_mU_k'\Sigma_k', -\beta_m q_m + Ap_{m+1}](R_{k+1}')^{-1}. \end{aligned}$$

In the derivation of our method, we assume only that the matrix B_m is upper triangular, because B_m has this property after restarts. Below we comment on possible simplifications when B_m is upper bidiagonal.

Let

$$\hat{Q}_k := Q_mU_k'. \tag{3.30}$$

The columns of this matrix are orthonormal, and we define the Fourier coefficients

$$\hat{c}_k = [\hat{\gamma}_1, \hat{\gamma}_2, \ldots, \hat{\gamma}_k]^T := \hat{Q}_k^T(-\beta_m q_m + A p_{m+1}).$$

The vector

$$\hat{\alpha}_{k+1}\hat{q}_{k+1} := -\beta_m q_m + A p_{m+1} - \hat{Q}_k \hat{c}_k$$

is orthogonal to the columns of $\hat{Q}_k$, and the scaling factor $\hat{\alpha}_{k+1} > 0$ is chosen so that $\hat{q}_{k+1}$ is of unit length. It follows that

$$A\hat{P}_{k+1} = \hat{Q}_{k+1}\hat{B}_{k+1}, \tag{3.31}$$

where

$$\hat{Q}_{k+1} := [\hat{Q}_k, \hat{q}_{k+1}] \in \mathbb{R}^{\ell\times(k+1)}, \tag{3.32}$$

$$\hat{B}_{k+1} := \begin{bmatrix} \sigma_1' & & & 0 & \hat{\gamma}_1 \\ & \sigma_2' & & & \hat{\gamma}_2 \\ & & \ddots & & \vdots \\ & & & \sigma_k' & \hat{\gamma}_k \\ 0 & & & & \hat{\alpha}_{k+1} \end{bmatrix} (R_{k+1}')^{-1} \in \mathbb{R}^{(k+1)\times(k+1)}. \tag{3.33}$$

Thus, $\hat{Q}_{k+1}$ has orthonormal columns, and $\hat{B}_{k+1}$ is the product of two upper triangular matrices, one of which has nonzero entries only on the diagonal and in the last column. In particular, $\hat{B}_{k+1}$ is upper triangular. The decomposition (3.31) is the desired analogue of (3.7). When B_m is given by (1.6), one can show that $\hat{\alpha}_{k+1} = \alpha_{m+1}$, $\hat{q}_{k+1} = q_{m+1}$, and $\hat{c}_k = 0$.

We now derive an analogue of the decomposition (3.11). Let $\hat{Q}_k$ be given by (3.30). Then (3.25) yields

$$A^T\hat{Q}_k = A^T Q_m U_k' = P_{m+1}B_{m,m+1}^T U_k' = P_{m+1}V_k'\Sigma_k'. \tag{3.34}$$

It follows from the decomposition on the left-hand side of (3.25) that

$$[I_m, \beta_m B_m^{-1} e_m]V_k' = B_m^{-1}U_k'\Sigma_k',$$

and therefore

$$V_k' = \begin{bmatrix} B_m^{-1}U_k'\Sigma_k' & -\beta_m B_m^{-1}e_m \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I_k \\ e_{m+1}^T V_k' \end{bmatrix}. \tag{3.35}$$

Substituting (3.35) into (3.34), using (3.28) and (3.29), gives

$$A^T\hat{Q}_k = P_{m+1}Q_{k+1}'R_{k+1}' \begin{bmatrix} I_k \\ e_{m+1}^T V_k' \end{bmatrix} \Sigma_k' = \hat{P}_{k+1}R_{k+1}' \begin{bmatrix} I_k \\ e_{m+1}^T V_k' \end{bmatrix} \Sigma_k'. \tag{3.36}$$

Let $\hat{B}_{k,k+1}$ be the leading $k \times (k+1)$ submatrix of the upper triangular matrix $\hat{B}_{k+1}$ in (3.31). Then (3.31) yields

$$\hat{Q}_k^T A\hat{P}_{k+1} = \hat{B}_{k,k+1}. \tag{3.37}$$

It follows from (3.36) that

$$\hat{P}_{k+1}^T A^T \hat{Q}_k = R'_{k+1} \begin{bmatrix} I_k \\ e_{m+1}^T V'_k \end{bmatrix} \Sigma'_k,$$

and comparison with (3.37) shows that

$$R'_{k+1} \begin{bmatrix} I_k \\ e_{m+1}^T V'_k \end{bmatrix} \Sigma'_k = \hat{B}_{k,k+1}^T.$$

Hence, (3.36) can be expressed as

$$A^T \hat{Q}_k = \hat{P}_{k+1} \hat{B}_{k,k+1}^T. \tag{3.38}$$

We turn to the last column, $A^T \hat{q}_{k+1}$, of $A^T \hat{Q}_{k+1}$. Equation (3.31) yields

$$\hat{P}_{k+1}^T A^T \hat{q}_{k+1} = \hat{B}_{k+1}^T \hat{Q}_{k+1}^T \hat{q}_{k+1} = \hat{B}_{k+1}^T e_{k+1} = \hat{\alpha}_{k+1} e_{k+1},$$

where $\hat{\alpha}_{k+1}$ denotes the last diagonal entry of $\hat{B}_{k+1}$. Thus,

$$A^T \hat{q}_{k+1} = \hat{\alpha}_{k+1} \hat{p}_{k+1} + \breve{r}_{k+1}, \tag{3.39}$$

where $\hat{P}_{k+1}^T \breve{r}_{k+1} = 0$. Combining (3.38) and (3.39) yields

$$A^T \hat{Q}_{k+1} = \hat{P}_{k+1} \hat{B}_{k+1}^T + \breve{r}_{k+1} e_{k+1}^T, \tag{3.40}$$

which is the desired analogue of (3.11). Note that the residual vector $\breve{r}_{k+1}$ can be computed from (3.39), because the other terms are explicitly known.

Let $\hat{\beta}_{k+1} := \|\breve{r}_{k+1}\|$ and $\hat{p}_{k+2} := \breve{r}_{k+1}/\hat{\beta}_{k+1}$, and define

$$\hat{P}_{k+2} := [\hat{P}_{k+1}, \hat{p}_{k+2}], \qquad \breve{B}_{k+1,k+2} := [\hat{B}_{k+1}, \hat{\beta}_{k+1} e_{k+1}].$$

Then (3.40) can be written in the form

$$A^T \hat{Q}_{k+1} = \hat{P}_{k+2} \breve{B}_{k+1,k+2}^T,$$

which is an analogue of (2.3).

Given the decompositions (3.31) and (3.40), we can proceed analogously as in subsection 3.1 to compute the decompositions

$$A\hat{P}_m = \hat{Q}_m \hat{B}_m, \qquad A^T \hat{Q}_m = \hat{P}_m \hat{B}_m^T + \breve{r}_m e_m^T, \tag{3.41}$$

where $\hat{P}_m \in \mathbb{R}^{n\times m}$ has orthonormal columns with leading $n \times (k+2)$ submatrix $\hat{P}_{k+2}$, $\hat{Q}_m \in \mathbb{R}^{\ell\times m}$ has orthonormal columns with leading $\ell \times (k+1)$ submatrix $\hat{Q}_{k+1}$, and

$$\hat{B}_m = \begin{bmatrix} \hat{B}_{k+1} & \hat{\beta}_{k+1} & & & & 0 \\ & \hat{\alpha}_{k+2} & \hat{\beta}_{k+2} & & & \\ & & \hat{\alpha}_{k+3} & \hat{\beta}_{k+3} & & \\ & & & & \ddots & \\ & & & & \ddots & \hat{\beta}_{m-1} \\ 0 & & & & & \hat{\alpha}_m \end{bmatrix} \in \mathbb{R}^{m\times m}$$

has the $(k+1)\times(k+1)$ leading principal submatrix $\hat{B}_{k+1}$ and an $(m-k-1)\times(m-k-1)$ upper bidiagonal trailing principal submatrix. The residual vector $\breve{r}_m$ is orthogonal to the columns of $\hat{P}_m$. The method of this subsection can be restarted by letting $B_m := \hat{B}_m$, $P_m := \hat{P}_m$, $Q_m := \hat{Q}_m$, $r_m := \breve{r}_m$, and $\beta_m := \|\breve{r}_m\|$.

We remark that the accurate computation of the vector $B_m^{-1}e_m$, used in (3.28), can be difficult when the matrix B_m has a large condition number $\kappa(B_m) := \sigma_m^{(B_m)}/\sigma_1^{(B_m)}$, where the singular values are enumerated as in (3.23). In this case, we switch from augmentation by harmonic Ritz vectors to augmentation by Ritz vectors. Details are provided in the following subsection.

3.3. An augmented Lanczos bidiagonalization algorithm. We describe an algorithm for the computation of a few of the largest or smallest singular triplets of a large matrix A. The algorithm is based on augmentation by Ritz vectors or harmonic Ritz vectors as described in the previous subsections. The Boolean variable *harmonic* suggests the type of augmentation used. If *harmonic* is true and B_m is not too ill-conditioned, then the augmentation scheme of subsection 3.2 is applied; otherwise augmentation is carried out according to subsection 3.1.

ALGORITHM 3.1. AUGMENTED LANCZOS BIDIAGONALIZATION.

$A \in \mathbb{R}^{\ell\times n}$
A A^T
$p_1 \in \mathbb{R}^n$:
m :
k :
δ : (2.13)
ϵ :
harmonic :

$\{\sigma_j, u_j, v_j\}_{j=1}^k$ A

1 (1.3)–(1.4) 2.1
2 (2.10) B_m
3 k (2.13)
4
4a *harmonic* $\kappa(B_m) > \epsilon^{-1/2}$
$P := \tilde{P}_{k+1}$ $Q := \tilde{Q}_{k+1}$ $B := \tilde{B}_{k+1}$
$r := \tilde{f}_{k+1}$ (3.2), (3.5), (3.6) (3.10)
4b *harmonic* $\kappa(B_m) \le \epsilon^{-1/2}$
(3.24) $B_{m,m+1}$
(3.28)
$P := \hat{P}_{k+1}$ $Q := \hat{Q}_{k+1}$ $B := \hat{B}_{k+1}$
$r := \breve{r}_{k+1}$ (3.29), (3.32), (3.33) (3.39)
5 P Q B r

$$AP = QB, \qquad A^TQ = PB^T + re_k^T.$$

$m-k$ P Q $m-k$
B P_m Q_m B_m
r_m
6 2

The above algorithm is a simplification of the actual computations carried out. For instance, the algorithm exits when a matrix B_m that is numerically singular has been detected, but the available decompositions of A can be used to determine singular triplets of A; see the discussion in section 1. Moreover, the number of augmented vectors used at each restart is typically larger than the number of desired singular triplets. Assume that k' of the desired k singular triplets have been found. We then augment by $k + k''$ (instead of k) singular triplets, where k'' is chosen as large as possible, such that $k'' \leq k'$ and $k + k'' \leq m - 3$. The term -3 secures that at least three orthogonalization steps can be carried out between restarts. This approach has been advocated by Lehoucq [20] in the context of the implicitly restarted Arnoldi method. It often yields faster convergence without increasing the memory requirement. Finally, our implementation enforces two-sided reorthogonalization when $\kappa(B_m) > \epsilon^{-1/2}$. MATLAB code is available at the authors' home pages.

4. Numerical examples. All computations were carried out using MATLAB version 6.5 R13 on a Dell 530 workstation with two 2.4 GHz (512k cache) Xeon processors and 2 GB (400 MHz) of memory running under the Windows XP operating system. Machine epsilon is $\epsilon = 2.2 \cdot 10^{-16}$.

We compare our methods, outlined by Algorithm 3.1, with methods recently proposed by Hochstenbach [14, 13] and Kokiopoulou, Bekas, and Gallopoulos [17], as well as with the MATLAB internal function `svds` and the scheme proposed in [2]. Hochstenbach [14, 13] presents a Jacobi–Davidson method. This is a powerful scheme when a good preconditioner for the linear system of equations that has to be solved is available. In our computed examples, we assume that no good preconditioner is known, and we apply the method without preconditioner. The linear system of equations is solved by the GMRES iterative method. The MATLAB implementation[1] `jdsvd` offers several extraction choices, such as standard, u-harmonic, v-harmonic, double-harmonic, and refined. In our numerical examples, refined extraction often gave best accuracy. This is consistent with results reported by Hochstenbach [14]. We refer to the Jacobi–Davidson method with refined extraction as `jdsvd(Ref)`. The code `jdsvd` is still under development, and we used the version available to us at the time of the numerical experiments.

Our methods are mathematically, but not numerically, equivalent to the methods proposed by Kokiopoulou, Bekas, and Gallopoulos [17]. We used the MATLAB implementation `irlanb` by Kokiopoulou, Bekas, and Gallopoulos [17] in our comparison. The code `irlanb` calls Larsen's MATLAB code `lanbpro` [18] to compute partial Lanczos bidiagonalizations with partial reorthogonalization. `irlanb` is designed for computing a few of the smallest singular triplets but not for computing a few of the largest ones. The code therefore is not used in Examples 4 and 5 below. Ritz values and harmonic Ritz values can be used as shifts. We refer to `irlanb` with these shift selections as `irlanb(R)` and `irlanb(H)`, respectively. The code `irlanb` is still under development, and we report results for the version available to us at the time of the numerical experiments.

The internal MATLAB function `svds` uses FORTRAN codes of ARPACK [22]. It calls an eigenvalue routine to compute eigenvalue-eigenvector pairs associated with positive eigenvalues of the symmetric matrix

$$Z := \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix} \in \mathbb{R}^{(\ell+n)\times(\ell+n)}. \tag{4.1}$$

[1] Code is available at http://www.case.edu/artsci/math/hochstenbach/software/jdsvd.html

The matrix Z has the eigenvalues

$$\pm\sigma_1^{(A)}, \pm\sigma_2^{(A)}, \ldots, \pm\sigma_n^{(A)},$$

as well as $\ell - n$ zero eigenvalues, where we as usual assume that $\ell \geq n$. The eigenvectors of Z yield both the right and left singular vectors of A. The method requires the computation of eigenpairs associated with eigenvalues near zero when determining the smallest singular triplets of A. This can be difficult when Z has many positive and negative eigenvalues of large magnitude. For this reason, the `svds` function uses a shift-and-invert approach for computing the smallest singular triplets. This requires factorization of the matrix Z, and therefore the function `svds` typically demands much more storage than the other methods in our comparison. Since `svds` does not yield the number of linear systems of equations that have to be solved, we report only the CPU time required.

The MATLAB code `irblsvds` implements an implicitly restarted block-Lanczos method applied to the matrix (4.1) for the computation of a few singular triplets of A. The method and code are described in [2, 3].[2] The matrix Z is used only for evaluation of matrix-vector products; in particular, the matrix Z is not factored. The code `irblsvds` is not, in general, well suited for computing the smallest singular triplets of A when $\ell - n$ is large, because then Z has $\ell - n$ zero eigenvalues, and the code may determine these eigenvalues and associated eigenvectors instead of eigenpairs associated with tiny singular triplets of A. Unless indicated otherwise, we use the default value 3 for the block-size in our experiments with `irblsvds`. The largest number of consecutive block-Lanczos steps is chosen so that `irblsvds` has about the same storage requirement as the other codes. We note that of the methods used in the examples of this section, only the ones of the present paper are based on augmented matrix formulations.

The schemes of subsections 3.1 and 3.2 are implemented by the MATLAB code `irlba`.[2] The execution of `irlba` is determined by certain user-specified parameters; see Table 4.1. We refer to the augmentation method of subsection 3.1, based on Ritz vectors, as `irlba(R)` and to the augmentation method of subsection 3.2, based on harmonic Ritz vectors, as `irlba(H)`. The scheme `irlba(R)` with one- and two-sided full reorthogonalization is referred to as `irlba(R1)` and `irlba(R2)`, respectively. The analogous implementations of `irlba(H)` are denoted by `irlba(H1)` and `irlba(H2)`. One-sided full reorthogonalization reorthogonalizes the columns of the smaller of the matrices P_m and Q_m. Thus, when $\ell \geq n$, the columns of P_m are reorthogonalized.

The codes `irblsvds`, `irlanb`, `jdsvd`, and `svds` allow a user to choose numerous parameters that affect their performance. Unless stated otherwise, we use the default values for the parameters. Except for the function `svds`, we choose the parameters that affect storage so that all codes require about the same maximum computer storage.

It is impossible to use the same starting vector for all the methods, since some routines work with A and A^T and others with the matrix Z. To make our comparison less dependent on the choice of starting vector, we record the best results for each method over five runs using default random starting vector(s) generated by each code. We use the same starting vector when the same method is applied with different parameter values. The reported number of matrix-vector products is the total number of matrix-vector product evaluations with A and A^T for `irlba`, `irlanb`, and `jdsvd` and with Z for `irblsvds`.

[2] Code is available at the authors' home pages.

TABLE 4.1
Parameters for `irlba`.

adjust	Initial number of vectors added to the k restart vectors to speed up convergence. Default value: $adjust = 3$.
aug	A 4-letter string. The value RITZ yields the augmentation described in subsection 3.1; the value HARM gives augmentation according to subsection 3.2. Default value: aug = HARM if $sigma$ = SS, and aug = RITZ if $sigma$ = LS.
disps	When $disps > 0$, available approximations of the k desired singular values and norms of associated residual errors are displayed each iteration; $disps = 0$ inhibits display of these quantities. Default value: $disps = 0$.
k	Number of desired singular triplets. Default value: $k = 6$.
maxit	Maximum number of restarts. Default value: $maxit = 100$.
steps	Maximum number of Lanczos bidiagonalization steps. The parameter specifies the largest value of steps m in (1.3)–(1.4) and determines the storage requirement of the method. Default value: $steps = 20$.
reorth	A 3-letter string. The value ONE yields one-sided full reorthogonalization of the "shorter" vectors; the value TWO gives two-sided full reorthogonalization. When our available estimate of $\kappa(A)$ (see the discussion following (2.13)) is larger than $\epsilon^{-1/2}$, two-sided full reorthogonalization is used. Default value: $reorth$ = ONE.
sigma	A 2-letter string (SS for smallest and LS for largest) which specifies which extreme singular triplets are to be computed. Default value: $sigma$ = LS.
δ	Tolerance used for convergence check; see (2.13). Default value: $\delta = 10^{-6}$.
v_0	Initial vector for Lanczos bidiagonalization. When $\ell \geq n$, $p_1 := v_0$; cf. Algorithm 2.1. Default value: v_0 is a random vector with normally distributed entries.

' . ,' 1 (smallest singular value). We would like to compute the smallest singular triplet of the diagonal matrices

$$A := \mathrm{diag}[1, 2, \ldots, n], \qquad n = 100, 200, 300, 400.$$

Each code was instructed to determine only one singular triplet, the smallest one. This corresponds to the parameter values $sigma$ = SS and $k = 1$ for `irlba`. We allowed each of the codes `irlba`, `irlanb`, `jdsvd`, and `irblsvds` about the same amount of storage as required for carrying out 20 Lanczos bidiagonalization steps. In particular, the largest number of consecutive block-Lanczos steps allowed by `irblsvds` was limited to 7 (with block-size 3). This corresponds to about the same storage requirement as 21 Lanczos bidiagonalization steps. We let $adjust := 4$ for `irbla`; see Table 4.1. This forces both the `irlba` and `irlanb` codes to apply the same number of bidiagonalization steps in the first restart. We chose the tolerance 10^{-6} for all codes, i.e., $\delta := 10^{-6}$ in (2.13). For the `jdsvd` code we report only refined extraction, because this extraction method was the fastest. Figure 4.1 displays the CPU times required for the different methods. The MATLAB `svds` function is seen to be fastest. This may depend on the fact that `svds` is not coded in MATLAB and that it is based on a shift-and-invert approach. For large problems shift-and-invert may require unacceptable amounts of memory and execution time; however, when computation and storage of factors of matrices of the form $Z - \tau I$, where τ is a scalar, is feasible, then this approach is attractive. Among the methods that use only the matrices A and A^T for evaluating matrix-vector products, `irlba` is seen to be competitive.

' . ,' 2 (smallest singular values). We are interested in determining the smallest singular triplet of four matrices generated in MATLAB with the commands

$$A = randn(n), \qquad A(:, 1) = A(:, 10),$$

for $n = 200, 400, 600, 800$. Here $A = randn(n)$ determines an $n \times n$ matrix with

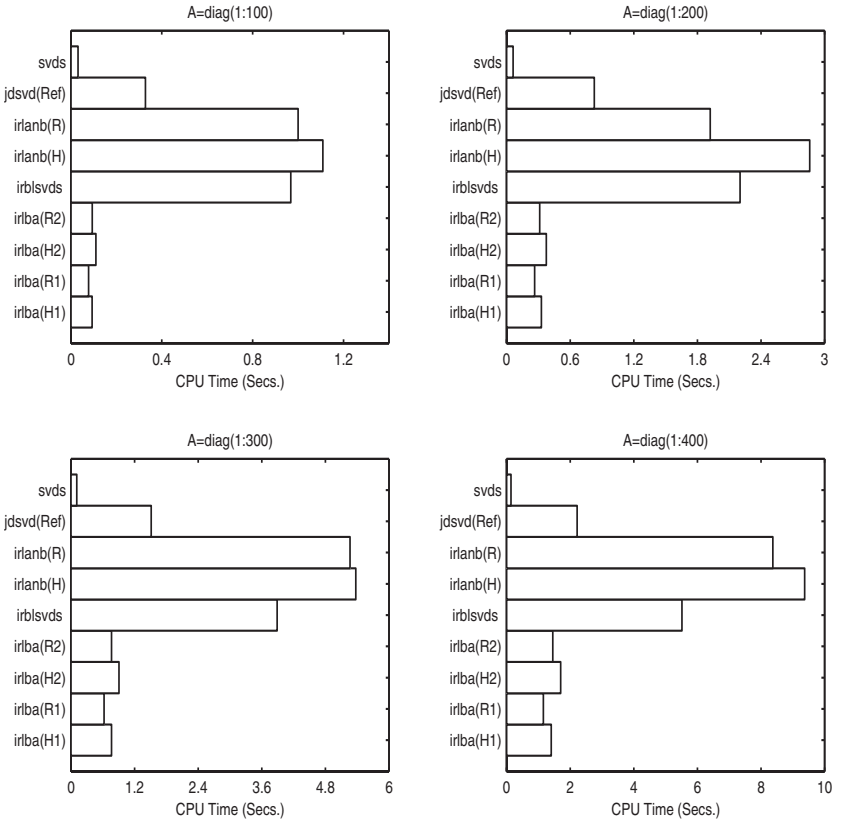


FIG. 4.1. *Example* 1: *CPU times for computing the smallest singular triplet.*

normally distributed entries with mean zero and variance one. The command $A(:,1) = A(:,10)$ overwrites column 1 by column 10 and secures that the matrix obtained has a zero singular value. This example illustrates that augmentation by harmonic Ritz vectors can give substantially faster convergence than augmentation by Ritz vectors.

The code `irlba` was used with $k = 1$, $\delta = 10^{-16}$, $reorth = \text{TWO}$, and $steps = 30$. Figure 4.2 shows the convergence to zero of the smallest computed singular value. The vertical axis displays the absolute error in the smallest computed singular value, i.e., the smallest computed singular value, and the horizontal axis shows the number of restarts. The cross-over label indicates when `irlba` switched from augmenting with harmonic Ritz vectors to augmenting with Ritz vectors. As mentioned in subsection 3.2, augmentation by harmonic Ritz vectors requires the solution of a linear system of equations with the matrix B_m (while augmentation by Ritz vectors does not). We switch from augmentation by harmonic Ritz vectors to augmentation by Ritz vectors when the condition number of the matrix B_m is larger than the square root of the reciprocal of machine epsilon; cf. Algorithm 3.1.

We remark that computing a singular triplet with a numerically vanishing singular value is not always possible since the approximated left singular vector $\tilde{u}_j^{(A)}$ lives in the Krylov subspace (2.8), which is restricted to the range of A. However, in the presence of round-off errors, `irlba` is often able to successfully compute singular triplets associated with zero singular values.

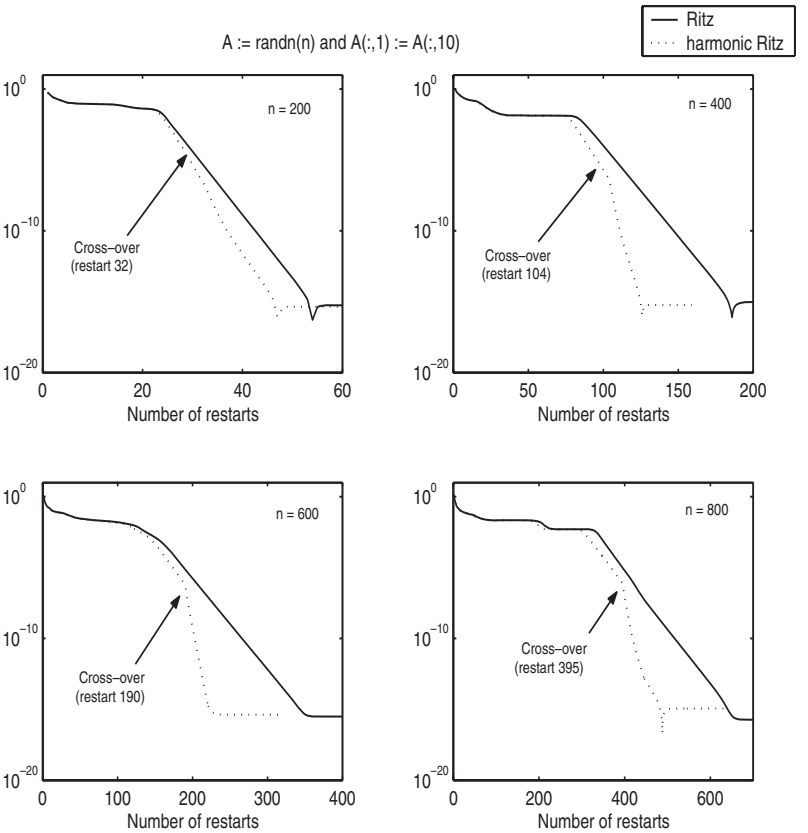


FIG. 4.2. *Example* 2: *Absolute error in the smallest singular value computed by* `irlba` *versus number of restarts using augmentation with Ritz vectors (solid curve) and augmentation with harmonic Ritz vectors (dotted curve). After the cross-over point augmentation is with Ritz vectors.*

3 (smallest singular values). We consider the 1033×320 matrix WELL1033 and the 1850×712 matrix WELL1850 from the set LSQ in the Harwell–Boeing sparse matrix collection [9]. These matrices arise from surveying problems. We would like to determine the six smallest singular triplets of these matrices and select the parameters for the different codes accordingly. For instance, for `irlba` we let *sigma* := SS and $k := 6$. All codes are allowed the amount of storage required for 40 Lanczos bidiagonalization steps. The number of implicit QR-steps in `irlanb` is chosen to be 31; this choice is consistent with the default value of the parameter *adjust* of `irlba`. It forces `irbla` and `irlanb` to apply the same number of Lanczos bidiagonalization steps in the first restart. The tolerance for all methods is set to $\delta = 10^{-6}$. `irlanb` with shifts chosen to be harmonic Ritz values gave better results than when Ritz values were used as shifts. We therefore report only results for the former. The restarted block-Lanczos method `irblsvds` used block-size 4 and was allowed to carry out at most 10 consecutive block-Lanczos steps between restarts. `jdsvd` gave the best results for refined extraction. Therefore we do not report results for other extraction methods. The MATLAB `svds` function was unable to find any of the desired singular triplets to requested accuracy for either of the matrices. Table 4.2 summarizes the computed results. The table shows that `irlba` was able to compute

TABLE 4.2
Example 3: *Computation of the six smallest singular triplets of the matrices WELL*1033 *and WELL*1850.

`irlba(H1)`

Matrix	# Matrix-vector products	CPU time	Magnitude of largest error
WELL1033	638	0.36s	$2.41 \cdot 10^{-15}$
WELL1850	1442	1.39s	$1.72 \cdot 10^{-13}$

`irlba(H2)`

Matrix	# Matrix-vector products	CPU time	Magnitude of largest error
WELL1033	638	0.59s	$2.14 \cdot 10^{-15}$
WELL1850	1442	2.41s	$1.72 \cdot 10^{-13}$

`irblsvds`

Matrix	# Matrix-vector products	CPU time	Magnitude of largest error
WELL1033	7520	18.67s	$8.53 \cdot 10^{-16}$
WELL1850	16080	77.53s	$1.11 \cdot 10^{-15}$

`irlanb(H)`

Matrix	# Matrix-vector products	CPU time	Magnitude of largest error
WELL1033*	–	–	–
WELL1850	1578	8.31s	$3.03 \cdot 10^{-10}$

* Method failed to convergence.

`jdsvd(Ref)`

Matrix	# Matrix-vector products	CPU time	Magnitude of largest error
WELL1033	1978	2.66s	$4.15 \cdot 10^{-10}$
WELL1850	4244	7.69s	$6.84 \cdot 10^{-12}$

all six singular values with the fewest matrix-vector product evaluations and the least CPU time for both matrices.

4 (largest singular values). The matrices MEDLINE, CRANFIELD, and CISI are standard term-by-document test matrices and can be obtained from the Cornell SMART FTP server[3] or the TMG web page.[4] They are of size 5735×1033, 4563×1398, and 5544×1460, respectively. We also consider the term-by-document matrix HYPATIA[5] of size 11390×1265 with 109056 nonzero terms from the web server at the Department of Mathematics, University of Rhode Island. HYPATIA was created in the same manner as standard term-by-document test matrices. All test matrices use the local term frequency (i.e., number of times a word occurs on a website) and have no global weighting or normalization. The parameter values chosen for the different methods are consistent with our desire to determine the 10 largest

[3] ftp://ftp.cs.cornell.edu/pub/smart.
[4] http://scgroup.hpclab.ceid.upatras.gr/scgroup/Projects/TMG/.
[5] The matrix is available at http://math.uri.edu/∼jbaglama.

TABLE 4.3
Example 4: *Computation of the* 10 *largest singular values of the matrices MEDLINE, CRANFIELD, CISI, and HYPATIA.*

`irlba(R1)`

Matrix	# Mat.-vec. products	CPU time			Magn. largest error
		mat.-vec. products	one-sided reorthog.	total	
MEDLINE	74	0.23s	0.05s	0.33s	$1.11 \cdot 10^{-10}$
CRANFIELD	78	0.35s	0.03s	0.48s	$1.29 \cdot 10^{-12}$
CISI	76	0.26s	0.08s	0.48s	$2.77 \cdot 10^{-12}$
HYPATIA	78	0.41s	0.17s	0.76s	$1.54 \cdot 10^{-11}$

`irlba(R2)`

Matrix	# Mat.-vec. products	CPU time			Magn. largest error
		mat.-vec. products	two-sided reorthog.	total	
MEDLINE	74	0.22s	0.11s	0.41s	$1.11 \cdot 10^{-10}$
CRANFIELD	78	0.35s	0.12s	0.53s	$1.39 \cdot 10^{-12}$
CISI	76	0.26s	0.14s	0.53s	$2.79 \cdot 10^{-12}$
HYPATIA	78	0.40s	0.28s	0.89s	$1.55 \cdot 10^{-11}$

`irblsvds`

Matrix	# Mat.-vec. products	CPU time	Magn. largest error
MEDLINE	436	4.55s	$5.09 \cdot 10^{-10}$
CRANFIELD	508	5.72s	$3.28 \cdot 10^{-10}$
CISI	564	6.39s	$2.41 \cdot 10^{-10}$
HYPATIA	432	9.36s	$2.15 \cdot 10^{-09}$

`jdsvd(Ref)`

Matrix	# Mat.-vec. products	CPU time	Magn. largest error
MEDLINE	218	2.25s	$7.32 \cdot 10^{-10}$
CRANFIELD	240	2.58s	$7.16 \cdot 10^{-11}$
CISI	218	2.39s	$2.93 \cdot 10^{-10}$
HYPATIA	240	4.70s	$4.43 \cdot 10^{-10}$

`svds`

Matrix	CPU time	Magn. largest error
MEDLINE	1.06s	$4.94 \cdot 10^{-12}$
CRANFIELD	1.48s	$2.64 \cdot 10^{-11}$
CISI	1.58s	$2.73 \cdot 10^{-12}$
HYPATIA	2.53s	$1.02 \cdot 10^{-11}$

singular triplets for each of these matrices; for `irlba` we let *sigma* :=LS and $k := 10$. The available storage is assumed to be large enough to simultaneously store all vectors generated during 20 consecutive steps of Lanczos bidiagonalization. The tolerance δ in the stopping criterion is 10^{-6}. Since we seek to determine the largest singular triplets, `irbla` uses the augmentation method of section 3.1. We report the results for one-sided and two-sided full reorthogonalization. The code `irblsvds` was used with block-size 4 and was allowed to carry out at most five consecutive block-Lanczos steps between restarts. The fastest extraction method for `jdsvd` was refined. Table 4.3 displays the performance of the methods and illustrates the competitiveness of

`irlba`. The code `irlanb` is not part of our comparison, since it is designed for the computation of only a few of the smallest singular triplets.

5 (condition number). We would like to determine the condition number $\kappa(A) := \sigma_1^{(A)}/\sigma_n^{(A)}$, where the singular values are enumerated according to (1.1), of the Läuchli matrix $A := L(n, \mu) \in \mathbb{R}^{(n+1)\times n}$ for $n = 20000$ and the default value of μ.[6] Thus, A has ones across the top row and μ on the subdiagonal; the remaining matrix entries are zero. This matrix often is used to illustrate the drawback of forming A^TA in least-squares computations; see [19]. The Läuchli matrix A is nonsingular; it has the simple singular value $\sqrt{n+\mu^2}$ and the singular value μ of multiplicity $n-1$ giving the condition number $\kappa(A) = 9.490724975767860 \cdot 10^9$, and therefore A^TA is numerically singular. `irlba` only implicitly works with the matrix A^TA and is able to compute $\kappa(A)$. Specifically, we let *sigma* :=LS and SS, $k := 1$, and we allow the storage required for all vectors generated by 20 consecutive steps of Lanczos bidiagonalization. The tolerance δ in the stopping criterion is set to machine epsilon ϵ. Two-sided full reorthogonalization was employed. We augmented by Ritz vectors when computing the largest singular value and, until the matrices B_m became ill-conditioned, by harmonic Ritz vectors when determining the smallest singular value. `irlba` required 0.703 and 0.796 seconds of CPU time to compute the largest and smallest singular values of A, respectively, and gave the condition number $9.490724975767925 \cdot 10^9$ with a relative error of $6.83 \cdot 10^{-15}$. Thus, the fact that A^TA is numerically singular does not cause `irlba` problems.

Acknowledgments. We would like thank Stratis Gallopoulos for a copy of the paper [17], Michiel Hochstenbach and Efi Kokiopoulou for providing copies of their codes, and Costas Bekas, Michiel Hochstenbach, and Efi Kokiopoulou for valuable comments and discussions.

REFERENCES

[1] J. Baglama, D. Calvetti, and L. Reichel, *Iterative methods for the computation of a few eigenvalues of a large symmetric matrix*, BIT, 36 (1996), pp. 400–421.

[2] J. Baglama, D. Calvetti, and L. Reichel, *IRBL: An implicitly restarted block-Lanczos method for large-scale Hermitian eigenproblems*, SIAM J. Sci. Comput., 24 (2003), pp. 1650–1677.

[3] J. Baglama, D. Calvetti, and L. Reichel, *Algorithm* 827: *irbleigs: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix*, ACM Trans. Math. Software, 29 (2003), pp. 337–348.

[4] Å. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, 1996.

[5] Å. Björck, E. Grimme, and P. Van Dooren, *An implicit shift bidiagonalization algorithm for ill-posed systems*, BIT, 34 (1994), pp. 510–534.

[6] J. R. Bunch and C. P. Nielsen, *Updating the singular decomposition*, Numer. Math., 31 (1978), pp. 111–129.

[7] D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21.

[8] P. Comon and G. H. Golub, *Tracking a few extreme singular values and vectors in signal processing*, Proc. IEEE, 78 (1990), pp. 1327–1343.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis, *User's Guide for the Harwell-Boeing Sparse Matrix Collection (Release* I*)*, Technical report TR/PA/92/86, CERFACS, Toulouse, France, 1992. The matrices are available at http://math.nist.gov/MatrixMarket/.

[10] G. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, J. Soc. Indust. Appl. Math. Ser. B Numer Anal., 2 (1965), pp. 205–224.

[6] The Läuchli matrix can be obtained from the Matrix Market website, http://math.nist.gov/MatrixMarket/deli/Lauchli/. We used $\mu = 1.4901006677403 \cdot 10^{-8}$.

[11] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[12] M. Gu and S. C. Eisenstat, *A Stable and Fast Algorithm for Updating the Singular Value Decomposition*, Technical report YALEU/DCS/RR-966, Department of Computer Science, Yale University, New Haven, CT, 1993.
[13] M. E. Hochstenbach, *A Jacobi–Davidson type SVD method*, SIAM J. Sci. Comput., 23 (2001), pp. 606–628.
[14] M. E. Hochstenbach, *Harmonic and refined extraction methods for the singular value problem, with applications to least-squares problems*, BIT, to appear.
[15] M. E. Hochstenbach, *Subspace Methods for Eigenvalue Problems*, Ph.D. thesis, Department of Mathematics, University of Utrecht, Utrecht, The Netherlands, 2003; also available online from http://www.case.edu/artsci/math/hochstenbach/.
[16] Z. Jia and D. Niu, *An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition*, SIAM J. Matrix Anal. Appl., 25 (2003), pp. 246–265.
[17] E. Kokiopoulou, C. Bekas, and E. Gallopoulos, *Computing smallest singular triplets with implicitly restarted Lanczos bidiagonalization*, Appl. Numer. Math., 49 (2004), pp. 39–61.
[18] R. M. Larsen, *Lanczos bidiagonalization with partial reorthogonalization*, Chapter of Ph.D. thesis, Department of Computer Science, University of Aarhus, Aarhus, Denmark, 1998; also available online from http://soi.stanford.edu/∼rmunk/.
[19] P. Läuchli, *Jordan-Elimination und Ausgleichung nach kleinsten Quadraten*, Numer. Math., 3 (1961), pp. 226–240.
[20] R. B. Lehoucq, *Implicitly restarted Arnoldi methods and subspace iteration*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 551–562.
[21] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 789–821.
[22] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools 6, SIAM, Philadelphia, 1998.
[23] The MathWorks, Inc., *MATLAB, Version* 6.5 *R*13, Natick, MA, 2002.
[24] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Linear Algebra Appl., 154/156 (1991), pp. 289–309.
[25] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[26] R. B. Morgan and M. Zeng, *Harmonic Restarted Arnoldi Algorithm for Calculating Eigenvalues and Determining Multiplicity*, preprint, 2003.
[27] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, *Approximate solutions and eigenvalue bounds from Krylov subspaces*, Numer. Linear Algebra Appl., 2 (1995), pp. 115–134.
[28] H. D. Simon and H. Zha, *Low-rank matrix approximation using the Lanczos bidiagonalization process with applications*, SIAM J. Sci. Comput., 21 (2000), pp. 2257–2274.
[29] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.
[30] D. C. Sorensen, *Numerical methods for large eigenvalue problems*, Acta Numer., 11 (2002), pp. 519–584.
[31] A. Stathopoulos and Y. Saad, *Restarting techniques for the (Jacobi-)Davidson symmetric eigenvalue methods*, Electron. Trans. Numer. Anal., 7 (1998), pp. 163–181.
[32] A. Stathopoulos, Y. Saad, and K. Wu, *Dynamic thick restarting of the Davidson, and the implicitly restarted Arnoldi methods*, SIAM J. Sci. Comput., 19 (1998), pp. 227–245.
[33] K. Wu and H. Simon, *Thick-restart Lanczos method for large symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 43–66
© 2005 Society for Industrial and Applied Mathematics

A PARALLEL EIGENSOLVER FOR DENSE SYMMETRIC MATRICES BASED ON MULTIPLE RELATIVELY ROBUST REPRESENTATIONS*

PAOLO BIENTINESI†, INDERJIT S. DHILLON†, AND ROBERT A. VAN DE GEIJN†

Abstract. We present a new parallel algorithm for the dense symmetric eigenvalue/eigenvector problem that is based upon the tridiagonal eigensolver, Algorithm MR^3, recently developed by Dhillon and Parlett. Algorithm MR^3 has a complexity of $O(n^2)$ operations for computing all eigenvalues and eigenvectors of a symmetric tridiagonal problem. Moreover the algorithm requires only $O(n)$ extra workspace and can be adapted to compute any subset of k eigenpairs in $O(nk)$ time. In contrast, all earlier stable parallel algorithms for the tridiagonal eigenproblem require $O(n^3)$ operations in the worst case, while some implementations, such as divide and conquer, have an extra $O(n^2)$ memory requirement. The proposed parallel algorithm balances the workload equally among the processors by traversing a matrix-dependent *representation tree* which captures the sequence of computations performed by Algorithm MR^3. The resulting implementation allows problems of very large size to be solved efficiently—the largest dense eigenproblem solved in-core on a 256 processor machine with 2 GBytes of memory per processor is for a matrix of size 128,000 × 128,000, which required about 8 hours of CPU time. We present comparisons with other eigensolvers and results on matrices that arise in the applications of computational quantum chemistry and finite element modeling of automobile bodies.

Key words. parallel computing, symmetric matrix, eigenvalues, eigenvectors, relatively robust representations

AMS subject classifications. 65F15, 65Y05, 68W10

DOI. 10.1137/030601107

1. Introduction.

The symmetric eigenvalue problem is ubiquitous in computational sciences; problems of ever-growing size arise in applications as varied as computational quantum chemistry, finite element modeling, and pattern recognition. In many of these applications, both time and space are limiting factors for solving the problem, and hence efficient parallel algorithms and implementations are needed. The best approach for computing all the eigenpairs (eigenvalues and eigenvectors) of a dense symmetric matrix involves three phases: (1) [illegible]—reduce the given symmetric matrix A to tridiagonal form T; (2) [illegible]—compute all the eigenpairs of T; (3) [illegible]—map T's eigenvectors into those of A. For an $n \times n$ matrix, the reduction and backtransformation phases require $O(n^3)$ arithmetic operations each. Until recently, all algorithms for the tridiagonal eigenproblem too had cubic complexity in the worst case; these include the remarkable QR algorithm [26, 27, 36], inverse iteration [42], and the divide and conquer method [10].

In fact, the tridiagonal problem can be the computational bottleneck for large problems taking nearly 70–80% of the total time to solve the entire dense problem. For example, on a 2.4 GHz Intel Pentium 4 processor the tridiagonal reduction and backtransformation of a 2000 × 2000 dense matrix takes about 32 seconds, while

*Received by the editors October 6, 2003; accepted for publication (in revised form) January 11, 2005; published electronically August 17, 2005. This research was supported in part by NSF CAREER Award ACI-0093404 (ISD) and NSF Award ACI-0203685.

http://www.siam.org/journals/sisc/27-1/60110.html

†Department of Computer Sciences, The University of Texas at Austin, Austin, TX 78712 (pauldj@cs.utexas.edu, inderjit@cs.utexas.edu, rvdg@cs.utexas.edu).

LAPACK's bisection and inverse iteration software takes 106 seconds to compute all the eigenpairs of the tridiagonal. The timings for a 4000×4000 matrix clearly show the $O(n^3)$ behavior: 290 seconds for tridiagonal reduction and backtransformation, and 821 seconds for bisection and inverse iteration to solve the tridiagonal eigenproblem. Timings for the tridiagonal QR algorithm are 86 seconds for $n = 2000$ and 1099 seconds for $n = 4000$. More detailed timing results are given in section 4.

Recently, Dhillon and Parlett proposed Algorithm MRRR or MR^3 (multiple relatively robust representations) [13, 18, 17], which gives the first stable $O(nk)$ algorithm to compute k eigenvalues and eigenvectors of a symmetric tridiagonal matrix. In this paper we present a parallel algorithm based on Algorithm MR^3 for computing any subset of eigenpairs of a dense symmetric matrix; this yields the first parallel implementation of Algorithm MR^3. We refer to the parallel algorithm as PMR^3 (parallel MR^3). As a consequence, the time spent by the proposed algorithm on the tridiagonal eigenproblem is negligible compared to the time spent on reduction and backtransformation. For example, to compute all the eigenpairs of a 15,000 × 15,000 matrix on 16 processors the new algorithm requires 546 seconds for reduction, 22.2 seconds for the tridiagonal solution, and 160 seconds for backtransformation. In comparison, the corresponding timings for existing implementations for the tridiagonal eigensolution are 2054 seconds for the QR algorithm and 92.4 seconds for the divide and conquer method. For a 32,000 × 32,000 matrix the timings for PMR^3 on 16 processors are 4876 seconds for the reduction, 118 seconds for the tridiagonal solution, and 1388 seconds for backtransformation. These timings clearly contrast the $O(n^2)$ complexity of Algorithm MR^3 as opposed to the $O(n^3)$ reduction and backtransformation phases.

Moreover, some of the earlier algorithms have extra memory requirements: the ScaLAPACK divide and conquer code (PDSTEVD) requires extra $O(n^2)$ storage, while the inverse iteration code (PDSTEIN) can lead to a memory imbalance on the processors depending upon the eigenvalue distribution. Thus neither PDSTEVD nor PDSTEIN can be used to solve the above-mentioned 32,000 × 32,000 eigenproblem on 16 processors. In contrast, our parallel algorithm requires only workspace that is linear in n and the memory needed to store the eigenvectors of the tridiagonal problem is evenly divided among processors, thus allowing us to efficiently solve problems of very large size. The largest dense problem we have solved "in-core" on a 256 processor machine with 2 GBytes of memory per processor is a matrix of size 128,000 × 128,000, which required about 8 hours of computation time. Section 4 contains further timing results for the parallel implementations.

The rest of the paper is organized as follows. Section 2 reviews previous work on algorithms for the dense symmetric eigenvalue problem. In section 3, we present the proposed parallel Algorithm PMR^3, which uses multiple relatively robust representations for the tridiagonal problem. Section 4 presents detailed timing results comparing Algorithm PMR^3 with existing software. These include results on matrices that arise in the real-life applications of computational quantum chemistry and finite element modeling of automobile bodies.

A word on the notation used throughout the paper. T indicates a symmetric tridiagonal matrix, n represents the size of a matrix, eigenvalues are denoted by λ, and eigenvectors are denoted by $\boldsymbol{v}$. Computed quantities will often be denoted by "hatted" symbols, for example, $\hat{\lambda}$ and $\hat{\boldsymbol{v}}$. The number of processors in a parallel computation is p, while the ith processor is denoted by p_i.

2. Related work. As mentioned earlier, most algorithms for the dense symmetric eigenvalue problem proceed in three phases. The first and third phases, House-

holder reduction and backtransformation, are fairly standard and described in section 3.2. The second stage, tridiagonal eigensolution, has led to a variety of interesting algorithms; we now give a quick overview of existing methods, emphasizing their parallel versions.

The QR algorithm, independently invented by Francis [26, 27] and Kublanovskaya [36], is an iteration that produces a sequence of similar matrices that converges to diagonal form. When the starting matrix is symmetric and tridiagonal, each iterate produced by the QR algorithm is also symmetric and tridiagonal. Convergence to diagonal form is rapid (ultimately cubic) with a suitable choice of shifts [38]. A fast square-root-free version of QR developed by Pal, Walker, and Kahan (PWK) is useful if only eigenvalues are desired [38]. Another attractive alternative, in the latter case, is to use the differential quotient-difference algorithm (dqds) that is based on the related LR iteration [24]. In practice 2–3 iterations, on average, are needed per eigenvalue in the QR algorithm where each iteration is composed of at most $n-1$ Givens rotations. Thus all eigenvalues can be computed at a cost of $O(n^2)$ operations, while the accumulation of Givens rotations required for computing orthogonal eigenvectors results in $O(n^3)$ operations (in practice, $6n^3$ to $9n^3$ operations are observed).

The inherent sequential nature of the QR algorithm makes the eigenvalue computation hard to parallelize. However, when eigenvectors are needed, an effective parallel algorithm that yields good speedups can be obtained as follows. First, the Householder reflections computed during the reduction are accumulated in approximately $\frac{4}{3}n^3$ operations to form a matrix Z, which is then evenly partitioned among the p processors so that each processor owns approximately n/p rows of Z. The tridiagonal matrix is duplicated on all processors and the $O(n^2)$ eigenvalue computation is redundantly performed on all processors, while the Givens rotations are directly applied on each processor to its part of Z. This accumulation achieves perfect speedup since all processors can simultaneously update their portion of Z without requiring any communication, thus leading to an overall parallel complexity of $O(n^3/p)$ operations. A faster algorithm (up to a factor of 2) can be obtained by using perfect shifts and inner deflations [16].

A major drawback of the QR algorithm is that it is hard to adapt to the case when only a subset of eigenvalues and eigenvectors is desired at a proportionately reduced operation count. Thus a commonly used parallel solution is to invoke the bisection algorithm followed by inverse iteration [32]. The bisection algorithm was first proposed by Givens in 1954 and allows the computation of k eigenvalues of a symmetric tridiagonal T in $O(kn)$ operations [28]. Once accurate eigenvalues are known, the method of inverse iteration may be used to compute the corresponding eigenvectors [42]. However, inverse iteration can guarantee only small residual norms. It cannot ensure orthogonality of the computed vectors when eigenvalues are close. A commonly used "remedy" is to orthogonalize each approximate eigenvector, using the modified Gram–Schmidt method, against previously computed eigenvectors of "nearby" eigenvalues—the LAPACK and EISPACK implementations orthogonalize when eigenvalues are closer than $10^{-3}\|T\|$. Unfortunately even this conservative strategy can fail to give accurate answers in certain situations [14]. The amount of work required by inverse iteration to compute all the eigenvectors of a symmetric tridiagonal matrix depends strongly upon the distribution of eigenvalues (unlike the QR algorithm, which always requires $O(n^3)$ operations). If eigenvalues are well separated (gaps greater than $10^{-3}\|T\|$), then $O(n^2)$ operations are sufficient. However, when eigenvalues are clustered, current implementations of inverse iteration can take up to $10n^3$ operations due to orthogonalization [34]. Unfortunately the latter sit-

uation is the norm rather than the exception for large matrices since even uniform eigenvalue spacings when n exceeds 1000 can lead to eigenvalue gaps smaller than $10^{-3}\|T\|$.

When eigenvalues are well separated, both bisection and inverse iteration can be effectively parallelized leading to a complexity of $O(n^2/p)$ operations. However, as remarked above, the common situation for large matrices is that inverse iteration requires $O(n^3)$ operations; see section 4.3 for some timings. Thus, parallel inverse iteration requires $O(n^3/p)$ operations in these situations. Moreover, considerable communication is required when Gram–Schmidt orthogonalization is done across processor boundaries. Indeed, to avoid communication, the current inverse iteration implementation in ScaLAPACK (PDSTEIN) computes all the eigenvectors corresponding to a cluster of eigenvalues on a single processor, thus leading to a parallel complexity of $O(n^3)$ in the worst case and also an imbalance in the memory required on each processor [8, p. 48].

The bisection algorithm to find eigenvalues has linear convergence and can be quite slow. To speed up bisection, there have been many attempts to employ faster zero-finders such as the Rayleigh quotient iteration [38], Laguerre's method [37, 39], and the zeroin scheme [11, 9]. These zero-finders can speed up the computation of isolated eigenvalues by a considerable amount, but they seem to stumble when eigenvalues cluster. In all these cases, the corresponding eigenvectors still need to be computed by inverse iteration.

The divide and conquer method proposed by Cuppen in 1981 is a method specially suited for parallel computation [10, 20]; remarkably this algorithm also yields a faster sequential implementation than QR. The basic strategy of the divide and conquer algorithm is to express the tridiagonal matrix as a low-rank modification of a direct sum of two smaller tridiagonal matrices. This modification may be a rank-one update or may be obtained by crossing out a row and column of the tridiagonal. The entire eigenproblem can then be solved in terms of the eigenproblems of the smaller tridiagonal matrices, and this process can be repeated recursively. For several years after its inception, it was not known how to guarantee numerically orthogonality of the eigenvector approximations obtained by this approach. However, Gu and Eisenstat found a solution to this problem, leading to robust software based on their strategy [30].

The main reason for the unexpected success of divide and conquer methods on serial machines is [illegible], which occurs when an eigenpair of a submatrix of T is an acceptable eigenpair of a larger submatrix. The greater the amount of deflation, which depends on the eigenvalue distribution and on the structure of the eigenvectors, the lesser the work required in these methods. For matrices with clustered eigenvalues, deflation can be extensive; however, in general, $O(n^3)$ operations are needed. The divide and conquer method is suited for parallelization since smaller subproblems can be solved independently on various processors. However, communication costs for combining subproblems are substantial, especially when combining the larger subproblems to get the solution to the full problem [43]. A major drawback of the divide and conquer algorithm is its extra $O(n^2)$ memory requirement—as we shall see later, this limits the largest problem that can be solved using this approach.

2.1. Other solution methods. The oldest method for solving the symmetric eigenproblem dates back to Jacobi [33]. Jacobi's method does not reduce the dense symmetric matrix to tridiagonal form, as most other methods do, but works on the dense matrix at all stages. It performs a sequence of plane rotations, each of which annihilates an off-diagonal element (which is filled in during later steps). There are

a variety of Jacobi methods that differ solely in their strategies for choosing the next element to be annihilated. All good strategies tend to diminish the off-diagonal elements, and the resulting sequence of matrices converges to the diagonal matrix of eigenvalues.

Jacobi's method fell out of favor with the discovery of the QR algorithm. The primary reason is that, in practice, the cost of even the most efficient variants of the Jacobi iteration is an order of magnitude greater than that of the QR algorithm. Nonetheless, it has periodically enjoyed a resurrection since it can be efficiently parallelized, and theoretical results show it to be more accurate than the QR algorithm [12].

The symmetric invariant subspace decomposition algorithm (SYISDA) formulates the problem in a dramatically different way [5]. The idea is to scale and shift the spectrum of the given matrix so that its eigenvalues are mapped to the interval $[0, 1]$, with the mean eigenvalue being mapped to $\frac{1}{2}$. Letting B equal the transformed matrix, a polynomial p is applied to B with the property that $\lim_{i\to\infty} p^i([0, 1]) = \{0, 1\}$. By applying this polynomial to B in the iteration $C_0 = B$, $C_{i+1} = p(C_i)$ until convergence, all eigenvalues of C_{i+1} eventually become arbitrarily close to 0 or 1. The eigenvectors of C_{i+1} and A are related in such a way that allows the computation of two subspaces that can then be used to decouple matrix A into two subproblems, each of size roughly half that of A. The process then continues with each of the subproblems. The benefit of the SYISDA approach is that the computation can be cast in terms of matrix-matrix multiplication, which can attain near-peak performance on modern microprocessors and parallelizes easily [19, 31]. Unfortunately, this benefit is accompanied by an increase in the operation count, to the point where SYISDA is not considered to be competitive.

2.2. Parallel libraries. A great deal of effort has been spent in building efficient parallel symmetric eigensolvers for distributed systems. Specially designed software for this problem has been developed as part of a number of numerical libraries. Among these the best known are the Scalable Linear Algebra Package (ScaLAPACK) [21, 8], Parallel Eigensolver (PeIGS) [23], the Parallel Research on Invariant Subspace Methods (PRISM) project [5], and the Parallel Linear Algebra Package (PLAPACK) [44]. All of these packages attempt to achieve portability by embracing the Message-Passing Interface (MPI) and the Basic Linear Algebra Subprograms (BLAS) [19].

The ScaLAPACK project is an effort to parallelize the Linear Algebra Package (LAPACK) [1] to distributed memory architectures. It supports a number of different algorithms, as further discussed in the experimental section. PeIGS supports a large number of chemistry applications that give rise to large dense eigenvalue problems. It includes a parallel tridiagonal eigensolver that is based on a very early version of Algorithm MR^3; this preliminary version does limited Gram–Schmidt orthogonalization; see [15]. The PRISM project implements the SYISDA approach outlined in section 2.1. PLAPACK currently supports a parallel implementation of the QR algorithm as well as the algorithm that is the topic of this paper.

3. The proposed algorithm. We now present the proposed parallel algorithm. Section 3.1 describes how the tridiagonal eigenproblem can be solved using the method of multiple relatively robust representations (MR^3), while section 3.2 briefly describes the phases of Householder reduction and backtransformation.

3.1. Tridiagonal eigensolver using multiple relatively robust representations. Algorithm MR^3 was recently introduced by Dhillon and Parlett [13, 18, 17] for the task of computing k eigenvectors of a symmetric tridiagonal matrix T, and

has a complexity of $O(nk)$ operations. The superior time complexity of the algorithm is achieved by avoiding Gram–Schmidt orthogonalization, which in turn is the result of high relative accuracy in intermediate computations.

3.1.1. The sequential algorithm. We provide the main ideas behind Algorithm MR^3; an in-depth technical description and justification of the algorithm can be found in [13, 18, 17]. There are three key ingredients that form the backbone of Algorithm MR^3:

1. A relatively robust representation (RRR) is a representation that determines its eigenvalues and eigenvectors to high relative accuracy; i.e., small componentwise changes to individual entries of the representation lead to small relative changes in the eigenvalues and small changes in the eigenvectors (modulo relative gaps between eigenvalues; see (2) below). Unfortunately, the traditional representation of a tridiagonal by its diagonal and off-diagonal elements does not form an RRR; see [18, sect. 3] for an example. However, the bidiagonal factorization $T = LDL^t$ of a positive definite tridiagonal is an RRR, and in many cases an indefinite LDL^t also forms an RRR [18]. We now make precise the conditions needed for LDL^t to be an RRR: Write l_i for $L(i+1,i)$ and d_i for $D(i,i)$. Define the relative gap of $\hat{\lambda}$, where $\hat{\lambda}$ is closer to λ than to any other eigenvalue of LDL^t, to be

$$\text{relgap}(\hat{\lambda}) := \text{gap}(\hat{\lambda})/|\hat{\lambda}|,$$

where $\text{gap}(\hat{\lambda}) = \min\{|\nu - \hat{\lambda}| : \ \nu \neq \lambda, \ \nu \in \text{spectrum}(LDL^t)\}$. We say that $(\lambda, \boldsymbol{v})$ is determined to high relative accuracy by L and D if small relative changes, $l_i \to l_i(1+\eta_i)$, $d_i \to d_i(1+\delta_i)$, $|\eta_i| < \xi$, $|\delta_i| < \xi$, $\xi \ll 1$, cause changes $\delta\lambda$ and $\delta\boldsymbol{v}$ that satisfy

$$\frac{|\delta\lambda|}{|\lambda|} \leq K_1 n\xi, \quad \lambda \neq 0, \tag{1}$$

$$|\sin\angle(\boldsymbol{v}, \boldsymbol{v}+\delta\boldsymbol{v})| \leq \frac{K_2 n\xi}{\text{relgap}(\lambda)} \tag{2}$$

for modest constants K_1 and K_2, say, smaller than 100. We call such an LDL^t factorization an RRR for $(\lambda, \boldsymbol{v})$. The advantage of an RRR is that the eigenvalues and eigenvectors can be computed to high relative accuracy as governed by (1) and (2). For more details see [18].

2. Once an accurate eigenvalue $\hat{\lambda}$ is known, its eigenvector may be computed by solving the equation $(LDL^t - \hat{\lambda}I)\boldsymbol{z} \approx 0$. However, it is not straightforward to solve this equation: the trick is to figure out what equation to ignore in this nearly singular system. Unable to find a solution to this problem, current implementations of inverse iteration in LAPACK and EISPACK solve $(LDL^t - \hat{\lambda}I)\boldsymbol{z}_{i+1} = \boldsymbol{z}_i$ and take $\boldsymbol{z}_0$ to be a random starting vector (this difficulty was known to Wilkinson [45, p. 318]). This problem was solved recently by using twisted factorizations that are obtained by gluing a top-down (LDL^t) and a bottom-up (UDU^t) factorization. The solution is presented in Algorithm Getvec below; see [18, 40, 25] for more details.

3. By using an RRR and Algorithm Getvec for computing eigenvectors, it can be shown that the computed eigenvectors are numerically orthogonal when the eigenvalues have large relative gaps [18]. However, when eigenvalues have small relative gaps, the above approach is not adequate. For the case of small relative gaps, Algorithm MR^3 uses

ALGORITHM Getvec(L,D, $\hat{\lambda}$).

Input: L is unit lower bidiagonal (l_i denotes $L(i+1,i)$, $1 \le i \le n-1$), and D is diagonal (d_i denotes $D(i,i)$, $1 \le i \le n$); LDL^t is the input tridiagonal matrix assumed to be irreducible. $\hat{\lambda}$ is an approximate eigenvalue.

Output: $\boldsymbol{z}$ is the computed eigenvector.

I. Factor $LDL^t - \hat{\lambda}I = L_+D_+L_+^t$ by the dstqds (differential stationary quotient-difference with shift) transform.

II. Factor $LDL^t - \hat{\lambda}I = U_-D_-U_-^t$ by the dqds (differential progressive quotient-difference with shift) transform.

III. Compute γ_k for $k = 1, \ldots, n$ by the formula $\gamma_k = s_k + \frac{d_k}{D_-(k+1)}p_{k+1}$ that involves the intermediate quantities s_k and p_{k+1} computed in the dstqds and dqds transforms (for details see [18, sect. 4.1]). Pick an r such that $|\gamma_r| = \min_k |\gamma_k|$. Form the twisted factors with twist index r, N_r and Δ_r, which satisfy $N_r\Delta_rN_r^t = LDL^t - \hat{\lambda}I$.

IV. Form the approximate eigenvector $\boldsymbol{z}$ by solving $N_r^t\boldsymbol{z} = \boldsymbol{e}_r$ ($\boldsymbol{e}_r$ is the rth column of the identity matrix I), which is equivalent to solving $(LDL^t - \hat{\lambda}I)\boldsymbol{z} = N_r\Delta_rN_r^t\boldsymbol{z} = \boldsymbol{e}_r\gamma_r$ since $N_r\boldsymbol{e}_r = \boldsymbol{e}_r$ and $\Delta_r\boldsymbol{e}_r = \gamma_r\boldsymbol{e}_r$:

$$z(r) = 1.$$

$$\text{For } i = r-1, \ldots, 1, \quad z(i) = \begin{cases} -L_+(i)z(i+1), & z(i+1) \neq 0, \\ -(d_{i+1}l_{i+1}/d_il_i)z(i+2), & \text{otherwise.} \end{cases}$$

$$\text{For } j = r, \ldots, n-1, \quad z(j+1) = \begin{cases} -U_-(j)z(j), & z(j) \neq 0, \\ -(d_{j-1}l_{j-1}/d_jl_j)z(j-1), & \text{otherwise.} \end{cases}$$

Note that d_il_i is the $(i, i+1)$ element of LDL^t.

V. Set $\boldsymbol{z} \leftarrow \boldsymbol{z}/\|\boldsymbol{z}\|$.

FIG. 1. *Algorithm* Getvec *for computing the eigenvector of an isolated eigenvalue.*

multiple RRRs, i.e., multiple factorizations $L_cD_cL_c^t = LDL^t - \tau_cI$, where τ_c is close to a cluster. The shifts τ_c are chosen to "break" clusters, i.e., to make relative gaps bigger (note that relative gaps change upon shifting by τ_c). After forming the new representation $L_cD_cL_c^t$, the eigenvalues in the cluster are "refined" so that they have high relative accuracy with respect to $L_cD_cL_c^t$. Finally the eigenvectors of eigenvalues that become relatively well separated after shifting are computed by Algorithm Getvec using $L_cD_cL_c^t$; the process is iterated for eigenvalues that still have small relative gaps. Details are given in Algorithm MR^3 below. The tricky theoretical aspects that address the relative robustness of intermediate representations and whether the eigenvectors computed using different RRRs are numerically orthogonal may be found in [41] and [17]. It is important to note that orthogonality of the computed eigenvectors is achieved without Gram–Schmidt being used in any of the procedures.

We first present Algorithm Getvec in Figure 1. Getvec takes an LDL^t factorization and an approximate eigenvalue $\hat{\lambda}$ as input and computes the corresponding eigenvector by forming the appropriate twisted factorization $N_r\Delta_rN_r^t = LDL^t - \hat{\lambda}I$. The twist index r in step III of Figure 1 is chosen so that $|\gamma_r| = \min_k |\gamma_k|$ and is followed by solving $(LDL^t - \hat{\lambda}I)\boldsymbol{z} = \gamma_r\boldsymbol{e}_r$; thus r is the index of the equation that is ignored and provides a solution to Wilkinson's problem mentioned above [40]. The result-

ing eigenvector is accurate since differential transformations are used to compute the twisted factorization, and the eigenvector is computed solely by multiplications (no additions or subtractions) in step IV of the algorithm. We assume that LDL^t is an irreducible tridiagonal, i.e., all off-diagonals are nonzero. Details on twisted factorizations, differential quotient-difference transforms, and Algorithm Getvec may be found in [18].

Algorithm Getvec computes a single eigenvector of an RRR; it was shown in [18] that the computed eigenvector is highly accurate and so is numerically orthogonal to all other eigenvectors if the corresponding eigenvalue has a large relative gap. However, if Getvec is invoked when the corresponding eigenvalue is part of a cluster, the computed vector will, in general, not be orthogonal to other eigenvectors in the cluster. The difficulty is that, as seen by (2), the eigenvectors of eigenvalues with small relative gaps are highly sensitive to tiny changes in L and D.

To overcome this problem, Algorithm MR^3, given in Figure 2, uses multiple LDL^t factorizations—the basic idea is that there will be an LDL^t factorization for each cluster of eigenvalues. A new LDL^t factorization is formed per cluster in order to increase relative gaps within the cluster. Once an eigenvalue has a large relative gap, Algorithm Getvec is invoked to compute the corresponding eigenvector, as seen in step II of Figure 2. Otherwise we are in the presence of a cluster Γ_c of eigenvalues and a new representation $L_cD_cL_c^t = LDL^t - \tau_c I$ needs to be computed. The shift τ_c is chosen in such a way such that (a) the new representation is relatively robust for the eigenvalues in Γ_c and (b) at least one of the shifted eigenvalues in Γ_c is relatively well separated from the others. The process is iterated if other clusters are encountered. The inputs to MR^3 are an index set Γ_0 that specifies the desired eigenpairs, the symmetric tridiagonal matrix T given by its traditional representation of diagonal and off-diagonal elements, and a tolerance *tol* for relative gaps. Note that the computational path taken by MR^3 depends on the relative gaps between eigenvalues. We again emphasize that MR^3 does not need any Gram–Schmidt orthogonalization of the eigenvectors.

3.1.2. Representation trees. The sequence of computations in Algorithm MR^3 can be pictorially expressed by a [illegible]. Such a tree contains information about how the eigenvalues are clustered (nodes of the tree) and what shifts are used to "break" a cluster (edges of the tree). A precise description of a representation tree can be found in [17]. Here we present a slightly simplified version of the tree, without specifying edge labels, which will facilitate the description of the parallel algorithm.

The root node of the representation tree is denoted by (L_0, D_0, Γ_0), where $L_0D_0L_0^t$ is the base representation obtained in step 1A of Algorithm MR^3; see Figure 2. An example representation tree is shown in Figure 3. Let Π_c be an internal node of the tree and let Π_p be its parent node. If Π_c is a nonleaf node, it will be denoted by (L_c, D_c, Γ_c), where the index set Γ_c is a proper subset of Γ_p, the index set of the parent node $\Pi_p = (L_p, D_p, \Gamma_p)$. Node Π_c indicates that $L_cD_cL_c^t$ is a representation that is computed by shifting, $L_pD_pL_p^t - \tau_c = L_cD_cL_c^t$, and will be used for computing the eigenvectors indexed by Γ_c. If Π_c is a leaf node instead, it will be denoted only by the singleton $\{c\}$, where $c \in \Gamma_p$. The singleton node $\{c\}$ signifies that the eigenvalue λ_c has a large relative gap with respect to the parent representation $L_pD_pL_p^t$, and its eigenvector will be computed by Algorithm Getvec.

Figure 3 gives an example of a representation tree for a matrix of size 11 for which all the eigenvectors are desired: the root contains the representation L_0, D_0 and the index set $\Gamma_0 = \{1, 2, \ldots, 11\}$. The algorithm begins by classifying the eigenvalues: in this example λ_1, λ_4, and λ_{11} are well separated, so in the tree they appear as singleton

ALGORITHM $\mathsf{MR}^3(T,\Gamma_0,tol)$.
Input: T is the given symmetric tridiagonal;
Γ_0 is the index set of desired eigenpairs;
tol is the input tolerance for relative gaps, usually set to 10^{-3}.
Output: $(\hat\lambda_j, \hat{\boldsymbol{v}}_j), j \in \Gamma_0$, are the computed eigenpairs.
1. Split T into irreducible subblocks $T_1, T_2, \ldots, T_\ell$.
For each subblock T_i, $i = 1, \ldots, \ell$, do:
A. Choose μ_i, and compute L_0 and D_0 such that $L_0 D_0 L_0^t = T_i + \mu_i I$ is a factorization that determines the desired eigenvalues and eigenvectors, λ_j and $\boldsymbol{v}_j$, $j \in \Gamma_0$, to high relative accuracy. In general, the shift μ_i can be in the interior of T_i's spectrum, but a safe choice is to make $T_i + \mu_i I$ positive or negative definite.
B. Compute the desired eigenvalues of $L_0 D_0 L_0^t$ to high relative accuracy by the dqds algorithm [24] or by bisection using a differential quotient-difference transform.
C. Form a work queue Q, and initialize $Q = \{(L_0, D_0, \Gamma_0)\}$. Call $\mathsf{MR}^3_\mathsf{Vec}(Q,tol)$.
end for
Subroutine $\mathsf{MR}^3_\mathsf{Vec}(Q,tol)$

While queue Q is not empty:
I. Remove an element (L, D, Γ) from the queue Q. Partition the computed eigenvalues $\hat\lambda_j, j \in \Gamma$, into clusters $\Gamma_1, \ldots, \Gamma_h$ according to their relative gaps and the input tolerance tol. The eigenvalues are thus designated as [illegible] (cluster size equals 1) or [illegible]. More precisely, if $\mathsf{rgap}(\hat\lambda_j) := \min_{i \neq j} |\hat\lambda_j - \hat\lambda_i|/|\hat\lambda_j| \geq tol$, then $\hat\lambda_j$ is isolated. On the other hand, all consecutive eigenvalues $\hat\lambda_{j-1}, \hat\lambda_j$ in a nontrivial cluster Γ_c ($|\Gamma_c| > 1$) satisfy $|\hat\lambda_j - \hat\lambda_{j-1}|/|\hat\lambda_j| < tol$.
II. For each cluster Γ_c, $c = 1, \ldots, h$, perform the following steps.
If $|\Gamma_c| = 1$ with eigenvalue $\hat\lambda_j$, i.e., $\Gamma_c = \{j\}$, **then** invoke Algorithm $\mathsf{Getvec}(L,D,\hat\lambda_j)$ to obtain the computed eigenvector $\hat{\boldsymbol{v}}_j$.
else
a. Pick τ_c near the cluster and compute $LDL^t - \tau_c I = L_c D_c L_c^t$ using the dstqds (differential form of stationary quotient-difference) transform; see [18, sect. 4.1] for details.
b. "Refine" the eigenvalues $\hat\lambda - \tau_c$ in the cluster so that they have high relative accuracy with respect to the computed $L_c D_c L_c^t$. Set $\hat\lambda \leftarrow (\hat\lambda - \tau_c)_{refined}$ for all eigenvalues in the cluster.
c. Add (L_c, D_c, Γ_c) to the queue Q.
end if
end for
end while

FIG. 2. *Algorithm* MR^3 *for computing orthogonal eigenvectors without using Gram–Schmidt orthogonalization.*

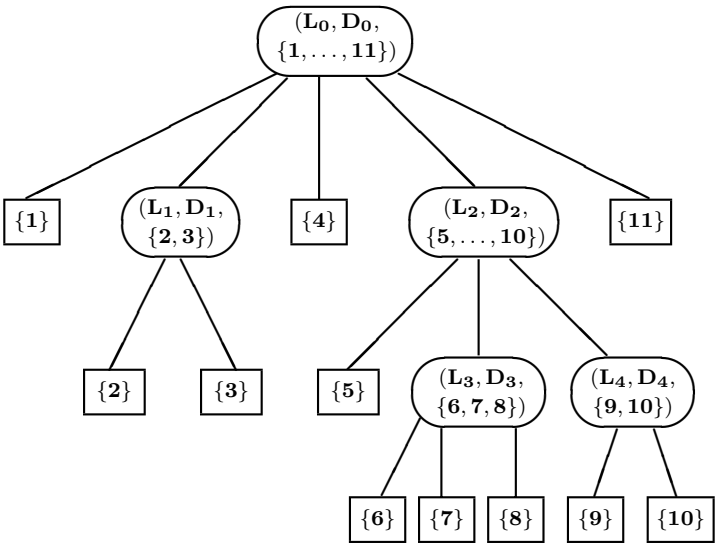


FIG. 3. *An example representation tree for a matrix of size* 11.

leaves (their eigenvectors can be directly computed by the calls $\mathsf{Getvec}(L_0, D_0, \hat{\lambda}_i)$, $i = 1, 4$, and 11). The second and third eigenvalues violate the condition $|(\hat{\lambda}_3 - \hat{\lambda}_2)/\hat{\lambda}_3| \geq tol$, and therefore they form a cluster; a new representation $L_1 D_1 L_1^t = L_0 D_0 L_0^t - \tau_1 I$ has to be computed and the two eigenvalues have to be refined to have high relative accuracy with respect to L_1 and D_1. This is represented by the node $(L_1, D_1, \{2, 3\})$. Similarly the eigenvalues $\{\lambda_5, \ldots, \lambda_{10}\}$ are clustered: a new representation $L_2 D_2 L_2^t = L_0 D_0 L_0^t - \tau_2 I$ is computed as shown by the node $(L_2, D_2, \{5, \ldots, 10\})$. This illustrates the working of Algorithm MR^3 for all the nodes at depth 1 in the representation tree of Figure 3.

The computation proceeds by classifying the eigenvalues of the two internal nodes $(L_1, D_1, \{2, 3\})$ and $(L_2, D_2, \{5, \ldots, 10\})$. From the tree it can be deduced that the two eigenvalues λ_2 and λ_3 in the first internal node are now relatively well separated, while the second node is further fragmented into a relatively well separated eigenvalue λ_5 and two nodes with clusters: nodes $(L_3, D_3, \{6, 7, 8\})$ and $(L_4, D_4, \{9, 10\})$. Finally these two clusters are further fragmented to yield singletons, and these eigenvectors are computed by Algorithm Getvec. Note that, as seen by the description of Algorithm MR^3 in Figure 2, the representation tree is processed in a breadth-first fashion.

For most matrices, the depth of the representation tree is quite small; we give here a sketch of the representation tree for two matrices that arise in the finite element modeling of automobile bodies (see section 4.1 for more details). The tree for the matrix auto.13786 (n = 13,786) has maximum depth 2; at depth 1 there are 12,937 singleton nodes, 403 clusters of size 2, 10 clusters of size 3, and 1 cluster of size 13. The tree for the matrix auto.12387 (n = 12,387) also has maximum depth 2 even though it has many more internal nodes: it has 5776 singletons and 1991 nodes corresponding to clusters with sizes ranging from 2 to 31.

A further note about reducible matrices: the solution is computed by iterating over the subblocks, and thus the sequence of computations can be captured by a forest of trees (one for each subblock) rather than by a single tree.

3.1.3. The parallel algorithm. We now describe Algorithm PMR^3. The input to the algorithm is a tridiagonal matrix T, an index set Γ_0 of desired eigenpairs, a tolerance parameter $_{,\prime}$, and the number of processors p that execute the algorithm. We target our algorithm to a distributed memory system, in which each processor has its own local main memory and communication is done by message-passing. We assume that the tridiagonal matrix is available on every processor before the algorithm is invoked.

We first discuss the parallelization strategy before describing the algorithm in detail. Let the size of the input index set Γ_0 be k; i.e., k eigenvalues and eigenvectors are to be computed. The total $O(kn)$ complexity of Algorithm MR^3 can be broken down into the work required at each node of the representation tree:

1. Each leaf node requires the computation of an eigenvector by Algorithm Getvec, which requires $O(n)$ operations (at most $2n$ divisions and $10n$ multiplications and additions).
2. Each internal node (L_c, D_c, Γ_c) requires (a) computation of the representation $L_c D_c L_c^t = L_i D_i L_i^t - \tau_c I$, and (b) refinement of the eigenvalues corresponding to the index set Γ_c so that they have high relative accuracy with respect to $L_c D_c L_c^t$. Computing the representation by the differential stationary quotient-difference transform requires $O(n)$ operations (n divisions, $4n$ multiplications and additions), while refinement of the eigenvalues can be done in $O(n|\Gamma_c|)$ operations using a combination of bisection and Rayleigh quotient iteration.

We aim for a parallel complexity of $O(\frac{nk}{p} + n)$ operations. Due to communication overheads, we will not attempt to parallelize $O(n)$ procedures, such as computing a single eigenvector or computing a new representation. Our strategy for the parallel algorithm will be to divide the leaf nodes equally among the processors, i.e., each processor will make approximately k/p calls to Algorithm Getvec. Thus each processor is assigned a set of eigenvectors that are to be computed locally.

However, before the leaf nodes can be processed the computation at the internal nodes needs to be performed. Each internal node (L_c, D_c, Γ_c) is associated with a subset of q processors, $q \leq p$, that are responsible for computing the eigenvectors in Γ_c. Since k eigenvectors are to be computed by a total of p processors, q approximately equals $\lceil p|\Gamma_c|/k \rceil$. Note that since $|\Gamma_c|$ is small in most practical applications (see comments towards the end of section 3.1.2), q is mostly small; in our examples, q is usually 1, sometimes 2, but rarely greater than 2. If q equals 1, the computation at each internal node is just done serially. If q is greater than 1, the parallel algorithm will process an internal node as follows: the representation $L_c D_c L_c^t$ is computed (redundantly) by each of the q processors. The eigenvalue refinement using bisection or Rayleigh quotient iteration ($O(n)$ per eigenvalue) is then parallelized over the q processors, resulting in a parallel complexity $(n|\Gamma_c|/q) = O(nk/p)$. Since many subsets of processors may be handling internal nodes at the same time, the overall parallel complexity is $O(\frac{nk}{p} + n)$. Note that due to communication overheads in a practical implementation, we impose a threshold on $|\Gamma_c|$ before the refinement is performed in parallel; if $|\Gamma_c|$ is below this threshold, the computation is carried out redundantly on all the q processors. On heterogeneous parallel machines, the redundant computation on every processor will need to be replaced by computation on a designated processor followed by a broadcast of the computed results to the other participating processors [7].

Figure 4 gives a description of Algorithm PMR^3 according to the strategy outlined

ALGORITHM $\mathsf{PMR}^3(T, \Gamma_0, tol, p)$. {executed by processor p_s}

Input: T is the given symmetric tridiagonal;
Γ_0 is the index set of desired eigenpairs;
tol is the input tolerance for relative gaps, usually set to 10^{-3};
p is the number of processors that execute the algorithm.

Output: $(\hat{\lambda}_j, \hat{\boldsymbol{v}}_j), j \in \Gamma_0$ are the computed eigenpairs.

1. Split T into irreducible subblocks $T_1, T_2, \ldots, T_\ell$.
For each subblock T_i, $i = 1, \ldots, \ell$, do:
A. Choose μ_i such that $L_0 D_0 L_0^t = T_i + \mu_i I$ is a factorization that determines the desired eigenvalues and eigenvectors, λ_j and $\boldsymbol{v}_j$, $j \in \Gamma_0$, to high relative accuracy.
B. Compute the desired eigenvalues Λ_i of $L_0 D_0 L_0^t$ to high relative accuracy by the dqds algorithm [24] or by parallel bisection using a differential quotient-difference transform. Let $\Gamma_i \subseteq \Gamma_0$ be the index set corresponding to Λ_i that contains the desired eigenvalues.
end for

2. Determine the subset $\Gamma_0^s \subseteq \Gamma_0$ of eigenvectors to be computed locally. Form a work queue $\bar{Q}$, and initialize it with all the subblocks (L_i, D_i, Γ_i) containing eigenvectors to be computed locally, i.e., the subblocks for which $\Gamma_i \cap \Gamma_0^s \neq \emptyset$, with $i = 1, \ldots, \ell$.

3. While queue $\bar{Q}$ is not empty:
I. Remove an element (L, D, Γ) from the queue $\bar{Q}$. Partition the computed eigenvalues $\hat{\lambda}_j, j \in \Gamma$, into clusters $\Gamma_1, \ldots, \Gamma_h$ according to their relative gaps and the input tolerance tol.
II. For each cluster Γ_c, $c = 1, \ldots, h$, perform the following steps:

If $\Gamma_c \subseteq \Gamma_0^{p_s}$ **then** all eigenvectors in Γ_c have to be computed locally. The eigenvectors are computed by invoking $\mathsf{MR}^3_\mathsf{Vec}((L, D, \Gamma_c), tol)$.

elseif $\Gamma_c \cap \Gamma_0^s = \emptyset$ **then** the cluster Γ_c does not contain any eigenvector that needs to be computed locally. Discard the cluster Γ_c.

elseif $\Gamma_c \cap \Gamma_0^s \neq \emptyset$ **then** the cluster Γ_c contains some eigenvectors to be computed locally, and needs to be further fragmented by the following steps.

- Pick τ_c near the cluster and compute $LDL^t - \tau_c I = L_c D_c L_c^t$ using the dstqds transform; see [18, sect. 4.1] for details.
- "Refine" the eigenvalues $\hat{\lambda} - \tau_c$ in the cluster so that they have high relative accuracy with respect to the computed $L_c D_c L_c^t$. Set $\hat{\lambda} \leftarrow (\hat{\lambda} - \tau_c)_{refined}$ for all eigenvalues in the cluster.
- Add (L_c, D_c, Γ_c) to the queue $\bar{Q}$.

end if

FIG. 4. *Algorithm* PMR^3 *for parallel computation of a subset* Γ_0 *of eigenvalues and eigenvectors.*

above. In order to show how subblocks of T are handled by the parallel algorithm, we do not assume that T is irreducible. Each processor will compute k/p eigenvectors, assuming k is divisible by p. Once the eigenvalues are grouped according to the subblocks and sorted (per subblock), work is assigned to the processors in a block cyclic manner; i.e., processor p_0 is assigned eigenvectors $1, 2, \ldots, k/p$, processor p_1 is assigned eigenvectors $k/p + 1, \ldots, 2k/p$, and so on. Thus the memory requirement to store the eigenvectors is exactly $(n \cdot k/p)$ floating point numbers per processor. The extra workspace required is linear in n, so problems of large size can be tackled. To give an idea of the limits of the sequential implementation, the size n of the largest problem that can be solved (with $k = n$) on a computer equipped with 1.5 GBytes of memory is about 14,000 when all the eigenvectors are required, while to solve a problem of size 30,000 a computer should be equipped with about 7 GBytes of main memory.

As seen in Figure 4, the eigenvectors are computed by invoking the sequential algorithms Getvec or MR^3_Vec (which in turn invokes Getvec). In terms of the representation tree, each processor maintains a local work queue $\bar{Q}$ of nodes (possibly leaves) which collectively index a superset of the eigenvectors to be computed locally. Initially all the processors have a single node in the queue corresponding to the desired index set Γ_0. The representation tree is traversed in a breadth-first fashion to fragment the clusters until all the eigenvectors of a node are local. Fragmenting a cluster is equivalent to descending one level in the representation tree. Nodes that contain eigenvectors all of which are associated with other processors are removed from the local queue of the processor. Once a node contains only eigenvectors to be computed locally, the sequential algorithms Getvec or MR^3_Vec are invoked depending on the size of the cluster. Recall that there is a tree for each subblock.

A word about the initial eigenvalue computation. The dqds algorithm for computing the eigenvalues is very fast but, like the QR algorithm, is inherently sequential. Moreover, the dqds algorithm cannot be adapted to compute k eigenvalues in $O(nk)$ time, instead always requiring $O(n^2)$ computations. On the other hand the bisection algorithm is easily parallelized [3]; however, bisection is rather slow. Thus in a parallel implementation, it is often preferable to redundantly compute the eigenvalues on each processor unless p is large or unless only a small subset of the n eigenvalues is desired. A quick calculation reveals the decision procedure to decide whether to use bisection or dqds. Bisection is linearly convergent, and finds one additional bit of an eigenvalue at every iteration. Thus, computing the desired eigenvalues by parallel bisection on p processors requires approximately $52n|\Gamma_0|/p$ operations in IEEE double precision arithmetic. On the other hand, dqds requires about 3 iterations, on average, to compute each eigenvalue to full relative accuracy. But dqds cannot be parallelized, so it requires approximately $3n^2$ operations irrespective of Γ_0. As a result, in our implementation, we use the dqds algorithm to redundantly compute all eigenvalues on the p processors when $3n^2 \leq 52n|\Gamma_0|/p$, i.e., $p \leq 17|\Gamma_0|/n$; otherwise we use parallel bisection.

We now illustrate the parallel execution of the algorithm on the matrix of Figure 3, assuming we want to compute all 11 eigenvectors on 3 processors. In Figure 5 we have annotated the representation tree of Figure 3 to show how the tree is processed by the 3 processors. Initially the eigenvalues $\Lambda_0 = \{\lambda_1, \ldots, \lambda_{11}\}$ are computed. Then based on the relative gaps between eigenvalues each processor determines whether a cluster is to be computed locally, has to be fragmented, or has to be discarded. The labels p_0, p_1, p_2 on the root node denote that each of the 3 processors is involved in the computation.

Processor p_0 classifies the eigenvalues λ_1 through λ_4, but discards all the clusters (possibly singletons) from λ_5 to the end of the spectrum as they do not contain

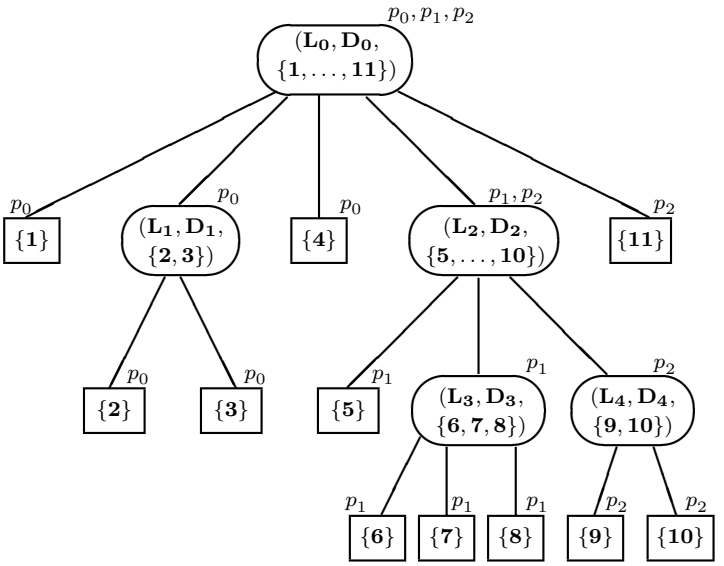


FIG. 5. *Representation tree annotated to describe the execution of the parallel algorithm. The matrix size is* 11*; the algorithm is run on* 3 *processors.*

eigenvectors to be computed locally. The clusters $\{\lambda_1\}$, $\{\lambda_2, \lambda_3\}$, $\{\lambda_4\}$ contain eigenvalues local to p_0, so the corresponding nodes in the tree are labeled p_0. In classifying the eigenvalues, both processors p_1 and p_2 find that the cluster $\{\lambda_5, \ldots, \lambda_{10}\}$ contains eigenvalues whose eigenvectors are to be computed locally: λ_5 through λ_8 for p_1 and λ_9, λ_{10} for p_2. The new representation is computed redundantly by both p_1 and p_2, and the refinement of eigenvalues λ_5 through λ_{10} is parallelized over p_1 and p_2. Thus the node is labeled p_1 as well as p_2 in Figure 5. The singleton λ_{11} is recognized as local by p_2 and therefore labeled p_2. The eigenvalue classification for node $(L_2, D_2, \{5, \ldots, 10\})$ is independently performed by processors p_1 and p_2: p_1 recognizes the clusters $\{\lambda_5\}$, $\{\lambda_6, \lambda_7, \lambda_8\}$ to contain local eigenvalues and discards the cluster $\{\lambda_9, \lambda_{10}\}$; vice versa for processor p_2. Nodes $\{5\}$ and $(L_3, D_3, \{6, 7, 8\})$ are therefore labeled p_1 while node $(L_4, D_4, \{9, 10\})$ is labeled p_2.

It is important to realize that the parallel algorithm traverses the sequential representation tree in parallel. Assuming identical arithmetic on all processors, this implies that the computed eigenvectors match exactly the ones computed by the sequential algorithm and therefore satisfy the same accuracy properties.

3.2. Householder reduction and backtransformation. To solve the dense, symmetric eigenproblem the solution to the tridiagonal eigenproblem is preceded by reduction to tridiagonal form and followed by a backtransformation stage to obtain the eigenvectors of the dense matrix. We will see in the performance section that Algorithm PMR^3 discussed in the previous section reduces the cost of the tridiagonal eigenproblem sufficiently so that it is the reduction and backtransformation stages that dominate the computation time. In this section, we give a brief overview of the major issues behind the parallel implementation of these stages. A more detailed discussion can be found in [4].

Reduction to tridiagonal form is accomplished through the application of a se-

quence of orthogonal similarity transformations; usually Householder transformations are preferred to Givens rotations. At the ith step in the reduction, a Householder transformation is computed that annihilates the elements in the ith column that lie below the first subdiagonal. This transformation is then applied to the matrix from the left and the right, after which the computation moves on to the next column of the updated matrix. Unfortunately, this simple "unblocked" algorithm is rich in matrix-vector operations (matrix-vector multiplications and symmetric rank-one updates, to be exact) which do not achieve high performance on modern microprocessors. Thus, a blocked version of the algorithm is derived from the unblocked algorithm by delaying updates to the matrix, accumulating those updates into a so-called symmetric rank-k update [22]. This casts the computation in terms of matrix-matrix multiplication, which can achieve much better performance. However, it is important to note that even for the blocked algorithm, approximately half the computation is in symmetric matrix-vector multiplication. This means that the best one can hope for is that implementations based on the blocked algorithm improve performance by a factor of 2 over implementations based on the unblocked algorithm.

The backtransformation stage applies the Householder transforms encountered during the reduction to tridiagonal form to the eigenvectors computed for the tridiagonal matrix. It is well known how to accumulate such Householder transforms into block Householder transforms so that the computation is again cast in terms of matrix-matrix multiplication [6]. This time essentially all computation involves matrix-matrix multiplication, allowing very high performance to be achieved.

Parallel implementation of both these stages now hinges on the fact that the parallel implementation of the symmetric matrix-vector multiplication, the symmetric rank-k update, and matrix-matrix multiplication is scalable, and can achieve high performance. Since these issues are well understood, we omit presenting them here and refer the reader to [32, 31]. Some subtle differences in the parallel implementations of these stages as supported by ScaLAPACK and PLAPACK are given in the appendix of [4]. Essentially, the ScaLAPACK and PLAPACK implementations are tuned for smaller and larger matrices, respectively.

4. Experimental results. This section presents timing results for the proposed algorithm. First we report results on the dense problem in section 4.2: it will be apparent that very large problems can now be tackled and that the tridiagonal eigenproblem is an order of magnitude faster than the reduction and backtransformation stages. In section 4.3 we focus on the tridiagonal eigensolvers and show that Algorithm PMR^3 achieves the best performance compared to previous algorithms.

4.1. Implementation details and test matrices. All experiments were conducted on a cluster of Linux workstations. Each node in the cluster consisted of a dual Intel Pentium 4 processor (2.4 GHz) with 2 GBytes of main memory. The nodes were connected via a high-performance network (2 Gigabit/s) from Myricom. In our experiments, only one processor per node was enabled. The reason for using one processor was primarily related to the fact that during early experiments it was observed that reliable timings were difficult to obtain when both processors were enabled. Notice that the qualitative behavior of the different algorithms and implementations is not affected by this decision, even if the quantitative results are.

We will often refer to our proposed parallel dense eigensolver as Dense PMR^3, and use PMR^3 to denote the tridiagonal eigensolver outlined in Figure 4; however, sometimes we just use PMR^3 when it is clear whether we are referring to the dense or tridiagonal eigensolver. Dense PMR^3 has been implemented using the PLAPACK

library for Householder reduction and backtransformation, while PMR^3 has been implemented in C and Fortran using the MPI library for communications and LAPACK for numerical routines.

We compare dense PMR^3 with the ScaLAPACK implementations of (a) bisection and inverse iteration (routine PDSYEVX) and (b) divide and conquer (routine PDSYEVD), and the PLAPACK implementation of (c) the QR algorithm (routine PLA_VDVt). All the routines have been compiled with the same optimization flags enabled and linked to the same high-performance BLAS library (the so-called GOTO BLAS, which in our experience achieve the highest performance on this machine [29]).

All dense eigensolvers have been tested on symmetric matrices of sizes ranging from 8000 to 128,000 with given eigenvalue distributions. We considered 4 types of diverse eigenvalue distributions:

1. UNIFORM (ε to 1):
$$\lambda_i = \varepsilon + (i-1) * \tau, \quad i = 1, 2, \ldots, n,$$
where $\tau = (1-\epsilon)/(n-1)$.
2. GEOMETRIC (ε to 1):
$$\lambda_i = \varepsilon^{(n-i)/(n-1)}, \quad i = 1, 2, \ldots, n.$$
3. RANDOM (ε to 1): the eigenvalues are drawn from a uniform distribution on the interval $[0, 1]$.
4. CLUSTERED at ε:
$$\lambda_1 \approx \lambda_2 \approx \cdots \approx \lambda_{n-1} \approx \varepsilon \text{ and } \lambda_n = 1.$$

In addition to the above "constructed" matrices, we also report timings for matrices arising in applications. We considered three matrices from computational quantum chemistry of sizes 966, 1687, and 2053, occurring, respectively, in modeling of the biphenyl molecule, study of bulk properties for the $SiOSi_6$ molecule, and solution of a nonlinear Schrödinger problem using the self-consistent Hartree–Fock method for the zeolite ZSM-5. More details on these matrices can be found in [2, 13].

We also considered three matrices (sizes 7923, 12387, 13786) that arise in frequency response analyses of automobile bodies. These matrices come from a symmetric matrix pencil arising from a finite element model of order 1 million or so, going through a process of dividing the entire structure into several thousand "substructures" using nested dissection and finding the "lowest" eigenvectors for each substructure. Projecting the matrix pencil onto the substructure eigenvector subspace and then converting to standard form followed by Householder reduction yields the test tridiagonal matrices. Details on producing these matrices can be found in [35].

Notice that the matrices for which we report results are at least one order of magnitude larger than the results reported in [43, 32].

4.2. Results for the dense problem. We now present performance results for computing all eigenpairs of a dense symmetric matrix highlighting the difference between the $O(n^3)$ reduction and backtransformation stages and the $O(n^2)$ tridiagonal computation of PMR^3.

When possible we compare the proposed algorithm against the ScaLAPACK implementation of divide and conquer (PDSYEVD) [43] since the latter routine is the fastest among the tridiagonal eigensolvers currently available in ScaLAPACK. All matrices considered in the following results have random distribution of eigenvalues, i.e.,

TABLE 1
Timings in seconds for different stages of the dense eigensolvers, $n = 8000$, random eigenvalue distribution.

Stage	Method	# of processors				
		2	4	8	16	32
Reduction	PLAPACK	549	289	156	98	64
	ScaLAPACK	_	_	155	85	50
Tridiagonal	PMR^3	**13**	**9.4**	**7.4**	**6.3**	**3.9**
	PDSTEDC	_	_	32.3	19	11.3
Backtransformation	PLAPACK	178	95	51	29	18
	ScaLAPACK	_	_	113	65	34

TABLE 2
Timings in seconds for different stages of the dense eigensolvers, $n = 15000$, random eigenvalue distribution.

Stage	Method	# of processors			
		8	16	32	64
Reduction	PLAPACK	996	546	340	257
	ScaLAPACK	_	487	263	128
Tridiagonal	PMR^3	**26.2**	**22.2**	**13.6**	**6.9**
	PDSTEDC	_	92.4	52.0	32.5
Backtransformation	PLAPACK	292	160	93	65.7
	ScaLAPACK	_	226	114	64.2

matrices designated RANDOM in section 4.1. Note that neither the reduction nor the backtransformation stage is affected by the distribution of eigenvalues in the input matrix. Comparisons with other tridiagonal eigensolvers (QR algorithm, bisection, and inverse iteration) on matrices with varying eigenvalue distributions are given in section 4.3.

In Tables 1 and 2 we report timings for matrices of sizes 8000 and 15,000, respectively. The stages of dense PMR^3 are labeled PLAPACK (for reduction and backtransformation) and PMR^3 (for the tridiagonal solution), while the stages for the routine PDSYEVD are similarly labeled ScaLAPACK and PDSTEDC (the tridiagonal divide and conquer routine). As mentioned in section 2, a major drawback of the divide and conquer algorithm is its extra $O(n^2)$ memory requirement. As a result, there are several instances where PMR^3 can be run on a particular matrix, but PDSTEDC cannot be run; the symbol "_" in the tables indicates that the experiment could not be run because of memory constraints.

Figure 6 gives a pictorial view of the dense PMR^3 timings in Table 2. It is easy to see from the figure that the tridiagonal stage is an order of magnitude faster than the reduction and backtransformation stages. As justified by the calculations in section 3.1.3, Algorithm PMR^3 uses dqds for the initial eigenvalue computation when $p \leq 17|\Gamma_0|/n$; here $|\Gamma_0| = n$, so dqds is used when $p \leq 17$; otherwise parallel bisection with a differential quotient-difference transform is used.

Next we present timings for an incremental test of dense PMR^3, keeping the memory allocation per processor constant while increasing the matrix size and the number of processors accordingly. To compute all eigenvectors, the memory requirement is quadratic in n; hence for fixed memory per processor when the matrix size is doubled, four times as many processors are needed. So we give results for dense PMR^3

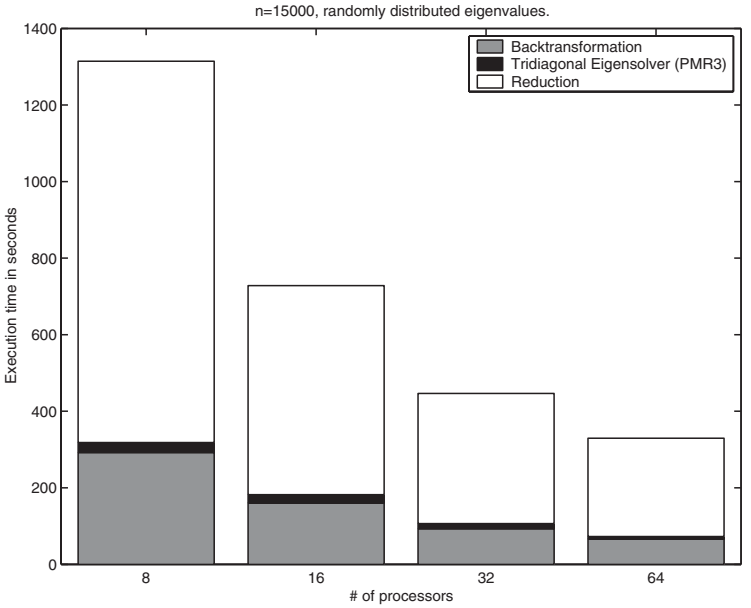


FIG. 6. *Timing breakdown for the dense* PMR^3 *algorithm,* $n = 15{,}000$.

TABLE 3

Timings for the dense eigensolvers. Columns 3 *to* 6 *give timings for dense* PMR^3*, while the last column reports extrapolated* (*) *numbers for a PLAPACK implementation of the QR algorithm. The numbers within parentheses in the fourth column* (PMR^3) *represent the time spent to compute eigenvalues. Entries in the last two columns are expressed in minutes.*

n	p	Reduction	PMR^3	Backtransformation	Total (dense PMR^3)	PLAPACK QR
8000	1	1081 s.	17.6 s. (5.4 s.)	348 s.	**24.1 min.**	100 min.*
16,000	4	2425 s.	38.8 s. (20.8 s.)	684 s.	**52.4 min.**	215 min.*
32,000	16	4876 s.	118 s. (97 s.)	1388 s.	**106.3 min.**	420 min.*
64,000	64	9638 s.	124 s. (104 s.)	2846 s.	**210.1 min.**	750 min.*
128,000	256	22,922 s.	128 s. (107 s.)	5827 s.	**481.3 min.**	

on matrices of sizes 8000, 16,000, 32,000, 64,000, and 128,000 using 1, 4, 16, 64, and 256 processors, respectively. Table 3 includes these timings in addition to extrapolated timings for the PLAPACK QR implementation.[1] We are unable to run divide and conquer (PDSYEVD) for this test, as it runs out of memory. The difference in complexity between the $O(n^2)$ tridiagonal eigensolver versus the $O(n^3)$ reduction and backtransformation is again obvious.

Indicating by $T_p(n)$ the total time from Table 3 to solve a problem of size n with p processors, we plot in Figure 7 the [illegible]: the ratio $T_p(n\sqrt{p})/T_1(n)$ with $p = 1, 4, 16, 64, 256$. Since the dense eigensolver has $O(n^3)$ complexity, doubling the matrix size and deploying four times as many processors, assuming constant

[1]The performance of the QR algorithm applied to tridiagonal matrices is very predictable; the extrapolated timings have been obtained by running the tridiagonal problem with a larger number of processors (thereby avoiding memory problems) and adjusting the times for reduction and backtransformation, assuming perfect parallelization.

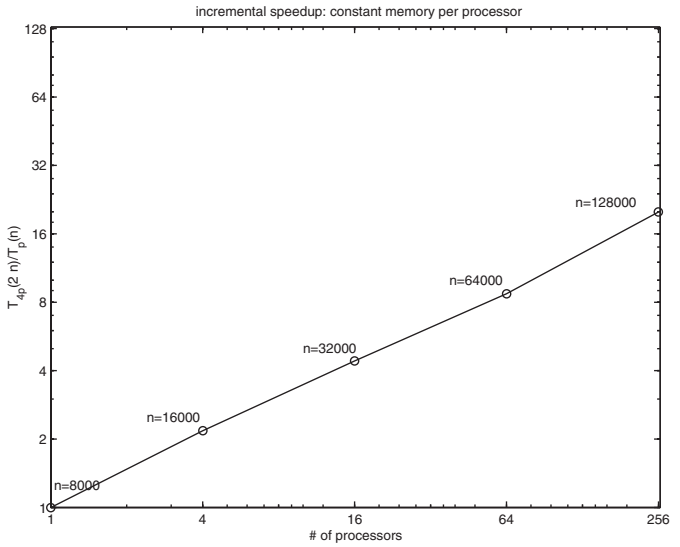


FIG. 7. *Incremental speedup.*

TABLE 4

Orthogonality and residual results for dense PMR^3. *Given the dense matrix A, the computed eigenvalues $\hat{\Lambda}$, and the computed eigenvectors $\hat{V}$, we display here* $\max_{i,j} |\hat{V}^T\hat{V} - I|_{ij}$ *(orthogonality) and* $\max_{i,j} |A\hat{V} - \hat{V}\hat{\Lambda}|_{ij}$ *(residual).*

Size		RANDOM	UNIFORM	GEOMETRIC	CLUSTERED
8000	Orthogonality	7.7e-11	2.0e-11	7.7e-12	2.5e-15
	Residual	3.7e-11	1.1e-11	1.5e-13	8.4e-16
15,000	Orthogonality	1.2e-10	1.0e-11	1.5e-11	1.7e-15
	Residual	1.9e-10	5.2e-12	6.8e-14	2.3e-16

performance per processor, the ratio $T_{4p}(2n)/T_p(n)$ equals

$$\frac{T_{4p}(2n)}{T_p(n)} = \frac{2^3 \ \frac{T_p(n)}{4}}{T_p(n)} = \frac{8}{4} = 2.$$

The log-log graph in Figure 7 testifies that the theoretical prediction is almost perfectly matched in practice. This demonstrates that dense PMR^3 scales up well in performance and larger problems can be tackled effectively when more processors are available.

Finally, in Table 4 we report on the accuracy of Algorithm PMR^3. Note that the eigenvalues and eigenvectors returned by PMR^3 are exactly the same as those computed by the sequential algorithm; therefore they satisfy the properties for residual and orthogonality described in [17].

4.3. Results for the tridiagonal problem. We now focus on the tridiagonal stage, comparing the performance of three different algorithms on matrices with differing eigenvalue distributions. With earlier QR or inverse iteration–based algorithms, the tridiagonal eigenproblem is often the bottleneck in the solution of the dense eigenproblem. For example, running sequential code for a matrix of size 2000 with uniformly distributed eigenvalues, the reduction and backtransformation stages take a total of about 32 seconds, while the tridiagonal eigenproblem takes 86 seconds

TABLE 5
Timings in seconds for tridiagonal eigensolvers, $n = 8000$.

Distribution	Method	# of processors					
		1	2	4	8	16	32
RANDOM	PMR^3	**17.4**	**13**	**9.4**	**7.4**	**6.2**	**3.9**
	PDSTEDC	_	_	_	32.3	19	11.3
	PLA_VDVt	*	2152	1117	532	273	142
	PDSTE(BZ+IN)	*	*	*	5844	5817	5656
UNIFORM	PMR^3	**15.3**	**12.4**	**9.1**	**7.4**	**6.5**	**3.8**
	PDSTEDC	_	_	_	35	20.4	12.2
	PLA_VDVt	*	*	1115	536	275	143
	PDSTE(BZ+IN)	*	*	*	143	141	132
GEOMETRIC	PMR^3	**11**	**9.4**	**6.3**	**4.9**	**4.1**	**3.5**
	PDSTEDC	_	_	_	16.7	9.9	6.4
	PLA_VDVt	*	*	945	458	247	128
	PDSTE(BZ+IN)	*	*	*	5297	5348	5233
CLUSTERED	PMR^3	**.101**	**.06**	**.051**	**.043**	**.04**	**.037**
	PDSTEDC	_	_	_	2.7	1.4	1.0
	PLA_VDVt	*	*	888	450	229	122
	PDSTE(BZ+IN)	*	*	*	6293	6501	6373

for the QR algorithm, 106 seconds for bisection and inverse iteration, 4 seconds for the divide and conquer algorithm, but only .8 seconds for MR^3. The timings for a matrix of size 4000 are 290 seconds for reduction and backtransformation, 1099 seconds for QR, 821 for bisection and inverse iteration, 24 seconds for divide and conquer, and 3.1 seconds for MR^3. Note the $O(n^2)$ behavior of MR^3 as compared to the other tridiagonal methods. The above timings use the optimized GOTO BLAS; however, it may not always be possible to obtain optimized BLAS, especially for newer CPUs. The timings for $n = 4000$ using Fortran BLAS are 1125 seconds for QR, 830 for bisection and inverse iteration, 176 seconds for divide and conquer, and 3.4 seconds for MR^3. Thus, divide and conquer suffers the most when using Fortran BLAS.

In the following we illustrate the parallel performance of Algorithm PMR^3. Tables 5 and 7 include the timings for PMR^3 when the number of processors and eigenvalue distributions are varied. We compare our results with ScaLAPACK's divide and conquer routine (PDSYEVD) whenever possible; again the symbol "_" in the table indicates that the experiment could not be run because of memory constraints. We also present timings for a PLAPACK implementation of QR (PLA_VDVt); we found the ScaLAPACK QR implementation (PDSYEV) to be slower than the equivalent algorithm implemented with PLAPACK, probably due to synchronization issues; for this reason we omit PDSYEV timings. Finally for the experiment on matrices of size 8000 we also show timings for the ScaLAPACK implementation of bisection and inverse iteration (PDSTEBZ+PDSTEIN). An asterisk indicates that the experiment has not been run because of excessive time needed.

Table 5 shows that for matrices of order 8000 the PMR^3 algorithm is faster than all the other algorithms. In particular it is several orders of magnitude faster than bisection and inverse iteration; notice that the performance of the latter does not improve when increasing the number of processors. This is due to the presence of a large cluster of eigenvalues as judged by PDSTEIN. Since PDSTEIN does not split clusters across processors, all eigenvectors of the large cluster end up being computed

TABLE 6
Time spent in the computation of the eigenvectors by PMR^3, $n = 8000$.

Distribution	Time (in seconds) to compute eigenvectors					
	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$
RANDOM	12	7.6	4.0	2.0	1.0	.5
UNIFORM	9.5	6.6	3.4	1.7	.9	.4
GEOMETRIC	7.8	6.0	3.0	1.6	.8	.4
CLUSTERED	.1	.059	.05	.042	.039	.037

TABLE 7
Timings in seconds for tridiagonal eigensolvers, $n = 15{,}000$.

Distribution	Method	# of processors			
		8	16	32	64
RANDOM	PMR^3	**26.2**	**22.2**	**13.6**	**6.9**
	PDSTEDC	_	92.4	52.0	32.5
	PLA_VDVt	*	2054	1000	565
UNIFORM	PMR^3	**26.4**	**22.7**	**13.6**	**6.8**
	PDSTEDC	_	101.5	56.8	35.3
	PLA_VDVt	*	2172	991	564
GEOMETRIC	PMR^3	**18.4**	**15.0**	**13.2**	**6.7**
	PDSTEDC	_	45.3	27.2	16.9
	PLA_VDVt	*	1890	839	467
CLUSTERED	PMR^3	**.16**	**.1**	**.1**	**.09**
	PDSTEDC	_	4.9	3.5	1.8
	PLA_VDVt	*	1448	827	440

on a single processor. Since PDSTE(BZ+IN) is not competitive, we do not present its results in Table 7. Routine PLA_VDVt is also several orders of magnitude slower than PMR^3, but it achieves perfect speedup and does not suffer from the memory problems of PDSTEDC.

Algorithm PMR^3 attains good performance, but for $p \leq 16$ it does not exhibit particularly good speedups due to the redundant serial eigenvalue computation. It is interesting to analyze the timings for just the eigenvector computations; we display these values in Table 6. Notice that the eigenvector computations attain very good speedups (the timings given for $p = 1$ are for the serial code, not the parallel code with $p = 1$). The exception is the case of clustered eigenvalues, for which the entire computation is extremely fast independent of the number of processors.

Results for matrices of size 15,000 are displayed in Table 7. The proposed algorithm is again the fastest in all the experiments. Finally, we consider six tridiagonal matrices coming from applications. Timings for PMR^3 are shown in Table 8; the parallel algorithm is again seen to be very fast; for example, it takes only $5 + 0.7$ seconds for a matrix of size 13,786 on 64 processors. Algorithm PMR^{3}achieves good speedups for the eigenvector computation in all cases, and the speedups for the eigenvalue computation are almost perfect when $p \geq 16$, i.e., when parallel bisection is used.

5. Conclusions. In this paper, we have presented a new parallel eigensolver for computing all or a subset of the eigenvectors of a dense symmetric matrix. The tridiagonal kernel is not only faster than previous algorithms, but also scales up well in memory requirements ($O(n)$ work space), allowing for very large eigenproblems to be solved. As a result, the time now spent in the tridiagonal eigenproblem is negligible

TABLE 8

Timings in seconds for PMR^3 *on matrices from applications. Biphenyl, $SiOSi_6$, and ZSM-5 are matrices from quantum chemistry, while the matrices* auto.7923, auto.12387, *and* auto.13786 *arise in the finite element modeling of automobile bodies.*

Matrix	Size	Time to compute eigenvalues			Time to compute eigenvectors						
		$p = 1, 2, 4, 8, 16$	$p = 32$	$p = 64$	$p = 1$	$p = 2$	$p = 4$	$p = 8$	$p = 16$	$p = 32$	$p = 64$
Biphenyl	966	.072	.064	.035	.133	.091	.05	.027	.014	.007	.004
$SiOSi_6$	1687	.235	.17	.09	.436	.274	.155	.085	.043	.022	.011
ZSM-5	2053	.339	.241	.12	.752	.445	.228	.122	.066	.033	.017
auto.7923	7923	4.3	3.4	1.75	11.2	7.3	3.9	2.0	1.0	.52	.26
auto.12387	12,387	15.8	8.3	4.2	31.6	20.0	10.2	5.2	2.6	1.3	.65
auto.13786	13,786	13.5	9.8	5.0	31.8	22.3	11.3	5.7	2.8	1.4	.7

compared to the stages of Householder reduction and backtransformation. Thus, the onus is now squarely on speeding up the latter two stages if the dense symmetric eigensolver is to be sped up further.

Acknowledgments. We are indebted to the Center for Computational Research at the University at Buffalo, SUNY, for the use of their 300 compute node Dell Linux Cluster. We also acknowledge the Texas Advanced Computing Center (TACC) for use of their 32 compute node IBM Pentium III Linux Cluster. We thank George Fann for providing the matrices from computational quantum chemistry, and Jeff Bennighof and Mintae Kim for the matrices arising from frequency analyses of automobile bodies.

REFERENCES

[1] E. ANDERSON, Z. BAI, C. BISCHOF, J. DEMMEL, J. DONGARRA, J. DU CROZ, A. GREENBAUM, S. HAMMARLING, A. MCKENNEY, S. OSTROUCHOV, AND D. SORENSEN, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, 1995.

[2] D. BERNHOLDT AND R. HARRISON, *Orbital invariant second order many-body perturbation on parallel computers: An approach for large molecules*, J. Chem. Physics, 102 (1995), pp. 9582–9589.

[3] H. J. BERNSTEIN AND M. GOLDSTEIN, *Parallel implementation of bisection for the calculation of eigenvalues of tridiagonal symmetric matrices*, Computing, 37 (1986), pp. 85–91.

[4] P. BIENTINESI, I. S. DHILLON, AND R. VAN DE GEIJN, *A Parallel Eigensolver for Dense Symmetric Matrices Based on Multiple Relatively Robust Representations*, Technical report TR-03-26, Department of Computer Sciences, University of Texas, Austin, TX, 2003.

[5] C. BISCHOF, S. HUSS-LEDERMAN, X. SUN, A. TSAO, AND T. TURNBULL, *Parallel performance of a symmetric eigensolver based on the invariant subspace decomposition approach*, in Proceedings of the Scalable High Performance Computing Conference, Knoxville, TN, 1994.

[6] C. BISCHOF AND C. VAN LOAN, *The WY representation for products of Householder matrices*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. s2–s13.

[7] L. BLACKFORD, A. CLEARY, J. DEMMEL, I. S. DHILLON, J. DONGARRA, S. HAMMARLING, A. PETITET, H. REN, K. STANLEY, AND R. WHALEY, *Practical experience in the numerical dangers of heterogeneous computing*, ACM Trans. Math. Software, 23 (1997), pp. 133–147.

[8] L. S. BLACKFORD, J. CHOI, A. CLEARY, E. D'AZEVEDO, J. DEMMEL, I. S. DHILLON, J. DONGARRA, S. HAMMARLING, G. HENRY, A. PETITET, K. STANLEY, D. WALKER, AND R. C. WHALEY, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, 1997.

[9] R. P. BRENT, *Algorithms for Minimization without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[10] J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[11] T. J. DEKKER, *Finding a zero by means of successive linear interpolation*, in Constructive Aspects of the Fundamental Theorem of Algebra, B. Dejon and P. Henrici, eds., Wiley-Interscience, New York, 1969, pp. 37–48.

[12] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.
[13] I. S. Dhillon, *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*, Ph.D. thesis, Computer Science Division, University of California, Berkeley, CA, 1997; available as UC Berkeley Technical report UCB//CSD-97-971.
[14] I. S. Dhillon, *Current inverse iteration software can fail*, BIT, 38 (1998), pp. 685–704.
[15] I. S. Dhillon, G. Fann, and B. N. Parlett, *Application of a new algorithm for the symmetric eigenproblem to computational quantum chemistry*, in Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing (Minneapolis, MN, 1997), CD-ROM, SIAM, Philadelphia, 1997.
[16] I. S. Dhillon and A. N. Malyshev, *Inner deflation for symmetric tridiagonal matrices*, Linear Algebra Appl., 358 (2002), pp. 139–144.
[17] I. S. Dhillon and B. N. Parlett, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra Appl., 387 (2004), pp. 1–28.
[18] I. S. Dhillon and B. N. Parlett, *Orthogonal eigenvectors and relative gaps*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 858–899.
[19] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling, *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.
[20] J. J. Dongarra and D. C. Sorensen, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. s139–s154.
[21] J. Dongarra, R. van de Geijn, and D. Walker, *A look at scalable dense linear algebra libraries*, in Proceedings of the Scalable High-Performance Computing Conference, IEEE Computer Society Press, Los Alamitos, CA, 1992, pp. 372–379.
[22] J. Dongarra, S. J. Hammarling, and D. C. Sorensen, *Block reduction of matrices to condensed forms for eigenvalue computations*, J. Comput. Appl. Math., 27 (1989), pp. 215–227.
[23] D. Elwood, G. Fann, and R. Littlefield, *PeIGS User's Manual*, Pacific Northwest National Laboratory, Richland, WA, 1993.
[24] K. Fernando and B. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.
[25] K. V. Fernando, *On computing an eigenvector of a tridiagonal matrix. Part* I: *Basic results*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1013–1034.
[26] G. J. F. Francis, *The QR transformation: A unitary analogue to the LR transformation.* I, Comput. J., 4 (1961/1962), pp. 265–271.
[27] G. J. F. Francis *The QR transformation.* II, Comput. J., 4 (1961/1962), pp. 332–345.
[28] W. J. Givens, *Numerical Computation of the Characteristic Values of a Real Symmetric Matrix*, Technical report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, 1954.
[29] K. Goto and R. A. van de Geijn, *On Reducing TLB Misses in Matrix Multiplication*, Technical report CS-TR-02-55, Department of Computer Sciences, University of Texas, Austin, TX, 2002; see also http://www.cs.utexas.edu/users/kgoto.
[30] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.
[31] J. Gunnels, C. Lin, G. Morrow, and R. van de Geijn, *A flexible class of parallel matrix multiplication algorithms*, in Proceedings of First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing, 1998, pp. 110–116.
[32] B. Hendrickson, E. Jessup, and C. Smith, *Toward an efficient parallel eigensolver for dense symmetric matrices*, SIAM J. Sci. Comput., 20 (1999), pp. 1132–1154.
[33] C. G. F. Jacobi, *Concerning an easy process for solving equations occurring in the theory of secular disturbances*, J. Reine Angew. Math., 30 (1846), pp. 51–94.
[34] E. R. Jessup and I. C. F. Ipsen, *Improving the accuracy of inverse iteration*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 550–572.
[35] M. F. Kaplan, *Implementation of Automated Multilevel Substructuring for Frequency Response Analysis of Structures*, Ph.D. thesis, University of Texas, Austin, TX, 2001.
[36] V. N. Kublanovskaya, *On some algorithms for the solution of the complete eigenvalue problem*, Zh. Vych. Mat., 1 (1961), pp. 555–570.
[37] T. Y. Li and Z. G. Zeng, *The Laguerre iteration in solving the symmetric tridiagonal eigenproblem, revisited*, SIAM J. Sci. Statist. Comput., 15 (1994), pp. 1145–1173.
[38] B. N. Parlett, *The Symmetric Eigenvalue Problem*, 2nd ed., SIAM, Philadelphia, 1997.
[39] B. N. Parlett, *Laguerre's method applied to the matrix eigenvalue problem*, Math. Comp., 18 (1964), pp. 464–485.
[40] B. N. Parlett and I. S. Dhillon, *Fernando's solution to Wilkinson's problem: An application of double factorization*, Linear Algebra Appl., 267 (1997), pp. 247–279.

[41] B. N. Parlett and I. S. Dhillon, *Relatively robust representations of symmetric tridiagonals*, Linear Algebra Appl., 309 (2000), pp. 121–151.

[42] G. Peters and J.H. Wilkinson, *The calculation of specified eigenvectors by inverse iteration, contribution* II/18, in Handbook of Automatic Computation, Vol. 2, Springer-Verlag, New York, Heidelberg, Berlin, 1971, pp. 418–439.

[43] F. Tisseur and J. Dongarra, *A parallel divide and conquer algorithm for the symmetric eigenvalue problem on distributed memory architectures*, SIAM J. Sci. Comput., 20 (1999), pp. 2223–2236.

[44] R. van de Geijn, *Using PLAPACK: Parallel Linear Algebra Package*, MIT Press, Cambridge, MA, 1997.

[45] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 67–92
© 2005 Society for Industrial and Applied Mathematics

MULTIGRID METHODS FOR IMPLICIT RUNGE–KUTTA AND BOUNDARY VALUE METHOD DISCRETIZATIONS OF PARABOLIC PDEs*

JAN VAN LENT† AND STEFAN VANDEWALLE†

Abstract. Advanced time discretization schemes for stiff systems of ordinary differential equations (ODEs), such as implicit Runge–Kutta and boundary value methods, have many appealing properties. However, the resulting systems of equations can be quite large and expensive to solve. Many techniques, exploiting the structure of these systems, have been developed for general ODEs. For spatial discretizations of time-dependent partial differential equations (PDEs) these techniques are in general not sufficient and also the structure arising from spatial discretization has to be taken into consideration. We show here that for time-dependent parabolic problems, this can be done by multigrid methods, as in the stationary elliptic case. The key to this approach is the use of a smoother that updates several unknowns at a spatial grid point simultaneously. The overall cost is essentially proportional to the cost of integrating a scalar ODE for each grid point. Combination of the multigrid principle with both time stepping and waveform relaxation techniques is described, together with a convergence analysis. Numerical results are presented for the isotropic heat equation and a general diffusion equation with variable coefficients.

Key words. parabolic partial differential equations, implicit Runge–Kutta methods, boundary value methods, multigrid, time stepping, waveform relaxation

AMS subject classifications. 65M55, 65L06

DOI. 10.1137/030601144

1. Introduction.

Large, stiff systems of ordinary differential equations (ODEs) are often solved using low order methods such as the implicit Euler scheme. This choice is usually dictated by implementation and computational complexity issues. Several more advanced time discretization schemes such as implicit Runge–Kutta (IRK) methods [9] and, more recently, boundary value methods (BVMs), block boundary value methods (BBVMs) [4, 11, 12], and general linear methods (GLMs) [6] have been developed. These methods offer high orders of accuracy together with favorable stability properties, error estimators, and other interesting features. Unfortunately, the linear algebra cost per time step is in general very large for these methods. When a general IRK method with three stages or a BVM with three steps is applied, for example, the dimension of the linear systems is three times that of the implicit Euler case. The cost of solving these systems with standard methods is prohibitively large.

For general ODEs many specialized linear algebra methods have been developed and implemented in ODE software. These methods exploit the structure arising from carefully chosen time discretization formulae, such as diagonally or singly implicit Runge–Kutta methods [5, 6, 9]. Since the systems arising from spatial discretization of a time-dependent partial differential equation (PDE) can be extremely large, it becomes important to use specialized methods that also exploit the structure arising from discretization in space. We will show here that for parabolic PDEs this can be

*Received by the editors October 8, 2003; accepted for publication (in revised form) October 21, 2004; published electronically August 17, 2005.
http://www.siam.org/journals/sisc/27-1/60114.html

†Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, B-3001 Leuven, Belgium (Jan.Vanlent@cs.kuleuven.ac.be, Stefan.Vandewalle@cs.kuleuven.ac.be). The first author is Research Assistant of the Fund for Scientific Research–Flanders (Belgium) (F.W.O.-Vlaanderen).

done very effectively by using multigrid methods. The methods exploiting structure arising from the time discretization formula can still be used as well, but with a good multigrid solver, this may no longer be absolutely necessary.

Some of the results in this paper, especially with respect to multigrid for IRK methods, have already been reported by other authors; see, e.g., [5, 19]. Their results are considerably extended here and cast into the more general framework of time stepping and block time stepping with GLMs. The use of multigrid for BVM and BBVM discretizations is new. This approach enables efficient solution of the BVM and BBVM linear systems, even for very large-scale semidiscrete PDE problems.

In section 2 we describe several high order time discretization schemes as well as the finite difference discretization used for the spatial domain. The multigrid iterative method is introduced in section 3, and we explain how it can be applied successfully for the systems arising from time stepping and block time stepping schemes. Section 4 shows how the convergence of the iterative methods for time stepping can be analyzed. The main theoretical results are presented in section 5 where waveform relaxation methods for block time stepping using GLMs and BBVMs are analyzed. Numerical results for two model problems, an isotropic and an anisotropic diffusion equation, are presented in section 6. A number of different IRK and BVM schemes are considered. The corresponding theoretical multigrid convergence rates are evaluated by means of a two-grid Fourier mode analysis and verified by extensive numerical experiments. One of the experiments involves the solution of an implicit linear system with more than 23 million unknowns, corresponding to a BVM discretization of the two-dimensional diffusion equation on a 256×256 spatial mesh, with 360 time steps computed simultaneously. One multigrid iteration step, gaining approximately one extra digit of solution accuracy, takes about 5 minutes on a 1.7 GHz Intel® Pentium® 4 system with 1 GB internal memory. Hence, the results clearly illustrate the effectiveness of the multigrid algorithm for use with implicit time integration methods, even for extremely large systems.

2. Time and space discretization schemes. In this article we consider parabolic equations of the form

$$\frac{\partial u}{\partial t} = \mathcal{L}u + f, \tag{2.1}$$

where $\mathcal{L}$ is an elliptic operator. To allow theoretical analysis and to keep the notation concise we assume that $\mathcal{L}$ is time-independent. All algorithms presented here are, however, applicable to the more general case where $\mathcal{L}$ is time-dependent. We also consider only discretization in time with a constant time step and finite difference discretizations in space on equidistant, rectangular grids with Dirichlet boundary conditions. The methods presented here can, however, be extended to three-dimensional problems and finite element discretizations on regular or irregular grids with more general boundary conditions.

2.1. Time discretization. We consider methods for stiff initial value problems of the form

$$\dot{y}(t) = f(t, y(t)), \tag{2.2}$$

where $t \in \Omega_t$, $y(t) \in V$, and $y(0) = y_0 \in V$ is given. The time domain can be bounded ($\Omega_t = [0, T]$) or unbounded ($\Omega_t = \mathbb{R}^+$). The space V can be, for example, $\mathbb{R}$ in the case of a scalar ODE, or $\mathbb{R}^m$ for a system of ODEs. Time-dependent PDEs also fit this setting with V a space of functions on some spatial domain Ω.

A brief description is given of linear multistep formulae, IRK methods, GLMs, and BVMs. For detailed information, we refer to the books [4, 5, 6, 8, 9]. The stability domains of the methods are described in section 5.1. The parameters of the schemes used in the numerical experiments can be found in Appendix A.

(LMFe) use solutions at several previous time steps in a formula of the form

$$\sum_{j=-k}^{0} \alpha_{k+j} y_{i+j} = \Delta t \sum_{j=-k}^{0} \beta_{k+j} f(t_{i+j}, y_{i+j}), \tag{2.3}$$

where $y_i \approx y(t_i) \in V$, $t_i = i\Delta t$, and α_j, β_j, $j = 0, \ldots, k$ are the parameters of the method. There is an equation for every $i = k, \ldots, n$. Since only y_0 is given, the values $y_1, \ldots, y_{k-1}$ have to be obtained by other methods. The index j ranges from $-k$ to 0 to indicate the similarity with the BVMs formulated below and to emphasize that the set of equations (2.3) for $i = k, \ldots, n$ can be described by a banded lower triangular matrix. For stiff initial value problems, implicit LMFe ($\beta_k \neq 0$) will be used. A well-known class of implicit LMFe are the so-called backward difference formulae (BDFe) [9, 16]. Note, however, that it is impossible to construct A-stable LMFe of order higher than 2. If $y(t) \in \mathbb{R}^m$, then in each step a system of m equations has to be solved. For the methods discussed below the systems can be much larger. Unlike the LMFe, the methods below can exhibit very good stability properties, even for high order discretizations.

(IRK) methods [5, 6, 9] use a number of intermediate quantities to advance one time step. These stage values $\tilde{y}_i \in V^s$ are calculated from

$$\tilde{y}_i = \mathbf{1}_s y_{i-1} + \Delta t A \tilde{f}(\tilde{t}_i, \tilde{y}_i), \tag{2.4}$$

where $\tilde{t}_i = \mathbf{1}_s t_{i-1} + c\Delta t$, $\mathbf{1}_s = [1 \cdots 1]^T \in \mathbb{R}^s$, and $\tilde{f}$ indicates componentwise application of f. The stage values $\tilde{y}_i$ are then used to calculate $y_i \in V$ using

$$y_i = y_{i-1} + \Delta t b^T \tilde{f}(\tilde{t}_i, \tilde{y}_i). \tag{2.5}$$

The parameters of the methods are contained in the matrix $A \in \mathbb{R}^{s \times s}$ and the vectors $b, c \in \mathbb{R}^s$, where s denotes the number of stage values. Some well-known methods are the Gauss, Radau, and Lobatto methods [5, 6, 9].

Both LMFe and IRK methods belong to the class of so-called (GLMs) [6, 9]. There are several ways to formulate this more general class of methods (see [6]). We choose to use the equations

$$\tilde{y}_i = C y_{i-1} + \Delta t A \tilde{f}(\tilde{t}_i, \tilde{y}_i), \tag{2.6}$$

$$y_i = D y_{i-1} + \Delta t B \tilde{f}(\tilde{t}_i, \tilde{y}_i), \tag{2.7}$$

where $A \in \mathbb{R}^{s \times s}$, $B \in \mathbb{R}^{r \times s}$, $C \in \mathbb{R}^{s \times r}$, and $D \in \mathbb{R}^{r \times r}$. Both $\tilde{y}_i \in V^s$ and $y_i \in V^r$ are vectors. This formulation highlights the analogy with IRK methods. The stage values $\tilde{y}_i$ are typically approximations of $y(t)$ for some t in the current time step. The vectors y_i can contain, for example, approximations of $y(t)$, brought forward from previous time steps or scaled approximations of derivatives of $y(t)$. In order to have a stable method, D has to be power bounded. Further conditions are needed to ensure that the method is consistent. The matrices C and D are typically chosen such that D is of rank one and also $D\mathbf{1}_r = \mathbf{1}_r$ and $C\mathbf{1}_r = \mathbf{1}_s$.

Within the family of GLMs many special methods have been developed that try to combine the efficiency of LMFe and the good stability properties of IRK methods. The computational complexity of IRK methods and GLMs depends mainly on the structure of A. In a Newton iteration for (2.6), the Jacobian is usually taken constant for all stages. An eigenvalue decomposition of A then reduces the system of order sm to s systems of order m. For general matrices A this will, however, involve using complex arithmetic. This can be avoided when A is lower triangular or diagonal. In some cases the cost can be further reduced if in addition the diagonal elements of A all have the same value. If A is diagonal, all stages can be solved in parallel. Increased efficiency can also be attained if A has only a single eigenvalue. For information and references about IRK and GLMs with special structures, see [5, 6, 9]. In the rest of the article, we will derive results for GLMs. The results for the subclass of IRK methods can be retrieved by taking $r = 1$, $B = b^T$, $C = \mathbf{1}_s$, and $D = 1$.

Another, more recent, type of method are the [illegible] (BVMs) [4, 11, 12], which can be interpreted as a generalization of the LMFe. The discretization is obtained by first assuming that the ODE is a boundary value problem and then imposing extra initial and final conditions on the values at the boundaries that are in fact unknown. A k-step BVM has for each $i = k_1, \ldots, n - k_2$ an equation of the form

$$\sum_{j=-k_1}^{k_2} \alpha_{k_1+j} y_{i+j} = \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} f(t_{i+j}, y_{i+j}). \tag{2.8}$$

The main difference with the traditional LMF is that j ranges from $-k_1$ to k_2 ($k = k_1 + k_2$). The extra $k_1 - 1$ initial and k_2 final equations are of the form

$$\sum_{j=0}^{k} \alpha_j^{(i)} u_j = \Delta t \sum_{i=0}^{k} \beta_j^{(i)} f(t_j, u_j), \qquad i = 1, \ldots, k_1 - 1, \tag{2.9}$$

$$\sum_{j=0}^{k} \alpha_j^{(i)} u_{n-k+j} = \Delta t \sum_{j=0}^{k} \beta_j^{(i)} f(t_{n-k+j}, u_{n-k+j}), \qquad i = n - k_2 + 1, \ldots, n, \tag{2.10}$$

where the coefficients $\alpha_j^{(i)}$ and $\beta_j^{(i)}$ are chosen such that the truncation errors for the initial and final conditions are of the same order as for the basic method (2.8). The matrices describing the systems involved are banded, as for LMFe, but no longer lower triangular. The n equations (2.8)–(2.10) can be written as

$$A_e y_e = \Delta t B_e f(t_e, y_e),$$

with $t_e \in \mathbb{R}^{n+1}$, $y_e \in V^{n+1}$, and A_e, $B_e \in \mathbb{R}^{n\times(n+1)}$. The e subscript indicates that the known initial value is included. The matrix A_e has the following structure:

$$A_e = \begin{bmatrix} \alpha_0^1 & \alpha_1^1 & \cdots & \alpha_k^1 & & & \\ \vdots & \vdots & \cdots & \vdots & & & \\ \alpha_0^{k_1-1} & \alpha_1^{k_1-1} & \cdots & \alpha_k^{k_1-1} & & & \\ & \alpha_0 & & \cdots & \alpha_k & & \\ & & \ddots & & \cdots & \ddots & \\ & & & \alpha_0 & & \cdots & \alpha_k \\ & & & \alpha_0^{n-k_2+1} & & \cdots & \alpha_k^{n-k_2+1} \\ & & & \vdots & & \cdots & \vdots \\ & & & \alpha_0^n & & \cdots & \alpha_k^n \end{bmatrix}. \tag{2.11}$$

Replacing α by β yields the structure for B_e. Using the partitions $A_e = [a_0|A]$ and $B_e = [b_0|B]$, where we split off the first columns, we can rewrite this as a system for the unknowns $y \in V^n$. We get

$$Ay = \Delta t B f(t, y) + g_0, \tag{2.12}$$

where $g_0 = -a_0 y_0 + \Delta t b_0 f(t_0, y_0)$ contains the initial condition.

We conclude this part on time discretization schemes with a method that can form an extra step in the process of transforming the continuous problem (2.2) to a discrete problem. Dividing the time interval into n subintervals leads to the sequence of subproblems

$$\begin{aligned} &\dot{y}_i(t) = f(t, y_i), \quad y_i : [t_{i-1}, t_i] \to V, \quad i = 1, \ldots, n, \\ &y_1(t_0) = y_0, \quad y_i(t_{i-1}) = y_{i-1}(t_{i-1}), \quad i = 2, \ldots, n, \end{aligned}$$

where the final value of one subinterval is used as the initial condition on the next. This procedure is called time windowing or block time stepping.

Applying time windowing and a k-step BVM using s time steps on each of the subintervals yields a [illegible] (BBVM) [4, 11, 12]:

$$Ay_i + A_0 y_{i-1} = \Delta t B f(t_i, y_i) + \Delta t B_0 f(t_{i-1}, y_{i-1}), \tag{2.13}$$

where $t_i \in \mathbb{R}^s$, $y_i \in V^s$, $A_0 = a_0 e_s^T$, $B_0 = b_0 e_s^T \in \mathbb{R}^{s \times s}$, and $e_s = [0 \cdots 0\, 1]^T \in \mathbb{R}^s$. This approach allows matrices A and $B \in \mathbb{R}^{s \times s}$ of much lower dimension. When $s \approx k$ these matrices can no longer be considered banded and the Toeplitz character is lost.

[illegible] 2.1 It is straightforward to formulate IRK methods as BBVMs and GLMs as so-called BBVMs with memory. The main difference in the latter case is that the matrices A_0 and B_0 are general. If y_i and $\tilde{y}_i$ are the values and stage values calculated by a GLM defined by A, B, C, and D, then using, for example,

$$\hat{y}_i = \begin{bmatrix} \tilde{y}_i \\ y_i \end{bmatrix},$$

we get a BBVM for $\hat{y}_i$ defined by the matrices

$$\hat{A} = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}, \quad \hat{A}_0 = \begin{bmatrix} 0 & -C \\ 0 & -D \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} A & 0 \\ B & 0 \end{bmatrix}, \quad \hat{B}_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

[illegible] 2.2 It is also possible to formulate BBVMs as GLMs. Using, for example,

$$\tilde{y}_i = y_i = \begin{bmatrix} \hat{y}_i \\ \hat{y}_{i-1} \end{bmatrix},$$

and assuming that $\hat{A}^{-1}$ exists, we get

$$A = B = \begin{bmatrix} \hat{A}^{-1}\hat{B} & \hat{A}^{-1}\hat{B}_0 \\ 0 & 0 \end{bmatrix}, \quad C = D = \begin{bmatrix} -\hat{A}^{-1}\hat{A}_0 & 0 \\ I & 0 \end{bmatrix}.$$

[illegible] 2.3 It is important to note that Δt has a different interpretation in GLMs and BBVMs. For BBVMs it is the distance between stage values; for GLMs it is the length of one time step, which contains s stage values.

For more information on BBVMs and GLMs we refer to [4, 6, 11, 12].

2.2. Space discretization. We consider equations of the form

$$\mathcal{L}u = (au_x)_x + (bu_y)_y + cu = f, \tag{2.14}$$

where u, a, b, c, and f are functions defined on a spatial domain. More specifically, we consider only the unit square $\Omega = [0,1]^2$ and define the equidistant rectangular grid

$$\begin{aligned} x_i &= i\Delta x, \qquad i = 0, \ldots, n_x, \qquad \Delta x = n_x^{-1}, \\ y_j &= j\Delta y, \qquad j = 0, \ldots, n_y, \qquad \Delta y = n_y^{-1}. \end{aligned}$$

Using central differences we can approximate (2.14) by

$$c_{i,j}^{0,0}u_{i,j} + c_{i,j}^{-1,0}u_{i-1,j} + c_{i,j}^{1,0}u_{i+1,j} + c_{i,j}^{0,-1}u_{i,j-1} + c_{i,j}^{0,1}u_{i,j+1} = f_{i,j},$$

where $i = 1, \ldots, n_x - 1$, $j = 1, \ldots, n_y - 1$, $u_{i,j} \approx u(x_i, y_j)$, and

$$\begin{aligned} c_{i,j}^{\pm 1,0} &= \Delta x^{-2} a(x_i \pm \Delta x/2, y_j), \qquad c_{i,j}^{0,\pm 1} = \Delta y^{-2} b(x_i, y_j \pm \Delta y/2), \\ c_{i,j}^{0,0} &= c(x_i, y_j) - c_{i,j}^{-1,0} - c_{i,j}^{1,0} - c_{i,j}^{0,-1} - c_{i,j}^{0,1}. \end{aligned}$$

Packing the unknowns at the interior points of the grid into a vector of dimension $m = (n_x - 1)(n_y - 1)$ using, for example, the ordering

$$u_k = u_{i,j}, \qquad k = (i-1)(n_y - 1) + j,$$

the discrete system can be represented in the matrix form

$$Lu = f, \tag{2.15}$$

with $u, f \in \mathbb{R}^m$ and $L \in \mathbb{R}^{m \times m}$. The boundary conditions are incorporated in f. The matrix L is sparse and has a block tridiagonal structure with tridiagonal subblocks.

3. Time integration: Time stepping and waveform relaxation.

3.1. Introduction: Classical iterative methods and multigrid. It is well known that for systems of equations involving large sparse matrices, such as the ones derived from discretizing PDEs, direct methods are often inefficient. Many iterative methods, on the other hand, deal very well with these kinds of problems. Two very simple iterative schemes are the Jacobi and the Gauss–Seidel methods. Both can be described using a splitting of the system matrix. Using $L = L^+ + L^-$ and (2.15) the iteration scheme becomes

$$L^+ u^{(\nu)} + L^- u^{(\nu-1)} = f.$$

The matrix L^+ is chosen such that this system can be easily solved. For the Jacobi method it contains only the diagonal of L. For the Gauss–Seidel method it is the lower triangular part of L. Unlike for the Jacobi method, the ordering of the unknowns is relevant for the Gauss–Seidel method. Common orderings on two-dimensional grids are lexicographical and red-black. For more variations and other iterative methods of this kind we refer to [26]. A description of methods appropriate for time-dependent parabolic equations with spatial anisotropy can be found in [27].

A Jacobi or Gauss–Seidel iteration step can be implemented very efficiently in the case of sparse matrices. The implementation is even simpler when uniform grids are

$\text{mg}(\mathbf{x}^{(0)}, \mathbf{A}, \mathbf{b}) \to \mathbf{x}^{(3)}$
- $\mathbf{x}^{(1)} \leftarrow \text{smooth}(\mathbf{x}^{(0)}, \mathbf{A}, \mathbf{b}, \mu_1)$
- $\bar{\mathbf{b}} \leftarrow \mathbf{R}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(1)})$
- $\bar{\mathbf{x}} \leftarrow \text{cgc}(\bar{\mathbf{A}}, \bar{\mathbf{b}})$
- $\mathbf{x}^{(2)} \leftarrow \mathbf{x}^{(1)} + \mathbf{P}\bar{\mathbf{x}}$
- $\mathbf{x}^{(3)} \leftarrow \text{smooth}(\mathbf{x}^{(2)}, \mathbf{A}, \mathbf{b}, \mu_2)$

$\text{cgc}(\bar{\mathbf{A}}, \bar{\mathbf{b}}) \to \bar{\mathbf{x}}$
- $\bar{\mathbf{x}} \leftarrow \bar{\mathbf{A}}^{-1}\bar{\mathbf{b}}$

$\text{smooth}(\mathbf{x}^{(0)}, \mathbf{A}, \mathbf{b}, \mu) \to \mathbf{x}^{(\mu)}$
- for $\nu = 1, \ldots, \mu$
  solve $\mathbf{A}^{+}\mathbf{x}^{(\nu)} = \mathbf{b} - \mathbf{A}^{-}\mathbf{x}^{(\nu-1)}$

ALGORITHM 3.1: Two-grid iteration for $\mathbf{A}\mathbf{x} = \mathbf{b}$, using the splitting $\mathbf{A} = \mathbf{A}^{+} + \mathbf{A}^{-}$.

$\text{cgc}(\bar{\mathbf{A}}, \bar{\mathbf{b}}) \to \bar{\mathbf{x}}$
- $\bar{\mathbf{x}} \leftarrow 0$
- γ times
  $\bar{\mathbf{x}} \leftarrow \text{mg}(\bar{\mathbf{x}}, \bar{\mathbf{A}}, \bar{\mathbf{b}})$

ALGORITHM 3.2: Coarse grid correction for multigrid iteration. V-cycle: $\gamma = 1$, W-cycle: $\gamma = 2$.

used. Unfortunately these simple splitting methods may converge very slowly. They do, however, form the basis of the very efficient multigrid methods discussed next.

Algorithm 3.1 gives a schematic overview of a two-grid iteration for a linear system of the form $\mathbf{A}\mathbf{x} = \mathbf{b}$. The quantities $\mathbf{x}$ and $\mathbf{b}$ are elements of a vector space approximating functions on a spatial domain. The operator $\mathbf{A}$ is a discrete approximation to an elliptic operator. The barred symbols indicate quantities and operators on a coarse grid. Restriction $\mathbf{R}$ and prolongation $\mathbf{P}$ are the intergrid operators, transferring quantities from the fine to the coarse level and vice versa. The whole iteration can be written as $\mathbf{x}^{(3)} = \mathbf{M}\mathbf{x}^{(0)} + \mathbf{q}$ or, equivalently, $\mathbf{e}^{(3)} = \mathbf{M}\mathbf{e}^{(0)}$ for the errors $\mathbf{e}^{(\cdot)} = \mathbf{x}^{(\cdot)} - \mathbf{x}$. It is straightforward to derive the following form for the two-grid iteration operator

$$\mathbf{M} = \mathbf{K}^{\mu_2}(\mathbf{I} - \mathbf{P}\bar{\mathbf{A}}^{-1}\mathbf{R}\mathbf{A})\mathbf{K}^{\mu_1},$$
$$\mathbf{K} = -\mathbf{A}^{+-1}\mathbf{A}^{-}.$$

In the two-grid iteration the equation $\bar{\mathbf{A}}\bar{\mathbf{x}} = \bar{\mathbf{b}}$ on the coarse grid is solved exactly. We can, however, solve this equation using the same iterative algorithm. Algorithm 3.2 shows how the two-grid iteration can be extended to a general multigrid iteration by applying this idea recursively. Discretizations on a hierarchy of increasingly coarser grids are used. The problem on the coarsest level can be solved exactly or approximately by applying a few smoothing steps.

For clarity we consider only linear problems. Nonlinear problems can be solved by embedding the multigrid algorithm as a linear solver in a Newton iteration or by using a nonlinear multigrid scheme [26]. Here, we use the multigrid method as a solver, not as a preconditioner for a Krylov subspace method [34]. Finally, we will consider only the standard geometric multigrid procedure for discretizations on regular, rectangular grids. For irregular grids and other multigrid techniques, such as the algebraic multigrid (AMG) method, we refer to [26].

The standard geometric multigrid method for discretization (2.15) of the elliptic equation (2.14) fits the framework described here with

$$\mathbf{x} = u, \quad \mathbf{b} = f,$$

$$\mathbf{A} = L, \quad \mathbf{A}^+ = L^+, \quad \mathbf{A}^- = L^-,$$

where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^m$. For a given coarsest grid, the hierarchy of grids is built up by successively doubling the number of intervals in both the x- and the y-direction. $\mathbf{R} = R$ and $\mathbf{P} = P$ are the standard full weighting restriction and bilinear interpolation operators. For details we refer to [26].

By changing the order in which the discretization and solution methods are applied, a whole variety of multigrid based solution methods for time-dependent problems can be derived. We now formulate the two families of methods we will focus on in the remainder of this article.

3.2. Iterative methods for time stepping schemes. If we apply any of the time discretization schemes discussed in section 2 to a time-dependent PDE, we get a so-called time stepping method. In the case of a parabolic initial value problem, the elliptic subproblems can be solved using a multigrid algorithm. One smoothing step consists of a loop over all spatial grid points, updating all unknowns for the considered time step simultaneously. For LMF methods there is only one unknown per grid point and only scalar computations are required. An s-dimensional system has to be solved at each grid point in the IRK, GLM, and BBVM cases. Since s is usually small, this can be done using a standard dense solver. It is still possible to use eigenvalue decompositions or methods with a special structure, as described in section 2, but this will generally not be necessary. The BVM case cannot be considered a time stepping method since the values at all time steps have to be obtained from a system of equations that is banded but not triangular or block triangular.

Applying to (2.1) first a time discretization from section 2.1 and then the space discretization and the splitting of the discretized elliptic operator introduced in sections 2.2 and 3.1, we get the following iterative methods, which can all be used as the smoother in a multigrid scheme.

Using an LMF (2.3) results in

$$\begin{aligned}\sum_{j=-k}^{0} \alpha_{k+j} u_{i+j}^{(\nu)} = \Delta t \sum_{j=-k}^{0} \beta_{k+j} L^+ u_{i+j}^{(\nu)} \\ + \Delta t \sum_{j=-k}^{0} \beta_{k+j} L^- u_{i+j}^{(\nu-1)} + \Delta t \sum_{j=-k}^{0} \beta_{k+j} f_{i+j}.\end{aligned} \tag{3.1}$$

Note that additional methods have to be provided to approximate the initial values $u_i \in \mathbb{R}^m, i = 1, \ldots, k-1$, to turn this into a practical method.

When we apply a BVM (2.8), we get

$$\begin{aligned}\sum_{j=-k_1}^{k_2} \alpha_{k_1+j} u_{i+j}^{(\nu)} = \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} L^+ u_{i+j}^{(\nu)} \\ + \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} L^- u_{i+j}^{(\nu-1)} + \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} f_{i+j}.\end{aligned} \tag{3.2}$$

The initial and final conditions can be derived in the same way from (2.9) and (2.10).

For a GLM described by (2.6) and (2.7), the smoother becomes

$$\begin{aligned}\tilde{u}_i^{(\nu)} = (C \otimes I_m) u_{i-1}^{(\nu)} + \Delta t (A \otimes L^+) \tilde{u}_i^{(\nu)} \\ + \Delta t (A \otimes L^-) \tilde{u}_i^{(\nu-1)} + \Delta t (A \otimes I_m) \tilde{f}_i,\end{aligned} \tag{3.3}$$

$$u_i^{(\nu)} = (D \otimes I_m)u_{i-1}^{(\nu)} + \Delta t(B \otimes L^+)\tilde{u}_i^{(\nu)} + \Delta t(B \otimes L^-)\tilde{u}_i^{(\nu-1)} + \Delta t(B \otimes I_m)\tilde{f}_i, \tag{3.4}$$

where "$\otimes$" denotes the matrix Kronecker product.

Discretizing time using a BBVM (2.13) results in the iterative method

$$\begin{aligned}(A\otimes I_m)u_i^{(\nu)} &+ (A_0 \otimes I_m)u_{i-1}^{(\nu)} \\ &= \Delta t(B \otimes L^+)u_i^{(\nu)} + \Delta t(B \otimes L^-)u_i^{(\nu-1)} + \Delta t(B \otimes I_m)f_i \\ &\quad + \Delta t(B_0 \otimes L^+)u_{i-1}^{(\nu)} + \Delta t(B_0 \otimes L^-)u_{i-1}^{(\nu-1)} + \Delta t(B_0 \otimes I_m)f_{i-1}.\end{aligned} \tag{3.5}$$

The BVM discretization corresponds to taking the number of subintervals $n = 1$.

3.3. Waveform relaxation methods for block time stepping schemes. An interesting modification of the time stepping schemes is to collect the equations associated with a consecutive set of time steps together into one big system of equations [10, 25, 35]. Combining four BDF1 time steps, for example, results in a system of equations of the form

$$(A \otimes I_m)u = \Delta t(B \otimes L)u + \Delta t(B \otimes I_m)f + g_0,$$

where $u, f, g_0 \in \mathbb{R}^{4m}$, $B = I_4$, and

$$A = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & -1 & 1 & \\ & & -1 & 1 \end{bmatrix}.$$

Information about the previous time step is incorporated in g_0 as in (2.12). The splitting $L = L^+ + L^-$ leads to an iteration of the form

$$(A \otimes I_m)u^{(\nu)} = \Delta t(B \otimes L^+)u^{(\nu)} + \Delta t(B \otimes L^-)u^{(\nu-1)} + \Delta t(B \otimes I_m)f + g_0.$$

This can be seen as a collective update for all the unknowns on the set of time levels considered and can be executed by solving a series of small systems of equations, one 4 x 4 system at each grid point in this particular example. We call this type of method a block time stepping scheme and the associated iteration a discrete waveform relaxation method. As mentioned in the previous section, "block time stepping" is the only option for BVMs. Similar extensions are possible for any other LMFe as well as for GLMs and BBVMs. For LMFe the subsystems are lower triangular and characterized by matrices A and B of the form

$$A = \begin{bmatrix} \alpha_k & & & & \\ \vdots & \ddots & & & \\ \alpha_0 & \dots & \alpha_k & & \\ & \ddots & \ddots & \ddots & \\ & & \alpha_0 & \dots & \alpha_k \end{bmatrix}.$$

For BVMs they are banded of the form (2.11), and in the case of GLMs or BBVMs they are block lower triangular. For all these cases, the systems that arise when using a discrete waveform relaxation method can be easily solved by direct methods.

. . . . 3 1 Another way of deriving the methods in this section is to first transform (2.1) into the system of ODEs

$$\dot{u} = Lu + f \tag{3.6}$$

by discretizing only the spatial domain. This fits the framework outlined in section 3.1 with $u, f : \Omega_t \to R^m$ and the operator $\mathbf{A} = \frac{d}{dt} - L$. The splitting $\mathbf{A}^+ = \frac{d}{dt} - L^+$, $\mathbf{A}^- = -L^-$ gives us the smoother

$$\dot{u}^{(\nu)} = L^+ u^{(\nu)} + L^- u^{(\nu-1)} + f \tag{3.7}$$

and thus a multigrid algorithm. This algorithm can be defined on a time interval that is finite, $\Omega_t = [0, T]$, or infinite, $\Omega_t = [0, \infty)$. In this continuous waveform relaxation method, one smoothing step consists of the solution of m scalar ODEs, one for each grid point. Applying any of the time discretization schemes to solve these brings us back to the discrete waveform relaxation methods discussed before.

3.4. Remarks. The equations to be solved for the time stepping and block time stepping schemes are exactly the same. The corresponding multigrid methods differ only in the order in which the unknowns are updated. In the time stepping case the ODE solver forms the outer loop and a multigrid solver is used within each time step. In the waveform relaxation case, the multigrid solver forms the outer loop and the ODE solver is used within its smoother. Summarizing, we can say that a time stepping method loops over time steps, then multigrid iterations, then multigrid levels, then smoothing steps, and finally grid points. A waveform relaxation method loops over multigrid iterations, then multigrid levels, then smoothing steps, then grid points, and finally time steps. In both cases the cost of one multigrid iteration is equivalent to the cost of solving a scalar ODE at each grid point.

The approach discussed here has to be contrasted with other approaches for the solution of systems arising from the discretization of general ODEs. One approach to solving systems with matrices of the form $A \otimes I_m - \Delta t B \otimes L$, obtained by using BVMs, is to use circulant approximations for the matrices A and B to precondition a Krylov subspace iteration (see, for example, [2, 3, 7, 13]). This essentially comes down to splitting the matrices A and B, whereas we split the matrix L. The approaches are, of course, not mutually exclusive, but for the small subsystems that have to be solved at each grid point in our multigrid smoothers, direct methods are hard to beat. Finally, we note that smoothers derived by directly applying the Jacobi or Gauss–Seidel methods to the system matrix give very unsatisfactory results.

4. Convergence analysis for the time stepping case. In this section we present a convergence analysis of the iterative methods for time stepping schemes discussed in the previous sections. We use the asymptotic convergence factor, i.e., the spectral radius of the iteration operator, as a measure for convergence. In the time stepping case the iterates are vectors and the iteration operator is a matrix. The convergence factors can therefore be derived using only linear algebra manipulations.

4.1. Classical iterative methods. For the LMF case, we obtain from (4.1) the error iteration

$$\alpha_k I_m e_i^{(\nu)} = \Delta t \beta_k L^+ e_i^{(\nu)} + \Delta t \beta_k L^- e_i^{(\nu-1)}, \tag{4.1}$$

where the iteration errors are $e_i^{(\nu)} = u_i^{(\nu)} - u_i \in \mathbb{R}^m$. Let $\mathcal{K}_{\Delta t}$ be the corresponding iteration operator. The convergence factor we are looking for is

$$\begin{aligned}\rho\left(\mathcal{K}_{\Delta t}\right) &= \rho\left((\alpha_k I_m - \Delta t \beta_k L^+)^{-1} \Delta t \beta_k L^-\right) \\ &= \rho\left(\left(\frac{1}{\Delta t}\frac{\alpha_k}{\beta_k} I_m - L^+\right)^{-1} L^-\right),\end{aligned}$$

where we assumed that $\beta_k \neq 0$ and $\frac{1}{\Delta t}\frac{\alpha_k}{\beta_k} \notin \sigma(L^+)$. Introducing the operator

$$K(z) = (z I_m - L^+)^{-1} L^-, \tag{4.2}$$

we can write this as

$$\rho\left(\mathcal{K}_{\Delta t}\right) = \rho\left(K\left(\frac{1}{\Delta t}\frac{\alpha_k}{\beta_k}\right)\right).$$

We can now state the following theorem.

THEOREM 4.1. $\mathcal{K}_{\Delta t}$ [illegible] (3.1) [illegible] $\Sigma = \left\{\frac{\alpha_k}{\beta_k}\right\}$ [illegible] $\frac{1}{\Delta t}\Sigma \cap \sigma(L^+) = \phi$ [illegible]

$$\rho(\mathcal{K}_{\Delta t}) = \left[\rho\big(K(z)\big)\right]_{z \in \frac{1}{\Delta t}\Sigma}. \tag{4.3}$$

Note that for the Gauss–Seidel and Jacobi methods, the operator $K(z)$ is the iteration matrix for the equivalent splitting method applied to

$$zu = Lu + f, \tag{4.4}$$

which is the Laplace transform of the system of ODEs (3.6).

For block BVMs (with standard BVMs and LMFe as special cases), we get

$$(A \otimes I_m) e_i^{(\nu)} = \Delta t (B \otimes L^+) e_i^{(\nu)} + \Delta t (B \otimes L^-) e_i^{(\nu-1)}, \tag{4.5}$$

where $e_i^{(\nu)} = u_i^{(\nu)} - u_i \in \mathbb{R}^{sm}$. Let $\mathcal{K}_{\Delta t}$ be the corresponding iteration operator. Assuming that B^{-1} exists and that $\frac{1}{\Delta t}\sigma(B^{-1}A) \cap \sigma(L^+) = \phi$, we get

$$\begin{aligned}\rho\left(\mathcal{K}_{\Delta t}\right) &= \rho\left((A \otimes I_m - \Delta t B \otimes L^+)^{-1} \Delta t (B \otimes L^-)\right) \\ &= \rho\left(\left(\frac{1}{\Delta t} B^{-1}A \otimes I_m - I_s \otimes L^+\right)^{-1} (I_s \otimes L^-)\right),\end{aligned}$$

which we can write as

$$\rho\left(\mathcal{K}_{\Delta t}\right) = \rho\left(K\left(\frac{1}{\Delta t} B^{-1} A\right)\right).$$

This leads us to the following theorem.

THEOREM 4.2. $\mathcal{K}_{\Delta t}$ [illegible] (3.5) [illegible] $\Sigma = \sigma(B^{-1}A)$ [illegible] $\frac{1}{\Delta t}\Sigma \cap \sigma(L^+) = \phi$ [illegible]

$$\rho(\mathcal{K}_{\Delta t}) = \max_{z \in \frac{1}{\Delta t}\Sigma} \rho(K(z)). \tag{4.6}$$

By choosing Σ an appropriate subset of $\mathbb{C}$, (4.6) will apply to all other time discretization methods considered here (max is replaced by sup when Σ is an infinite set).

In the IRK/GLM time stepping case, the iterative method is used to find the solutions of the equations for the stage values. The values u_i are calculated directly from $\tilde{u}_i$ and do not take part in the iteration. The iteration for the errors $\tilde{e}_i^{(\nu)} = \tilde{u}_i^{(\nu)} - \tilde{u}_i \in \mathbb{R}^{sm}$ becomes

$$\tilde{e}_i^{(\nu)} = \Delta t(A \otimes L^+)\tilde{e}_i^{(\nu)} + \Delta t(A \otimes L^-)\tilde{e}_i^{(\nu-1)}. \tag{4.7}$$

The convergence factor of the corresponding iteration operator $\mathcal{K}_{\Delta t}$ can be written as

$$\begin{aligned}\rho\left(\mathcal{K}_{\Delta t}\right) &= \rho\left(\left(I_s \otimes I_m - \Delta t A \otimes L^+\right)^{-1}(\Delta t A \otimes L^-)\right) \\ &= \rho\left(K\left(\frac{1}{\Delta t}A^{-1}\right)\right),\end{aligned}$$

where we assume that A^{-1} exists and $\frac{1}{\Delta t}\sigma(A^{-1}) \cap \sigma(L^+) = \phi$.

THEOREM 4.3. [illegible] $\mathcal{K}_{\Delta t}$ [illegible] (3.3) [illegible] $\Sigma = \sigma(A^{-1})$ [illegible] $\frac{1}{\Delta t}\Sigma \cap \sigma(L^+) = \phi$ [illegible]

$$\rho(\mathcal{K}_{\Delta t}) = \max_{z \in \frac{1}{\Delta t}\Sigma} \rho(K(z)). \tag{4.8}$$

4.2. Multigrid. A standard approach for the convergence analysis of geometric multigrid methods is to consider a two-grid method applied to an equation with constant coefficients and periodic boundary conditions for the spatial domain. These simplifications make it possible to do a Fourier mode analysis. This so-called two-grid local Fourier mode analysis provides an indication of what convergence to expect for the multigrid iteration. For more information on convergence analysis of multigrid methods we refer to [26, 30, 34]. A three-grid analysis can be found in [33]. As an example we show how the analysis for classical iterative methods can be extended to a two-grid time stepping method using a BBVM time discretization. The results for the other time stepping schemes are analogous.

We first derive the equivalent to the operator $K(z)$ by applying the framework in section 3.1 to (4.4). Using $\mathbf{A} = zI_m - L$, $\mathbf{A}^+ = zI_m - L^+$, $\mathbf{A}^- = -L^-$, $\mathbf{R} = R$, $\mathbf{P} = P$, and $\bar{\mathbf{A}} = zI_{\bar{m}} - \bar{L}$, we obtain the two-grid iteration matrix

$$M(z) = K(z)^{\mu_2}(I - P(zI_{\bar{m}} - \bar{L})^{-1}R(zI_m - L))K(z)^{\mu_1}, \tag{4.9}$$

where $K(z)$ is defined as before.

Doing the same for $\mathbf{A} = A \otimes I_m - \Delta t B \otimes L$, $\mathbf{A}^+ = A \otimes I_m - \Delta t B \otimes L^+$, $\mathbf{A}^- = -\Delta t B \otimes L^-$, $\mathbf{R} = I_s \otimes R$, $\mathbf{P} = I_s \otimes P$, and $\bar{\mathbf{A}} = A \otimes I_{\bar{m}} - \Delta t B \otimes \bar{L}$, we derive that the iteration operator for the two-grid method using (3.5) as smoother can be written as

$$\mathcal{M}_{\Delta t} = M\left(\frac{1}{\Delta t}B^{-1}A\right).$$

The results from the previous section therefore hold for the two-grid iteration if we replace $K(z)$ by $M(z)$ and extend the condition on the set Σ to

$$\frac{1}{\Delta t}\Sigma \cap (\sigma(L^+) \cup \sigma(\bar{L})) = \phi.$$

5. Convergence analysis for the block time stepping case. Waveform relaxation schemes were first introduced and analyzed in a qualitative way in the electrical engineering literature [17, 31, 32]. Here we follow the more quantitative convergence analysis introduced in [21, 22]. This analysis was extended to multigrid waveform relaxation [19, 28], IRK time discretizations [5, 19], time periodic problems [28], and space discretizations using finite elements [14, 15]. In this section we present an analysis for GLMs and BBVMs. Notation and presentation are mainly taken from the last two references, except that for clarity we will not consider finite element discretizations here. As usual, we consider methods defined on infinite time intervals since this provides results that correspond better to numerically observed behavior. For further motivation of this approach we refer to [14, 21, 28]. Weighted norms or pseudospectra can also be used to obtain meaningful information about convergence. In [20] the convergence analysis on infinite time intervals is shown to be a limiting case of the pseudospectral analysis.

5.1. Preliminaries: The stability domain and its complement. Stability domains of time discretization schemes play an important part in the convergence analysis of discrete waveform relaxation methods. We recall the definitions of the stability domains of the methods considered here. For each method we also introduce a set $\Sigma \subset \mathbb{C}$ that we will later encounter in the convergence analysis. For the case of LMFe a close relation exists between Σ and the stability domain. We show that the same relation holds for GLMs (and therefore also for IRK methods), BVMs, and BBVMs. As a bonus, this relation provides us with a convenient way to calculate points on a curve containing the boundary of the stability domain.

The linear stability of a time discretization scheme is studied using the Dahlquist test equation [9]

$$y' = \lambda y, \quad y(0) = y_0. \tag{5.1}$$

The stability domain consists of all $z = \Delta t\lambda \in \mathbb{C}$ for which the discrete solution is bounded. The interior of the stability domain, also called the open stability domain, consists of all z for which the discrete solution converges to 0. Only the definitions for open stability domains are given since they are simpler and sufficient for the rest of the discussion.

For an LMF (2.3) we define the polynomials

$$a(w) = \sum_{j=0}^{k} \alpha_j w^j \quad \text{and} \quad b(w) = \sum_{j=0}^{k} \beta_j w^j \tag{5.2}$$

and make the common assumptions of irreducibility, consistency, and zero stability [6, 8]. Furthermore, we consider only implicit methods ($\beta_k \neq 0$). We can use the same definitions for BVMs [4]. Again, the methods are assumed to be irreducible, consistent, and zero stable. BVMs are implicit by nature.

DEFINITION 5.1. [illegible] (k_1, k_2, k_2) [illegible] k_1 [illegible] k_2 [illegible] k_3 [illegible]

By applying each of the schemes in section 2.1 to the test equation (5.1), we obtain the following definitions.

DEFINITION 5.2. [illegible] S [illegible] k [illegible] $z \in \mathbb{C}$ [illegible] $a(w) - zb(w)$ [illegible] $(k, 0, 0)$

A generalization of the stability concept to BVMs can be found in [1, 4].

DEFINITION 5.3. (k_1, k_2) S $z \in \mathbb{C}$ $a(w) - zb(w)$ $(k_1, 0, k_2)$

DEFINITION 5.4. S $z \in \mathbb{C}$ $D + B(z^{-1}I_s - A)^{-1}C$ $(s, 0, 0)$ [6, 9]

DEFINITION 5.5. S $z \in \mathbb{C}$ $-(A - zB)^{-1}(A_0 - zB_0)$ $(s, 0, 0)$ [4, 12]

An LMF, GLM, or BBVM is called A-stable if the left half-plane is contained in its stability domain. A BVM is called A_{k_1,k_2}-stable if the left half-plane is contained in its (k_1, k_2)-stability domain. These are desirable, if not essential, properties when solving stiff systems of ODEs. An A-stable LMF cannot have order higher than 2. Many families of IRK methods exist that are A-stable for any order. Generalized Adams methods (GAMs) and generalized backward difference formulae (GBDFe) are A_{k_1,k_2}-stable BVMs. When used as BBVMs, GAMs are A-stable up to order 8 and GBDFe up to order 4. By adding a few points near the edges of the equidistant grid in a time step, A-stable methods of any order can be constructed.

In the convergence analysis of LMFe we encounter the set

$$\Sigma = \left\{ z \in \mathbb{C} : z = \frac{a}{b}(w), |w| \geq 1 \right\}. \tag{5.3}$$

In the BVM case we have

$$\Sigma = \{ z \in \mathbb{C} : a(w) - zb(w) \text{ is not of type } (k_1, 0, k_2) \}. \tag{5.4}$$

It is clear that in each case Σ is the complement in $\mathbb{C}$ of the open stability domain S, i.e.,

$$\Sigma = \mathbb{C} \setminus S. \tag{5.5}$$

We now introduce the sets Σ that we encounter in the convergence analysis of GLMs and BBVMs and show that (5.5) still holds.

LEMMA 5.6. S 5.4 Σ

$$\Sigma = \{ z \in \mathbb{C} : z^{-1} \in \sigma(A + C(wI_r - D)^{-1}B), |w| \geq 1 \}; \tag{5.6}$$

(5.5)

. We first prove the following lemma.

LEMMA 5.7. $A \in \mathbb{C}^{s \times s}$ $B \in \mathbb{C}^{r \times s}$ $C \in \mathbb{C}^{s \times r}$ $D \in \mathbb{C}^{r \times r}$ $p, q \in \mathbb{C}$

1. $q \in \sigma(A + C(pI_r - D)^{-1}B)$,
2. $p \in \sigma(D + B(qI_s - A)^{-1}C)$,

$(p \notin \sigma(D),\ q \notin \sigma(A))$

. Because of symmetry, it is enough to prove that the first statement implies the second. Using the definition of an eigenvalue, the first statement results in the following sequence of implications:

$$\begin{aligned}
&\exists x \neq 0 : C(pI_r - D)^{-1}Bx + Ax = qx && \wedge \quad Bx \neq 0, \\
&\exists x \neq 0 : C(pI_r - D)^{-1}Bx = (qI_s - A)x && \wedge \quad Bx \neq 0, \\
&\exists x \neq 0 : (qI_s - A)^{-1}C(pI_r - D)^{-1}Bx = x && \wedge \quad Bx \neq 0,
\end{aligned}$$

$$\begin{aligned}
&\exists x \neq 0 : B(qI_s - A)^{-1}C(pI_r - D)^{-1}Bx = Bx \quad \wedge \quad Bx \neq 0,\\
&\exists y \neq 0 : B(qI_s - A)^{-1}Cy = (pI_r - D)y \quad \wedge \quad Cy \neq 0,\\
&\exists y \neq 0 : B(qI_s - A)^{-1}Cy + Dy = py \quad \wedge \quad Cy \neq 0.
\end{aligned}$$

By the definition of an eigenvalue, this yields the second statement. □

Using Lemma 5.7 we see that $z \in \Sigma$ if and only if there exists a w such that

$$w \in \sigma(D + B(z^{-1}I_s - A)^{-1}C) \quad \text{and} \quad |w| \geq 1$$

or, equivalently, $z \notin S$. □

5 8 Lemma 5.7 can also be proved by applying the identity

$$\begin{aligned}
\det\left(\begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}\right) &= \det(M_{11})\det(M_{22} - M_{21}M_{11}^{-1}M_{12})\\
&= \det(M_{11} - M_{12}M_{22}^{-1}M_{21})\det(M_{22})
\end{aligned}$$

to the matrix

$$\begin{bmatrix} qI_s - A & -C \\ -B & pI_r - D \end{bmatrix}.$$

LEMMA 5.9. S

5.5 Σ

$$\Sigma = \{z \in \mathbb{C} : z \in \sigma\big((B + w^{-1}B_0)^{-1}(A + w^{-1}A_0)\big); |w| \geq 1\}; \tag{5.7}$$

(5.5)

. We first prove the following lemma.

LEMMA 5.10. $A, B, A_0, B_0 \in \mathbb{C}^{s\times s}$ $p, q \in \mathbb{C}$

1. $q \in \sigma((B_0 + pB)^{-1}(A_0 + pA))$
2. $p \in \sigma(-(A - qB)^{-1}(A_0 - qB_0))$

. Because of symmetry, it is enough to prove that the first statement implies the second. Using the definition of an eigenvalue, the first statement results in the following sequence of implications:

$$\begin{aligned}
&\exists x \neq 0 : (B_0 + pB)^{-1}(A_0 + pA)x = qx,\\
&\exists x \neq 0 : (A_0 + pA)x = q(B_0 + pB)x,\\
&\exists x \neq 0 : (A_0 - qB_0)x = -p(A - qB)x,\\
&\exists x \neq 0 : -(A - qB)^{-1}(A_0 - qB_0)x = px.
\end{aligned}$$

By the definition of an eigenvalue, this yields the second statement. □

Using Lemma 5.10 we see that $z \in \Sigma$ if and only if there exists a w such that

$$w \in \sigma\big(-(A - zB)^{-1}(A_0 - zB_0)\big) \quad \text{and} \quad |w| \geq 1$$

or, equivalently, $z \notin S$. □

5 11 Taking $w = e^{\epsilon + i\theta}$ with $\epsilon > 0$ in the definition of Σ provides a convenient way to calculate points outside the stability domain. If Σ is used in a formula of the form (4.6) and if all poles of $K(z)$ lie in S, then the maximum (supremum) can be taken over the curve defined by $\epsilon = 0$ instead of over the whole set Σ.

5.2. Convergence analysis of discrete waveform relaxation. We first state without proof two lemmas regarding the convergence of convolution operators. In what follows $\mathcal{H}_{\Delta t}$ is a discrete linear convolution operator with a matrix-valued kernel $h_{\Delta t}$, i.e.,

$$(\mathcal{H}_{\Delta t}e)_i = \sum_{j=0}^{i} h_{i-j}e_j,$$

where $e_i \in \mathbb{C}^d$ and $h_i \in \mathbb{C}^{d\times d}$.

The first lemma relates the convergence factor of a discrete convolution operator acting on finite sequences to the convergence factor of the first term of the discrete Laplace transform of its kernel (see [15, 22]).

LEMMA 5.12. [illegible] $\mathcal{H}_{\Delta t}$ [illegible] $l_p(n+1;\mathbb{C}^d)$ [illegible] $1 \le p \le \infty$ [illegible] n [illegible] $\mathcal{H}_{\Delta t}$ [illegible]

$$\rho(\mathcal{H}_{\Delta t}) = \rho(h_0) = \rho(H_{\Delta t}(\infty)), \tag{5.8}$$

[illegible] $H_{\Delta t}(w) = \sum_{i=0}^{n} h_i w^{-i}$ [illegible] $h_{\Delta t}$

For discrete convolution operators acting on infinite sequences we get the following lemma (see [22, 15]).

LEMMA 5.13. [illegible] $\mathcal{H}_{\Delta t}$ [illegible] $l_p(\infty;\mathbb{C}^d)$ [illegible] $1 \le p \le \infty$ [illegible] $h_{\Delta t} \in l_1(\infty;\mathbb{C}^{d\times d})$ [illegible] $\mathcal{H}_{\Delta t}$ [illegible]

$$\rho(\mathcal{H}_{\Delta t}) = \sup_{|w|\ge 1} \rho(H_{\Delta t}(w)) = \sup_{|w|=1} \rho(H_{\Delta t}(w)), \tag{5.9}$$

[illegible] $H_{\Delta t}(w) = \sum_{i=0}^{\infty} h_i w^{-i}$ [illegible] $h_{\Delta t}$

We now use Lemma 5.13 to derive convergence factors for the discrete waveform relaxation method on infinite time intervals using the LMF, BVM, GLM, and BBVM time discretization schemes. By taking $|w| \to \infty$, we can then see that the results for finite time intervals are the same as for time stepping.

To apply Lemma 5.13 we will have to prove the boundedness of the convolution kernels involved. To this end we will use a matrix version of Wiener's inversion theorem.

THEOREM 5.14 (Wiener's inversion theorem). [illegible] $A_{\Delta t}$ [illegible] $A_{\Delta t} \in l_1(\infty;\mathbb{C}^{d\times d})$ [illegible]

$$\det \sum_{i=0}^{\infty} A_i w^{-i} \neq 0$$

[illegible] $|w| \ge 1$ [illegible] $\sum_{i=0}^{\infty} B_i w^{-i} = (\sum_{i=0}^{\infty} A_i w^{-i})^{-1}$ [illegible] $B_{\Delta t} \in l_1(\infty;\mathbb{C}^{d\times d})$

For a proof we refer to [18] or [22].

5.2.1. LMF. From (3.1) we find the error iteration

$$\sum_{j=-k}^{0} \alpha_{k+j} e_{i+j}^{(\nu)} = \Delta t \sum_{j=-k}^{0} \beta_{k+j} L^{+} e_{i+j}^{(\nu)} + \Delta t \sum_{j=-k}^{0} \beta_{k+j} L^{-} e_{i+j}^{(\nu-1)}.$$

By taking the discrete Laplace transform of these equations and assuming that $e_0 = \cdots = e_{k-1} = 0$, we get

$$\big(a(w)I_m - \Delta t b(w) L^{+}\big) e^{(\nu)}(w) = \Delta t b(w) L^{-} e^{(\nu-1)}(w),$$

where $e^{(\nu)}(w) = \sum_{i=k}^{\infty} e_i^{(\nu)} w^{-i}$ and a and b are the stability polynomials as defined by (5.2). The discrete Laplace transform of the operator $\mathcal{K}_{\Delta t}$, described by (3.1), is therefore

$$\begin{aligned} K_{\Delta t}(w) &= \left(a(w)I_m - \Delta t b(w) L^+\right)^{-1} \Delta t b(w) L^- \\ &= K\left(\frac{1}{\Delta t}\frac{a}{b}(w)\right). \end{aligned} \tag{5.10}$$

The following lemma is proved in [15, 22]. We omit the proof since it is very similar to the ones given below for GLMs and BBVMs. The open stability domain S is the one defined by Definition 5.2.

LEMMA 5.15. $\sigma(\Delta t L^+) \subset S$ $\mathcal{K}_{\Delta t}$ $l_p(\infty; \mathbb{C}^m)$

Combining Lemmas 5.13 and 5.15 with (5.10) and (5.3) results in the following theorem.

THEOREM 5.16. $\mathcal{K}_{\Delta t}$ $l_p(\infty; \mathbb{C}^m)$ $1 \le p \le \infty$ $\sigma(\Delta t L^+) \subset S$

$$\rho(\mathcal{K}_{\Delta t}) = \sup_{z \in \frac{1}{\Delta t}\Sigma} \rho(K(z)) = \sup_{z \in \frac{1}{\Delta t}\partial\Sigma} \rho(K(z)). \tag{5.11}$$

We can take $|w| \to \infty$ and retrieve the results for time stepping and discrete waveform relaxation on finite intervals.

5.2.2. BVM. The convergence analysis for BVM waveform relaxation on infinite sequences can formally proceed as in the finite case (see (4.5)). As in the LMF case, we assume $e_0, \ldots, e_{k_1-1} = 0$. The error equation becomes

$$(A \otimes I_m) e^{(\nu)} = \Delta t (B \otimes L^+) e^{(\nu)} + \Delta t (B \otimes L^-) e^{(\nu-1)}, \tag{5.12}$$

where A and B are no longer matrices but infinite banded Toeplitz operators of the form

$$A = \begin{bmatrix} \alpha_{k_1} & \cdots & \alpha_k & & \\ \vdots & \ddots & & \ddots & \\ \alpha_0 & & & & \\ & \ddots & & & \end{bmatrix}, \qquad B = \begin{bmatrix} \beta_{k_1} & \cdots & \beta_k & & \\ \vdots & \ddots & & \ddots & \\ \beta_0 & & & & \\ & \ddots & & & \end{bmatrix}. \tag{5.13}$$

A relation of the form (4.6) can again be derived where the set Σ is now the spectrum of the operator $B^{-1}A$. It can be seen that this corresponds to Σ as defined in 5.4 (for more information we refer to [4]). The BVM case, of course, reduces to the LMF case for $k_2 = 0$.

5.2.3. IRK and GLM. From (3.3) and (3.4) we derive that the error iteration for the discrete waveform relaxation method using a GLM can be described by

$$\tilde{e}_i^{(\nu)} = (C \otimes I_m) e_{i-1}^{(\nu)} + \Delta t (A \otimes L^+) \tilde{e}_i^{(\nu)} + \Delta t (A \otimes L^-) \tilde{e}_i^{(\nu-1)}, \tag{5.14}$$

$$e_i^{(\nu)} = (D \otimes I_m) e_{i-1}^{(\nu)} + \Delta t (B \otimes L^+) \tilde{e}_i^{(\nu)} + \Delta t (B \otimes L^-) \tilde{e}_i^{(\nu-1)}. \tag{5.15}$$

Applying the discrete Laplace transform to these equations results in

$$P(w) \hat{e}^{(\nu)}(w) = Q(w) \hat{e}^{(\nu-1)}(w), \tag{5.16}$$

with

$$\hat{e}^{(\nu)}(w) = \begin{bmatrix} \tilde{e}^{(\nu)}(w) & e^{(\nu)}(w) \end{bmatrix}^T,$$
$$P(w) = \begin{bmatrix} I_s \otimes I_m - \Delta t A \otimes L^+ & -w^{-1}C \otimes I_m \\ -\Delta t B \otimes L^+ & I_r \otimes I_m - w^{-1}D \otimes I_m \end{bmatrix},$$
$$Q(w) = \begin{bmatrix} \Delta t A \otimes L^- & 0 \\ \Delta t B \otimes L^- & 0 \end{bmatrix}.$$

By eliminating $e^{(\nu)}(w)$ from (5.16) we get the following iteration for $\tilde{e}^{(\nu)}(w)$:

$$\begin{aligned} \big[I_s \otimes I_m - (\Delta t A + w^{-1}C(I_r - w^{-1}D)^{-1}\Delta t B) \otimes L^+\big]\tilde{e}^{(\nu)}(w) \\ = \big[(\Delta t A + w^{-1}C(I_r - w^{-1}D)^{-1}\Delta t B) \otimes L^-\big]\tilde{e}^{(\nu-1)}(w). \end{aligned}$$

If we denote by $\mathcal{K}_{\Delta t}$ the operator mapping $\tilde{e}^{(\nu-1)}$ to $\tilde{e}^{(\nu)}$, then its discrete Laplace transform is

$$K_{\Delta t}(w) = K\left(\frac{1}{\Delta t}\big(A + w^{-1}C(I_r - w^{-1}D)^{-1}B\big)^{-1}\right). \tag{5.17}$$

The following lemma provides a sufficient condition for $\mathcal{K}_{\Delta t}$ to be bounded. The open stability domain S is the one defined by Definition 5.4.

LEMMA 5.17. [illegible] $\sigma(\Delta t L^+) \subset S$ [illegible] $\mathcal{K}_{\Delta t}$ [illegible] $l_p(\infty; \mathbb{C}^{sm})$ [illegible]. It is sufficient to prove that $P(w)^{-1}Q(w)$ is the discrete Laplace transform of an l_1-sequence. It is clear that both $P(w)$ and $Q(w)$ are discrete Laplace transforms of l_1-sequences. By Wiener's theorem we have that $P(w)^{-1}$ is the Laplace transform of an l_1-sequence if

$$\det(P(w)) \neq 0 \quad \text{for } |w| \geq 1. \tag{5.18}$$

If this is true, then $P(w)^{-1}Q(w)$ is the discrete Laplace transform of the convolution of two l_1-sequences and therefore the operator corresponding to (5.14) and (5.15) is bounded. Condition (5.18) can hold only if the condition

$$\det\left(\begin{bmatrix} I_s - zA & -w^{-1}C \\ -zB & I_r - w^{-1}D \end{bmatrix}\right) \neq 0 \tag{5.19}$$

holds for all $z \in \sigma(\Delta t L^+)$ and all $|w| \geq 1$. By considering the LU factorization of the matrix, we see that condition (5.19) is equivalent to the conditions

$$\det(I_s - zA) \neq 0, \tag{5.20}$$
$$\det(I_r - w^{-1}D - zB(I_s - zA)^{-1}w^{-1}C) \neq 0. \tag{5.21}$$

For z to be in S, the matrix $D + B(z^{-1}I_s - A)^{-1}C$ has to be well defined and therefore condition (5.20) is satisfied. Suppose equality holds for some $z \in \sigma(\Delta t L^+)$ in (5.21). This would mean that

$$w \in \sigma(D + zB(I_s - zA)^{-1}C)$$

with $|w| \geq 1$ and therefore $z \notin S$. This contradicts the assumption of the lemma. Hence, (5.21) is satisfied. ∎

Combining Lemmas 5.13 and 5.17 with equations (5.17) and (5.6) results in the following theorem.

THEOREM 5.18. [illegible] $\mathcal{K}_{\Delta t}$ [illegible] $l_p(\infty; \mathbb{C}^{sm})$ [illegible] $1 \leq p \leq \infty$ [illegible] $\sigma(\Delta t L^+) \subset S$ [illegible]

$$\rho(\mathcal{K}_{\Delta t}) = \sup_{z \in \frac{1}{\Delta t}\Sigma} \rho(K(z)) = \sup_{z \in \frac{1}{\Delta t}\partial\Sigma} \rho(K(z)). \tag{5.22}$$

5.2.4. BBVM. Using (3.5) and $e_i^{(\nu)} = u_i^{(\nu)} - u_i \in \mathbb{C}^{sm}$, we find the error iteration

$$\begin{aligned}(A\otimes I_m)e_i^{(\nu)} + (A_0 \otimes I_m)e_{i-1}^{(\nu)} \\ = \Delta t(B \otimes L^+)e_i^{(\nu)} + \Delta t(B \otimes L^-)e_i^{(\nu-1)} \\ + \Delta t(B_0 \otimes L^+)e_{i-1}^{(\nu)} + \Delta t(B_0 \otimes L^-)e_{i-1}^{(\nu-1)}.\end{aligned}$$

Applying the discrete Laplace transform to these equations results in

$$\begin{aligned}\left[(A + w^{-1}A_0) \otimes I_m\right] e^{(\nu)}(w) = \left[\Delta t(B + w^{-1}B_0) \otimes L^+\right] e^{(\nu)}(w) \\ + \left[\Delta t(B + w^{-1}B_0) \otimes L^-\right] e^{(\nu-1)}(w).\end{aligned} \tag{5.23}$$

The discrete Laplace transform of the operator $\mathcal{K}_{\Delta t}$ described by (3.5) is therefore

$$K_{\Delta t}(w) = K\left(\frac{1}{\Delta t}(B + w^{-1}B_0)^{-1}(A + w^{-1}A_0)\right).$$

In order to apply Lemma 5.13, the operator $\mathcal{K}_{\Delta t}$ must be bounded. A sufficient condition is provided by the following lemma. The open stability domain S is the one defined by Definition 5.5.

LEMMA 5.19. [illegible] $\sigma(\Delta t L^+) \subset S$ [illegible] $\mathcal{K}_{\Delta t}$ [illegible] $l_p(\infty;\mathbb{C}^{sm})$ [illegible]. It is sufficient to prove that $K_{\Delta t}(w)$ is the discrete Laplace transform of an l_1-sequence. Following the same reasoning as for Lemma 5.17, we arrive at the condition

$$\det\left((A + w^{-1}A_0) \otimes I_m - \Delta t(B + w^{-1}B_0) \otimes L^+\right) \neq 0 \tag{5.24}$$

for $|w| \geq 1$. This condition can hold only if

$$\det\left((A + w^{-1}A_0) - z(B + w^{-1}B_0)\right) \neq 0 \tag{5.25}$$

for $z \in \sigma(\Delta t L^+)$. If we assume that equality holds, then using Lemma 5.10 we get

$$w \in \sigma(-(A - zB)^{-1}(A_0 - zB_0)).$$

Because $|w| \geq 1$ this would mean that $z \notin S$, which contradicts the assumption of the lemma and therefore (5.25) is true. □

Combining Lemmas 5.13 and 5.19 with equations (5.23) and (5.7) results in the following theorem.

THEOREM 5.20. [illegible] $\mathcal{K}_{\Delta t}$ [illegible] $l_p(\infty;\mathbb{C}^{sm})$ [illegible] $1 \leq p \leq \infty$ [illegible] $\sigma(\Delta t L^+) \subset S$ [illegible]

$$\rho(\mathcal{K}_{\Delta t}) = \sup_{z\in\frac{1}{\Delta t}\Sigma} \rho(K(z)) = \sup_{z\in\frac{1}{\Delta t}\partial\Sigma} \rho(K(z)). \tag{5.26}$$

5.3. Multigrid convergence analysis. As in the time stepping case the results for the two-grid iteration can be found by substituting $M(z)$ for $K(z)$. The condition on the set Σ becomes $(\sigma(\Delta t L^+) \cup \sigma(\bar{L})) \subset S$. For proofs we refer to [14, 15, 19, 28].

TABLE 5.1
The sets Σ to be maximized over in (4.6).

LMF	GLM	BVM	BBVM
$\frac{\alpha_k}{\beta_k}$	$\sigma(A^{-1})$	$\sigma(B^{-1}A)$	$\sigma(B^{-1}A)$
$\frac{a(w)}{b(w)}$	$\sigma\left((A+C(wI_r-D)^{-1}B)^{-1}\right)$	$\frac{a(w)}{b(w)}$	$\sigma\left((B_0+wB)^{-1}(A_0+wA)\right)$

5.4. Relation to a continuous in time method. The asymptotic convergence factor for the continuous multigrid waveform relaxation method on infinite intervals using (3.7) as smoother is given by

$$\rho = \sup_{z\in\mathbb{C}^+} \rho(M(z)) = \sup_{z\in i\mathbb{R}} \rho(M(z)),$$

assuming that $(\sigma(L^+) \cup \sigma(\bar{L})) \subset \mathbb{C}^-$ [14, 19, 28]. Therefore, when $\Sigma \subset \mathbb{C}^+$, the convergence factor of discrete waveform relaxation is bounded by that of continuous waveform relaxation. This is the case for LMFe, GLMs, and BBVMs that are A-stable and for A_{k_1,k_2}-stable BVMs.

5.5. Summary. The asymptotic convergence factor of the multigrid methods for time-dependent problems described above can be found from

$$\rho = \sup_{z\in\frac{1}{\Delta t}\partial\Sigma} \rho(M(z)). \tag{5.27}$$

The appropriate sets Σ for different time discretization schemes are listed in Table 5.1. The first row refers to time stepping and discrete waveform relation on finite intervals; the second row refers to discrete waveform relation on infinite intervals. Note that except for the BVM case the first row can be derived by taking $|w| \to \infty$. The function $\rho(M(z))$ can be estimated by a local mode Fourier analysis of the two-grid iteration applied to the elliptic equation

$$zu = \mathcal{L}u + f.$$

6. Numerical results. The time discretization schemes used for the numerical experiments are implicit Euler (=BDF1=GBDF1=RadauIIA1), the backward difference formula with 5 steps (BDF5), the Radau IIA implicit Runge–Kutta method with 3 stages (RAD3), the generalized Adams methods with 4 steps (GAM4), and the generalized backward difference formula with 5 steps (GBDF5). All but the implicit Euler method are fifth order methods. The GAM4 and GBDF5 boundary value methods were also used as block BVMs (BGAM4, BGBDF5). The coefficients for the discretization schemes used can be found in Appendix A.

We prefer to report convergence rates instead of convergence factors. The convergence rate R is defined by

$$R = -\log_{10}(\rho)$$

and can be interpreted as the average number of additional correct digits obtained per iteration. The asymptotic convergence rate can be estimated using

$$\rho_n^m = \frac{\|e_n\|_\infty}{\|e_{n-m}\|_\infty}, \qquad R_n^m = -\frac{\log_{10}(\rho_n^m)}{m}, \tag{6.1}$$

where e_i is the error after the ith iteration. We used the average of R_i^1 for $i = 10, \ldots, 20$ for all numerical experiments reported here. This corresponds to taking the geometric mean of ρ_i^1 or taking $n = 20$, $m = 10$ in (6.1).

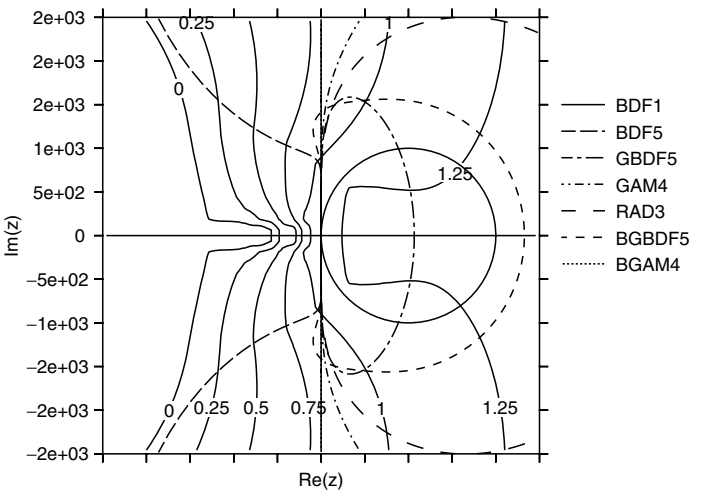


FIG. 6.1. *Contour lines of the convergence rate* $R(z) = -\log_{10}\rho(M(z))$ *and boundaries of the sets* Σ *for discrete waveform relaxation on infinite time domains. The boundary for the BGAM4 discretization coincides with the imaginary axis.*

TABLE 6.1
Convergence rates obtained by a two-grid local Fourier analysis (heat equation).

	BDF1	BDF5	GBDF5	GAM4	RAD3	BGBDF5	BGAM4
TS	1.29	1.32	-	-	1.16	0.92	1.08
WR	1.13	0.06	0.90	0.86	0.89	0.85	0.79

6.1. Heat equation. As a first example we consider the heat equation

$$u_t = u_{xx} + u_{yy} + f \tag{6.2}$$

on the unit square. The problem is discretized on a grid with $\Delta x = \Delta y = 2^{-5}$. The two-grid analysis was conducted for full weighting restriction, bilinear interpolation, and 1 pre- and 1 postsmoothing red-black Gauss–Seidel step. Time integration is over $\Omega_t = [0, 1]$; i.e., $\Delta t = 3 \cdot 10^{-3}$ for the RAD3 discretization and $\Delta t = 10^{-3}$ for all others. Figure 6.1 shows contour lines of the predicted convergence rate $R(z)$ together with the boundaries of the regions Σ of the discretization schemes used in waveform relaxation on infinite time domains. We expect convergence for the BDF5 discretization to be slow since the boundary of the corresponding set Σ nearly touches the line $R = 0$. If these lines were to touch or intersect, convergence would no longer be guaranteed. Note that all but the BDF5 and BGBDF5 method are A-stable. The BGBDF5 method is, however, very close to being A-stable and we expect very good convergence. The results are presented in Table 6.1. The first row contains the results for time stepping or discrete waveform relaxation on a finite domain. These results are obtained by maximizing $R(z)$ over Σ, which, in this case, is a discrete set of points. The BVMs (GAM4 and GBDF5) cannot be used in a time stepping sense. The second row of Table 6.1 contains the results for discrete waveform relaxation on infinite time domains. The results were obtained by maximizing over a number of points on the curves depicted in Figure 6.1. The points were found by taking $w = \exp(\frac{2\pi ik}{n})$, $k = 0, \ldots, n-1$, $n = 200$ in the definitions of the sets Σ. Since $\Sigma = \mathbb{C}^+$ for the

TABLE 6.2
Number of time steps and number of unknowns per time step for the time discretizations used in the numerical experiments.

	BDF1	BDF5	GBDF5	GAM4	RAD3	BGBDF5	BGAM4
TS	1,1	1,5	-	-	1,3	1,5	1,4
WR	1020,1	1020,1	1020,1	1020,1	360,3	204,5	255,4

TABLE 6.3
Convergence rates obtained from numerical experiments (heat equation).

	BDF1	BDF5	GBDF5	GAM4	RAD3	BGBDF5	BGAM4
TS	1.18	1.11	-	-	1.33	1.11	1.10
WR	0.92	0.03	0.87	0.87	0.87	0.87	0.88

BGAM4 discretization, the last result in the table provides a good approximation to the convergence rate for continuous waveform relaxation. From these results we expect good convergence for all methods except waveform relaxation with BDF5 time discretization.

Decreasing Δt corresponds to a proportional increase in the size of the sets Σ in Figure 6.1. Similarly, when Δx and Δy are decreased simultaneously, the contour lines of $\rho(M(z))$ have approximately the same shape but on a larger scale (the dependency is quadratic in this case). This can be explained by observing that scaling L, L^+, and L^- in the formulae for $K(z)$ and $M(z)$ is equivalent to the inverse scaling of z. Decreasing Δx and Δy will, of course, also increase the size of all the matrices involved, but this can be neglected for fine discretizations (large matrices). Figure 6.1 can thus be used to estimate the convergence for other discretization resolutions by changing the relative size of the curves depicted.

For the numerical experiments the initial conditions, the Dirichlet boundary conditions, and the source term f are chosen such that the exact solution is $u = 0$. We use a standard W-cycle geometric multigrid algorithm with 5 levels, full weighting restriction, bilinear interpolation and 1 pre- and 1 postsmoothing red-black Gauss–Seidel step. The spatial domain is discretized using finite differences. The grid sizes are $\Delta x = \Delta y = 2^{-1}$ on the coarsest grid and $\Delta x = \Delta y = 2^{-5}$ on the finest grid. As in the analysis the time steps are $\Delta t = 3 \cdot 10^{-3}$ for RAD3 and $\Delta t = 10^{-3}$ otherwise. The number of time steps for each method was chosen such that the number of unknowns per spatial grid point was minimal in the time stepping case and the same for all methods in the waveform relaxation case. Table 6.2 shows the number of time steps and the number of unknowns per time step for each method. All unknowns were set to 1 for the initial approximation.

The results are presented in Table 6.3. It is clear that they correspond very well to the results obtained from the theoretical analysis. The theoretical convergence rates are not strictly lower bounds (upper bounds on the convergence factors), since they are asymptotic convergence rates obtained using a local Fourier mode two-grid approximation. Similar results with poor convergence for the BDF5 method and convergence rates close to 1 for the other methods were obtained for waveform relaxation on grids of dimensions $2^5 \times 2^5 \times 10^4$, $2^8 \times 2^8 \times 360$, and $2^9 \times 2^9 \times 60$. We can conclude that, for this problem, multigrid methods are very efficient even when using high order time discretization schemes.

TABLE 6.4
Convergence rates for the MGS iteration (diffusion equation with variable coefficients).

	BDF1	BDF5	GBDF5	GAM4	RAD3	BGBDF5	BGAM4
TS	1.50	1.34	-	-	1.91	1.34	1.47
WR	1.22	0.69	1.22	1.22	1.22	1.22	1.22

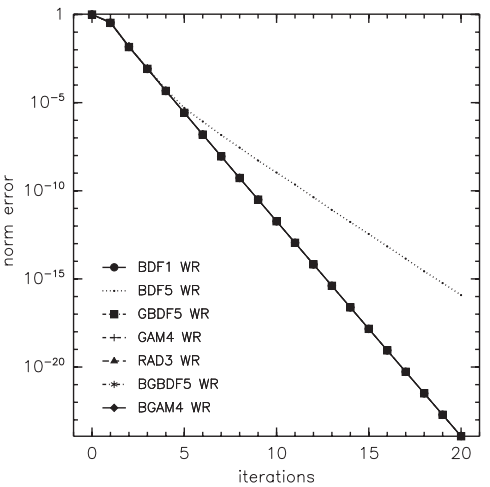


FIG. 6.2. *Convergence of MGS method (diffusion equation with variable coefficients).*

6.2. Diffusion equation with variable coefficients. The next example is a more general diffusion equation of the form

$$u_t = (au_x)_x + (bu_y)_y + f,$$
$$a(x, y, t) = \exp(10(x - y)),$$
$$b(x, y, t) = \exp(-10(x - y)).$$

This is a so-called anisotropic problem since the diffusion coefficients in a certain point can be very different depending on the direction of diffusion. Standard multigrid methods do not handle this type of problem well and yield very slow convergence (e.g., $R = 0.1$). However, just as we did with the original multigrid algorithm, we can extend multigrid methods developed for time-independent anisotropic problems to time-dependent ones. Here we used the "multigrid as smoother" (MGS) method which uses the same simple smoothers as before, together with an extended hierarchy of coarse grids. For more information on multigrid for anisotropic problems, we refer to [23, 24, 26, 27, 29].

The same discretizations, initial and boundary conditions, and initial approximation were used as for the previous example. The multigrid algorithm uses full weighting restriction, bilinear and linear interpolation, and 1 pre- and 1 postsmoothing step. The smoothing consists of a multigrid semicoarsening step in the x-direction followed by one in the y-direction, both using only 1 presmoothing red-black Gauss–Seidel step. A hierarchy of 5×5 grids is used. The results are presented in Table 6.4. The convergence history of the MGS waveform relaxation method is visualized in Figure 6.2 for different time discretization schemes. Except for the BDF5 method,

the convergence histories are indistinguishable. We see once more that it is possible to obtain very efficient multigrid methods even when high order time discretization schemes are used.

7. Concluding remarks. We have shown by theoretical analysis and numerical experiments that it is possible to develop efficient multigrid methods for high order time discretizations of parabolic equations. The analysis shows that the stability of the time discretization scheme is important for the convergence of multigrid methods. Among schemes with good stability properties (such as A-stability), the convergence is comparable. Depending on the problem at hand, a suitable scheme can be chosen taking into account other properties such as implementation cost, efficiency, and availability of error estimators. Since the cost of a multigrid iteration in a time stepping or block time stepping scheme is proportional to the cost of solving a sequence of scalar ODEs over a time step or over a block of time steps, it suffices to consider the properties of the time discretization scheme when applied to scalar equations. Furthermore, the convergence for parabolic problems is roughly equivalent to the convergence of the corresponding multigrid methods for elliptic equations. This means that both theoretical analysis and experience with elliptic equations can be used as guidelines to determine whether it is worthwhile to invest in the implementation of a multigrid method. Experience with the elliptic case also gives an indication of how the multigrid method will compare to other iterative methods such as GMRES or BiCGstab. It is also possible to use multigrid as a preconditioner for these methods [34]. Considering the success of multigrid methods for stationary equations it is clear that multigrid methods for time-dependent equations show great promise.

Appendix A. Coefficient matrices.

A.1. Implicit Runge–Kutta methods. We considered only the Radau IIA method with 3 stages.

RAD3:

$$A = \begin{bmatrix} \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{bmatrix}, \quad b = \begin{bmatrix} \frac{16-\sqrt{6}}{36} \\ \frac{16+\sqrt{6}}{36} \\ \frac{1}{9} \end{bmatrix}, \quad c = \begin{bmatrix} \frac{4-\sqrt{6}}{10} \\ \frac{4+\sqrt{6}}{10} \\ 1 \end{bmatrix}.$$

A.2. Boundary value methods. The coefficient matrices A_e and B_e for the LMFe and BBVMs we used are given below. The boldface rows correspond to the coefficients α_j and β_j of the basic method (2.8). The rows above correspond to the initial conditions and the rows below to the final conditions. The coefficient matrices for $s > k$ are obtained by repeating the boldface rows as in (2.11).

BDF1:

$$A_e = \left[\begin{array}{c|c} \mathbf{-1} & \mathbf{1} \end{array}\right], \quad B_e = \left[\begin{array}{c|c} \mathbf{0} & \mathbf{1} \end{array}\right].$$

GAM4:

$$A_e = \left[\begin{array}{c|ccc} -1 & 1 & & \\ \mathbf{-1} & \mathbf{1} & & \\ & -1 & 1 & \\ & & -1 & 1 \end{array}\right], \quad B_e = \frac{1}{720}\left[\begin{array}{c|cccc} 251 & 646 & -264 & 106 & -19 \\ \mathbf{-19} & \mathbf{346} & \mathbf{456} & \mathbf{-74} & \mathbf{11} \\ 11 & -74 & 456 & 346 & -19 \\ -19 & 106 & -264 & 646 & 251 \end{array}\right].$$

GBDF5:

$$A_e = \frac{1}{60}\left[\begin{array}{c|ccccc} -12 & -65 & 120 & -60 & 20 & -3 \\ 3 & -30 & -20 & 60 & -15 & 2 \\ \mathbf{-2} & \mathbf{15} & \mathbf{-60} & \mathbf{20} & \mathbf{30} & \mathbf{-3} \\ 3 & -20 & 60 & -120 & 65 & 12 \\ -12 & 75 & -200 & 300 & -300 & 137 \end{array}\right], \quad B_e = \left[\begin{array}{c|ccccc} 0 & 1 & & & & \\ 0 & & 1 & & & \\ \mathbf{0} & & & \mathbf{1} & & \\ 0 & & & & 1 & \\ 0 & & & & & 1 \end{array}\right].$$

BDF5. The last rows of the previous matrices contain the coefficients of the BDF5 method.

RAD3. The Radau IIA methods can be easily formulated as a BBVM. It is important to note that the points within a time step are not equidistant as in the previous BVMs and that the time step Δt has a different interpretation:

$$A_e = \left[\begin{array}{c|ccc} -1 & 1 & & \\ -1 & & 1 & \\ -1 & & & 1 \end{array}\right], \quad B_e = \left[\begin{array}{c|ccc} 0 & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ 0 & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ 0 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array}\right].$$

Acknowledgments. Some of the results were verified using a version of the Fourier analysis code `LFA00_2D_scalar` [33] extended to complex numbers in collaboration with its author. We would like to thank the anonymous referees for their helpful comments.

REFERENCES

[1] P. Amodio and F. Mazzia, *A boundary value approach to the numerical solution of initial value problems by multistep methods*, J. Differ. Equations Appl., 1 (1995), pp. 353–367.

[2] D. Bertaccini, *A circulant preconditioner for the systems of LMF-based ODE codes*, SIAM J. Sci. Comput., 22 (2000), pp. 767–786.

[3] D. Bertaccini and M. K. Ng, *Block $\{\omega\}$-circulant preconditioners for the systems of differential equations*, Calcolo, 40 (2003), pp. 71–90.

[4] L. Brugnano and D. Trigiante, *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Stability and Control: Theory, Methods and Applications 6, Gordon and Breach Science Publishers, Amsterdam, 1998.

[5] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations*, Numerical Mathematics and Scientific Computation, Oxford Science Publications, The Clarendon Press, Oxford University Press, New York, 1995.

[6] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, Ltd., Chichester, UK, 2003.

[7] R. H. Chan, M. K. Ng, and X.-Q. Jin, *Strang-type preconditioners for systems of LMF-based ODE codes*, IMA J. Numer. Anal., 21 (2001), pp. 451–462.

[8] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations.* I. *Nonstiff Problems*, Springer Ser. Comput. Math. 8, 2nd ed., Springer-Verlag, Berlin, 1993.

[9] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations.* II. *Stiff and Differential-Algebraic Problems*, Springer Ser. Comput. Math. 14, 2nd ed., Springer-Verlag, Berlin, 1996.

[10] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.

[11] F. Iavernaro and F. Mazzia, *Solving ordinary differential equations by generalized Adams methods: Properties and implementation techniques*, Appl. Numer. Math., 28 (1998), pp. 107–126.

[12] F. Iavernaro and F. Mazzia, *Block-boundary value methods for the solution of ordinary differential equations*, SIAM J. Sci. Comput., 21 (1999), pp. 323–339.

[13] F. Iavernaro and D. Trigiante, *Preconditioning and conditioning of systems arising from boundary value methods*, Nonlinear Dyn. Syst. Theory, 1 (2001), pp. 59–79.

[14] J. Janssen and S. Vandewalle, *Multigrid waveform relaxation on spatial finite element meshes: The continuous-time case*, SIAM J. Numer. Anal., 33 (1996), pp. 456–474.

[15] J. Janssen and S. Vandewalle, *Multigrid waveform relaxation on spatial finite element meshes: The discrete-time case*, SIAM J. Sci. Comput., 17 (1996), pp. 133–155.

[16] J. D. Lambert, *Computational Methods in Ordinary Differential Equations*, John Wiley & Sons, London, New York, Sydney, 1973.

[17] E. Lelarasmee, A. Ruehli, and A. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large-scale integrated circuits*, IEEE CAD for IC Systems, 1 (1982), pp. 131–145.

[18] C. Lubich, *On the stability of linear multistep methods for Volterra convolution equations*, IMA J. Numer. Anal., 3 (1983), pp. 439–465.
[19] C. Lubich and A. Ostermann, *Multigrid dynamic iteration for parabolic equations*, BIT, 27 (1987), pp. 216–234.
[20] A. Lumsdaine and D. Wu, *Spectra and pseudospectra of waveform relaxation operators*, SIAM J. Sci. Comput., 18 (1997), pp. 286–304.
[21] U. Miekkala and O. Nevanlinna, *Convergence of dynamic iteration methods for initial value problems*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 459–482.
[22] U. Miekkala and O. Nevanlinna, *Sets of convergence and stability regions*, BIT, 27 (1987), pp. 554–584.
[23] C. W. Oosterlee, *The convergence of parallel multiblock multigrid methods*, Appl. Numer. Math., 19 (1995), pp. 115–128.
[24] C. W. Oosterlee and P. Wesseling, *On the robustness of a multiple semi-coarsened grid method*, Z. Angew. Math. Mech., 75 (1995), pp. 251–257.
[25] J. H. Saltz and V. K. Naik, *Towards developing robust algorithms for solving partial differential equations on MIMD machines*, Parallel Comput., 6 (1988), pp. 19–44.
[26] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2001.
[27] J. Van lent and S. Vandewalle, *Multigrid waveform relaxation for anisotropic partial differential equations*, Numer. Algorithms, 31 (2002), pp. 361–380.
[28] S. Vandewalle, *Parallel Multigrid Waveform Relaxation for Parabolic Problems*, Teubner Skripten zur Numerik., B. G. Teubner, Stuttgart, Germany, 1993.
[29] T. Washio and C. W. Oosterlee, *Flexible multiple semicoarsening for three-dimensional singularly perturbed problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1646–1666.
[30] P. Wesseling, *An Introduction to Multigrid Methods*, Pure and Applied Mathematics, John Wiley & Sons Ltd., Chichester, UK, 1992.
[31] J. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*, Kluwer Academic Publishers, Boston, 1987.
[32] J. White, A. Sangiovanni-Vincentelli, F. Odeh, and A. Ruehli, *Waveform relaxation: Theory and practice*, Trans. Soc. for Comput. Simulation, 2 (1985), pp. 95–133.
[33] R. Wienands and C. W. Oosterlee, *On three-grid Fourier analysis for multigrid*, SIAM J. Sci. Comput., 23 (2001), pp. 651–671.
[34] R. Wienands, C. W. Oosterlee, and T. Washio, *Fourier analysis of* GMRES(m) *preconditioned by multigrid*, SIAM J. Sci. Comput., 22 (2000), pp. 582–603.
[35] D. E. Womble, *A time-stepping algorithm for parallel computers*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 824–837.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 93–107
© 2005 Society for Industrial and Applied Mathematics

EFFICIENTLY COMPUTING MANY ROOTS OF A FUNCTION*

D. J. KAVVADIAS†, F. S. MAKRI†, AND M. N. VRAHATIS†

Dedicated to the memory of Árpád Elbert (1939–2001)

Abstract. We present a new bisection based method for counting and computing roots of a function in a given interval. Our method is focused on very large problems, i.e., instances with the number of roots of the order of hundreds. The method draws its power from the fact that the roots are expected to be many, and is able to discover a large percentage of them very efficiently. Its main advantage, apart from its efficiency, is the fact that it requires only the sign of the function at a certain point and not its actual value. Also, its simplicity makes it a suitable preprocessing step for reducing the size of the problem, prior to more robust but also more demanding methods. The algorithm is accompanied by a probabilistic analysis of its behavior, which shows that a simple existence criterion like Bolzano's rule can be a powerful tool in the zerofinding process.

Key words. zerofinding, expected behavior, bisection based methods, counting and computing the roots of a function, very large problems, Riemann's hypothesis, zeta-function, special functions, Elbert's conjecture

AMS subject classifications. 65H05, 65Y20

DOI. 10.1137/S1064827502406531

1. Introduction.

In this paper we present a bisection based method for the problem of counting and computing the simple roots of a single equation:

$$f(x) = 0 \tag{1.1}$$

in a given interval (a, b) where $f: [a, b] \subset \mathbb{R} \to \mathbb{R}$ is continuous. The only computable information required by the proposed method is the algebraic signs of the function values. The method is focused on very large instances of the problem (with roots in the order of hundreds or more).

Pieces of information concerning all of the roots (or a large fraction of them), as well as all the extrema of a function $f: [a, b] \subset \mathbb{R} \to \mathbb{R}$, are of major importance in many different fields in science and technology such as mechanics, physical sciences, statistics, and operations research. For instance, the problem of locating local maxima and minima of a function (or zeros of the derivative of a function) from approximate measurement results is vital for many physical applications. In spectral analysis, chemical species are identified by locating local maxima of the spectra. In radioastronomy, sources of celestial radio emission and their subcomponents are identified by locating local maxima of the measured brightness of the radio sky. Elementary particles are identified by locating local maxima of the experimental curves (for a discussion of the importance of the above mentioned problems for applications, see [28]). The importance of the problem has attracted the attention of many research efforts, and as a result many different approaches to the problem exist. Regarding special functions, for example, Lozier [17] and Lozier and Olver [18] have provided a survey of algorithms and software for the numerical evaluation of special functions.

*Received by the editors April 30, 2002; accepted for publication (in revised form) November 4, 2004; published electronically August 17, 2005.
http://www.siam.org/journals/sisc/27-1/40653.html

†Department of Mathematics, University of Patras, GR–26110 Patras, Greece, and University of Patras Artificial Intelligence Research Center, University of Patras, GR–26110 Patras, Greece (djk@math.upatras.gr, makri@math.upatras.gr, vrahatis@math.upatras.gr).

They have pointed out that the available software for special functions (including the zerofinding approach) exhibits gaps and defects with respect to the needs of modern high-performance computing. In particular, the software regarding the zeros of Bessel functions has a low cumulative score as reported in [17, p. 351], and thus the computation of the zeros of special functions is an area of particular need.

Apart from its practical significance, the problem of computing zeros of a function poses theoretical challenges that have attracted the attention of the scientific community for many years. A characteristic paradigm is the famous [illegible], which is one of the main open problems of mathematics (and was included by Hilbert as Problem No. 8 in his famous 1900 list of 23 problems). According to this hypothesis, all complex roots $s_1 = \Re(s_1) + i\,\Im(s_1), s_2 = \Re(s_2) + i\,\Im(s_2), \ldots$, of Riemann's ζ-function (zeta-function)

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s}$$

(i.e., the values for which $\zeta(s_k) = 0$) are located on the straight line $\Re(s_k) = 1/2$ in the complex plane (except for the known zeros which are negative integers). It is known that the imaginary parts of the roots of Riemann's ζ-function are uniformly distributed. Riemann's ζ-function has proved to be of fundamental importance not only in the theory of prime numbers, but also in the higher theory of the Gamma-function and allied functions. Numerical computation of the roots of the ζ-function in a specific interval may gather evidence that hopefully will help in resolving the problem.

In addition, the task of massive calculation of zeros of functions in one variable also emerges naturally in recently posed problems. Such a problem is [illegible]. More specifically, in [3, p. 75] Elbert pointed out that the density property of the zeros of Bessel functions plays an important role, and better insight into the distribution of these zeros is required (see also [11, 12], where Joó dealt with oscillation of circular membranes). Elbert considered the set

$$\mathcal{S} = \bigcup_{n,k=1}^{\infty} \{j_{nk}\} = \bigcup_{j=1}^{\infty} \{x_j\},$$

where j_{nk} is the kth positive zero of the Bessel function of first kind, $J_n(x)$, and $x_1 < x_2 < \cdots$, and he conjectured that [3]

$$\limsup_{j\to\infty} x_j(x_{j+1} - x_j) < \infty.$$

For results on the massive calculation and more information on the values $x_j(x_{j+1} - x_j)$, see [23].

The proposed method takes advantage of the abundance of roots, in an attempt to very inexpensively (in terms of computing resources) locate a large number of roots, thus reducing the size of the problem. The algorithm almost "blindly" searches for roots by employing only Bolzano's existence criterion (see next section). It turns out, however, that even this simple rule is sufficient to guide the algorithm in discovering a large proportion of the total set of roots. Thus with very few sign determinations, the problem can be considerably reduced. But this is already more than could be expected from such a simple method: While new roots are being discovered, the cost

gets increasingly high to a point where it becomes unprofitable to continue. The algorithm stops when a predetermined fraction of the roots has been discovered. This requires a method of estimating the number of roots as the algorithm proceeds. After the termination of the algorithm the unsearched parts of the interval must be searched by a different, more demanding, method. The main advantage of the proposed method is its simplicity, which follows precisely from the expectation that with high probability even a simple search for a root will prove successful due to the abundance of roots.

In this paper we also study analytically the expected behavior of the method and give theoretical justification of its good performance.

We mention that there has been a surge of interest concerning the expected behavior of numerical algorithms [6, 7, 8, 20, 24, 27] and [13, 14]. The traditional approach in evaluating a numerical method usually involves a number of experiments on a number of inputs, either of individual interest or randomly constructed by altering certain parameters of the problem. Very few numerical methods exist that are accompanied by a robust analysis of their expected behavior.

A by-product of the algorithm is an estimation of the number of roots in an interval. This problem is interesting on its own, either to a priori evaluate the size of a problem or, as in this case, to establish a stopping criterion for the method.

Other approaches that have been used, aiming to find all solutions of systems of equations as well as the global optimum of a function, are based on interval analysis (see, e.g., [1, 9, 15, 19]). The corresponding existence tool of these methods is the availability of the range of the function in a given interval, which can be implemented using interval arithmetic, although range overestimation may occur, and hence efficiency problems must be resolved. This tool will, with mathematical rigor, give either a "no" or an "unknown" answer. The former case proceeds by subdividing the interval into two halves and employing additional criteria. The way the evaluation of functions is encoded influences the answer, which is usually pessimistic (i.e., "unknown"). In the vicinity of a root, interval Newton methods (see, e.g., [15, 19]) may, however, with slightly more computational effort, give an unambiguous "yes" answer.

In the next section we give some background material on the bisection method, as was modified in [29, 30]. In section 3 we briefly discuss the main steps of the algorithm to present a (central to our analysis) probabilistic result which follows in section 4. Then in section 5 we give a method for estimating the total number of roots. A more detailed description of the method, which also refers to the theoretical analysis, is presented in section 6. We end in section 7 by presenting some conclusions and future work.

2. Background material. A simple oracle on the existence of a solution of $f(x) = 0$ in some interval (a, b) where the function f is continuous in $[a, b]$ is the following criterion:

$$f(a)\, f(b) < 0 \quad \text{or} \quad \operatorname{sgn} f(a) \operatorname{sgn} f(b) = -1,$$

where sgn is the well-known three-valued sign function. This criterion is known as Bolzano's existence criterion. (For a generalization of this criterion to higher dimensions, see [31].) Note that this oracle may introduce a one-sided error; i.e., a positive response means that at least one root exists, but a negative response may correspond to the existence of an even number of simple roots.

The simplicity of this criterion is what makes it attractive even though it has the above disadvantage. Thus it will be our main tool in what follows. More elaborate relations can give more information on the existence of roots. For example, interval

analysis uses the range of the function to decide whether a root exists (see, e.g., [15, 19]). Another approach is given in [16]. An even more complicated oracle which gives the exact number of roots $\mathcal{N}^r$ is based on topological degree theory using Kronecker's integral on a Picard's extension [10, 21]. This oracle was used in [14] as part of the first phase of an algorithm for the isolation of all of the simple roots of a function $f(x)$ in an interval (a, b) and returns the number of roots $\mathcal{N}^r$ using the formula

$$\mathcal{N}^r = -\frac{\gamma}{\pi}\int_a^b \frac{f(x)f''(x) - f'^2(x)}{f^2(x) + \gamma^2 f'^2(x)}\,dx + \frac{1}{\pi}\arctan\left(\frac{\gamma\,[f(a)f'(b) - f(b)f'(a)]}{f(a)f(b) + \gamma^2 f'(a)f'(b)}\right), \tag{2.1}$$

where γ is an arbitrary small real positive constant, i.e., $\gamma \simeq 1$. (For a variation of the CPU time for the computation of $\mathcal{N}^r$ versus γ, see [36].) It was explicitly shown by Picard [21, 22] that relation (2.1) is independent of the value of γ.

The algorithm of [14] uses bisection in its second phase to compute the roots. Specifically, it uses the following simplified version described in [29]:

$$x_{i+1} = x_i + c\,\mathrm{sgn}f(x_i)/2^{i+1}, \quad i = 0, 1, \ldots, \tag{2.2}$$

where $c = \mathrm{sgn}f(a)\,(b - a)$. The sequence (2.2) converges to a root $r \in (a, b)$ if for some x_i,

$$\mathrm{sgn}f(x_0)\,\mathrm{sgn}f(x_i) = -1, \quad \text{for} \quad i = 1, 2, \ldots.$$

Furthermore, the number of iterations ν which are required to obtain an approximate root r^* such that $|r - r^*| \leqslant \varepsilon$ for some $\varepsilon \in (0, 1)$ is given by

$$\nu = \left\lceil \log(b - a)\,\varepsilon^{-1} \right\rceil, \tag{2.3}$$

where the logarithm in the above relation and also in the rest of the paper is taken with base two. Instead of the iterative formula (2.2) we can also use the following one:

$$x_{i+1} = x_i - c\,\mathrm{sgn}f(x_i)/2^{i+1}, \quad i = 0, 1, \ldots, \tag{2.4}$$

where $c = \mathrm{sgn}f(b)\,(b - a)$.

The reason for choosing the bisection method is that it always converges within the given interval (a, b) and is a globally convergent method. Moreover it has a great advantage since it is worst-case optimal; i.e., it possesses asymptotically the best possible rate of convergence in the worst case [25, 26]. This means that it is guaranteed to converge within the predefined number of iterations, and, moreover, no other method has this property. Therefore, using relation (2.3) it is easy to have beforehand the number of iterations that are required for the attainment of an approximate root to a predetermined accuracy. Finally, the bisection method requires only the algebraic signs of the function values to be computed, as is evident from (2.2) or (2.4); thus it can be applied to problems with imprecise function values. As a consequence for problems where the function value follows as a result of an infinite series (e.g., Bessel or Airy functions), it can be shown [35, 37] that the sign stabilizes after a relatively small number of terms of the series and the calculations can be sped up considerably.

3. A high level description of the method. Here is an informal description of the algorithm. For the detailed algorithm, see subsection 6.1. The algorithm gets as input the fraction of the roots that are required to be computed and begins by

subdividing the interval into two equal subintervals. Alternatively, it may get as input a predetermined "budget" of function evaluations. The main body of the algorithm consists of three steps.

1. A number of subintervals stored from previous steps, enter Step 1. These subintervals resulted from dividing the original interval, and at the first iteration these are just the two halves mentioned above. We determine the sign of the function at the endpoints of these subintervals. Depending on the number of simple roots in each subinterval, its endpoints will have the same signs (in the case of even roots) or opposite signs (in the case of odd roots). We therefore have certainty on the existence of at least one simple root in an interval with opposite signs, so we proceed in discovering a root in those intervals using the bisection method.

2. Note that at this point our interval has been divided into subintervals whose endpoints are of the same sign. These subintervals are kept for further examination if our stopping criterion has not been fulfilled yet. We propose two variants of a stopping criterion. The first (which we consider to be more interesting and suggest as the main variant of the algorithm) requires that a fraction of the total number of roots must be discovered before terminating the algorithm. The second terminates the algorithm if a predefined budget of function evaluations has expired.

3. If the stopping criterion has not been fulfilled, we subdivide the set of subintervals with maximum length (among all subintervals that have been stored) into two halves and go back to Step 1. Otherwise we output the discovered roots and halt.

Observe that the algorithm uses the function only to compute its sign at specific points during its execution. Thus, if the sign is available by other means, apart from a direct calculation (and this mainly refers to an inexact calculation using some kind of approximation of the function value), the algorithm can still run without any problem [4, 5, 35, 37]. As a result, the actual problem that the algorithm solves is that of discovering N hidden points in the interval, using only an that answers queries of the following form: "Is the number of points in a specific subinterval (a, b) odd or even?"

4. The distribution of odd subintervals. In this section we find the probability of having k specific subintervals containing an odd number of roots. The probability space on which we make our calculations and the whole framework can be found in [14]. We briefly mention here that we view each root as a random point in the interval $(0, 1)$. That is, we assume without loss of generality that the given interval is normalized, i.e., its length is 1. This assumption is followed throughout this paper. We also make the assumption that the roots are and distributed; i.e., intervals of equal length have equal probability of containing a root. Consequently, the length ℓ of a subinterval also gives the probability of a randomly chosen point to lie within this subinterval.

Consider the following situation: N points are chosen randomly and uniformly in the interval $(0, 1)$. Assume that we have chosen m nonoverlapping subintervals all of the same length ℓ. We view the remaining part of the interval (which may not be connected) as the $(m + 1)$-st subinterval.

Let us denote by E the event that each one of k specific subintervals contains an odd number of roots and the remaining $m - k$ contain an even number of roots. The following theorem gives the exact probability of the event E.

THEOREM 4.1. $(0,1)$ N m

ℓ [illegible] k [illegible] m [illegible]

$$\mathrm{P}(E) = \frac{1}{2^m} \sum_{S} \binom{N}{S} \ell^S (1 - m\ell)^{N-S} \sum_{j=0}^{m} (m - 2j)^S \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.1}$$

[illegible]. We associate with the ith subinterval, $i = 1, 2, \ldots, m$, a random variable X_i, $i = 1, 2, \ldots, m$, that takes the value 1 if the number of points in the subinterval is odd and 0 if it is even. These variables are identically distributed since we have assumed that the roots are uniformly distributed and the subintervals have equal length. The probability of the event E is given by

$$\mathrm{P}(E) = \mathrm{P}(X_{i_1} = 1, \ldots, X_{i_k} = 1, X_{i_{k+1}} = 0, \ldots, X_{i_m} = 0),$$

where $i_1, i_2, \ldots, i_m$ is a permutation of the set $\{1, 2, \ldots, m\}$. Notice that $\mathrm{P}(E)$ is independent of the specific permutation $i_1, i_2, \ldots, i_m$ since a point has the same probability equal to ℓ of being in each of the m subintervals.

The arrangement of the N points in the $m + 1$ subintervals can therefore be seen as a repetition N times of an experiment with $m + 1$ possible outcomes, the first m of which each have probability ℓ, and the last $(1 - m\ell)$. Let S, $0 \leqslant S \leqslant N$ be the total number of points in the m subintervals. The number of ways the subintervals $i_1, i_2, \ldots, i_m$ will contain, respectively, $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ points is given by the multinomial coefficient $\binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}}$. Hence we get for $\mathrm{P}(E)$

$$\mathrm{P}(E) = \sum_{S} \binom{N}{S} \sum \binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}} \ell^{\nu_{i_1} + \cdots + \nu_{i_m}} (1 - m\ell)^{N - (\nu_{i_1} + \cdots + \nu_{i_m})}, \tag{4.2}$$

where the inner summation is over all nonnegative integers $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ satisfying the conditions $\nu_{i_1} + \nu_{i_2} + \cdots + \nu_{i_m} = S$ with $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_k}$ being odd numbers and $\nu_{i_{k+1}}, \nu_{i_{k+2}}, \ldots, \nu_{i_m}$ being even. The above equation therefore becomes

$$\mathrm{P}(E) = \sum_{S} \binom{N}{S} \ell^S (1 - m\ell)^{N-S} \sum \binom{S}{\nu_{i_1}, \ldots, \nu_{i_m}}. \tag{4.3}$$

The proof now rests in finding a more easily computable form for the sum of multinomial coefficients, since the number of different tuples of $\nu_{i_1}, \nu_{i_2}, \ldots, \nu_{i_m}$ that fulfill the requirements is very large for any reasonable problem size, not to mention for computing large factorials.

Notice that the sum of multinomial coefficients (which we denote by $C(S, m, k)$) gives the number of permutations of S out of m distinct objects with repetitions such that k specific objects are each selected an odd number of times and the remaining $m - k$ are selected an even number of times.

Let us denote by $B(u, v)$ the number of v-permutations of u objects with repetition where each object is selected an odd number of times. In [2, p. 227] $B(u, v)$ is given

by

$$B(u,v) = \begin{cases} 2^{v-u+1} \displaystyle\sum_{j=0}^{\frac{u}{2}-1} (-1)^j \binom{u}{j} \left(\frac{u}{2}-j\right)^v, & u \text{ even, } v \text{ even,} \\ 0, & u \text{ even, } v \text{ odd,} \\ 2^{v-u+1} \displaystyle\sum_{j=0}^{\frac{u-1}{2}} (-1)^j \binom{u}{j} \left(\frac{u}{2}-j\right)^v, & u \text{ odd, } v \text{ odd,} \\ 0, & u \text{ odd, } v \text{ even.} \end{cases} \tag{4.4}$$

Let us also denote by $A(u,v)$ the number of v-permutations of u objects with repetition where each object is selected an even number of times. In the same reference, $A(u,v)$ is given by

$$A(u,v) = \begin{cases} 2^{-u} \displaystyle\sum_{j=0}^{u} \binom{u}{j} (u-2j)^v, & v \text{ even,} \\ 0, & v \text{ odd.} \end{cases} \tag{4.5}$$

So if we assume that ν points in total lie in the k subintervals which are expected to have an odd number of points, the remaining $S-\nu$ lie in the remaining $m-k$ subintervals. We may therefore express $C(S,m,k)$ in terms of $A(u,v)$ and $B(u,v)$ as follows:

$$C(S,m,k) = \sum_{\nu=k}^{S} \binom{S}{\nu} B(k,\nu) A(m-k, S-\nu). \tag{4.6}$$

This sum can now be transformed using (4.4) and (4.5), resulting in

$$C(S,m,k) = \frac{1}{2^m} \sum_{j=0}^{m} (m-2j)^S \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.7}$$

Substituting the above in (4.3) we get (4.1). □

When the initial interval is partitioned into m equal subintervals, i.e., when $m\ell = 1$, then all N roots must necessarily lie in the m subintervals. Let us denote E' the event that k specific subintervals have an odd number of roots under this assumption. We have the following corollary.

COROLLARY 4.1. [illegible] E' [illegible]

$$\mathrm{P}(E') = \frac{1}{2^m m^N} \sum_{j=0}^{m} (m-2j)^N \sum_{i=0}^{j} (-1)^i \binom{k}{i} \binom{m-k}{j-i}. \tag{4.8}$$

[illegible]. In this situation there are only m possible outcomes of the experiment, all of the same probability ℓ. The analysis above that resulted in (4.1) now gives the above formula. □

5. Estimating the total number of roots. Our stopping criterion in the main variant of the algorithm requires the knowledge of the total number of roots in the interval. For this purpose, the algorithm, in parallel with its main task of computing roots, also estimates the total number of roots. The estimation of the total number of

roots is revised after each iteration, when new roots have been discovered. Hence, at the beginning of the algorithm there is only a rough estimation of the number of roots, but as we proceed the additional information allows us to be more accurate in our prediction. For our estimation method we shall need the following two propositions.

PROPOSITION 5.1. [illegible] N [illegible] p_{odd} [illegible] ℓ [illegible]

$$p_{\text{odd}} = \frac{1-(1-2\ell)^N}{2}. \tag{5.1}$$

[illegible]. Let p_{even} be the probability that a subinterval of length ℓ contains an even number of roots. Then clearly $p_{\text{odd}} = 1 - p_{\text{even}}$. Recall that ℓ is the probability of having a specific root in an interval of length ℓ. Now

$$p_{\text{even}} = (1-\ell)^N + \binom{N}{2}\ell^2(1-\ell)^{N-2} + \binom{N}{4}\ell^4(1-\ell)^{N-4} + \cdots.$$

Observe that the right-hand side equals $\frac{1}{2}\left[(x+y)^N + (x-y)^N\right] = \frac{1}{2}\left[1+(1-2\ell)^N\right]$ for $x = 1-\ell$ and $y = \ell$. □

Notice that for a number of roots N on the order of 20 or so, p_{odd} is very close to $1/2$ for large enough ℓ. Therefore, roughly half of the intervals have at least one root. This is natural, since when the number of roots is large, the intervals with odd roots and the intervals with even roots are expected to be about the same. In such a case, after computing the roots, it is worthwhile continuing by subdividing the largest intervals. When, however, we have discovered a number of roots close to the total (and thus the remaining roots are few), the number of intervals with no roots begins to increase, and thus the required sign evaluations per root also increase.

To quantify the above, consider the random variables X_i, $i = 1, \ldots, m$, introduced in the previous section. These variables are not independent since, for example, if $m = N+1$ and we know that $X_i = 1$ for $i = 1, 2, \ldots, m-1$, the event $X_m = 0$ has probability 1; i.e., the probability of an X_i taking a certain value is influenced by the values of the other variables. This dependency is, however, very weak for large values of N in the sense that the joint probability distribution function of the X_i's is very close, yet not equal, to the product of the marginal distribution functions.

If the number of roots in any of the subintervals were independent from the number of roots in the other subintervals, then the probability of the event E (see the previous section) would be given by the formula $p_2 = (p_{\text{odd}})^k(1-p_{\text{odd}})^{m-k}$. This follows from the observation that the probability of having odd roots is p_{odd} for all subintervals of length ℓ, which along with the assumption of independence gives the above. What we show next is that the exact probability p_1 of E as given by (4.1) is actually very close to p_2. We demonstrate this by comparing p_1 and p_2 numerically.

In Table 5.1 we have included a small fraction of extensive numerical comparisons of p_1 and p_2 for several values of the parameters N, m, k, and ℓ. To reduce the size of the table, we present here results for ℓ fixed to a typical value of $m/0.8$. The results show that p_1 and p_2 are close enough to be considered equal for the purpose of estimating the number of roots N.

It is therefore a satisfactory approximation to consider the X_i's as independent Bernoulli random variables with probability given by (5.1). The following is a straightforward estimation of this probability.

TABLE 5.1
Numerical comparison of p_1 and p_2 for several values of the parameters N, m, and k.

N	m	k	p_1	p_2
100	20	2	0.952×10^{-6}	0.953×10^{-6}
100	20	15	0.959×10^{-6}	0.959×10^{-6}
100	50	5	0.367×10^{-14}	0.404×10^{-14}
100	50	40	0.246×10^{-15}	0.268×10^{-15}
100	120	12	0.101×10^{-27}	0.153×10^{-26}
100	120	70	0.333×10^{-41}	0.514×10^{-40}
500	100	10	0.808×10^{-30}	0.808×10^{-30}
500	600	60	0.816×10^{-134}	0.161×10^{-133}
1000	200	40	0.152×10^{-64}	0.454×10^{-64}
1000	200	160	0.596×10^{-60}	0.596×10^{-60}
1000	500	100	0.213×10^{-145}	0.390×10^{-145}
1000	500	400	0.393×10^{-156}	0.104×10^{-155}

PROPOSITION 5.2. [illegible] m [illegible] ℓ k [illegible] $(1-\alpha)$ [illegible] α [illegible] 0 [illegible] 1 [illegible] p_{odd} [illegible] ℓ [illegible]

$$p_{\text{lower}} \leqslant p_{\text{odd}} \leqslant p_{\text{upper}},$$

[illegible]

$$p_{\text{lower}} = \frac{k - z_{\alpha/2}\sqrt{\frac{k(m-k)}{m}}}{m},$$

$$p_{\text{upper}} = \frac{k + z_{\alpha/2}\sqrt{\frac{k(m-k)}{m}}}{m},$$

[illegible] $z_{\alpha/2}$ [illegible]

$$\text{Prob}(-z_{\alpha/2} \leqslant Z \leqslant z_{\alpha/2}) = 1 - \alpha,$$

[illegible] Z [illegible]. Consider the m subintervals. Let X_i be a random variable assuming value 1 if the ith subinterval has opposite signs and 0 otherwise, $i = 1, \ldots, m$. By the discussion above, the $X_i, i = 1, \ldots, m$, are considered to be independent Bernoulli random variables $B(1, p_{\text{odd}})$. A confidence interval for p_{odd} with approximate confidence coefficient $(1-\alpha)$ is given by

$$\frac{k}{m} - z_{\alpha/2}\sqrt{\frac{\frac{k}{m}\left(1-\frac{k}{m}\right)}{m}} \leqslant p_{\text{odd}} \leqslant \frac{k}{m} + z_{\alpha/2}\sqrt{\frac{\frac{k}{m}\left(1-\frac{k}{m}\right)}{m}},$$

which proves the proposition. □

For example, when $\alpha = 0.05$, then $z_{\alpha/2} = 1.96$. We use the above confidence interval to estimate p_{odd} and then use (5.1) to estimate N. This is done by computing

two values for N, calling them N_{lower} and N_{upper}. The first is computed by solving the equation (with respect to N)

$$p_{\text{lower}} = \frac{1-(1-2\ell)^N}{2}, \tag{5.2}$$

and the second is computed by solving the equation

$$p_{\text{upper}} = \frac{1-(1-2\ell)^N}{2}. \tag{5.3}$$

Notice that in some cases N_{upper} and even N_{lower} can be infinite. This will happen when insufficient data are available for determining exact values for the confidence interval. But for our purposes this is immaterial: in either of these cases we continue subdividing the intervals.

When after some iterations, both ends of the confidence interval are well defined, we choose some preferable value for N. A good choice seems to be $(N_{\text{lower}}+N_{\text{upper}})/2$. Or, we may decide to be on the safe side and choose N to be N_{upper}. Whichever we choose, we use it to decide whether it is time to terminate.

6. The method and its theoretical analysis. In this section we study the expected behavior of the algorithm as a function of the problem size N and the required fraction of roots λ. Our assumption is that, using the method described in the previous section, we know the total number of roots N, even though this can only be determined approximately within the confidence interval. Knowing N, the algorithm will run until a fraction λ of the roots has been discovered, i.e., λN roots. Clearly λ can vary from values close to 0, resulting in a quick termination of the algorithm, to values close to 1, which increase the cost of discovering new roots to forbiddingly high levels, after a large percentage has been discovered. Such values of λ will render the algorithm very costly and should not be used. We next give the expected cost of the algorithm for specific λ. To this end, we first give a technical definition that facilitates our analysis.

DEFINITION 6.1. [illegible] iteration number [illegible] i [illegible] ℓ [illegible] $\ell = 2^{-i}$

The reason for this definition is that at Step 1 of the algorithm, after dividing the intervals, we would typically expect some subintervals to have an even number and some to have an odd number of roots. If at some stage of execution of the algorithm all previous executions of Step 1 gave subintervals of both types or only even ones, then the iteration number i would coincide with what one would normally expect as iteration, since one execution of the three steps results in dividing the length of the largest subintervals by 2. But there is also the possibility that the intervals entering Step 1 are all of odd roots. In this case all subintervals would be further divided by bisection in Step 1 in the same "ordinary" iteration. We therefore give the above definition to relate the length of the largest even subintervals with the iteration number. Notice that the event of having all subintervals with an odd number of roots has small probability (actually it could only happen with some meaningful probability in the first few iterations). Therefore the iteration i as described above is an integer very close to the "ordinary" iteration. Note also that if the algorithm is in iteration i, then the effect on the original interval is the same as directly dividing it into 2^i equal subintervals and executing bisection in those subintervals that have an odd number of roots.

The next theorem gives the expected number of odd subintervals and hence the expected number of discovered roots in iteration i.

THEOREM 6.1. [illegible] $(0,1)$ [illegible] m [illegible]

$$\mathrm{E}_d = \frac{m^N - (m-2)^N}{2m^{N-1}}. \tag{6.1}$$

[illegible]. The probability of getting, after the division into m subintervals, k odd subintervals is given by $\binom{m}{k}\mathrm{P}(E')$. E' is the event where, after the division into m subintervals, k [illegible] intervals have an odd number of roots and the remaining $m-k$ have an even number. This event was formally defined in section 4 where its probability also was derived and is given by (4.8). Hence we have

$$\begin{aligned}
\mathrm{E}_d &= \sum_{k=0}^{m} k\binom{m}{k}\mathrm{P}(E') \\
&= \frac{1}{2^m m^N}\sum_{k=0}^{m} k\binom{m}{k}\sum_{j=0}^{m}(m-2j)^N\sum_{i=0}^{j}(-1)^i\binom{k}{i}\binom{m-k}{j-i} \\
&= \frac{1}{2^m m^N}\sum_{j=0}^{m}(m-2j)^N\binom{m}{j}\sum_{i=0}^{j}(-1)^i\binom{j}{i}\sum_{k=0}^{m} k\binom{m-j}{k-i}.
\end{aligned}$$

Now the rightmost sum is easily seen to be $(m-j)2^{m-j-1} + i2^{m-j}$. By substituting and after some algebraic manipulations, the theorem follows. □

Assume now that a fraction λ of the roots is required. In the ith iteration we have $m = 2^i$ and by substituting we get that the iteration where this is achieved, on average, is the solution rounded above, for i, of the equation

$$\lambda N = \frac{2^{iN} - (2^i - 2)^N}{2^{i(N-1)+1}}. \tag{6.2}$$

Table 6.1 gives the solution of (6.2) for various values of N and λ.

TABLE 6.1
Iteration and expected work for achieving a fraction of roots λ.

		N = 100		500		1000		5000	
		i	w	i	w	i	w	i	w
λ	0.5	7	776	10	3508	11	6515	13	25522
	0.7	8	1093	11	5175	12	9650	14	37146
	0.9	10	1919	13	11313	14	21724	16	83230
	0.95	11	2898	14	19203	15	37454	17	144998

As for the expected work for achieving the fraction λ, notice that at iteration i, where d odd subintervals have been discovered, $(2^i + 1)$ sign determinations are required to split the interval into 2^i subintervals, and if d of those are odd, then bisection will be applied to each of them, requiring $\log(\ell/\varepsilon)$ additional sign determinations where ℓ is the length of the subinterval. But $\ell = 2^{-i}$, and i is the solution of (6.2). Hence, we have the following corollary.

COROLLARY 6.1.

$$w = 2^i + \lambda N \log \frac{\ell}{\varepsilon} + 1 \equiv 2^i - \lambda N(i + \log \varepsilon) + 1, \tag{6.3}$$

(6.2)

Table 6.1 also gives the value of the expected work for various values of N and λ.

Another interesting aspect of the behavior of the algorithm is the increase of as a function of the iteration. Since an exact calculation of this seems to be hard, we approximate it as follows.

Let E_{d_i} be the expected number of discovered roots by the end of iteration i. Then, the expected number of discovered roots in iteration $i+1$ is $\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}$. Also, using Corollary 6.1, the expected work in iteration $i+1$ is $2^i - (i + \log \varepsilon)(\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}) - \mathrm{E}_{d_{i+1}}$. We approximate the expected work per root in iteration $i+1$ by the fraction of the expected work in iteration $i+1$ divided by the expected number of discovered roots in the same iteration. We call this fraction E^*_{i+1}. By the above discussion this is given by

$$E^*_{i+1} = \frac{2^i - \mathrm{E}_{d_{i+1}}}{\mathrm{E}_{d_{i+1}} - \mathrm{E}_{d_i}} - i - \log \varepsilon. \tag{6.4}$$

In Figure 6.1 we plot E^*_i for $N = 1000$ and $\varepsilon = 10^{-6}$ as a function of i. We have chosen to present the plot as a continuous curve, even though i is an integer, to better demonstrate its shape. It is interesting to note that the expected cost per root (as approximated by E^*) first decreases slightly and, after reaching a minimum, subsequently increases rapidly. The first part of the plot is explained since in the first iterations, discovering odd intervals is easy (about half of the intervals are odd), but computing the roots in those intervals costs more to the bisection (computing a root in an interval requires one more sign determination than in an interval of half the length). The subsequent rapid increment is caused by the fact that after some iteration, the odd intervals are becoming rare and increasingly more function evaluations are spent dividing an even interval and getting two even subintervals. After that point the behavior of the algorithm deteriorates rapidly and it soon becomes inefficient. This is the point to stop execution.

6.1. A detailed description of the proposed algorithm. In this section we give a detailed description of the proposed algorithm based on the previous results and analysis. The algorithm takes as input the desired fraction of the roots, λ.

1. Divide the interval into two equal subintervals. Set $i = 1$ and $m = 2$.
2. Let A be the set of subintervals with opposite signs at their endpoints and let k be its cardinality, i.e., $k = |A|$. Let B be the set of subintervals with the same signs at their endpoints.
3. Find one root in each interval in A using bisection (2.2) or (2.4). As bisection proceeds, it leaves out intervals with an even number of roots. Add those into set B.
4. Estimate the total number of roots using relations (5.2) and (5.3), where p_{lower} and p_{upper} are given in Proposition 5.2. In these equations take $\ell = 2^{-i}$, where i is the number of the current iteration, and use the parameters $m = 2^i$ and k defined above.
5. If both N_{lower} and N_{upper} are finite, check whether $d \geqslant \lambda N$. $N = (N_{\text{lower}} + N_{\text{upper}})/2$, and d is the number of discovered roots up to this point.

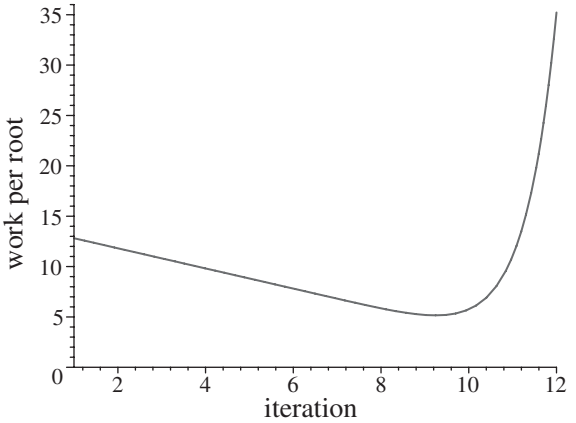


FIG. 6.1. *Expected work per root vs. iteration ($N = 1000$).*

6. If the criterion is not fulfilled or at least one of the N is infinite, then replace in B each subinterval [illegible] by its two halves. Set $i = i + 1$ and $m = 2^i$ and go to step 2.
7. If the stopping criterion of step 5 is fulfilled, then output the roots and terminate.

A useful observation that helps in improving the efficiency of the code (though it is not our goal to give implementation details here) is that set B is best implemented as a [illegible], i.e., a data structure that allows easy insertions and removals of its items and, what is most important, easy location of the largest element (in constant time). This is needed in steps 3 (insertion) and 6 (find max and removal). Actually, B is best viewed as a set consisting of subsets, where each subset contains subintervals of the same size. These sets are kept sorted by the heap structure, to minimize access time to the largest ones, as required in step 6.

The subdivision into intervals which are fractions that are a power of two is justified as follows: Recall that after we identify an interval with opposite signs at its endpoints, we proceed in discovering a root using bisection (this is step 3 above). But this has the side effect that these intervals will be subdivided further in the process of discovering the root. Now if we choose the initial subintervals to be a power of two, all of these subdivisions that emerge from bisection can be used in the future should the algorithm proceed to subsequent iterations, and thus some sign determinations can be saved. Moreover, this simplifies our arithmetic since each subinterval may be represented not by two real numbers but by its left endpoint and its size, which, in turn, may be represented by two integers, the exponents of actual numbers.

7. Conclusion and further research. As a conclusion, we have addressed the problem of computing the roots of a function when the number of roots is very large. This is a formidable problem with many applications in various fields of science. It seems possible, however, to attack the problem precisely by taking advantage of its size. To this end we have presented an algorithm that effectively discovers roots using an almost "blind" search up to a point where the original size of the problem has been greatly reduced. Then, more robust and expensive methods can be used to completely solve the problem. We have also given theoretical justification of its good performance, based on a probabilistic framework. The main advantages of the

proposed methodology are its simplicity, which results in fairly simple programming, and its efficiency, which increases with the problem size.

There are several possible directions to which this work may be extended. One very natural one is to consider an arbitrary distribution of roots, along the lines of [13]. An interesting problem in this case is to transform the method of estimating the number of roots that we presented here, in a way that takes into account the given distribution.

Moreover, there is no reason to limit the search space to a single dimension. With appropriate assumptions and, most of all, an appropriate existence criterion similar to Bolzano's, for instance, the Poincaré–Miranda hypercubes, Sperner simplices, or the characteristic polyhedra [16, 29, 30, 31, 32, 33, 34], it might be possible to extend the method to higher dimensions.

Finally, the most interesting extension is, of course, to apply the algorithm on a natural, real problem like the ones mentioned in the introduction.

Acknowledgment. The authors wish to thank the anonymous referees for their constructive comments, suggestions, and valuable criticisms, which helped us to improve the paper.

REFERENCES

[1] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[2] Ch. Charalambides, *Enumerative Combinatorics*, CRC Press Ser. Discrete Math. Appl., Chapman & Hall/CRC, Boca Raton, FL, 2002.

[3] Á. Elbert, *Some recent results on the zeros of Bessel functions and orthogonal polynomials*, J. Comput. Appl. Math., 133 (2001), pp. 65–83.

[4] I.Z. Emiris, B. Mourrain, and M.N. Vrahatis, *Sign methods for counting and computing real roots of algebraic systems*, Rapport de recherche 3669, INRIA (Institut National de Recherche en Informatique et en Automatique), Sophia Antipolis, France, 1999.

[5] I.Z. Emiris, B. Mourrain, and M.N. Vrahatis, *Sign methods for enumerating solutions of nonlinear algebraic systems*, in Hellenic European Research on Computer Mathematics and its Applications (HERCMA 2001), Vol. 2, E.A. Lipitakis, ed., L.E.A. Press, Athens, 2001, pp. 469–473.

[6] S. Graf, R.D. Mauldin, and S.C. Williams, *Random homeomorphisms*, Adv. Math., 60 (1986), pp. 239–359.

[7] S. Graf and E. Novak, *The average error of quadrature formulas for functions of bounded variation*, Rocky Mountain J. Math., 20 (1990), pp. 107–716.

[8] S. Graf, E. Novak, and A. Papageorgiou, *Bisection is not optimal on the average*, Numer. Math., 55 (1989), pp. 481–491.

[9] E.R. Hansen, *Global Optimization Using Interval Analysis*, Marcel Dekker, New York, 1992.

[10] B.J. Hoenders and C.H. Slump, *On the calculation of the exact number of zeros of a set of equations*, Computing, 30 (1983), pp. 137–147.

[11] I. Joó, *Exact Controllability and Oscillation Properties of Circular Membranes*, Dissertation for the title doctor of science, Budapest, Hungary, 1992.

[12] I. Joó, *On the control of a circular membrane* I, Acta Math. Hungar., 61 (1993), pp. 303–325.

[13] D.J. Kavvadias, F.S. Makri, and M.N. Vrahatis, *Locating and computing arbitrarily distributed zeros*, SIAM J. Sci. Comput., 21 (1999), pp. 954–969.

[14] D.J. Kavvadias and M.N. Vrahatis, *Locating and computing all the simple roots and extrema of a function*, SIAM J. Sci. Comput., 17 (1996), pp. 1232–1248.

[15] R.B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.

[16] B. Mourrain, M.N. Vrahatis, and J.C. Yakoubsohn, *On the complexity of isolating real roots and computing with certainty the topological degree*, J. Complexity, 18 (2002), pp. 612–640.

[17] D.W. Lozier, *Software needs in special functions*, J. Comput. Appl. Math., 66 (1996), pp. 345–358.

[18] D.W. Lozier and F.W.J. Olver, *Numerical evaluation of special functions*, in Mathematics of Computation 1943–1993: A Half-Century of Computational Mathematics, Proc. Sympos. Appl. Math. 48, W. Gautschi, ed., AMS, Providence, RI, 1994, pp. 79–125.

[19] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, UK, 1990.

[20] E. Novak, K. Ritter, and H. Woźniakowski, *Average-case optimality of a hybrid secant-bisection method*, Math. Comp, 64 (1995), pp. 1517–1539.

[21] E. Picard, *Sur le nombre des racines communes à plusieurs équations simultanées*, J. Math. Pures Appl. (4^e série), 8 (1892), pp. 5–24.

[22] E. Picard, *Traité d'analyse*, 3rd ed., Gauthier–Villars, Paris, 1922, chap. 4.7.

[23] V.P. Plagianakos, N.K. Nousis, and M.N. Vrahatis, *Locating and computing in parallel all the simple roots of special functions using PVM*, J. Comput. Appl. Math., 133 (2001), pp. 545–554.

[24] K. Ritter, *Average Case Analysis of Numerical Problems*, Lecture Notes in Math. 1733, Springer-Verlag, Berlin, 2000.

[25] K. Sikorski, *Bisection is optimal*, Numer. Math., 40 (1982), pp. 111–117.

[26] K. Sikorski, *Optimal Solution of Nonlinear Equations*, Oxford University Press, New York, 2001.

[27] J.F. Traub, G.W. Wasilkowski, and H. Woźniakowski, *Information–based complexity*, Academic Press, New York, 1988.

[28] K. Villaverde and V. Kreinovich, *A linear-time algorithm that locates local extrema of a function of one variable from interval measurements results*, Interval Computations, 4 (1993), pp. 176–194.

[29] M.N. Vrahatis, *Solving systems of nonlinear equations using the nonzero value of the topological degree*, ACM Trans. Math. Software, 14 (1988), pp. 312–329.

[30] M.N. Vrahatis, *CHABIS: A mathematical software package for locating and evaluating roots of systems of nonlinear equations*, ACM Trans. Math. Software, 14 (1988), pp. 330–336.

[31] M.N. Vrahatis, *A short proof and a generalization of Miranda's existence theorem*, Proc. Amer. Math. Soc., 107 (1989), pp. 701–703.

[32] M.N. Vrahatis, *An efficient method for locating and computing periodic orbits of nonlinear mappings*, J. Comput. Phys., 119 (1995), pp. 105–119.

[33] M.N. Vrahatis, *A generalized bisection method for large and imprecise problems*, in Scientific Computing and Validated Numerics, G. Alefeld, A. Frommer, and B. Lang, eds., Akademie Verlag, Berlin, 1996, pp. 186–192.

[34] M.N. Vrahatis, *Simplex bisection and Sperner simplices*, Bull. Greek Math. Soc., 44 (2000), pp. 171–180.

[35] M.N. Vrahatis, T.N. Grapsa, O. Ragos, and F.A. Zafiropoulos, *On the localization and computation of zeros of Bessel functions*, Z. Angew. Math. Mech., 77 (1997), pp. 467–475.

[36] M.N. Vrahatis, O. Ragos, T. Skiniotis, F.A. Zafiropoulos, and T.N. Grapsa, *RFSFNS: A portable package for the numerical determination of the number and the calculation of roots of Bessel functions*, Comput. Phys. Comm., 92 (1995), pp. 252–266; erratum, Comput. Phys. Comm., 117 (1999), p. 290.

[37] M.N. Vrahatis, O. Ragos, F.A. Zafiropoulos, and T.N. Grapsa, *Locating and Computing Zeros of Airy Functions*, Z. Angew. Math. Mech., 76 (1996), pp. 419–422.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 108–129
© 2005 Society for Industrial and Applied Mathematics

ON QUANTITATIVE ANALYSIS METHODS FOR MULTIGRID SOLUTIONS*

BORIS DISKIN†, JAMES L. THOMAS‡, AND RAYMOND E. MINECK‡

Abstract. Limitations of the classical local-mode Fourier (LMF) analysis in application to multigrid solutions of variable-coefficient problems are well known. This paper introduces alternative, very general, quantitative analysis methods for multigrid solutions of partial differential equations. The methods are applied to available, nonperfect multigrid solvers that deal with practical problems. The analysis methods considered in this paper focus on the main complementary parts of a multigrid cycle: relaxation and coarse-grid correction. Tasks for these parts, e.g., error smoothing for relaxation and reduction of smooth errors for coarse-grid correction, are assumed to be assigned in advance. Idealized relaxation (IR) and idealized coarse-grid (ICG) iterations are introduced. In these iterations, one part of the cycle (relaxation for IR iterations and coarse-grid correction for ICG iterations) is replaced with an idealized imitation known to be efficient for the assigned task; its complementary part is the actual part of a two-grid cycle. The analysis compares performances of the actual cycle and idealized iterations.

The IR and ICG iterations are very general and can be directly applied in the most complicated simulations including highly variable (or nonlinear) coefficients, complex geometries, and unstructured grids. The results of this analysis are not single-number estimates; they are rather convergence patterns of the iterations that may either confirm or refute expectations indicating what part of the actual solver is inefficient in carrying out the assigned task. The generality of the analysis makes it a valuable tool for analyzing complicated large-scale computational problems, where no other analysis methods are currently available.

In this paper, the IR and ICG iterations are applied to model problems of progressive complexity. The analysis proved sensitive to very delicate details of the actual multigrid cycle, pointing clearly to each instance when a part of the algorithm was inefficient.

Key words. multigrid methods, quantitative analysis, partial differential equations

AMS subject classifications. 65N12, 65N55, 65T99, 76B99

DOI. 10.1137/030601521

1. Introduction.

The subject of this paper is new, very general quantitative analysis methods for multigrid solutions of partial differential equation (PDE) problems. Multigrid solvers have proven to be the most efficient iterative methods for discretized PDE systems. Brandt [3, 7] defines a multigrid solver as having textbook multigrid efficiency (TME) if the solutions to the governing system of equations are attained in a total computational work that does not exceed a few (less than 10) minimal work units; a work unit is defined as the computational work required for one residual evaluation. The TME descriptor stems from the efficient multigrid solutions that have been demonstrated for elliptic problems and are available in multigrid textbooks [12, 27, 28]. Recently, TME methods have been developed for several nonelliptic problems in computational fluid dynamics (CFD) [26].

The development of quantitative analysis methodologies for iterative solutions of large-scale computational problems has lagged behind the multigrid practices. Some

*Received by the editors October 29, 2003; accepted for publication (in revised form) September 8, 2004; published electronically August 22, 2005.

http://www.siam.org/journals/sisc/27-1/60152.html

†National Institute of Aerospace (NIA), 144 Research Dr., Hampton, VA 23681 (bdiskin@nianet.org). This author's research was supported by the National Institute of Aerospace under NASA LaRC Cooperative Agreement NCC1-02043.

‡Computational Modeling and Simulation Branch, NASA Langley Research Center, Mail Stop 128, Hampton, VA 23681-2199 (James.L.Thomas@nasa.gov, Raymond.E.Mineck@nasa.gov).

methods, such as direct analyses of matrices corresponding to iterative solutions of discretized differential equations, may be useful for small model problems (see, e.g., [13] for the matrix analysis of one-dimensional defect-correction iterations), but are prohibitively expensive for multidimensional large-scale problems. Other methods, such as [18, 29], are qualitative and cannot provide sharp convergence estimates for practical algorithms. For many years, the local-mode Fourier (LMF) analysis has been the major practical tool for quantitative evaluation of multigrid convergence. The details of the analysis can be found in the literature [3, 6, 25, 27, 28]. Certain limitations of the LMF analysis, such as inability to account for boundary conditions and irregular coefficients, have also been known for a long time. These limitations make the LMF analysis inapplicable to many important computational problems, such as problems with highly variable coefficients, complex geometries, and unstructured grids, where other, more general, analysis techniques must be applied.

A typical approach to analyzing complex problems is based on the following steps: (1) identify contributors to different aspects of the problem, (2) formulate simplified models to isolate each identified contributor, (3) solve the simplified models, (4) extend the solutions to actual problems that cannot be directly analyzed. When applied to iterative methods, such an approach often leads to dramatic improvements of convergence properties of iterations. However, if the actual difficulty is represented not by a single object or phenomenon, but rather by interactions between several (or many) objects and phenomena, analytically solvable models of such interactions are scarce.

An alternative, more generally applicable approach is to employ an available, nonperfect solver to deal with a practical problem in its entirety, and then to isolate, identify, and improve the parts of the solver responsible for the less-than-optimal performance. This approach closely relates to the [illegible] concept introduced in [8] to evaluate multigrid efficiency of a given combination of a relaxation scheme with a set of coarse-grid variables.

The analysis methods introduced in this paper focus on the main complementary parts of a multigrid cycle: relaxation and coarse-grid correction. The full-weighting restriction and bilinear prolongation operators are fixed and assumed suitable for efficient multigrid solution. The novel analysis techniques, idealized relaxation (IR) and idealized coarse-grid (ICG) iterations, are applied to two-grid cycles solving PDE problems. Each part of the cycle is assigned with a task known in advance, e.g., relaxation may be assigned to smooth errors, coarse-grid correction may be assigned to reduce smooth error components.

To apply the new analysis, we first choose a desired sample fine-grid solution (zero is a natural choice for linear problems) and substitute it into the equations to generate the corresponding right-hand side and boundary data. Then we form an initial guess (for example, a random perturbation of the solution); thus, the fine-grid algebraic error is known. In the proposed analysis, idealized iterations probe the actual two-grid cycle to identify parts limiting the overall efficiency. In these iterations, one part of the cycle is actual, and its complementary part is replaced with an idealized imitation. The idealized imitations do not depend on the operators to be solved. They are numerical procedures acting directly on the known algebraic error to efficiently fulfill the task assigned to the corresponding part of the two-grid cycle. The results of the analysis are not single-number estimates; they are rather convergence patterns of the iterations that may either confirm or refute our expectations indicating what part of the actual cycle is not efficient in carrying out the assigned task. The possible implications from this analysis are at least twofold: one can seek a better

implementation of the identified part of the solver leaving other parts unchanged or one can adjust the algorithmic design by changing assignments to some (or all) parts of the solver. As an illustration of design adjustments, in some multigrid solvers for nonelliptic PDE problems, the relaxation assignment is extended beyond smoothing to reduce some smooth error components as well.

The new analysis methods can be regarded as a numerical extension of the LMF analysis to the problems where the classical LMF analysis is inapplicable. The idealized iterations are very general and can be directly applied in the most complicated situations including highly variable (or nonlinear) coefficients, global solvers, complex geometries, and unstructured grids. Similar to the LMF analysis, the idealized iterations are not constructive; they indicate algorithmic inefficiency but do not suggest a remedy. On the other hand, easily available additional information, such as the frequency content (in the LMF analysis) or shape and patterns (in the IR and ICG analysis) of slowly converging error, may hint at possible improvements to the algorithm. As shown by examples in section 5, the new analysis methods are sensitive to very delicate details of the tested two-grid cycle, clearly indicating what part of the algorithm is inefficient.

The material in this paper is organized as follows. Section 2 introduces the IR and ICG iterations as analysis methods for a two-grid cycle. In section 3, a motivating stagnation-flow problem, which is very difficult for standard analytical methods, is formulated. Section 4 discusses the limitations of LMF analysis in application to variable-coefficient problems. Section 5 presents examples of IR and ICG analysis applications; the limits of applicability of the LMF analysis are also demonstrated. An efficient multigrid solver for the motivating stagnation-flow problem that capitalizes on the findings of the new analysis is reported in section 6. The paper is concluded with some discussions in section 7.

2. IR and ICG iterations. IR and ICG iterations are analysis methods that test computational efficiency of a two-grid cycle. The two-grid cycle amplification matrix, $\mathbf{M}_h$, transforms the initial fine-grid algebraic error, $\mathbf{e}_h^{\text{old}}$, into the after-cycle error, $\mathbf{e}_h^{\text{new}}$,

$$\mathbf{e}_h^{\text{new}} = \mathbf{M}_h \mathbf{e}_h^{\text{old}}. \tag{2.1}$$

The amplification matrix can be defined as

$$\mathbf{M}_h = \mathbf{S}_h^{\nu_2} \mathbf{C}_h \mathbf{S}_h^{\nu_1}. \tag{2.2}$$

Here, ν_1 and ν_2 are small nonnegative integers representing the number of pre- and postrelaxation sweeps, $\mathbf{S}_h$ is the relaxation amplification matrix, and $\mathbf{C}_h$ is the amplification matrix of the coarse-grid correction:

$$\mathbf{C}_h = \mathbf{E}_h - \mathbf{P}_H^h \mathbf{L}_H^{-1} \mathbf{R}_h^H \mathbf{L}_h, \tag{2.3}$$

where $\mathbf{L}_h$ and $\mathbf{L}_H$ are the fine- and coarse-grid operator matrices, $\mathbf{P}_H^h$ and $\mathbf{R}_h^H$ are the prolongation and restriction matrices, and $\mathbf{E}_h$ is the fine-grid identity matrix.

For IR iterations, the coarse-grid correction part is actual and the relaxation is idealized. Assuming that the relaxation assignment is smoothing only, the idealized relaxation may be defined as an explicit error-averaging procedure. Some efficient averaging procedures can be derived from relaxation matrices, e.g., Gauss–Seidel or Jacobi, for a discretized Laplace operator. In this paper, we employ two IR procedures.

The first error-averaging procedure,

$$e_{i_1,i_2} \longleftarrow \frac{1}{4}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} e_{i_1,i_2}, \tag{2.4}$$

is performed in a lexicographic order and corresponds to the Gauss–Seidel relaxation for the five-point Laplacian. The second IR procedure corresponds to standard full-weighting averaging with stencil

$$e_{i_1,i_2} \longleftarrow \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} e_{i_1,i_2}. \tag{2.5}$$

The averaging (2.5) is simultaneously performed at all interior points. This procedure can be considered as a damped Jacobi relaxation for a nine-point Laplacian.

In IR analysis of a two-grid cycle, at each relaxation step, the known exact solution, if not zero, is subtracted from the current approximation to obtain the algebraic error function. The explicit averaging procedure is applied directly to the error function. The exact solution is then added back. The coarse-grid correction remains unchanged. If the convergence of IR iterations is much better than the convergence of the actual cycle, we conclude that the actual relaxation is not a sufficiently good smoother. Slow convergence of IR iterations indicates insufficient coarse-grid correction.

In ICG iterations, the relaxation scheme is actual and coarse-grid correction is idealized. Assuming that the actual restriction and prolongation operators are suitable for efficient multigrid solution, the ICG correction involves ideal operators $\mathbf{L}_H$ and $\mathbf{L}_h$, such that $\mathbf{L}_H^{-1}$ is an accurate approximation to $\mathbf{L}_h^{-1}$ for smooth components. The best ideal operators are the corresponding identity matrices. With this choice, the ICG correction becomes

$$\mathbf{C}_h^{\text{ideal}} = \mathbf{E}_h - \mathbf{P}_H^h \mathbf{R}_h^H. \tag{2.6}$$

In ICG analysis of a two-grid cycle, the ideal coarse-grid correction replaces the actual one. The idealized $\mathbf{C}_h^{\text{ideal}}$ is applied directly to the known algebraic errors obtained after pre-relaxation sweep(s) of the actual relaxation. If the performance of the ICG iterations is much better than the performance of actual two-grid cycle, we can conclude that the actual coarse-grid correction is lacking. Slow convergence observed in the ICG iterations is a sign of poor smoothing in relaxation.

The intended application of the IR and ICG iterations is to identify inefficient parts of an unacceptably slow actual cycle. It is assumed that the efficiency goal is defined in advance together with the assignments for each part of the cycle. The overall efficiency considerations set some additional requirements for the idealized imitations: in particular, the "reference cycle," which uses both IR and ICG iterations, has to meet the efficiency goal. In general, the convergence rate of the reference cycle represents a sensitivity threshold, i.e., the IR and ICG iterations, with given idealized imitations, may suggest some meaningful improvements only for actual cycles with convergence rates significantly slower than this threshold. The numerical tests where actual cycle converges with the rates superior (or comparable) to the threshold should be regarded as indications of sufficient efficiency of the actual cycle.

Note that the idealized iterations are not restricted to structured-grid applications. All multigrid algorithms, including unstructured-grid solvers, have inherently

available averaging procedures as well as restriction and prolongation operators that can be used for the idealized iterations.

3. Motivating problem. The set of two-dimensional nonconservative incompressible Euler equations is defined as

$$\begin{aligned} uu_x + vu_y + p_x &= f_u, \\ uv_x + vv_y + p_y &= f_v, \\ u_x + v_y &= f_c, \end{aligned} \tag{3.1}$$

where variables u, v, and p represent the x-directional velocity, the y-directional velocity, and the pressure, respectively. The first and second equations are referred as the u- and v-momentum equations, respectively; the third equation is the continuity equation. Standard boundary conditions for this formulation are the velocity components specified at the inflow boundary, the pressure specified at the outflow boundary, and the tangency condition (where the normal velocity component is set to zero) defined at the body surface.

The [illegible] is defined as a flow against a solid plane located at $x = 0$. The exact solutions have the form

$$\begin{aligned} u &= -ax, \\ v &= ay, \\ p &= -a^2 \tfrac{x^2+y^2}{2}, \end{aligned} \tag{3.2}$$

where a is an arbitrary constant.

A typical computational domain for the plane stagnation flow is a square $(x, y) \in [-D, 0] \times [-D/2, D/2]$. The inflow boundary is defined at the left vertical edge, $x = -D$, the outflow boundaries are at the horizontal edges, $y = \pm D/2$, and the tangency condition is defined at the plane surface, $x = 0$. If the boundary conditions are symmetric with respect to the axis $y = 0$, the stagnation point is located at $(x, y) = (0, 0)$.

The pressure-equation formulation [22, 23] is obtained from (3.1) by replacing the continuity equation with an equation for the pressure

$$\begin{aligned} uu_x + vu_y + p_x &= f_u, \\ uv_x + vv_y + p_y &= f_v, \\ (u_x)^2 + 2u_y v_x + (v_y)^2 + \Delta p &= f_p. \end{aligned} \tag{3.3}$$

The major advantage of this formulation is an easily available h-elliptic collocated-grid discretization. (See definition and discussions related to h-ellipticity in [2, 3, 27].) Besides the standard set of boundary conditions (given above), the pressure-equation formulation (3.3) requires an additional boundary condition at the inflow boundary. To ensure the same solution for the problems (3.1) and (3.3), this additional boundary condition is taken as the continuity equation enforced at the inflow boundary. This boundary condition can be implemented in a number of ways; a convenient choice is through the Neumann condition for the pressure derivative obtained by manipulating the normal (to the boundary) momentum equation and the continuity equation at the inflow boundary. That is,

$$\partial_x p = f_u - u\partial_x u - v\partial_y u = f_u + u(\partial_y v - f_c) - v\partial_y u, \tag{3.4}$$

where $f_u + u(\partial_y v - f_c) - v\partial_y u$ is a known quantity at the inflow boundary.

The second-order accurate interior discretization, $\mathbf{R}(\mathbf{q})$, of (3.3) is defined as upwind differencing for the convection operators of the momentum equations and central difference approximations for the pressure gradient; in the pressure equation, the standard h-elliptic five-point discretization, Δ^h, is applied for the Laplacian of the pressure, and central differencing is applied for the velocity derivatives. That is,

$$\mathbf{R}(\mathbf{q}) \equiv \begin{pmatrix} u\partial_x^u u + v\partial_y^u u + \partial_x^c p \\ u\partial_x^u v + v\partial_y^u v + \partial_y^c p \\ (\partial_x^c u)^2 + 2(\partial_y^c u)(\partial_x^c v) + (\partial_y^c v)^2 + \Delta^h p \end{pmatrix} = \begin{pmatrix} f_u \\ f_v \\ f_p \end{pmatrix}, \tag{3.5}$$

where $\mathbf{q} = (u, v, p)^T$, and superscripts u and c denote second-order accurate upwind and central differences for the first derivatives, respectively. Discretization of the Dirichlet conditions of given velocity components at the inflow boundary and given pressure at the outflow is straightforward. The Neumann conditions for the pressure can be implemented explicitly by employing a one-sided difference approximation for the normal derivative. With this implementation, the discrete pressure equation is not defined at the boundary. An alternative implementation uses two discrete equations for pressure defined at the same boundary point: a central approximation for the normal pressure derivative and a modified pressure equation. The pressure value located in the exterior of the computational domain is eliminated from these two equations. In the modified pressure equation, some velocity derivatives, $u_y, v_y, u_x = f_c - v_y$, are computed analytically from the Dirichlet conditions for the velocities; the derivative v_x is discretized with a one-sided stencil. An advantage of the latter implementation is a more compact stencil obtained for differencing the pressure at the boundary. For the discrete formulation (3.5), two numerical closure equations are required at the inflow boundary; one numerical closure equation is required at each outflow boundary and at the plane surface. The closure equations are usually modified discrete approximations to the target differential equations different from the discretizations in the interior. Detailed discussions and numerical tests for different boundary conditions can be found in [17].

The Newton linearization, $\frac{\partial \mathbf{R}}{\partial \mathbf{q}}$, of the operator (3.5) around the solution $\mathbf{q}$ is defined as

$$\frac{\partial \mathbf{R}}{\partial \mathbf{q}} \equiv \begin{pmatrix} Q^u + (\partial_x^u u) & (\partial_y^u u) & \partial_x^c \\ (\partial_x^u v) & Q^u + (\partial_y^u v) & \partial_y^c \\ 2(\partial_x^c u)\partial_x^c + 2(\partial_x^c v)\partial_y^c & 2(\partial_y^c v)\partial_y^c + 2(\partial_y^c u)\partial_x^c & \Delta^h \end{pmatrix}, \tag{3.6}$$

where $Q^u = u\partial_x^u + v\partial_y^u$ is a second-order accurate upwind discretization of the linear convection operator.

4. Limitations of LMF analysis. In this section, we discuss some limitations of the LMF analysis in applications to variable-coefficient problems. A rigorous LMF analysis for uniformly elliptic variable-coefficient problems has been considered in [5].

The classical LMF analysis applied to a variable-coefficient problem is defined in the following three steps.

1. Form a set of constant-coefficient approximations by freezing the variable coefficients at some locations, e.g., grid nodes.
2. Analyze each constant-coefficient problem on the infinite (or periodic) domain.
3. Take the final estimate of the analysis as the worst estimate among all the constant-coefficient problems.

The infinite grid assumption makes the LMF analysis intrinsically incapable of predicting the effects of boundary conditions. On finite computational domains, the choice of discrete boundary conditions and the way these conditions are treated in iterations can dramatically affect the convergence properties of iterations. A poor choice of boundary conditions may lead to an ill-posed discrete problem, even if the corresponding differential problem is well posed. Significant convergence deterioration may occur in the simplest well-known problems, e.g., the Laplace equation, as a result of improper handling of the boundary conditions. Some modifications of the LMF analysis, such as a half-space analysis of the first differential approximation (FDA analysis) [1, 11] and a discrete half-space analysis [16], have been developed to take the boundary conditions into account.

For the same reason, the LMF analysis cannot account for the downstream accuracy propagation that is an important convergence mechanism for some nonelliptic problems. In the presence of this mechanism, the accuracy is first achieved near the inflow boundary and then propagates downstream into the interior of the domain. (For illustration, see Figure 5.3 in section 5.2.) The computational domain affected by the downstream accuracy propagation may become very large, growing with each iteration. In the case of a downstream marching, it may expand to the entire domain in just one iteration. The estimates of the LMF analysis for nonelliptic flow problems can, however, be accurate in the areas that have yet to be affected by the downstream accuracy propagation from the inflow boundary. For iterative solvers of hyperbolic problems, e.g., nonmarching multigrid methods for the convection equation, LMF analysis estimates are usually accurate for the initial convergence; eventually, the accuracy propagated downstream from the inflow boundary extends over the entire computational domain.

Another serious limitation of the LMF analysis is the constant-coefficient framework itself. Accuracy of the LMF analysis extension to general variable-coefficient (or nonlinear) problems is limited by the accuracy of constant-coefficient approximations to the target problem. In this context, approximation means that the values of actual (variable) coefficients do not deviate significantly from a constant state.

There are many ways to measure how well a constant-coefficient formulation approximates a variable-coefficient one in a certain region. Here, we adopt a strong requirement that limits variation in each variable coefficient separately. Let $C_k(\mathbf{x})$, $k = 1, 2, \ldots$, be variable coefficients of the operator $\mathbf{L}_v$, and let

$$C_k^0 = C_k(\mathbf{x}_0) \tag{4.1}$$

be values of $C_k(\mathbf{x})$ frozen at the point $\mathbf{x}_0 = (x_0, y_0)$. We regard the constant-coefficient operator $\mathbf{L}_c$ obtained by freezing coefficients of $\mathbf{L}_v$ as approximating $\mathbf{L}_v$ in the vicinity of the point $\mathbf{x}_0$ with a relative accuracy ϵ, if there is a nonvanishing neighborhood $\Omega_{\mathbf{x}_0}^{s_1,s_2} = \{\mathbf{x} = (x, y), |x - x_0| < s_1, |y - y_0| < s_2\}$ such that, within this neighborhood,

$$|C_k^0 - C_k(\mathbf{x})| < \epsilon |C_k^0| \tag{4.2}$$

for any coefficient $C_k(\mathbf{x})$ of $\mathbf{L}_v$. The rectangular domain $\Omega_{\mathbf{x}_0}^{s_1,s_2}$ is called the ϵ-neighborhood of the point $\mathbf{x}_0$.

For any chosen ϵ, the values of s_1 and s_2 provide the scales at which the constant-coefficient problem approximates the variable-coefficient one with relative ϵ-accuracy. Reasonable values for ϵ are between zero and one, with $\epsilon = 0.5$ intuitively considered as a practical value. See Figure 4.1 for illustration of typical ϵ-neighborhoods for a

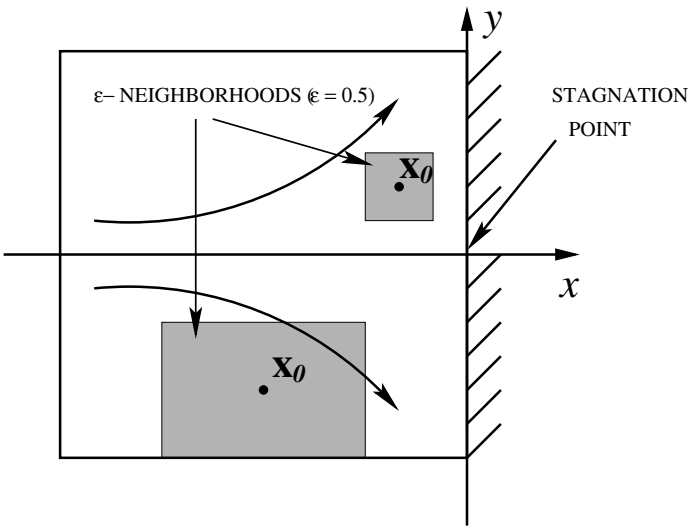


FIG. 4.1. *Model variable coefficient operator* (4.3)*: ϵ-neighborhoods corresponding to $\epsilon = 0.5$.*

variable-coefficient convection equation

$$-xu_x + yu_y = 0. \tag{4.3}$$

For the variable coefficients frozen at the point $\mathbf{x}_0 = (x_0, y_0)$, one should confine the LMF analysis to a computational subdomain that is located entirely within the ϵ-neighborhood of the point $\mathbf{x}_0$. If an ϵ-neighborhood is smaller than several mesh sizes in each dimension, the ϵ-neighborhood is considered vanishing; the LMF analysis should not be applied at this neighborhood. Implementation of the ϵ-neighborhood concept together with an assumption on the minimal number of mesh intervals sets some lower bounds on the ratios of coefficient to mesh size, e.g., the bound $\frac{y_0}{h} > 4$ occurs for (4.3) with $\epsilon = 0.5$ and a minimum of four mesh intervals in the y-direction per ϵ-neighborhood. Within this ϵ-neighborhood concept, some regions, such as the stagnation stream line, are inaccessible to the LMF analysis.

5. Examples of idealized iteration analysis.

5.1. Example 1: Skew discretization of Laplace equation. The first model problem is a two-grid cycle for a skew discretization of the Laplace operator. The computational domain is a unit square with the 17×17 uniform isotropic Cartesian grid. The discretization

$$\mathbf{L}u_{i_1,i_2} \equiv \frac{1}{2h^2}\left(u_{i_1+1,i_2+1} + u_{i_1-1,i_2+1} + u_{i_1+1,i_2-1} + u_{i_1-1,i_2-1} - 4u_{i_1,i_2}\right) = 0 \tag{5.1}$$

is subject to the Dirichlet boundary conditions. This formulation lacks h-ellipticity; therefore, a typical local relaxation exhibits poor smoothing properties. The tested solver is a two-grid (1, 2)-cycle with lexicographic point Gauss–Seidel relaxation, full coarsening, bilinear prolongation, and full-weighting residual restriction.

Classical LMF analysis can easily identify the problem. The smoothing factor, μ, of the Gauss–Seidel relaxation is computed as

$$\mu = \max_{\theta^{\mathrm{hf}}} \left| \frac{e^{i(\theta_y+\theta_x)} + e^{i(\theta_y-\theta_x)}}{4 - e^{-i(\theta_y+\theta_x)} - e^{-i(\theta_y-\theta_x)}} \right|, \tag{5.2}$$

where the maximum is taken over high-frequency modes

$$\theta^{\mathrm{hf}} = \left\{ (\theta_x, \theta_y), |\theta_x| \le \pi, |\theta_y| \le \pi, \max(|\theta_x|, |\theta_y|) \ge \frac{\pi}{2} \right\}. \tag{5.3}$$

The smoothing factor approaches unity for $|\theta_x| = |\theta_y| = \pi$.

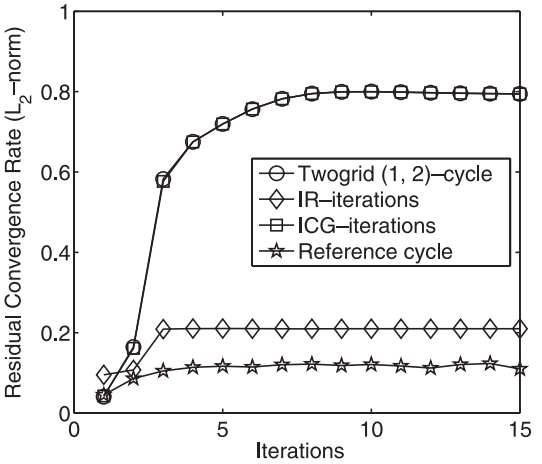


FIG. 5.1. *Convergence rates for skew Laplacian.*

The idealized iterations are effective indicators as well. The averaging procedure used in the IR iterations is given by (2.4). The convergence rates per iteration for actual cycle, reference cycle, IR, and ICG iterations are shown in Figure 5.1. The convergence rate of the two-grid cycle is well above the sensitivity threshold defined by the reference cycle. Recall that reference cycle employs both IR and ICG correction. Both IR and ICG iterations clearly point to poor smoothing in the actual cycle.

Regarding possible convergence improvements, the best advice is to avoid using non-h-elliptic discretization schemes. If one must solve the skew-Laplacian formulation, a good strategy is to consider this formulations as two uncoupled problems. Using a checkerboard analogy, the solver for the equations defined at black nodes should be independent of the solver for the formulation at white nodes. Considered at nodes of the same color, the discretization (5.1) is h-elliptic.

5.2. Example 2: Convection equation. In this section, we analyze a two-grid solver for the standard first-order accurate upwind discretization of the constant-coefficient convection equation

$$\partial_x^h u + \partial_y^h u = 0, \tag{5.4}$$

where ∂_x^h and ∂_y^h are two-point upwind differences for the x- and y-derivatives, respectively. A rectangular computational domain is covered by a uniform Cartesian

grid with the characteristics aligning with the diagonal direction. This is a nonelliptic problem known to represent difficulties for standard multigrid solvers. The solver to be tested is a standard full-coarsening two-grid $(1, 2)$-cycle including a red-black relaxation scheme with an underrelaxation parameter $w = 0.8$, full-weighting residual restriction, and bilinear prolongation. The coarse-grid discretization is identical to the discretization on the fine-grid. The LMF analysis for this problem is available in [27]. The relaxation smoothing factor is predicted as 0.52; the convergence of the two-grid cycle is limited by coarse-grid correction and is estimated as 0.5 per cycle. The slowly converging error components are smooth components that are much smoother in the characteristic direction than in other directions. These components are called characteristic components.

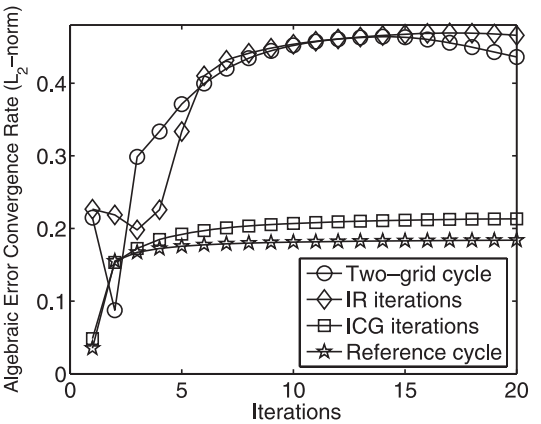


FIG. 5.2. *Convergence rates for a first-order discretization of the convection equation with periodic boundary conditions in the y-direction. Standard coarse-grid discretization.*

The numerical tests including the two-grid $(1, 2)$-cycle, IR and ICG iterations, and reference cycle have been performed on uniform grid, 1025×65 (domain 16×1), with the zero inflow boundary condition at $x = 0$ and periodic boundary conditions in the y-direction. The x-size of the domain has been chosen to be long enough for observing convergence slowdown caused by characteristic error components. The initial approximation is random. The results are shown in Figure 5.2. The averaging procedure used in IR iterations is given by (2.5).

Convergence rates of the two-grid $(1, 2)$-cycle are much slower than the rates of the reference cycle and close to the predictions of the LMF analysis. Convergence rates of IR and ICG iterations identify the coarse-grid correction as a part of the cycle limiting the overall efficiency. Similar to the frequency content in the LMF analysis, the profile of the algebraic errors after many iterations (see Figure 5.3 for errors after 20 and 40 iterations) clearly indicates the type of error that is slow to converge, namely smooth characteristic error. Moreover, the error profile demonstrates the effect of accuracy propagation from the inflow boundary. This important convergence mechanism for nonelliptic problems eludes the LMF analysis.

To improve convergence rates of multigrid solvers for convection-dominated (and other nonelliptic) problems, one can facilitate the accuracy propagation from the inflow boundary by employing a "global" relaxation scheme, e.g., downstream marching [11, 16] or implicit line relaxation [20]. In this context, "global" relaxation means that the relaxation scheme strongly affects at least some smooth errors, as opposed

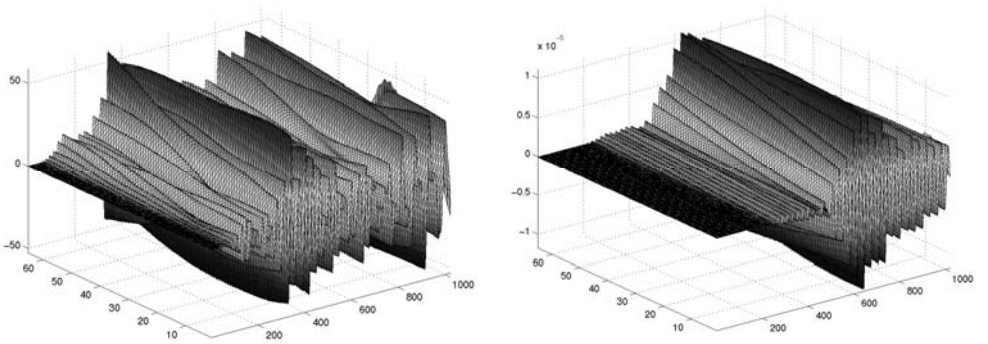


FIG. 5.3. *Error surface after* 20 *(left) and* 40 *(right) two-grid* $(1,2)$*-cycles for a first-order discretization of the convection equation with periodic boundary conditions in the* y*-direction. Grid* 1025×65.

to "local" relaxation that mostly affects high-frequency errors. Within such an approach, the relaxation assignment is extended to reduce both oscillatory and characteristic errors. The approach requiring characteristic error reduction in relaxation can be efficient as long as the characteristics of the nonelliptic operator are not closed. For problems with closed characteristics, the characteristic errors should be reduced in coarse-grid correction.

For efficient coarse-grid correction, the coarse-grid discretization should be adjusted to better approximate the fine-grid operator. One way to adjust the discretization is to employ a Galerkin coarse-grid operator with intergrid transfers satisfying certain accuracy conditions [30]. Another way is to explicitly add some adjustment terms to the coarse-grid operator to match the fine-grid (characteristic-component) first differential approximation [9, 14, 15, 19]. For the fine-grid discretization (5.4), the coarse-grid operator can be modified as

$$\bar{\partial}_x^h = \partial_x^h + \frac{h}{4}\partial_{xx}^u, \tag{5.5}$$

where $\bar{\partial}_x^h$ is a modified discretization of the x-derivative, ∂_x^h is the first-order upwind discretization, and ∂_{xx}^u is the first-order accurate three-point upwind discretization of the second derivative. The discretization of the y-derivative has been modified accordingly. With this modified coarse-grid operator, convergence rates of the two-grid $(1,2)$-cycle are improved significantly to 0.27 per cycle, as shown in Figure 5.4; the rates of IR iterations also improved. ICG iterations and reference cycle are unaffected, as expected.

5.3. Example 3: Laplace equation with Neumann boundary condition. In this section, the IR and ICG analyses are applied to the Laplace equation defined on a square domain $(x, y) \in [0,1] \times [0,1]$ with one Neumann boundary condition at $x = 1$ and the Dirichlet conditions at other boundaries. The Laplacian is discretized on a uniform 65×65 grid with the standard five-point stencil. The Neumann boundary condition is discretized with a three-point, one-sided differencing. The solution method is a full-coarsening two-grid $(1,2)$-cycle with lexicographic point Gauss–Seidel relaxation, full-weighting residual restriction, and bilinear prolongation. At the end of the relaxation sweep, the residuals of the Neumann equations at the boundary are zeros.

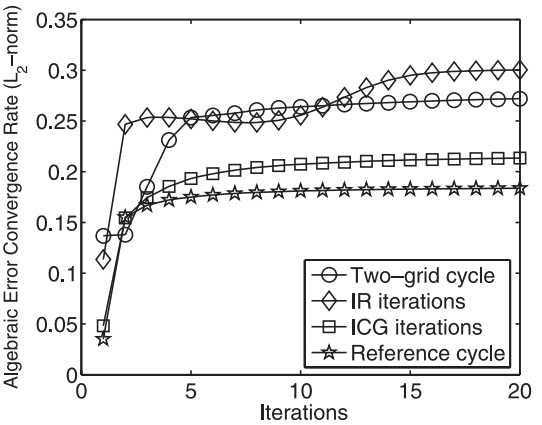


FIG. 5.4. *Convergence rates for a first-order discretization of the convection equation with periodic boundary conditions in the y-direction. Grid* 1025×65. *Modified coarse-grid discretization.*

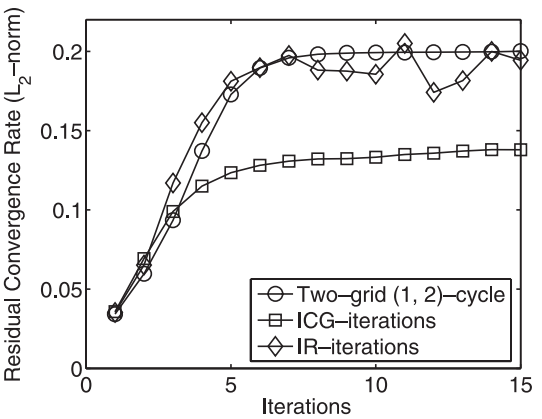


FIG. 5.5. *Convergence rates for the Laplace equation with Neumann boundary conditions. No boundary relaxation.*

For the two-grid $(1,2)$-cycle, the LMF analysis predicts the convergence rate of 0.12 per cycle. Figure 5.5 shows the residual convergence rates for the two-grid $(1,2)$-cycle and for the corresponding IR and ICG iterations. IR iterations employ the averaging procedure (2.4) in the interior performed in the lexicographic order. At the Neumann boundary, the error is averaged as

$$e_{i_1,i_2} \longleftarrow \frac{1}{2}\left(e_{i_1,i_2+1} + e_{i_1,i_2-1}\right). \tag{5.6}$$

In this test, convergence rates of reference cycle and ICG iterations are identical. The asymptotic convergence rate of the actual two-grid cycle is 0.20, noticeably worse than the rate predicted by the LMF analysis and observed in computations with all-Dirichlet boundary conditions.

The difficulty associated with the Neumann boundary relates to large interior residuals created after relaxing boundary conditions. The standard full-weighting

residual restriction near the Neumann boundary is inaccurate and degrades the quality of the coarse-grid correction. The LMF analysis cannot account for this effect. The ICG and IR iterations clearly indicate that the coarse-grid correction should be improved to accelerate convergence of the two-grid cycle.

There are several approaches to overcome this convergence slowdown; e.g., one can design a more accurate restriction operator near the Neumann boundary [27] or a local relaxation scheme that does not create large interior residuals. A more general approach, directly extendible to more complicated computational problems, is to apply a block relaxation near the boundary relaxing simultaneously all the equations defined at a certain number (, .) of grid lines adjacent to the Neumann boundary. With (almost) zero residuals in a boundary vicinity, an accurate residual restriction is no longer important. We refer to this special block relaxation as . The boundary relaxation procedure, performed once per cycle before transferring residuals to the coarse grid, can be considered a part of an improved coarse-grid correction scheme. Figure 5.6 illustrates the sensitivity of the two-grid $(1, 2)$-cycle and the IR iterations to the depth of the boundary relaxation region. The ICG iterations are not sensitive to the presence of boundary relaxation.

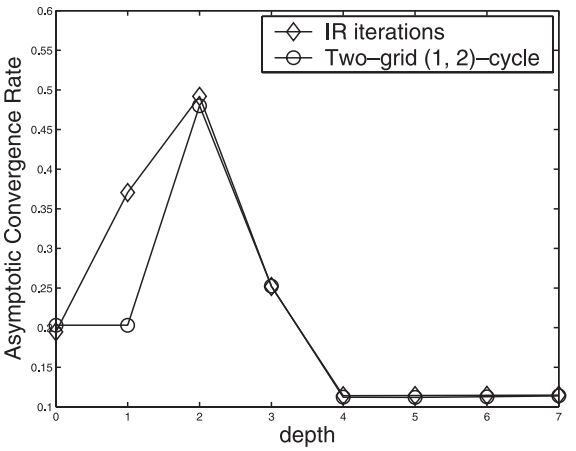


FIG. 5.6. *Convergence rates for the Laplace equation with Neumann boundary conditions. Boundary relaxation is applied once before residual restriction.*

The test results can be summarized as follows: For small depths of the boundary relaxation region, $0 \leq depth \leq 3$, the convergence rates of both iterations are not satisfactory. For $depth \geq 4$ the desired convergence rates of 0.12 per cycle are achieved. The performance of iterations is not sensitive to further expansions of the boundary relaxation region. Thus, the value $depth = 4$ is considered optimal. The optimal depth depends on the discretization stencils for boundary and interior equations, but does not depend on the grid size. A smaller optimal boundary relaxation depth can be achieved with a more compact discretization of boundary conditions. For example, a two-point wide discretization of the Neumann condition, derived from joint formulation of the interior five-point Laplacian and the central approximation for the normal derivative at the boundary, leads to the optimal $depth = 3$. The optimal $depth = 3$ can also be achieved for noncompact discretization of the Neumann boundary condition if the boundary relaxation procedure is performed at the end of each relaxation sweep rather than once per cycle. Such an arrangement softens

the residual jump at the interface between the boundary relaxation region and the interior. The latter relaxation-based implementation should not obscure the major role of boundary relaxation as facilitating coarse-grid correction. In practice, we are following an empirical rule to relax simultaneously all the discrete (physical and numerical) boundary conditions as well as all the interior equations that share data with the boundary condition equations.

5.4. Example 4: System of constant-coefficient PDEs. In this section we analyze a two-grid solver for a system of constant-coefficient equations related to the full Newton linearization (3.6) for the plane-flow solution (3.2) with $a = 1$. The system of constant-coefficient differential equations is formulated as

$$\mathbf{L}(\mathbf{q}) \equiv \begin{pmatrix} \bar{Q}^u - 1 & 0 & \partial_x^c \\ 0 & \bar{Q}^u + 1 & \partial_y^c \\ -2\partial_x^c & 2\partial_y^c & \Delta^h \end{pmatrix} \mathbf{q} = 0, \tag{5.7}$$

where $\mathbf{q} = (u, v, p)^T$, $\bar{Q}^u = \bar{u}\partial_x^u + \bar{v}\partial_y^u$, and $\bar{u}$ and $\bar{v}$ are positive constants. The computational domain is a rectangle $(x, y) \in [-\bar{u} - X, -\bar{u} + X] \times [\bar{v} - Y, \bar{v} + Y]$ with

$$0 < X \le \frac{\bar{u}}{2}, \qquad 0 < Y \le \frac{\bar{v}}{2}. \tag{5.8}$$

The latter conditions, referred as [illegible], are introduced to ensure that the constant-coefficient problem provides a good relative ϵ-accuracy ($\epsilon = 0.5$) in approximating the full-Newton linearization of the plane flow in the vicinity of $\mathbf{x}_0 = (-\bar{u}, \bar{v})$. A uniform Cartesian grid with a minimum of four mesh intervals in each spatial direction is applied. The physical boundary conditions are the following: u, v, and $\partial_x p$ are given at the inflow $x = -\bar{u} - X$; p is given at the outflow $x = -\bar{u} + X$; and periodicity is assumed in the y-direction ($\mathbf{q}(x, \bar{v} - Y) = \mathbf{q}(x, \bar{v} + Y)$). The normal derivative of the pressure, $\partial_x p$, at the inflow boundary is discretized with the second-order accurate, three-point, one-sided stencil. The numerical closure equations include central differencing for the x-derivative in the operator $\bar{Q}^u$ at the interior grid points adjacent to the inflow boundary; at the outflow, the x-derivative of the pressure in the first momentum equation is discretized with the three-point upwind stencil. The exact solution of the problem is $\mathbf{q} \equiv 0$.

The relaxation scheme consists of three steps.

(i) Line-implicit Gauss–Seidel relaxation is performed for the Laplace operator in the pressure equation with frozen u and v. In this relaxation, all the pressure equations defined at a vertical grid line with the same y-coordinate are relaxed simultaneously; this y-implicit relaxation scheme proceeds in the [illegible] direction, from the outflow to the inflow.

(ii) The boundary relaxation ($depth = 5$) is performed near the inflow boundary. All the equations defined at the five-grid lines adjacent to the inflow boundary (the boundary included) are solved simultaneously. Values of u, v, and p in the interior beyond the boundary relaxation region remain unchanged.

(iii) The momentum equations are [illegible] marched for u and v with p being fixed.

At the end of the relaxation sweep, the residuals of the momentum equations are zero.

The LMF analysis for this solver provides the following estimates. The Fourier symbol of the relaxation scheme is computed as

(5.9)
$$\hat{\mathbf{S}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} \hat{Q}-1 & 0 & \hat{\partial}_x^c \\ 0 & \hat{Q}+1 & \hat{\partial}_y^c \\ 0 & 0 & G\hat{S}^h \end{pmatrix}^{-1} \begin{pmatrix} \hat{Q}-1 & 0 & \hat{\partial}_x^c \\ 0 & \hat{Q}+1 & \hat{\partial}_y^c \\ -2\hat{\partial}_x^c & 2\hat{\partial}_y^c & \hat{\Delta}^h \end{pmatrix},$$

where

(5.10)
$$\begin{aligned} &\hat{Q} = \tfrac{\bar{u}}{h}\left(\tfrac{3}{2} - 2e^{-i\theta_x} + \tfrac{1}{2}e^{-i2\theta_x}\right) + \tfrac{\bar{v}}{h}\left(\tfrac{3}{2} - 2e^{-i\theta_y} + \tfrac{1}{2}e^{-i2\theta_y}\right), \\ &\hat{\partial}_x^c = \tfrac{i}{h}\sin(\theta_x), \\ &\hat{\partial}_y^c = \tfrac{i}{h}\sin(\theta_y), \\ &\hat{\Delta}^h = \tfrac{1}{h^2}\left(2\cos(\theta_x) + 2\cos(\theta_y) - 4\right), \\ &G\hat{S}^h = \tfrac{1}{h^2}\left(e^{i\theta_x} + 2\cos(\theta_y) - 4\right), \end{aligned}$$

θ_x and θ_y are normalized Fourier frequencies, $|\theta_x| \le \pi$, $|\theta_y| \le \pi$. The smoothing factor, μ, is

(5.11)
$$\mu = \max_{\theta^{\mathrm{hf}}} \rho(\hat{\mathbf{S}}),$$

where $\rho(\hat{\mathbf{S}})$ is the spectral radius of the matrix $\hat{\mathbf{S}}$, and the maximum is taken over the set of high-frequency Fourier modes θ^{hf} defined in (5.3).

The unrestricted LMF analysis considers the infinite domain with no compatible-coefficient restrictions (5.8) applied. This analysis predicts divergence of the relaxation scheme for high-frequency error components. For $\theta_x = \pi/2$, $\theta_y = -\pi/2$, and $\frac{\bar{u}}{h} = \frac{\bar{v}}{h} = \frac{1}{2}$, the convection symbol $\hat{Q} = 1$ and the smoothing factor $\rho(\mathbf{S}_h) = \infty$. Imposing the compatible-coefficient restrictions (5.8) together with the assumption that there are at least four mesh intervals in each dimension effectively implies $\frac{\bar{u}}{h} \ge 4$, $\frac{\bar{v}}{h} \ge 4$. With these restrictions, the LMF analysis predicts the smoothing factor of 0.53 on the 4×4 grid and factors better than 0.5 at finer grids converging to $\frac{1}{\sqrt{5}}$; this latter value is the smoothing factor of line relaxation for the scalar Laplace equation.

TABLE 5.1
Smoothing and amplification factors for the ϵ-neighborhood corresponding to $\bar{u} = \bar{v} = 1$, $\epsilon = 0.5$.

	Relaxation		Two-grid
N	Smoothing	Amplification	Amplification
8	0.51	1.00	0.14
16	0.51	1.14	1.71
32	0.50	1.17	2.60
58	0.50	18.66	31703.
64	0.50	1.46	8.15
128	0.50	1.80	24.23

For $\bar{u} = \bar{v} = 1$, the relaxation amplification factors are bounded by 1 everywhere except for the characteristic errors. As an example, Table 5.1 collects the relaxation smoothing and amplification factors (the pressure equation is relaxed with lexicographic pointwise Gauss–Seidel relaxation), as well as the amplifications factors of the corresponding two-grid (1, 2)-cycle. As noted previously, the smoothing factors are good. The amplification factors are erratic. On finer grids, the amplification factor

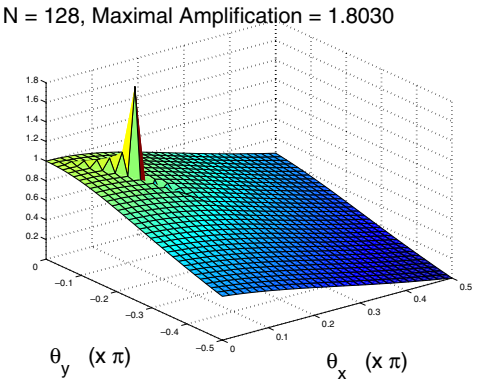


FIG. 5.7. *Relaxation amplification factor.*

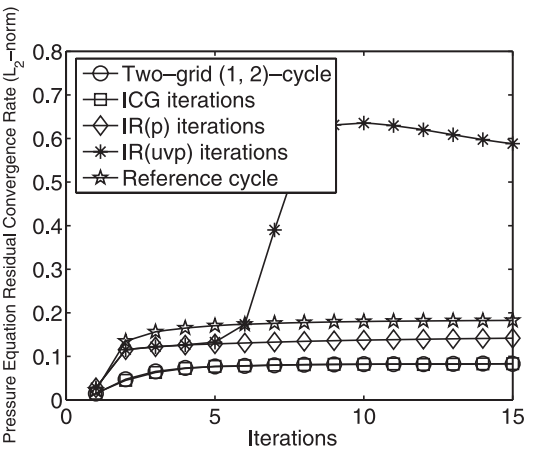


FIG. 5.8. *Convergence rates of the pressure equation residuals in solutions of the constant-coefficient problem* (5.7). *Grid* 257×33.

tends to infinity for certain smooth (but not the smoothest) characteristic frequencies of the convection operator ($\theta_x = -\theta_y$), for which the symbol $\hat{Q}$ approaches 1. On coarse grids, the smallest possible value of $|\hat{Q} - 1|$ is determined by the mesh size h and the frequency increment $\delta\theta$ and may vary nonmonotonically; that explains the irregular behavior of the amplification factors. For illustration purposes, the results for a nonstandard 59×59 grid are added to the table. At this grid, $h = 1/58$, and, for $\theta_x = -\theta_y = \frac{8}{58}\pi$, $\hat{Q} = 0.9909$. Figure 5.7 illustrates the relaxation amplification factor over a part of the frequency domain. More details of the analysis can be found in [17]. The LMF analysis does not account for the accuracy propagation from the inflow boundary, that is a major convergence mechanism for this two-grid cycle. Employment of a downstream marching in the relaxation scheme on a domain restricted as (5.8) ensures that the smooth characteristic errors, for which the LMF analysis predicts divergence, are strongly affected by the inflow boundary.

Figure 5.8 shows the convergence rates of the L_2-norm of the pressure-equation residuals for the four iterative methods solving problem (5.7) with $\bar{u} = \bar{v} = 1$ on the computational domain $(x, y) \in [-\frac{3}{2}, -\frac{1}{2}] \times [\frac{15}{16}, \frac{17}{16}]$ covered by the uniform isotropic

257×33 Cartesian grid. The size and shape of the domain have been chosen to satisfy compatible-coefficient restriction (5.8) and to have an isotropic grid with a large number of grid nodes between the inflow and outflow boundaries. The plot marked "Two-grid (1, 2)-cycle" shows the convergence rates of a two-grid cycle solving (5.7). The cycle includes the three-step relaxation scheme described above, full coarsening, full-weighting residual restriction, and bilinear prolongation. We use the coarse-grid discretization identical to (5.7), therefore, the assignment for relaxation is extended. That is, we suggest that downstream marching of the momentum equations efficiently reduces the smooth characteristic errors that cannot be approximated on the coarse grid. To illustrate the importance of this extended assignment for relaxation, we include two different IR iterations: IR iteration of the first type, IR(p) iteration, replaces step (i) in the relaxation scheme with the averaging procedure (2.5) for the pressure only. Boundary relaxation and downstream marching at steps (ii) and (iii) are performed as in actual relaxation. The second type of IR iterations is IR(uvp) iterations, where steps (i), (ii), and (iii) are replaced by averaging (2.5) for u, v, and p, and no downstream marching is performed.

Convergence rates of the actual cycle are better than convergence rate of the reference cycle with IR(uvp) iterations, indicating the efficiency goal is achieved. ICG iterations and IR(p) iterations provide an accurate estimate for the convergence of the actual cycle. The convergence rates of the ICG iterations and the two-grid cycle are virtually indistinguishable; the IR(p) convergence rates are predictably somewhat worse because line relaxation damps certain intermediate (neither high-frequency nor very smooth) error components significantly better than the ideal averaging procedure (2.5). With progressive grid refinement, the (not shown) convergence rates of the IR(uvp) iterations tend to the value 0.75; this is in agreement with the LMF analysis predictions for convergence rates of full-coarsening multigrid methods applied to a second-order discretization of the convection equation.

Although not shown in the figure, on larger domains (e.g., $[-5, 3] \times [\frac{1}{2}, \frac{3}{2}]$), accuracy propagation from the inflow boundary does not dominate the entire computational domain; divergence of the two-grid cycle is indeed observed in numerical tests, as predicted by the LMF analysis. However, on such domains, the compatible-coefficient restrictions (5.8) are violated; therefore, the formulation (5.7) does not provide a relevant constant-coefficient approximation to the variable-coefficient problem (3.6).

6. Multigrid solver for plane stagnation flow. The basic cycle employed in the multigrid algorithm for solution of the discrete equations (3.5) is an FV(ν_1, ν_2) cycle sketched in Figure 6.1. The FV cycle represents a modification of the well-known multigrid F cycle (see, e.g., [27]). The multigrid solutions have been obtained as follows: Random initial approximation has been assigned at each grid, and a full approximation scheme (FAS) version of FV(1,2) multigrid cycle has been used. The nonlinear equations at the fixed coarsest grid have been solved exactly; thus, finer grids used progressively more levels of multigrid. Compact formulations for discrete boundary conditions at the inflow and tangency boundaries have been implemented (see section 3 and [17]). The relaxation scheme included four steps: (1) point Gauss–Seidel relaxation of the pressure equation performed in a downstream order, (2) downstream relaxation of momentum equations, (3) boundary relaxation ($depth = 3$) at the inflow boundary, and (4) boundary relaxation ($depth = 3$) at the tangency boundary. Within the FAS cycle, the four-step relaxation scheme, full coarsening, full-weighting restriction of residuals, and bilinear prolongation of corrections have been used with direct injection of the nonlinear solution to the coarser level.

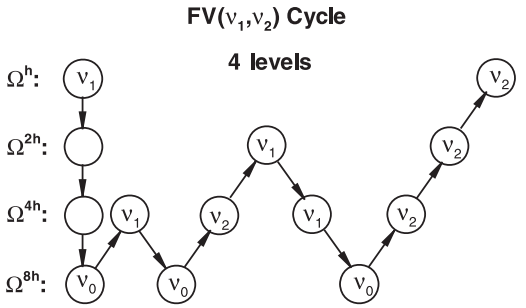


FIG. 6.1. *Schematic of the FV cycle for 4-level multigrid where ν_0 denotes the number of relaxations on the coarsest mesh (Ω^{8h}).*

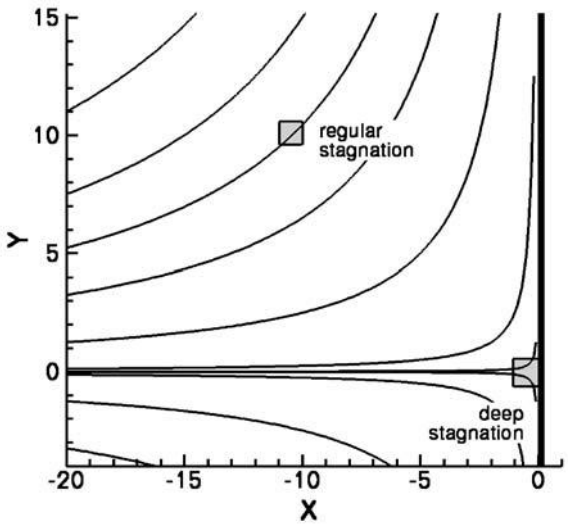


FIG. 6.2. *Plane stagnation flow streamlines and locations of the computational domains.*

Multigrid convergence has been studied for two plane stagnation flow regimes illustrated in Figure 6.2: (1) deep stagnation regime, where stagnation point is a part of the computational boundary; and (2) the regular stagnation flow, where the computational domain is shifted away from stagnation. For efficient solutions of stagnation flows in general curvilinear geometries, we refer readers to [17].

The convergence of the L_2-norm of residuals for the regular stagnation regime is shown in the left plot of Figure 6.3. Diamonds correspond to the residuals of the pressure equation; circles and squares denote residuals of the u- and v-momentum equations, respectively. The largest residuals at each cycle are those of the pressure equation with the momentum residuals generally 1-2 orders of magnitude smaller. The computational results are obtained on the unit-square domain centered at $(x, y) = (-10.5, 10)$. The finest isotropic grid is 65×65 and the coarsest grid level was 5×5. Because there is no discretization error for the plane-flow equations, the initial approximations on all grids is taken as random perturbations of the exact solutions for all variables. The convergence rates are grid-independent and very fast, more than an order of magnitude per cycle.

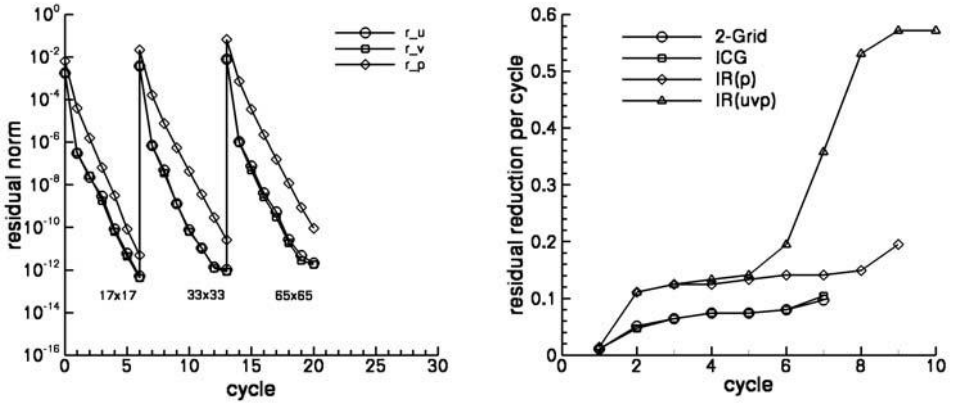


FIG. 6.3. *Residual convergence history and idealized analysis for regular stagnation regime.*

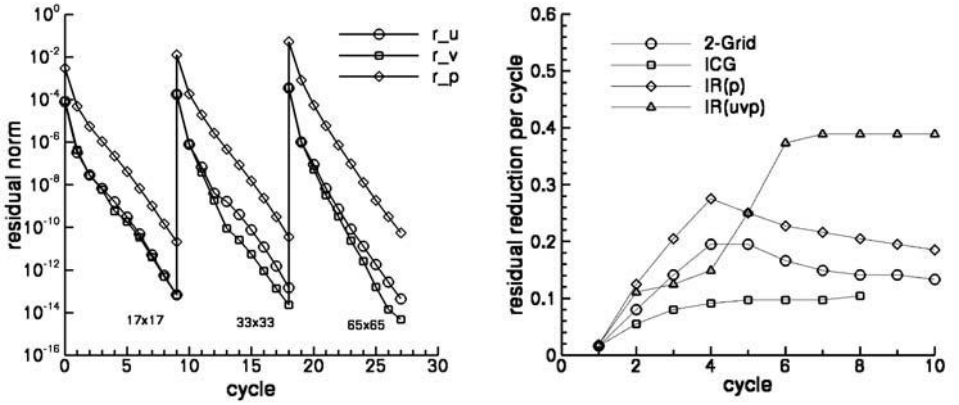


FIG. 6.4. *Residual convergence history and idealized analysis for deep stagnation regime.*

The right plot of Figure 6.3 illustrates the results of the IR and ICG iteration analysis in application to the nonlinear regular stagnation flow equations. Convergence rates of four iterations are compared on the 65×65 grid: (1) the actual two-grid cycle, (2) the ICG iterations, (3) IR(p) iterations, where explicit full-weighting averaging of algebraic error in the pressure replaces the relaxation of the pressure equation, and (4) IR(uvp) iterations where the entire relaxation sweep is replaced with full-weighting averaging of algebraic errors in u, v, and p. The convergence rates of the IR(uvp) iterations, which are significantly worse than the convergence rates of other iterations, point to some smooth (characteristic) error components that cannot be reduced in the coarse-grid correction. Fast convergence rates in IR(p) iterations confirm that the slow components are efficiently reduced in the downstream marching of the momentum equations. The reference cycle for these nonlinear computations is the same as for constant-coefficient equations in section 5.4, providing the reference convergence rates of 0.18 per cycle. The actual cycle, IR(p) and ICG iterations converge with better asymptotic rates. Close similarity between convergence rates of the actual two-grid cycle, ICG iterations, and IR(p) iterations indicate the optimal efficiency of the actual cycle.

Figure 6.4 presents convergence results for the deep stagnation flow regime. The

convergence rates are grid-independent but show a slight deterioration to approximately 0.14 per cycle asymptotically and a maximal (worst) convergence rate of approximately 0.21. The same four iterations described above have been performed for analysis of convergence in the deep stagnation regime. IR(uvp) iterations again converge slowly; convergence rates of other iterations are faster than rates of the reference cycle and very close to each other, confirming that the developed multigrid algorithm is optimally efficient.

7. Discussion. Some limitations of the classical LMF analysis that make the LMF analysis inapplicable to stagnation-flow problems have been discussed. A concept of ϵ-neighborhood has been introduced to restrict the computational domain to subdomains where there are accurate constant-coefficient approximations to the target variable-coefficient (or nonlinear) problem. With this restriction, some areas of the stagnation flow are inaccessible to the LMF analysis.

A new general approach to quantitative analysis of sophisticated multigrid solvers has been presented. The approach emphasizes the design stage of the process of developing efficient multigrid solutions. Each part of a multigrid solver is assumed to perform a certain task that is well understood and clearly formulated. The new analysis introduces IR and ICG iterations as tools to probe the complementary parts of a multigrid cycle, relaxation and coarse-grid correction. In a practical multigrid solver, one of these parts is replaced with an idealized imitation. The idealized imitation is not an optimally efficient procedure; it is rather a procedure that assuredly fulfills the task assigned to the corresponding part of the multigrid solver. The tests presented in this paper show that comparisons of the convergence rates of IR and ICG iterations with the actual two-grid cycle clearly detect deficiencies of the cycle. The convergence of the actual cycle can be improved either by adjusting parts of the cycle to better perform the assigned tasks or by redesigning the algorithm (redefining some tasks to parts of the cycle) to efficiently reduce slowly converging errors.

The IR and ICG iterations can be viewed as a numerical extension of the LMF analysis to problems where direct analytical methods are unavailable. The concept of isolating and probing parts of an algorithm by implementing idealized imitations for other parts is very general. The major disadvantage of this new analysis (shared by the LMF analysis as well) is that the analysis is not constructive; it detects problems but does not suggest remedies. However, problem detection is a big step forward for complicated applications where no other quantitative analysis methods are available. ICG and IR iterations have already been applied to some complicated nonlinear CFD problems with complex geometries on unstructured grids. The results of these applications have been encouraging in identifying the reasons for relatively slow convergence of practical solvers. In some cases, we have been able to cure the problems, e.g., IR and ICG analysis helped to develop TME solvers for stagnation-flow problems in general geometries [17]; in other cases (unstructured grid applications), we are still in search for adequate remedies.

There are many possible directions to extend the IR and ICG analysis. One can use the spectral radius of the idealized iterations as an estimate for the asymptotic convergence of the actual cycle. Such an extension may add rigorousness to the analysis, but probably limits its scope to problems of relatively low complexity, where computing the spectral radius is viable. Another possible extension is toward algebraic multigrid (AMG) [4, 10, 21, 24]. A usual assignment for an AMG relaxation is to reduce errors producing large residuals, i.e., the errors corresponding to large eigenvalues of the corresponding matrix; an AMG "coarse-grid" correction is typically

assigned to reduce algebraically smooth errors that are not necessarily geometrically smooth. The assignments extensively rely on the specific matrix; it is hardly possible to design general idealized iterations that would work for any AMG solver. However, assuming restriction, $\mathbf{R}_h^H$, and prolongation, $\mathbf{P}_h^H$, that preserve algebraically smooth functions, the ICG correction (2.6) is a good choice for general AMG applications. This choice emphasizes that errors that do not change much when transferred to the coarse grid, and then back to the fine grid, are designated as "smooth" to be handled by the coarse-grid operator. It follows that errors that do change much by the intergrid transfers are "nonsmooth" and need to be handled well by the smoother. A good idealized smoother must provide fast convergence of the reference cycle; a possible choice for IR is the composition of restriction and prolongation operators, $\mathbf{P}_H^h\mathbf{R}_h^H$. With this choice, the eigenvalues, λ_{ref}, of the iteration matrix of the reference cycle,

$$\left(\mathbf{P}_H^h\mathbf{R}_h^H\right)^{\nu_1}\left(\mathbf{E}_h - \mathbf{P}_H^h\mathbf{R}_h^H\right)\left(\mathbf{P}_H^h\mathbf{R}_h^H\right)^{\nu_2}, \tag{7.1}$$

are all in the form

$$\lambda_{\text{ref}} = \lambda^\nu(1-\lambda), \tag{7.2}$$

where λ denotes any eigenvalue of the matrix $\mathbf{P}_H^h\mathbf{R}_h^H$, and $\nu = \nu_1 + \nu_2$ is the total number of relaxations performed per one coarse-grid correction. The expression (7.2) achieves its local maximum when $\lambda = \nu/(\nu+1)$, yielding

$$\left|\lambda_{\text{ref}}\right| = \frac{\nu^\nu}{(\nu+1)^{(\nu+1)}}, \tag{7.3}$$

e.g., $\frac{1}{4}$, $\frac{4}{27}$, and $\frac{27}{256}$ for $\nu = 1$, 2, and 3, respectively, which seems very satisfactory.

Of course, $|\lambda_{\text{ref}}|$ might attain its global maximum on the "boundary," i.e., on the convex hull of the eigenvalues of $\mathbf{P}_H^h\mathbf{R}_h^H$ in the complex plane. If all eigenvalues of $\mathbf{P}_H^h\mathbf{R}_h^H$ are real (e.g., when $\mathbf{P}_H^h\mathbf{R}_h^H$ is symmetric), to find the global maximum, one only needs to consider the largest and smallest values of λ. In the particular common case where $\mathbf{R}_h^H$ is the (suitably scaled) transpose of $\mathbf{P}_H^h$, $\mathbf{P}_H^h\mathbf{R}_h^H$ is symmetric, positive semidefinite, so all of its eigenvalues are real and nonnegative. Furthermore, if $\mathbf{P}_H^h$ and $\mathbf{R}_h^H$ are properly scaled, the largest value of λ does not exceed 1, i.e., no eigenfunction is amplified by restriction followed by prolongation. Therefore, the spectral radius of the reference cycle is indeed given by (7.3). This will generally be the case, when the eigenvalues of $\mathbf{P}_H^h\mathbf{R}_h^H$ are real, except in unrealistic cases where $\mathbf{P}_H^h\mathbf{R}_h^H$ amplifies some eigenfunction significantly or changes signs of its elements. If $\mathbf{P}_H^h\mathbf{R}_h^H$ has complex eigenvalues, there may be some (often slight) deterioration of convergence of the reference cycle (which can be analyzed in a similar way). In this case, performing IR iterations which alternate between $(\mathbf{P}_H^h\mathbf{R}_h^H)^T$ and $\mathbf{P}_H^h\mathbf{R}_h^H$ may yield improved results.

REFERENCES

[1] A. Brandt, *Multigrid Solvers for Non-Elliptic and Singular-Perturbation Steady-State Problems*, unpublished manuscript, The Weizmann Institute of Science, Rehovot, Israel, 1981.

[2] A. Brandt, *Guide to multigrid development*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 220–312.

[3] A. Brandt, *Multigrid techniques:* 1984 *guide with applications to fluid dynamics*, in Lecture Notes for the Computational Fluid Dynamics, Lecture Series at the Von-Karman Institute for Fluid Dynamics, The Weizmann Institute of Science, Rehovot, Israel, 1984.

[4] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 32–56.
[5] A. Brandt, *Rigorous local mode analysis of multigrid*, in Proceedings of the 4th Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, 1989.
[6] A. Brandt, *Rigorous quantitative analysis of multigrid,* I: *Constant coefficients two-level cycle with L_2-norm*, SIAM J. Numer. Anal., 31 (1994), pp. 1695–1730.
[7] A. Brandt, *Appendix C: Recent developments in multigrid efficiency in computational fluid dynamics*, in Multigrid, U. Trottenberg, C. W. Oosterlee, and A. Schüler, Academic Press, London, 2000, pp. 573–589.
[8] A. Brandt, *General highly accurate algebraic coarsening*, Electron. Trans. Numer. Anal., 10 (2000), pp. 1–32.
[9] A. Brandt and B. Diskin, *Multigrid solvers for nonaligned sonic flows*, SIAM J. Sci. Comput., 21 (1999), pp. 473–501.
[10] A. Brandt, S. F. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, 1984, pp. 257–284.
[11] A. Brandt and I. Yavneh, *On multigrid solution of high-Reynolds incompressible entering flow*, J. Comput. Phys., 101 (1992), pp. 151–164.
[12] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[13] J. A. Désidéri and P. W. Hemker, *Convergence analysis of the defect-correction iteration for hyperbolic problems*, SIAM J. Sci. Comput., 16 (1995), pp. 88–118.
[14] B. Diskin, *Multigrid algorithm with conditional coarsening for the non-aligned sonic flow*, Electron. Trans. Numer. Anal., 6 (1997), pp. 106–119.
[15] B. Diskin, *Efficient multigrid methods for solving upwind-biased discretizations of the convection equation*, Appl. Math. Comput., 123 (2001), pp. 343–379.
[16] B. Diskin and J. L. Thomas, *Half-space analysis of the defect-correction method for Fromm discretization of convection*, SIAM J. Sci. Comput., 22 (2000), pp. 633–655.
[17] B. Diskin, J. L. Thomas, and R. E. Mineck, *Textbook Multigrid Efficiency for Leading Edge Stagnation*, NASA/TM-2004-213037, National Aeronautics and Space Administration, Langley Research Center, Hampton, VA, 2004.
[18] W. Hackbusch, *Multi-grid convergence theory*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 177–219.
[19] I. M. Llorente, M. Prieto-Matias, and B. Diskin, *An efficient parallel multigrid solver for 3-D convection and convection-diffusion problems*, Parallel Comput., 27 (2001), pp. 1715–1741.
[20] C. W. Oosterlee, F. J. Gaspar, T. Washio, and R. Wienands, *Multigrid line smoothers for higher order upwind discretizations of convection-dominated problems*, J. Comput. Phys., 139 (1998), pp. 274–307.
[21] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, Frontiers Appl. Math. 3, S. F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.
[22] D. Sidilkover, *Some approaches toward constructing optimally efficient multigrid solvers for the inviscid flow equations*, Comput. & Fluids, 28 (1999), pp. 551–571.
[23] D. Sidilkover and U. Asher, *A multigrid solver for the steady-state Navier-Stokes equations using the pressure-Poisson formulation*, Mat. Apl. Comput., 14 (1995), pp. 21–35.
[24] K. Stüben, *Appendix A: An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. W. Oosterlee, and A. Schüler, Academic Press, London, 2000, pp. 413–532.
[25] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and application*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, Berlin, 1982, pp. 1–176.
[26] J. L. Thomas, B. Diskin, and A. Brandt, *Textbook multigrid efficiency for fluid simulations*, Ann. Rev. Fluid Mech., 35 (2003), pp. 317–340.
[27] U. Trottenberg, C. W. Oosterlee, and A. Schüler, *Multigrid*, Academic Press, London, 2000.
[28] P. Wesseling, *An Introduction to Multigrid Methods*, Pure Appl. Math. (N.Y.), John Wiley & Sons, Chichester, 1992.
[29] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.
[30] I. Yavneh, *Coarse-grid correction for nonelliptic and singular perturbation problems*, SIAM J. Sci. Comput., 19 (1998), pp. 1682–1699.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 130–158
© 2005 Society for Industrial and Applied Mathematics

TWO NUMERICAL METHODS FOR RECOVERING SMALL INCLUSIONS FROM THE SCATTERING AMPLITUDE AT A FIXED FREQUENCY*

HABIB AMMARI†, EKATERINA IAKOVLEVA†, AND DOMINIQUE LESSELIER‡

Abstract. In this paper two noniterative algorithms for locating small electromagnetic inclusions from the scattering amplitude at a fixed frequency are developed. In particular, a variety of numerical results is presented to highlight their potential and their limitations.

Key words. inverse scattering problem, scattering amplitude, Helmholtz equation, electromagnetic inclusions, reconstruction, MUSIC algorithm, analytic continuation, FFT

AMS subject classifications. 35R30, 78A46

DOI. 10.1137/040612518

1. Introduction.

In this paper, we consider two-dimensional electromagnetic scattering from a collection of small electromagnetic inclusions. We suppose that there is a finite number of electromagnetic inclusions in $\mathbb{R}^2$, each of the form $z_j+\alpha B_j$, where $B_j \subset \mathbb{R}^2$ is a bounded, smooth ($\mathcal{C}^\infty$) domain containing the origin. The total collection of inclusions thus takes the form $\mathcal{I}_\alpha = \bigcup_{j=1}^m z_j + \alpha B_j$. The points $z_j \in \mathbb{R}^2$, $j = 1, \ldots, m$, which determine the location of the inclusions, are assumed to satisfy the following: $|z_j - z_l| \geq d_0 > 0$ for all $j \neq l$. The value of α, the common order of magnitude of the diameters of the inclusions, is taken as a small fraction of the wavelength of the wavefield at the operation frequency, which enables us to speak of small inclusions.

Let μ_0 and ε_0 denote the magnetic permeability and the electric permittivity of the free space; we shall assume that these are positive constants. Let μ^j and ε^j denote the permeability and the permittivity of the jth inclusion, $z_j + \alpha B_j$; these are also assumed to be positive constants. Using this notation, we introduce the piecewise constant magnetic permeability

$$\mu_\alpha(x) := \begin{cases} \mu^j & \text{in } z_j + \alpha B_j,\ j = 1, \ldots, m, \\ \mu_0 & \text{in } \mathbb{R}^2 \backslash \overline{\mathcal{I}_\alpha}. \end{cases}$$

The piecewise constant electric permittivity $\varepsilon_\alpha(x)$ is defined analogously by

$$\varepsilon_\alpha(x) := \begin{cases} \varepsilon^j & \text{in } z_j + \alpha B_j,\ j = 1, \ldots, m, \\ \varepsilon_0 & \text{in } \mathbb{R}^2 \backslash \overline{\mathcal{I}_\alpha}. \end{cases}$$

Contrasts, to be used later in these specific forms, between the electromagnetic parameters of the inclusions and those of the free space thereupon follow as

$$\gamma_\mu^j = \frac{\mu^j}{\mu_0} - 1, \quad \gamma_\varepsilon^j = \frac{\varepsilon^j}{\varepsilon_0} - 1.$$

*Received by the editors July 29, 2004; accepted for publication (in revised form) January 18, 2005; published electronically August 22, 2005.
http://www.siam.org/journals/sisc/27-1/61251.html

†Centre de Mathématiques Appliquées, CNRS UMR 7641 and Ecole Polytechnique, 91128 Palaiseau Cedex, France (ammari@cmapx.polytechnique.fr, iakov@cmapx.polytechnique.fr).

‡Département de Recherche en Électromagnétisme, Laboratoire des Signaux et Systèmes (CNRS-SUPÉLEC-UPS), Plateau de Moulon, 91192 GIF-sur-Yvette, France (lesselier@lss.supelec.fr).

Let $k = \omega\sqrt{\varepsilon_0\mu_0} > 0$ be the wavenumber, where $\omega > 0$ is a given frequency of operation, and let $u_0(x,\hat{\xi}) = e^{ik\hat{\xi}\cdot x}$, for $x \in \mathbb{R}^2$, denote a plane wave in $\mathbb{R}^2$ of direction $\hat{\xi} \in$ ¸ ^{1}, where ¸ ^{1} is the unit sphere in $\mathbb{R}^2$.

Let the electric field u_α be defined as the solution to the Helmholtz equation

$$\left(\nabla \cdot \frac{1}{\mu_\alpha}\nabla + \omega^2\varepsilon_\alpha\right) u_\alpha = 0 \quad \text{in } \mathbb{R}^2,$$

with the Rellich–Sommerfeld radiation condition, as $r = |x| \to +\infty$,

$$\lim_{r\to\infty} \sqrt{r}\left(\partial_r(u_\alpha - u_0) - ik(u_\alpha - u_0)\right) = 0.$$

Define the scattering amplitude at the fixed frequency ω, $A_\alpha(x/|x|,\hat{\xi})$, to be the function which satisfies

$$u_\alpha(x) = u_0(x) + A_\alpha\left(\frac{x}{|x|},\hat{\xi}\right)\frac{e^{ik|x|}}{\sqrt{|x|}} + o\left(\frac{1}{\sqrt{|x|}}\right)$$

as $|x| \to \infty$, uniformly with respect to $\hat{x} = x/|x|$ and $\hat{\xi}$ on S^1. From [3], the following asymptotic formula for the scattering amplitude A_α holds uniformly:

$$A_\alpha(\hat{x},\hat{\xi}) = \alpha^2\frac{k^2(1+i)}{4\sqrt{\pi k}}\sum_{j=1}^{m} e^{-ik(\hat{x}-\hat{\xi})\cdot z_j}\left(\gamma_\mu^j\,\hat{x}\,M^j\,\hat{\xi} + \gamma_\varepsilon^j|B_j|\right) + o(\alpha^2) \tag{1.1}$$

for any $\hat{\xi}$ and $\hat{x}$ on ¸ ^{1}, where the remainder $o(\alpha^2)$ is independent of the set of points $\{z_j\}_{j=1}^m$. The matrix $M^j, j = 1,\ldots,m$, is the polarization tensor associated with the domain B_j and the magnetic permeability μ^j; see [7]. The matrix M^j is given by

$$(M^j)_{kl} = \frac{\mu_0}{\mu^j}\int_{B_j} \partial_{y_k}\phi_l\,dy, \tag{1.2}$$

where ϕ_l is defined as the unique solution to

$$\begin{cases} \Delta\phi_l = 0 \quad \text{in } B_j \text{ and in } \mathbb{R}^2 \setminus \overline{B_j}, \\ \phi_l \text{ is continuous across } \partial B_j, \\ \dfrac{\mu^j}{\mu_0}(\partial_\nu\phi_l)^+ - (\partial_\nu\phi_l)^- = 0 \quad \text{on } \partial B_j, \\ \phi_l(y) - y_l \to 0 \quad \text{as } |y| \to \infty. \end{cases}$$

Here ν denotes the outward unit normal to ∂B_j; superscripts + and − indicate the limiting values as we approach ∂B_j from outside B_j and from inside B_j. Note that if the inclusion B_j is a disk, then its polarization tensor M^j has the following explicit form:

$$M^j = \frac{2\mu_0}{\mu_0 + \mu^j}|B_j|\,I_2 = m^j I_2, \tag{1.3}$$

where I_2 is the 2×2 identity matrix .

For convenience, we drop the constant $k^2(1+i)/(4\sqrt{\pi k})$ from the above expansion of A_α and denote by $A_\alpha^{(1)}$ the leading-order term in (1.1), that is,

$$A_\alpha^{(1)}(\hat{x},\hat{\xi}) = \alpha^2 \sum_{j=1}^{m} e^{-ik(\hat{x}-\hat{\xi})\cdot z_j} \left(\gamma_\mu^j\, \hat{x}\, M^j\, \hat{\xi} + \gamma_\varepsilon^j |B_j|\right). \tag{1.4}$$

Let $\{\hat{\xi}_p\}_{p=1}^n \subset\, ^1$ be a finite subset of incidence directions and let $\{\hat{x}_l\}_{l=1}^n \subset\, ^1$ be a finite subset of observation directions. Our aim in this paper is to develop two noniterative algorithms to determine the locations $z_j, j = 1,\dots,m$, and some geometric features of the inclusions from $A_\alpha^{(1)}(\hat{x}_p,\hat{\xi}_l)\big|_{\hat{x}_p=-\hat{\xi}_p}$ for $l,p = 1,\dots,n$. We first develop a robust numerical method of MUSIC (short for multiple-signal classification) type for identifying $\mathcal{I}_\alpha$. We briefly sketch the theoretical foundation of this method for solving our inverse scattering problem and provide a mathematical study of the focusing properties of the eigenvectors of the so-called multistatic response (MSR) matrix $(A_\alpha^{(1)}(\hat{x}_p,\hat{\xi}_l)\big|_{\hat{x}_p=-\hat{\xi}_p})_{l,p=1,\dots,n}$. Then we discuss the second method, which is based on analytic continuation and a fast Fourier transform (FFT) algorithm. This method has been introduced in [3] for three-dimensional scattering problems. Its adaptation to the two-dimensional case and key steps of its numerical implementation are described here. Finally we present a variety of numerical results to highlight the potential and the limitations of these two algorithms.

In concluding this introduction, we would like to refer to the survey paper by Colton, Coyle, and Monk [10] for a presentation of a number of other reconstruction algorithms in inverse scattering for the Helmholtz equation, emphasizing the linear-sampling method. This approach has some similarities to the MUSIC-type algorithm developed here. It is also worth mentioning that in the case where the data are discrete versions of the Neumann-to-Dirichlet boundary map, Brühl, Hanke, and Vogelius [6] used a corresponding asymptotic perturbation formula for small conductivity inclusions in combination with MUSIC (and linear-sampling) ideas to design a very effective algorithm for determining the locations of the inclusions. We also refer to the closely related nice works by Vogelius and Volkov [24] and Volkov [25] and to the monograph [1] for a systematic exposition of recent developments on the reconstruction of small inclusions.

2. MUSIC algorithm. MUSIC (multiple-signal classification) is generally used in signal processing problems as a method for estimating the individual frequencies of multiple-harmonic signals [23]. The MUSIC algorithm makes use of the eigenvalue structure of the so-called multistatic response (MSR) matrix. A more detailed description of this algorithm can be found in [11] and [8]. See also [21, 16, 22, 19].

Following [2] we first briefly recall how the MUSIC algorithm can be applied to the problem of locating m small inclusions with different electromagnetic parameters from those of the free space from the MSR matrix $(A_\alpha^{(1)}(\hat{x}_p,\hat{\xi}_l)\big|_{\hat{x}_p=-\hat{\xi}_p})_{l,p=1,\dots,n}$. Then we present a complete mathematical study of the focusing properties of the eigenvectors of this matrix. To the best of our knowledge, these qualitative results are new. The eigenvectors corresponding to significant eigenvalues span some kind of signal subspace in the sense that they contain nearly all the information about the inclusions which can be extracted from the MSR matrix. The others span some kind of noise subspace. The aim of the MUSIC type of algorithm is to use the eigensystem analysis of the MSR matrix to determinine the location and recover some geometric features (namely, the polarization tensors defined by (1.2)) of the small inclusions from the signal space.

We consider separately the following three cases, which we believe to be representative of the pros and cons of our MUSIC-type reconstruction:

(1) $\gamma_\mu^j = 0$ for all $j = 1, \dots, m$.
(2) $\gamma_\varepsilon^j = 0$ for all $j = 1, \dots, m$.
(3) $\gamma_\mu^j \neq 0$ and $\gamma_\varepsilon^j \neq 0$ for all $j = 1, \dots, m$.

2.1. The first case: Dielectric inclusions. We suppose $n \geq m$ and define the MSR matrix $\mathrm{A} = (\mathrm{A}_{pl})_{p,l=1}^n \in \mathbb{C}^{n\times n}$ by

$$\mathrm{A}_{pl} = A_\alpha^{(1)}(\hat{x}_p, \hat{\xi}_l)\big|_{\hat{x}_p = -\hat{\xi}_p} = \sum_{j=1}^m \tau_j e^{ik(\hat{\xi}_p + \hat{\xi}_l)\cdot z_j}$$

for $p, l = 1, \dots, n$, where $\tau_j = \alpha^2 \gamma_\varepsilon^j |B_j|$. Defining the matrices $\mathrm{T} \in \mathbb{R}^{m\times m}$ and $\mathrm{S} = [\mathrm{s}_1 \dots \mathrm{s}_m] \in \mathbb{C}^{n\times m}$ by

$$\mathrm{T} = \mathrm{diag}(\tau_j) \quad \text{and} \qquad \mathrm{s}_j = (e^{ik\hat{\xi}_1\cdot z_j}, \dots, e^{ik\hat{\xi}_n\cdot z_j})^T \text{ for } j = 1, \dots, m,$$

we then observe that A admits the decomposition

$$\mathrm{A} = \mathrm{STS}^T, \tag{2.1}$$

where $\mathrm{S}^T \in \mathbb{C}^{m\times n}$ denotes the transpose of S. Using (2.1), we immediately see that A is symmetric and $\bar{\mathrm{A}}^T = \bar{\mathrm{A}}$, where $\bar{\mathrm{A}}$ is the complex conjugate of A.

Throughout this section, we consider $\{\hat{\xi}_p\}_{p\in\mathbb{N}} \subset S^1$ to be a countable set of directions with the property that any analytic function which vanishes in $\hat{\xi}_p$ for all $p \in \mathbb{N}$ vanishes identically on S^1. This technical assumption is necessary to rigorously prove Propositions 2.1, 2.2, and 2.3. See, for example, [16].

Now a standard argument from linear algebra yields that if $n \geq m$ and if the matrix S has maximal rank m, then the ranges $\mathcal{R}(\mathrm{S})$ and $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}})$ coincide. Thus

$$\mathcal{R}(\mathrm{A}\bar{\mathrm{A}}) = \mathcal{R}(\mathrm{S}) = \mathrm{span}\{\mathrm{s}_1, \dots, \mathrm{s}_m\}.$$

But since the locations z_j are well separated, then using the property that any analytic function which vanishes in $\hat{\xi}_p$ for all $p \in \mathbb{N}$ vanishes identically, it can be immediately seen that there exists $n_0 \geq m$ such that for any $n \geq n_0$ the matrix S has maximal rank m.

Let $a \in \mathbb{R}\backslash\{0\}$; for any point $z \in \mathbb{R}^2$ we define the vector $g_{z,a} \in \mathbb{C}^n$ by

$$g_{z,a} = (ae^{ik\hat{\xi}_1\cdot z}, \dots, ae^{ik\hat{\xi}_n\cdot z})^T. \tag{2.2}$$

We note that $g_{z_1,a}, \dots, g_{z_m,a}$ are the columns of the matrix aS.

The main tool for the identification of the locations $z_j, j = 1, \dots, m$, is the following [16].

PROPOSITION 2.1. [illegible] $n_0 \in \mathbb{N}$ [illegible] $n \geq n_0$ [illegible] $g_{z,a}$ [illegible] (2.2), [illegible] $\mathcal{R}(\mathrm{S})$ [illegible] $z \in \{z_1, \dots, z_m\}$

Next let the singular value decomposition (SVD) of the matrix A be defined by $\mathrm{A} = \mathrm{V}\Sigma\overline{\mathrm{W}}^T$, where $\mathrm{V}, \mathrm{W} \in \mathbb{C}^{n\times n}$ are unitary and $\Sigma = \mathrm{diag}\{\sigma_1, \sigma_2, \dots, \sigma_n\}$ is a real nonnegative diagonal matrix, where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m > 0$ and σ_i, $i = m+1, \dots, n$,

are all zero (or [illegible]).[1] The numbers $\{\sigma_i\}$ are the nonnegative square roots of the eigenvalues of $\mathrm{A}\bar{\mathrm{A}}^T = \mathrm{A}\bar{\mathrm{A}}$. The columns of V are the eigenvectors of $\mathrm{A}\bar{\mathrm{A}}$ and the columns of W are the eigenvectors of $\bar{\mathrm{A}}\mathrm{A}$ (arranged in the same order as the corresponding eigenvalues σ_i^2). We note that the SVD of A can be written in the form $\mathrm{A} = \mathrm{V}\Sigma\mathrm{V}^{\mathrm{T}}$ (which is Takagi's factorization or an SVD for symmetric matrices in which $\mathrm{V} = \overline{\mathrm{W}}$). The first m columns of V, $\{v_1, v_2, \ldots, v_m\}$, provide an orthonormal basis for the space of A which is denoted by V_S, and the rest of the matrix V, $\{v_{m+1}, v_{m+2}, \ldots, v_n\}$, provides a basis for the left null (or the [illegible]) space of A, denoted by V_N. Then the best rank approximation for A is $(\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$ with error $E = \mathrm{A} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$.

From Proposition 2.1 we have that a test point z coincides with one of the positions z_j if and only if $g_{z,a} \in \mathcal{R}(\mathrm{A}\bar{\mathrm{A}})$ or, equivalently, if $\mathrm{P}g_z = 0$, where $\mathrm{P} = \mathrm{I} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)$ is the orthogonal projection onto the null (or the [illegible]) space of A. Thus we can form an image of z_j, $j = 1, \ldots, m$, by plotting, at each point z, the quantity

$$W_a(z) = \frac{1}{||\mathrm{P}g_{z,a}||}.$$

The resulting plot will have large peaks at the positions z_j, $j = 1, \ldots, m$.

2.1.1. SVD of the MSR matrix. By virtue of (2.1), the MSR matrix A can be written as

$$\mathrm{A} = \mathrm{STS}^T = \sum_{j=1}^{m} \tau_j \mathrm{s}_j \mathrm{s}_j^T. \tag{2.3}$$

We first consider m uncorrelated inclusions and introduce

$$\mathrm{A}_j = \tau_j \mathrm{s}_j \mathrm{s}_j^T, \quad j = 1, \ldots, m. \tag{2.4}$$

The matrix A_j therefore satisfies

$$\mathrm{A}_j\bar{\mathrm{A}}_j = \tau_j^2 ||\bar{\mathrm{s}}_j||^2 \mathrm{s}_j\bar{\mathrm{s}}_j^T, \quad \bar{\mathrm{A}}_j\mathrm{A}_j = \tau_j^2 ||\mathrm{s}_j||^2 \bar{\mathrm{s}}_j\mathrm{s}_j^T$$

for $j = 1, \ldots, m$, where $||\mathrm{s}_j||^2 = \bar{\mathrm{s}}_j^T\mathrm{s}_j = n$. The eigenvectors v_j and w_j of the matrices $\mathrm{A}_j\bar{\mathrm{A}}_j$ and $\bar{\mathrm{A}}_j\mathrm{A}_j$, having nonzero singular values σ_j, are given by

$$v_j = \frac{\mathrm{s}_j}{||\mathrm{s}_j||} = \frac{\mathrm{s}_j}{\sqrt{n}}, \quad w_j = \frac{\bar{\mathrm{s}}_j}{||\bar{\mathrm{s}}_j||} = \frac{\bar{\mathrm{s}}_j}{\sqrt{n}}, \tag{2.5}$$

and

$$\sigma_j = n|\tau_j|. \tag{2.6}$$

Now, let us turn to the matrix A, defined by (2.3), and consider the equations

$$\mathrm{A}\bar{\mathrm{A}}\hat{v}_j = \hat{\sigma}_j^2\hat{v}_j, \quad \bar{\mathrm{A}}\mathrm{A}\hat{w}_j = \hat{\sigma}_j^2\hat{w}_j. \tag{2.7}$$

Since $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}}) = \mathrm{span}\{\mathrm{s}_1, \ldots, \mathrm{s}_m\} = \mathrm{span}\{v_1, \ldots, v_m\}$, then the singular vector $\hat{v}_j$ are linear combinations of the array vectors v_j, while $\hat{w}_j$ are linear combinations of the array vectors $w_j = \bar{v}_j$, i.e.,

$$\hat{v}_j = \sum_{l=1}^{m} a_{jl} v_l, \quad \hat{w}_j = \sum_{l=1}^{m} \bar{a}_{jl} \bar{v}_l, \tag{2.8}$$

[1] That is, if A is the noisy MSR matrix.

where $a_j = (a_{j1}, \ldots, a_{jm})^T \in \mathbb{C}^m$, with $||a_j|| = 1$ as a normalization.

Introduce

$$H_{jl} = \bar{v}_j^T v_l, \quad j, l = 1, \ldots, m. \tag{2.9}$$

If we substitute (2.3) and (2.8) into (2.7), we obtain that

$$\mathrm{L}\bar{\mathrm{L}}a_j = \hat{\sigma}_j^2 a_j, \quad \bar{\mathrm{L}}\mathrm{L}\bar{a}_j = \hat{\sigma}_j^2 \bar{a}_j, \tag{2.10}$$

where matrix $\mathrm{L} \in \mathbb{C}^{m\times m}$ is given by $\mathrm{L}_{pq} = n\tau_p \bar{H}_{pq}$, $p, q = 1, \ldots, m$.

Following [11, 12], we first consider the case where the quantity H_{jl} is approximately $\delta_{jl}, j, l = 1, \ldots, m$. Here δ_{jl} is the Kronecker delta function. From (2.10) we get $\hat{\sigma}_j \approx n|\tau_j|$ and $a_{jl} \approx \delta_{jl}$. It then follows that the singular vectors, having nonzero singular values, are approximately given by $\hat{v}_j \approx v_j$, $\hat{w}_j \approx w_j$ for $j = 1, \ldots, m$. From the discussion above, we conclude that A has m nonzero singular values, each corresponds to the nonzero singular value of one of the matrices A_j. Moreover, the signal space of A is the space spanned by the eigenvectors of $\mathrm{A}_j, j = 1, \ldots, m$.

In the general case $(H_{pq})_{p,q=1}^n$ will not be a diagonal matrix and the spectral analysis of A does not simplify from its general form given in (2.3). In this case, from (2.9) we have

$$H_{jl} = \bar{v}_j^T v_l = \frac{1}{n}\sum_{p=1}^{n} e^{ik(z_l - z_j)\cdot\hat{\xi}_p} \approx J_0(k|z_j - z_l|), \tag{2.11}$$

$j, l = 1, \ldots, m$, for sufficiently large n. Here, J_0 is the Bessel function of the first kind and order zero. It is easy to see from the above approximation that H_{jl} is always nonzero. However, if $k|z_l - z_j|$ is large enough, then H_{jl} is small.

Using (2.11) the matrix L is approximated by a real matrix and (2.10) can be reduced to

$$\mathrm{L}a_j = \lambda_j a_j, \quad j = 1, \ldots, m, \tag{2.12}$$

where λ_j is the eigenvalue of the matrix L associated with the eigenvector $a_j \in \mathbb{R}^m$, and $|\lambda_j| = \hat{\sigma}_j$.

Let $\tau_j = |\tau_j|e^{2i\phi_j}$, $j = 1, \ldots, m$. Since in our case the constants $\tau_j \in \mathbb{R}$, then $\phi_j = 0$ if τ_j is positive and $\phi_j = \pi/2$ if τ_j is negative. The matrix L admits the decomposition $\mathrm{L} = \mathrm{D}\tilde{\mathrm{L}}\mathrm{D}^{-1}$, where $\mathrm{D} = \mathrm{diag}(\sqrt{|\tau_1|}e^{i\phi_1}, \ldots, \sqrt{|\tau_m|}e^{i\phi_m})$ and $\tilde{\mathrm{L}} \in \mathbb{C}^{m\times m}$ is given by

$$\tilde{\mathrm{L}}_{pq} = n\sqrt{|\tau_p||\tau_q|}e^{i(\phi_p+\phi_q)}H_{pq}, \quad p, q = 1, \ldots, m.$$

Therefore, the matrices L and $\tilde{\mathrm{L}}$ are similar, and thus they have the same eigenvalues, counted with their multiplicity.

We are now ready to investigate the singular values and eigenvectors of the matrix L, which is similar to $\tilde{\mathrm{L}}$ written as $\tilde{\mathrm{L}} = \Lambda + \mathrm{E}$, where $\Lambda = \mathrm{diag}(n\tau_1, \ldots, n\tau_m)$ and E is some (symmetric) perturbation matrix.

Let us assume that the values $|\tau_j|$ are arranged in decreasing order $|\tau_1| \geq \cdots \geq |\tau_m|$. Let

$$R_p(\tilde{\mathrm{L}}) = \sum_{\substack{q=1\\ q\neq p}}^{m} \frac{\sqrt{|\tau_p||\tau_q|}}{|\tau_1|}|H_{pq}|, \quad p, q = 1, \ldots, m. \tag{2.13}$$

Then by Geršgorin's theorem [15] all eigenvalues of $\tilde{\mathrm{L}}$ (which are in our case real) are located in the union of m intervals

$$\bigcup_{p=1}^{m}\left\{x'=\frac{x}{n}\in\mathbb{R}:\left|\frac{x'}{|\tau_1|}-\frac{\tau_p}{|\tau_1|}\right|\le R_p(\tilde{\mathrm{L}})\right\}\equiv G(\tilde{\mathrm{L}}). \tag{2.14}$$

Furthermore, if a union of p of the m intervals forms a connected region that is disjoint from all the remaining $m-p$ intervals, then there are precisely p eigenvalues of $\tilde{\mathrm{L}}$ in this region. Now, from (2.13) we conclude that if

$$|\tau_1|\gg\cdots\gg|\tau_m|, \tag{2.15}$$

then all Geršgorin's intervals are disjoint, $\lambda_p\to n\tau_p$, or, equivalently, $\hat{\sigma}_p\to\sigma_p=n|\tau_p|$, and the eigenvector $\hat{v}_p$ is approximately v_p for $p=1,\dots,m$. The eigenvectors of the MSR matrix then produce selective focusing onto the dielectric inclusions.

On the other hand, if $\tau_1=\cdots=\tau_m=\tau$, then all eigenvalues are located in the interval

$$\left\{x'=\frac{x}{n}\in\mathbb{R}:|x'-\tau|\le|\tau|\sum_{\substack{q=1\\q\neq p}}^{m}|H_{pq}|\right\},\quad p=1,\dots,m.$$

Let $\lambda_1(\mathrm{E})\ge\cdots\ge\lambda_m(\mathrm{E})$ are the ordered eigenvalues of E; then $\lambda_p=n\tau+n\tau\lambda_p(\mathrm{E})$ for all $p=1,\dots,m$. The maximal error is $n|\tau|\rho(\mathrm{E})$, where $\rho(\mathrm{E})$ is the spectral radius of E, which is the radius of the smallest disk centered at the origin in the complex plane that includes all the eigenvalues of E; see [15].

The MSR matrix may admit in this situation nonsimple eigenvalues. The diagonalization of the latter cannot choose the selective focusing densities among all their linear combinations which compose the corresponding eigenspace. The nice focusing properties will disappear, at least for some groups of eigenvalues. See [18, 14].

2.2. The second case: Permeable inclusions. In this case the MSR matrix $\mathrm{A}=(\mathrm{A}_{pl})_{p,l=1}^{n}\in\mathbb{C}^{n\times n}$ is defined by

$$\mathrm{A}_{pl}=A_{\alpha}^{(1)}(\hat{x}_p,\hat{\xi}_l)\big|_{\hat{x}_p=-\hat{\xi}_p}=-\sum_{j=1}^{m}e^{ik(\hat{\xi}_p+\hat{\xi}_l)\cdot z_j}\hat{\xi}_p\cdot\beta_j\hat{\xi}_l$$

for $p,l=1,\dots,n$, where $\beta_j=\alpha^2\gamma_\mu^j\mathrm{M}^j\in\mathbb{R}^{2\times2}$.

We define the matrix $\mathrm{D}=[\mathrm{d}_1\dots\mathrm{d}_{2m}]\in\mathbb{C}^{n\times2m}$ by

$$\mathrm{d}_{2(j-1)+p}=(\mathrm{e}_p\cdot\hat{\xi}_1e^{ik\hat{\xi}_1\cdot z_j},\dots,\mathrm{e}_p\cdot\hat{\xi}_ne^{ik\hat{\xi}_n\cdot z_j})^T,\quad j=1,\dots,m,p=1,2,$$

where $\{\mathrm{e}_p\}_{p=1,2}$ denotes an orthonormal basis in $\mathbb{R}^2$.

We observe that A admits the decomposition

$$\mathrm{A}=\mathrm{DBD}^T, \tag{2.16}$$

where $\mathrm{B}=\mathrm{diag}(-\beta_j)\in\mathbb{R}^{2m\times2m}$. It then follows immediately that A is symmetric.

Analogously to case 1, if $n\ge2m$ and if the locations z_j are well separated, then the matrix D has maximal rank $2m$, and the ranges $\mathcal{R}(\mathrm{D})$ and $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}})$ coincide. Thus

$$\mathcal{R}(\mathrm{A}\bar{\mathrm{A}})=\mathcal{R}(\mathrm{D})=\mathrm{span}\{\mathrm{d}_1,\dots,\mathrm{d}_{2m}\}.$$

Let $b \in \mathbb{R}^2 \backslash \{0\}$; for any point $z \in \mathbb{R}^2$ we define the vector $g_{z,b} \in \mathbb{C}^n$ by

$$g_{z,b} = \left(b \cdot \hat{\xi}_1 e^{ik\hat{\xi}_1 \cdot z}, \ldots, b \cdot \hat{\xi}_n e^{ik\hat{\xi}_n \cdot z} \right)^T. \tag{2.17}$$

We note that $g_{z_j,b}$ is the linear combination of the vector columns d_{2j-1} and d_{2j}, $j = 1, \ldots, m$, of the matrix D. The following proposition holds (see [2]).

PROPOSITION 2.2. [illegible] $b \in \mathbb{R}^2 \backslash \{0\}$ [illegible] $g_{z,b} \in \mathbb{C}^n$ [illegible] (2.17) [illegible] [illegible] $n_0 \in \mathbb{N}$ [illegible] $n \geq n_0$ [illegible] $g_{z,b} \in \mathcal{R}(\mathrm{D})$ [illegible] $z \in \{z_1, \ldots, z_m\}$

Once again let the SVD of the matrix A be defined by $\mathrm{A} = \mathrm{V}\Sigma\overline{\mathrm{W}}^T$. Since the rank of A is $2m$, then the first $2m$ columns of V, $\{v_1, v_2, \ldots, v_{2m}\}$, provide an orthonormal basis for the space of A which is denoted by V_S, and the rest of the matrix V, $\{v_{2m+1}, v_{2m+2}, \ldots, v_n\}$, provides a basis for left null space of A, denoted by V_N. Then the best rank approximation for A is $(\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$ with error $E = \mathrm{A} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$.

From Proposition 2.2 we have that a test point z coincides with one of the positions z_j if and only if $g_{z,b} \in \mathcal{R}(\mathrm{A}\bar{\mathrm{A}})$ or, equivalently, if $\mathrm{P}g_{z,b} = 0$, where $\mathrm{P} = \mathrm{I} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)$ is the orthogonal projection onto the null space of A.

Thus we can form an image of $z_j, j = 1, \ldots, m$, by plotting, at each point z, the quantity

$$W_b(z) = \frac{1}{||\mathrm{P}g_{z,b}||}.$$

The resulting plot will have large peaks at the positions z_j, $j = 1, \ldots, m$.

2.2.1. SVD of the MSR matrix. Through this subsection we assume for simplicity that all the matrices M^j are diagonal. This holds when all the B_j are disks or ellipses with the same axes. Then let the matrix $\beta_j = \alpha^2\gamma_\mu^j\mathrm{M}^j \in \mathbb{R}^{2\times 2}$ be represented as $\beta_j = \mathrm{diag}(\beta_{1,j}, \beta_{2,j})$. By virtue of (2.16) the MSR matrix A can be written as

$$\mathrm{A} = \mathrm{DBD}^T = \sum_{j=1}^{m} \beta_{1,j}\mathrm{d}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}\mathrm{d}_{2j}\mathrm{d}_{2j}^T. \tag{2.18}$$

Analogously to the first case, we first consider m independent problems, each corresponding to one inclusion, and introduce

$$\mathrm{A}_j = \beta_{1,j}\mathrm{d}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}\mathrm{d}_{2j}\mathrm{d}_{2j}^T, \quad j = 1, \ldots, m. \tag{2.19}$$

Let $\hat{\xi}_t = (\hat{\xi}_{1,t}, \hat{\xi}_{2,t}) = (\cos\theta_t, \sin\theta_t)$, where $\theta_t = \frac{2\pi(t-1)}{n}$ is the angle of observation, $t = 1, \ldots, n$. From the definition of the vector columns d of the matrix D we have

$$\bar{\mathrm{d}}_{2j-1}^T\mathrm{d}_{2j-1} = \sum_{t=1}^{n} \hat{\xi}_{1,t}^2 = \sum_{t=1}^{n} \cos^2\theta_t = \frac{n}{2}, \tag{2.20}$$

$$\bar{\mathrm{d}}_{2j}^T\mathrm{d}_{2j} = \sum_{t=1}^{n} \hat{\xi}_{2,t}^2 = \sum_{t=1}^{n} \sin^2\theta_t = \frac{n}{2}, \tag{2.21}$$

$$\bar{\mathrm{d}}_{2j-1}^T\mathrm{d}_{2j} = \sum_{t=1}^{n} \hat{\xi}_{1,t}\hat{\xi}_{2,t} = \sum_{t=1}^{n} \cos\theta_t \sin\theta_t = 0. \tag{2.22}$$

Then from (2.19), (2.20), (2.21), and (2.22) we obtain that

$$\begin{aligned}\mathrm{A}_j\bar{\mathrm{A}}_j &= \beta_{1,j}^2||\bar{\mathrm{d}}_{2j-1}||^2\mathrm{d}_{2j-1}\bar{\mathrm{d}}_{2j-1}^T + \beta_{2,j}^2||\bar{\mathrm{d}}_{2j}||^2\mathrm{d}_{2j}\bar{\mathrm{d}}_{2j}^T,\\ \bar{\mathrm{A}}_j\mathrm{A}_j &= \beta_{1,j}^2||\mathrm{d}_{2j-1}||^2\bar{\mathrm{d}}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}^2||\mathrm{d}_{2j}||^2\bar{\mathrm{d}}_{2j}\mathrm{d}_{2j}^T\end{aligned}$$

for $j = 1, \dots, m$. It then follows that the eigenvectors v_{2j-1}, v_{2j} and w_{2j-1}, w_{2j} of the matrices $\mathrm{A}_j\bar{\mathrm{A}}_j$ and $\bar{\mathrm{A}}_j\mathrm{A}_j$, having nonzero singular values σ_j, are given by

$$\begin{aligned} v_{2(j-1)+p} &= \frac{\mathrm{d}_{2(j-1)+p}}{||\mathrm{d}_{2(j-1)+p}||} = \sqrt{\frac{2}{n}}\,\mathrm{d}_{2(j-1)+p},\\ w_{2(j-1)+p} &= \frac{\bar{\mathrm{d}}_{2(j-1)+p}}{||\bar{\mathrm{d}}_{2(j-1)+p}||} = \sqrt{\frac{2}{n}}\,\bar{\mathrm{d}}_{2(j-1)+p},\\ \sigma_{2(j-1)+p} &= \frac{n}{2}|\beta_{p,j}| \quad \text{for } p = 1,2 \text{ and } j = 1, \dots, m.\end{aligned} \tag{2.23}$$

Now, let us turn to matrix A, defined by (2.18), and consider the equations

$$\mathrm{A}\bar{\mathrm{A}}\hat{v}_{l'} = \hat{\sigma}_{l'}^2\hat{v}_{l'}, \quad \bar{\mathrm{A}}\mathrm{A}\hat{w}_{l'} = \hat{\sigma}_{l'}^2\hat{w}_{l'}. \tag{2.24}$$

Since $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}}) = \mathrm{span}\{\mathrm{d}_1, \dots, \mathrm{d}_{2m}\} = \mathrm{span}\{v_1, \dots, v_{2m}\}$, then the singular vectors $\hat{v}_{l'}$ are linear combinations of the array vectors $v_{l'}$, while the vectors $\hat{w}_{l'}$ are linear combinations of the array vectors $w_{l'} = \bar{v}_{l'}$ for $l' = 1, \dots, 2m$. Let

$$H_{l',q'} = \bar{v}_{l'}^T v_{q'}, \quad l', q' = 1, \dots, 2m. \tag{2.25}$$

Define

$$\begin{aligned} l' &= 2(l-1) + p,\\ q' &= 2(q-1) + p', l, q = 1, \dots, m, \;\; p, p' = 1, 2.\end{aligned}$$

Using the polar coordinates $z_q - z_l = |z_l - z_q|(\cos\alpha_{lq}, \sin\alpha_{lq})$, we compute

$$H_{l',q'} = \frac{\sum_{t=1}^n \hat{\xi}_{p,t}\,\hat{\xi}_{p',t}\,e^{ik|z_l - z_q|\cos(\theta_t - \alpha_{lq})}}{\sqrt{\sum_{t=1}^n \hat{\xi}_{p,t}^2}\sqrt{\sum_{t=1}^n \hat{\xi}_{p',t}^2}}.$$

For $l = q$ we compute $H_{l',q'}$ from (2.20), (2.21), and (2.22), while for $l \neq q$ we have

$$\begin{aligned} &H_{l',q'} \approx J_0(k|z_l - z_q|) - \cos(2\alpha_{lq})J_2(k|z_l - z_q|) && \text{if } p = p' = 1,\\ &H_{l',q'} \approx J_0(k|z_l - z_q|) + \cos(2\alpha_{lq})J_2(k|z_l - z_q|) && \text{if } p = p' = 2,\\ &H_{l',q'} \approx \sin(2\alpha_{lq})J_2(k|z_l - z_q|) && \text{if } p \neq p'.\end{aligned} \tag{2.26}$$

Here, J_0 and J_2 are the Bessel functions of the first kind and of orders 0 and 2, respectively.

Analogously to the first case, we first consider the case where $H_{l',q'} \approx \delta_{l',q'}$, $l', q' = 1, \dots, 2m$. In this situation, we obtain $\hat{\sigma}_{l'} = \hat{\sigma}_{2(j-1)+p} \approx \frac{n}{2}|\beta_{p,j}|$. Moreover, the singular vectors, having nonzero singular values, are given by $\hat{v}_{l'} \approx v_{l'}$, $\hat{w}_{l'} \approx w_{l'}$ for $l' = 1, \dots, 2m$.

We also obtain that the $2m$ nonzero singular values of A are those of m uncorrelated matrices A_j. Similarly, the signal space of A is spanned by the eigenvalues of

$A_j, j = 1, \ldots, m$. Note that for each inclusion two singular values correspond. A natural question would be whether one could localize the inclusions independently of each other. In the general case the answer is no. However, if all the inclusions are disks, then, because $\beta_{1,j} = \beta_{2,j}, j = 1, \ldots, m$, the two singular values corresponding to the same inclusion coincide, and thus it would be possible to localize them independently.

Finally, the following conclusion can be made if all the quantities $\gamma_\mu^j M^j$ are different enough. If $|\beta_{1,1}| \geq |\beta_{2,1}| \gg |\beta_{1,2}| \geq |\beta_{2,2}| \gg \cdots \gg |\beta_{1,m}| \geq |\beta_{2,m}|$, then $\hat{\sigma}_{2(j-1)+p}$ is close to $\sigma_{2(j-1)+p}$, $j = 1, \ldots, m$, $p = 1, 2$, and the two first singular values correspond to the inclusion with the highest $\gamma_\mu^j M^j$. Following [6], the quantities $\gamma_\mu^j M^j$ can be recovered from the singular values of the MSR matrix once the positions of the inclusions have been determined.

2.3. The third case: Dielectric and permeable inclusions. In this case the MSR matrix $\mathrm{A} = (\mathrm{A}_{pl})_{p,l=1}^n \in \mathbb{C}^{n\times n}$ is defined by

$$\begin{aligned}\mathrm{A}_{pl} &= A_\alpha^{(1)}(\hat{x}_p, \hat{\xi}_l)\big|_{\hat{x}_p = -\hat{\xi}_p} \\ &= -\sum_{j=1}^m e^{ik(\hat{\xi}_p+\hat{\xi}_l)z_j}\hat{\xi}_p\beta_j\hat{\xi}_l + \sum_{j=1}^m \tau_j e^{ik(\hat{\xi}_p+\hat{\xi}_l)\cdot z_j}\end{aligned}$$

for $p, l = 1, \ldots, n$, where $\beta_j = \alpha^2\gamma_\mu^j \mathrm{M}^j \neq 0$ and $\tau_j = \alpha^2\gamma_\varepsilon^j |B_j| \neq 0$ for $j = 1, \ldots, m$.

By virtue of (2.1) and (2.16) we have

$$\mathrm{A} = \mathrm{DBD}^T + \mathrm{STS}^T = \mathrm{HEH}^T,$$

where $\mathrm{H} = [\mathrm{d}_1 \ldots \mathrm{d}_{2m} | \mathrm{s}_1 \ldots \mathrm{s}_m] \in \mathbb{C}^{n\times 3m}$ and $\mathrm{E} = \mathrm{diag}\{\mathrm{B}, \mathrm{T}\} \in \mathbb{C}^{3m\times 3m}$.

We assume that for any set of m distinct points $z_1, \ldots, z_m$, the $3m$ vectors $\{\mathrm{d}_{2j-1}, \mathrm{d}_{2j}, \mathrm{s}_j,\ j = 1, \ldots, m\}$ are linearly independent.

As previously, observe that if $n \geq 3m$ and if the locations z_j are well separated, then the matrix H has maximal rank $3m$, and the ranges of $\mathcal{R}(\mathrm{H})$ and $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}})$ coincide. Thus

$$\mathcal{R}(\mathrm{A}\bar{\mathrm{A}}) = \mathrm{span}\{\mathrm{d}_{2j-1}, \mathrm{d}_{2j}, \mathrm{s}_j,\ j = 1, \ldots, m\}.$$

Let $c \in \mathbb{R}^3\backslash\{0\}$; for any point $z \in \mathbb{R}^2$ we define the vector $g_{z,c} \in \mathbb{C}^n$ by

$$g_{z,c} = \left(c \cdot h_{\hat{\xi}_1} e^{ik\hat{\xi}_1\cdot z}, \ldots, c \cdot h_{\hat{\xi}_n} e^{ik\hat{\xi}_n\cdot z}\right)^T, \tag{2.27}$$

where $h_{\hat{\xi}_p} = (\mathrm{e}_1 \cdot \hat{\xi}_p, \mathrm{e}_2 \cdot \hat{\xi}_p, 1)^T \in \mathbb{R}^3$, $p = 1, \ldots, n$. We note that $g_{z_j,c}$ is a linear combination of the vector columns d_{2j-1}, d_{2j}, and s_j, $j = 1, \ldots, m$, of the matrix H.

Following arguments similar to those for cases 1 and 2, we can show the following result [2].

PROPOSITION 2.3. [illegible] $c \in \mathbb{R}^3\backslash\{0\}$ [illegible] $g_{z,c} \in \mathbb{C}^n$ [illegible] (2.27) [illegible] [illegible] $n_0 \in \mathbb{N}$ [illegible] $n \geq n_0$ [illegible] $g_{z,c} \in \mathcal{R}(\mathrm{H})$ [illegible] $z \in \{z_1, \ldots, z_m\}$

Analogously to cases 1 and 2, let the SVD of matrix A be defined by $\mathrm{A} = \mathrm{V}\Sigma\overline{\mathrm{W}}^T$. Since the rank of A is $3m$, then the first $3m$ columns of V, $\{v_1, \ldots, v_{3m}\}$, provide an orthonormal basis for the space of A which is denoted by V_S, and the rest of the matrix V, $\{v_{3m+1}, v_{3m+2}, \ldots, v_n\}$, provides the basis for left null space of A, denoted by V_N. Then the best rank approximation for A is $(\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$ with error $E = \mathrm{A} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)\mathrm{A}$.

From Proposition 2.3 we have that a test point z coincides with one of the positions z_j if and only if $\mathrm{P}g_{z,c} = 0$, where $\mathrm{P} = \mathrm{I} - (\mathrm{V}_S\overline{\mathrm{V}}_S^T)$ is the orthogonal projection onto the null space of A.

Thus we can form an image of $z_j, j = 1, \ldots, m$, by plotting, at each point z, the quantity

$$W_c(z) = \frac{1}{||\mathrm{P}g_{z,c}||}.$$

The resulting plot will have large peaks at the locations of $z_j, j = 1, \ldots, m$.

2.3.1. The SVD of the MSR matrix. This case is the superposition of the first and second cases. As in the previous case, we assume for simplicity that all the matrices M^j are diagonal. The MSR matrix A can be written as

$$\mathrm{A} = \sum_{j=1}^{m} \beta_{1,j}\mathrm{d}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}\mathrm{d}_{2j}\mathrm{d}_{2j}^T + \tau_j \mathrm{s}_j \mathrm{s}_j^T. \tag{2.28}$$

Introduce

$$\mathrm{A}_j = \beta_{1,j}\mathrm{d}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}\mathrm{d}_{2j}\mathrm{d}_{2j}^T + \tau_j \mathrm{s}_j \mathrm{s}_j^T, \quad j = 1, \ldots, m. \tag{2.29}$$

Set again $\hat{\xi}_t = (\hat{\xi}_{1,t}, \hat{\xi}_{2,t}) = (\cos\theta_t, \sin\theta_t)$, where $\theta_t = \frac{2\pi(t-1)}{n}$ is the angle of observation, $t = 1, \ldots, n$; then

$$\bar{\mathrm{d}}_{2j-1}^T \mathrm{s}_j = \sum_{t=1}^{n} \hat{\xi}_{1,t} = \sum_{t=1}^{n} \cos\theta_t = 0, \tag{2.30}$$

$$\bar{\mathrm{d}}_{2j}^T \mathrm{s}_j = \sum_{t=1}^{n} \hat{\xi}_{2,t} = \sum_{t=1}^{n} \sin\theta_t = 0. \tag{2.31}$$

From (2.29), (2.20), (2.21), (2.22), (2.30), and (2.31) it then follows that

$$\mathrm{A}_j\bar{\mathrm{A}}_j = \beta_{1,j}^2||\bar{\mathrm{d}}_{2j-1}||^2\mathrm{d}_{2j-1}\bar{\mathrm{d}}_{2j-1}^T + \beta_{2,j}^2||\bar{\mathrm{d}}_{2j}||^2\mathrm{d}_{2j}\bar{\mathrm{d}}_{2j}^T + \tau_j^2||\bar{\mathrm{s}}_j||^2\mathrm{s}_j\bar{\mathrm{s}}_j^T,$$
$$\bar{\mathrm{A}}_j\mathrm{A}_j = \beta_{1,j}^2||\mathrm{d}_{2j-1}||^2\bar{\mathrm{d}}_{2j-1}\mathrm{d}_{2j-1}^T + \beta_{2,j}^2||\mathrm{d}_{2j}||^2\bar{\mathrm{d}}_{2j}\mathrm{d}_{2j}^T + \tau_j^2||\mathrm{s}_j||^2\bar{\mathrm{s}}_j\mathrm{s}_j^T$$

for $j = 1, \ldots, m$. Therefore, the eigenvectors $v_{3j-2}, v_{3j-1}, v_{3j}$ and $w_{3j-2}, w_{3j-1}, w_{3j}$ of the matrices $\mathrm{A}_j\bar{\mathrm{A}}_j$ and $\bar{\mathrm{A}}_j\mathrm{A}_j$, having nonzero singular values σ_j, are given by

$$\{v_{3j-2}, v_{3j-1}, v_{3j}\} = \left\{\frac{\mathrm{d}_{2j-1}}{||\mathrm{d}_{2j-1}||}, \frac{\mathrm{d}_{2j}}{||\mathrm{d}_{2j}||}, \frac{\mathrm{s}_j}{||\mathrm{s}_j||}\right\}, \tag{2.32}$$

$$\{w_{3j-2}, w_{3j-1}, w_{3j}\} = \left\{\frac{\bar{\mathrm{d}}_{2j-1}}{||\bar{\mathrm{d}}_{2j-1}||}, \frac{\bar{\mathrm{d}}_{2j}}{||\bar{\mathrm{d}}_{2j}||}, \frac{\bar{\mathrm{s}}_j}{||\bar{\mathrm{s}}_j||}\right\}, \tag{2.33}$$

and

$$\{\sigma_{3j-2}, \sigma_{3j-1}, \sigma_{3j}\} = \left\{\frac{n}{2}|\beta_{1,j}|, \frac{n}{2}|\beta_{2,j}|, n|\tau_j|\right\}. \tag{2.34}$$

Since $\mathcal{R}(\mathrm{A}\bar{\mathrm{A}}) = \mathrm{span}\{\mathrm{d}_1, \ldots, \mathrm{d}_{2m}, \mathrm{s}_1, \ldots, \mathrm{s}_m\} = \mathrm{span}\{v_1, \ldots, v_{3m}\}$, then the singular vectors $\hat{v}_{l'}$ are linear combinations of the array vectors $v_{l'}$, while $\hat{w}_{l'}$ are linear combinations of the array vectors $w_{l'} = \bar{v}_{l'}$ for $l' = 1, \ldots, 3m$.

Let

$$H_{l',q'} = \bar{v}_{l'}^T v_{q'}, \quad l', q' = 1, \dots, 3m. \tag{2.35}$$

Define

$$\begin{aligned} l' &= 3(l-1)+p, \\ q' &= 3(q-1)+p', l, q = 1, \dots, m, \;\; p, p' = 1, 2, 3. \end{aligned}$$

Using the polar coordinates $z_q - z_l = |z_l - z_q|(\cos\alpha_{lq}, \sin\alpha_{lq})$, we compute

$$H_{l',q'} = \frac{\sum_{t=1}^n \hat{\xi}_{p,t}\, e^{ik|z_l - z_q|\cos(\theta_t - \alpha_{lq})}}{\sqrt{n}\sqrt{\sum_{t=1}^n \hat{\xi}_{p,t}^2}}, \;\; p = 1, 2, \; p' = 3, \; l \neq q. \tag{2.36}$$

For $l = q$ we compute $H_{l',q'}$ from (2.20), (2.21), (2.22), (2.30), and (2.31). Suppose $l \neq q$. For $p, p' = 1, 2$ we refer to section 2.2.1. Otherwise, the following holds:

$$\begin{aligned} &H_{l',q'} \approx i\sqrt{2}\,\cos(\alpha_{lq}) J_1(k|z_l - z_q|) && \text{if } p = 1, p' = 3, \\ &H_{l',q'} \approx i\sqrt{2}\,\sin(\alpha_{lq}) J_1(k|z_l - z_q|) && \text{if } p = 2, p' = 3, \\ &H_{l',q'} \approx J_0(k|z_l - z_q|) && \text{if } p = p' = 3, \end{aligned} \tag{2.37}$$

where J_1 is the Bessel function of the first kind and order one.

To conclude this subsection, we note that in the case where $H_{l',q'} \approx \delta_{l',q'}$ for $l', q' = 1, \dots, 3m$, we get $\hat{\sigma}_{3(j-1)+p} \approx \frac{n}{2}|\beta_{p,j}|$, $p = 1, 2$; $\hat{\sigma}_{3j} \approx n|\tau_j|$. Moreover, the singular vectors, having nonzero singular values, are given by $\hat{v}_{3(j-1)+p} \approx v_{3(j-1)+p}$, $\hat{w}_{3(j-1)+p} \approx w_{3(j-1)+p}$ for $j = 1, \dots, m$ and $p = 1, 2, 3$.

Suppose that $\gamma_\mu^j M^j$ and $\gamma_\varepsilon^j |B_j|$ are all different enough and α_{lq} are such that the right-hand sides of the two first equations in (2.37) take small values. Then we may claim that if $|\beta_{1,1}| \geq |\beta_{2,1}| \gg |\beta_{1,2}| \geq |\beta_{2,2}| \gg \cdots \gg |\beta_{1,m}| \geq |\beta_{2,m}|$ and $|\tau_1| \gg \cdots \gg |\tau_m|$, then $\hat{\sigma}_{2(j-1)+p}$ is close to $\sigma_{2(j-1)+p}$ and the two first singular values correspond to the inclusion with the highest $\gamma_\mu^j M^j$.

3. Fourier algorithm. In this section we present a second linear (noniterative) method to determine the location of the small inclusions from scattering amplitude measurements for a fixed frequency. This method, like the variational algorithm introduced in [4], reduces the reconstruction problem of the small inclusions to the calculation of an inverse Fourier transform. Using (1.4) we will see that the points z_j, which determine the location of the inclusions, can be recovered by performing an appropriate Fourier inversion.

The general lines of our method were first introduced in [3] for the three-dimensional scattering problem and used by Volkov [25] together with a projection technique on two-dimensional planes for recovering the positions of the inclusions. Here we consider the two-dimensional case and explain in detail how this algorithm applies by using explicit Fourier series representation of the scattering amplitude. The key steps of its numerical implementation are also described.

We assume that we are in possession of $A_\alpha^{(1)}(\hat{x}_l, \hat{\xi}_{l'})$ for a collection of pairs $(\hat{x}_l, \hat{\xi}_{l'})$, where $l = 1, \dots, L$ and $l' = 1, \dots, L'$. For simplicity we suppose that all the inclusions are disks. We then rewrite (1.4) as follows:

$$A_\alpha^{(1)}(\hat{x}, \hat{\xi}) = \alpha^2 \sum_{j=1}^m e^{-ik(\hat{x}-\hat{\xi})\cdot z_j} \left(\gamma_\mu^j\, m^j \hat{x}\cdot\hat{\xi} + \gamma_\varepsilon^j |B_j| \right), \quad \hat{x}, \; \hat{\xi} \in \text{¸}^{\,1}, \tag{3.1}$$

where m^j is defined by (1.3). We first observe that

$$A_\alpha^{(1)}(\hat{x},\hat{\xi}) = A_\alpha^{(1)}(-\hat{\xi},-\hat{x}), \quad A_\alpha^{(1)}(\hat{x},\hat{\xi}) = \overline{A}_\alpha^{(1)}(-\hat{x},-\hat{\xi}) \quad \forall \hat{x},\hat{\xi} \in \ ^1,$$

where $\overline{A}_\alpha^{(1)}$ denotes the complex conjugate of $A_\alpha^{(1)}$.

Define, for $l = 1,\dots,L$ and $l' = 1,\dots,L'$, the coefficients $a_{l,l'}$ by $a_{l,l'} = A_\alpha^{(1)}(\hat{x}_l,\hat{\xi}_{l'})$. Our reconstruction procedure is divided into three steps.

1. Given that $a_{l,l'}$, we can compute using the FFT as an accurate approximation of $A_\alpha^{(1)}(\hat{x},\hat{\xi})$ on $^1 \times \ ^1$.

2. Let $\mathcal{M}$ denote the following complex variety: $\mathcal{M} = \{\xi \in \mathbb{C}^2,\ \xi\cdot\xi = 1\}$. It is easy to see that $A_\alpha^{(1)}(\hat{x},\hat{\xi})$ has an analytic continuation to $\mathcal{M}\times\mathcal{M}$. Let $(Y_p)_{p=0,1,\dots,}$ denote the normalized (in $L^2(\ ^1)$) spherical harmonics. Denote by A_p the Fourier coefficients of $A_\alpha^{(1)}$:

$$A_\alpha^{(1)}(\hat{x},\hat{\xi}) = \sum_p A_p(\hat{x})Y_p(\hat{\xi}) \quad \forall \hat{x},\hat{\xi} \in \ ^1. \tag{3.2}$$

Recall from Step 1 that we are in fact in possession of an accurate approximation on ^{1} of $A_p(\hat{x})$, $p \le P$ for some P. In view of (3.2), the analytic continuation of the truncated Fourier series

$$\sum_{p;\, p\le P} A_p(\hat{x})Y_p(\hat{\xi})$$

of $A_\alpha^{(1)}(\hat{x},\hat{\xi})$ on $\mathcal{M}\times\mathcal{M}$ can be obtained by using the standard analytic continuation of the spherical harmonics $(Y_p(\hat{\xi}))_p$ on the complex variety $\mathcal{M}$ followed by another analytic continuation of the Fourier expansion in $\hat{x}$. We know that the analytic continuation of $A_\alpha^{(1)}$ from $^1\times\ ^1$ to $\mathcal{M}\times\mathcal{M}$ is unique. In fact, $\mathcal{M}$ is a one-dimensional complex variety and S^1 is a totally real one-dimensional real submanifold of $\mathcal{M}$. Thus an analytic function which vanishes on S^1 must be 0; see, for example, [5].

3. Recall that, given $a_{l,l'}$ for $l = 1,\dots,L$ and $l' = 1,\dots,L'$, we have constructed through Steps 1 and 2 an accurate approximation of the function $A_\alpha^{(1)}(\hat{x},\hat{\xi})$ that is analytic on $\mathcal{M}\times\mathcal{M}$ and is such that $A_\alpha^{(1)}(\hat{x}_l,\hat{\xi}_{l'}) = a_{l,l'}$ for all $l = 1,\dots,L$ and $l' = 1,\dots,L'$. However, for any $\xi \in \mathbb{R}^2$, we know that there exist ξ_1 and ξ_2 in $\mathcal{M}$ such that $\xi = k(\xi_1 - \xi_2)$; for example, if $|\xi| > 2k$, then

$$\xi_1 = \frac{\xi}{2k} + i\eta, \quad \xi_2 = -\frac{\xi}{2k} + i\eta, \tag{3.3}$$

where $\eta \in \mathbb{R}^2$ such that $\xi\cdot\eta = 0$, and

$$|\eta|^2 = \begin{cases} 0 & \text{for } |\xi| \le 2k, \\ \dfrac{|\xi|^2}{4k^2} - 1 & \text{for } |\xi| > 2k. \end{cases}$$

In particular, if $\xi = 0$, we may choose $\xi_1 = \xi_2 = 1/\sqrt{2}$. Let us now view $(a_{l,l'})$ as a function of $\xi \in \mathbb{R}^2$. Since $\xi_1\cdot\xi_2 = 1 - |\xi|^2/(2k^2)$, we can rewrite $A_\alpha^{(1)}(\xi_1,\xi_2)$ as

$$A_\alpha^{(1)}(\xi) := A_\alpha^{(1)}(\xi_1,\xi_2) = \alpha^2 \sum_{j=1}^m e^{-i\xi\cdot z_j}\left(\gamma_\mu^j m^j\left(1 - \frac{|\xi|^2}{2k^2}\right) + \gamma_\varepsilon^j |B_j|\right), \tag{3.4}$$

and therefore we are now in possession of an approximation to $A_\alpha^{(1)}(\xi)$ for any $\xi \in \mathbb{R}^2$. Here we rely on the fact that the analytic continuation is unique.

Recall that $e^{-i\xi\cdot z_j}$ (up to a multiplicative constant) is exactly the Fourier transform of the Dirac function δ_{z_j} (a point mass located at z_j). Multiplication by powers of ξ in Fourier space corresponds to differentiation of the Dirac function. Therefore, using the inverse Fourier transform, $\mathcal{F}^{-1}$, we obtain

$$\mathcal{F}^{-1}(A_\alpha^{(1)}) = \sum_{j=1}^{m} L_j(\delta_{z_j}),$$

where L_j are, in view of (3.2), second-order constant coefficient differential operators. Hence $A_\alpha^{(1)}(\xi)$ is the inverse Fourier transform of a distribution with its support at the locations of the centers of inclusions z_j.

We now give an explicit and useful Fourier representation of $A_\alpha^{(1)}$. In the polar system of coordinates, let $\hat{x} = (\cos\phi, \sin\phi), \hat{\xi} = (\cos\theta, \sin\theta)$, and $z_j = |z_j|(\cos\zeta_j, \sin\zeta_j)$. We write

$$A_\alpha^{(1)}(\theta,\phi) = \alpha^2 \sum_{j=1}^{m} e^{-ik|z_j|(\cos(\phi-\zeta_j)-\cos(\theta-\zeta_j))} \left(\gamma_\mu^j\, m^j \cos(\phi-\theta) + \gamma_\varepsilon^j |B_j|\right). \tag{3.5}$$

Let $k|z_j| = x_j$. Since $e^{iz\sin\alpha} = \sum_{n=-\infty}^{+\infty} J_n(z)e^{in\alpha}$, we can write

$$e^{-ix_j\cos(\phi-\zeta_j)} = e^{ix_j\sin(\phi-\zeta_j-\frac{\pi}{2})} = \sum_{p=-\infty}^{+\infty} J_p(x_j)e^{ip\phi}e^{-ip(\zeta_j+\frac{\pi}{2})}, \tag{3.6}$$

where $J_q(z)$ denotes the Bessel function of the first kind and order q. Similarly, we have

$$e^{ix_j\cos(\theta-\zeta_j)} = e^{ix_j\sin(\theta-\zeta_j+\frac{\pi}{2})} = \sum_{l=-\infty}^{+\infty} J_l(x_j)e^{il\theta}e^{-il(\zeta_j-\frac{\pi}{2})}. \tag{3.7}$$

By multiplying (3.6) by (3.7) we get

$$e^{-ix_j(\cos(\phi-\zeta_j)-\cos(\theta-\zeta_j))} = \sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{c}_{pl}J_p(x_j)J_l(x_j)e^{ip\phi}e^{il\theta}, \tag{3.8}$$

where

$$\tilde{c}_{pl} = e^{-ip(\zeta_j+\frac{\pi}{2})}e^{-il(\zeta_j-\frac{\pi}{2})}. \tag{3.9}$$

Making use of the trigonometric formula $\cos(\phi-\theta) = \frac{1}{2}(e^{i\phi}e^{-i\theta} + e^{-i\phi}e^{i\theta})$ we get

$$\begin{aligned}&\sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \tilde{c}_{pl}J_p(x_j)J_l(x_j)e^{ip\phi}e^{il\theta}\cos(\phi-\theta)\\ &\quad= \frac{1}{2}\sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} \left(\tilde{c}_{p-1,l+1}J_{p-1}(x_j)J_{l+1}(x_j) + \tilde{c}_{p+1,l-1}J_{p+1}(x_j)J_{l-1}(x_j)\right)e^{ip\phi}e^{il\theta}.\end{aligned}$$

From the recurrence formulae

$$J_{q-1}(z) + J_{q+1}(z) = \frac{2q}{z}J_q(z), \quad J_{q-1}(z) - J_{q+1}(z) = 2J_q'(z), \quad q \in \mathbb{Z},$$

together with the fact that $\tilde{c}_{p-1,l+1} = \tilde{c}_{pl}e^{i\pi}$, $\tilde{c}_{p+1,l-1} = \tilde{c}_{pl}e^{-i\pi}$, it immediately follows that

$$\begin{aligned}(3.10)\qquad &\sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty}\tilde{c}_{pl}J_p(x_j)J_l(x_j)e^{ip\phi}e^{il\theta}\cos(\phi-\theta)\\ &\quad=\sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty}\tilde{c}_{pl}\left(-J_p'(x_j)J_l'(x_j)+\frac{pl}{{x_j}^2}J_p(x_j)J_l(x_j)\right)e^{ip\phi}e^{il\theta}.\end{aligned}$$

Finally, using (3.5) and (3.10), we obtain that for θ and ϕ on the real axis, $A_\alpha^{(1)}(\theta,\phi)$ is given by its Fourier series,

$$(3.11)\qquad A_\alpha^{(1)}(\theta,\phi)=\sum_{p=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty}C_{pl}e^{ip\phi}e^{il\theta},$$

where the Fourier coefficients C_{pl} are given by

$$(3.12)\qquad C_{pl}=\alpha^2\tilde{c}_{pl}\sum_{j=1}^{m}c_{pl}^j,$$

and

$$c_{pl}^j=\left(-J_p'(x_j)J_l'(x_j)+J_p(x_j)J_l(x_j)\frac{pl}{x_j^2}\right)\gamma_\mu^j\,m^j+J_p(x_j)J_l(x_j)\gamma_\varepsilon^j|B_j|.$$

The reconstruction procedure described below can be reformulated as follows. Recall that for θ and ϕ on the real axis, $A_\alpha^{(1)}(\phi,\theta)$ is given by its Fourier series (3.11). Then, for any $\xi\in\mathbb{R}^2$, one determines the values for $\tilde{\phi}$ and $\tilde{\theta}$ (via the vectors ξ_2 and ξ_1) in order to satisfy (3.3); plugging these values into the truncated Fourier series for $A_\alpha^{(1)}$ yields C_{pl} the plth Fourier coefficient of $A_\alpha^{(1)}$ as

$$(3.13)\qquad A_\alpha^{(1)}(\tilde{\phi},\tilde{\theta})=\sum_{p,l=0}^{n-1}C_{pl}e^{ip\tilde{\phi}}e^{il\tilde{\theta}}.$$

Then, letting $\tilde{\phi}_I=\max|\mathrm{imag}\,(\tilde{\phi})|$ and $\tilde{\theta}_I=\max|\mathrm{imag}\,(\tilde{\theta})|$ for the complex values of $\tilde{\phi}$ and $\tilde{\theta}$, the sufficient condition for convergence of (3.13) is

$$(3.14)\qquad \left|C_{n/2-1,n/2-1}\right|e^{(n/2-1)\tilde{\phi}_I}e^{(n/2-1)\tilde{\theta}_I}\to 0\quad\text{as } n\to\infty.$$

It follows that one can estimate $A_\alpha^{(1)}$ for those complex values $\tilde{\phi}$ and $\tilde{\theta}$ that are close to the real axis if n is large enough, emphasizing the fact that the choice of the number n depends upon the value of k, and upon the locations of the inclusions themselves (i.e., $|z_j|, j=1,\ldots,m$). Note that to (numerically) compute (3.13), the regularization technique developed by Franklin in [13] can be used.

Let R be such that $2R$ is the radius of the disk where (3.14) holds. The function $A_\alpha^{(1)}$ is then estimated within the square $\Omega(R)=[-\sqrt{2}R,\sqrt{2}R]^2$, letting the spectral radius R be larger than k. In view of the above, we have two cases of interest:

- there exist $\tilde{\phi}$ and $\tilde{\theta}$ which are complex-valued, and one is zero-padding the Fourier domain by $A_\alpha^{(1)}(\tilde{\phi},\tilde{\theta}):=0$ for all complex values of $\tilde{\phi}$ and $\tilde{\theta}$;

- there exist $\tilde{\phi}$ and $\tilde{\theta}$ which are complex-valued, and one applies the method of analytic continuation by FFT in order to estimate $A_\alpha^{(1)}$.

As will be shown by the numerical results, if $k|z_j|$ is small, then n is not sufficiently large, which implies that $R \gg k$, while if $k|z_j|$ is large, then R tends to k and thus would be more appropriate to apply the zero-padding algorithm.

The number of data (sampling) points N^2 needed for an accurate discrete Fourier inversion of $A_\alpha^{(1)}$ follows from the so-called Shannon theorem [11]. We need (conservatively), of order $(2L/\delta)^2$, sampled values to reconstruct a collection of inclusions that lies inside a square of side $2L$, where the (spatial) resolution $\delta = \pi/2k$ for the zero-padding algorithm, while it is given by $\delta = \pi/\sqrt{2}R$ for the method of analytic continuation.

4. Numerical experimentation. The configuration of interest is schematized in Figure 4.1. One is considering an array $\{A_j\}$ of n ideal antennas that are hypothetically radiating from and receiving at infinity in $\mathbb{R}^2$. That is, one assumes that, at one (and only one, which is a rather adverse situation) circular frequency ω one knows the values a_{lp}—those may be affected by noise, as seen in one subsection later on—of the far field pattern $A_\alpha^{(1)}(-\hat{\xi}_l, \hat{\xi}_p)$, $l, p = 1, \ldots, n$, for a small finite number (it will be set to 20 in all examples) of equidistantly distributed directions of illumination $\hat{\xi}_p = -(\cos\theta_p, \sin\theta_p)$, and for the opposite set of directions of observation, letting $\theta_p = 2\pi(p-1)/n$, $p = 1, \ldots, n$.

These directions are taken all around (i.e., one is speaking of a full-view inverse problem) an unknown finite number of homogeneous inclusions located at unknown positions within some prescribed domain of free space (air). As considered before these inclusions may have their dielectric permittivity (first case) or magnetic permeability (second case) or both (third case) that are differing (then, always with superior real-positive values) from those, $\mu_0 = 1$ and $\varepsilon_0 = 1$, of the embedding medium.

As is evident from the previous theoretical material, what matter are the real-positive valued contrasts γ_ε^j, and/or the real-positive valued contrasts γ_μ^j (then, mul-

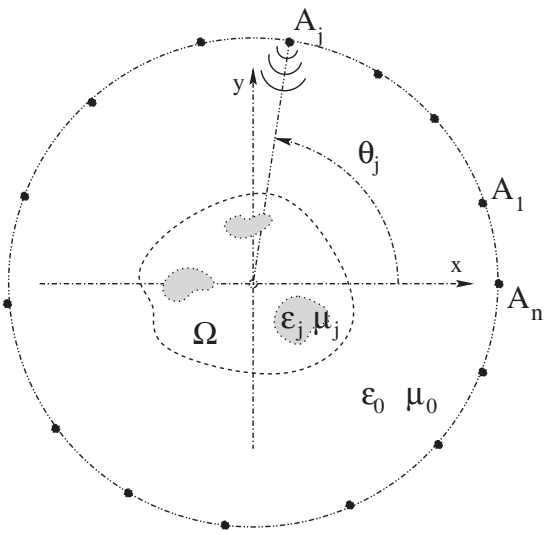


FIG. 4.1. *The full-view measurement setup.*

tiplied by the amplitude of the polarization tensor for a disk), noting that the higher the relative permittivity or permeability, the larger the magnitude of the resulting weight factor.

As for the single circular frequency of operation, it is set between the low-frequency value $\omega = 1$ and the high one $\omega = 10$, associated to wavenumbers k between 1 and 10 and to wavelengths in air between 2π and 0.2π, these figures being recalled when weighting in the usually expected half-a-wavelength far field resolutions with respect to those reached thereafter via the MUSIC algorithm (section 4.1) and Fourier reconstruction (section 4.2).

4.1. Examples of MUSIC reconstructions. As discussed in the theoretical examination, the nature (dielectric and/or magnetic) of the inclusions with respect to that of the embedding medium (here a purely dielectric and non magnetic medium) is essential to the results expected. Next, one is displaying and briefly analyzing a small number of numerical results from synthetically generated data in the three generic scattering cases previously studied.

The configuration involves two small homogeneous circular disks of diameter $\alpha = 0.1$ denoted B_1 and B_2 and, respectively, centered at $z_1 = (-0.45, 0.43)$ and $z_2 = (0.52, -0.44)$, to be retrieved using 20 illuminations and the corresponding observations within a prescribed search box, being the square one $\Omega = [-2, 2]\times[-2, 2] \subset \mathbb{R}^2$.

Corresponding dielectric permittivities ε^j, $j = 1, 2$, are equated to 5 and 2, and magnetic permeabilities μ^j, $j = 1, 2$, are equated to 5 and 2. Specific examples tailored to the analysis of the singular values assume identical values $\varepsilon^j = 3$ and/or $\mu^j = 3$, $j = 1, 2$. Unless stated otherwise for the study of the behavior of the approach versus frequency, the circular frequency of operation is fixed at $\omega = 4$.

Within the above setting, the retrieval of the inclusions involves the calculation of the SVD $\mathrm{A} = \mathrm{U}\Sigma\bar{\mathrm{V}}^T$ of the matrix $\mathrm{A} = (a_{l,p}) \in \mathbb{C}^{n\times n}$. Then, for each discrete location $z \in \Omega$ (the sampling step henceforth is $h = 0.05$), the identifiers of interest are

- in the first case: $W_a(z)$, $a = 1$;
- in the second case: $W_b(z)$, $b = \{(1, 0), (0, 1), (1, 1)\}$;
- in the third case: $W_c(z)$, $c = \{(1, 1, 0), (0, 0, 1), (1, 1, 1)\}$,

calculated within Ω, the specific values of b and c here having being inferred from the numerical experimentation. Plots of $z \to W_\cdot(z)$ illustrate the result achieved, sharp peaks being expected to occur at the locations of the inclusions, z_j, $j = 1, 2$. Other accompanying results displayed consist of the singular values of A, using a standard log scale, and also of grayscale maps of appropriate norms of "back-propagated" singular vectors or combinations.

4.1.1. The first case: Dielectric inclusions. In this purely dielectric case, the singular values are displayed in Figure 4.2; the identifier $W_a(z)$, where one has, for simplicity, let $a = 1$, is displayed in Figure 4.3; and the amplitudes of the scalar products of the singular vectors V_j with $g_{z,a}$ are displayed in Figure 4.4. The distribution of the nonzero singular values of A as a function of frequency $\omega = 1 : 10$ is shown in Figure 4.5 either for different inclusions ($\varepsilon^1 = 5$, $\varepsilon^2 = 2$) or for identical ones ($\varepsilon^1 = 3$, $\varepsilon^2 = 3$).

Results obtained as such are easy to interpret. Two singular values emerge from the 18 others in the noise subspace, and they range in accordance with the contrast of the corresponding inclusions. The two said inclusions are sharply retrieved, with a resolution no more than a small fraction of the wavelength (and isotropic, in tune with the fact that the search space is illuminated and seen from all around). As for the

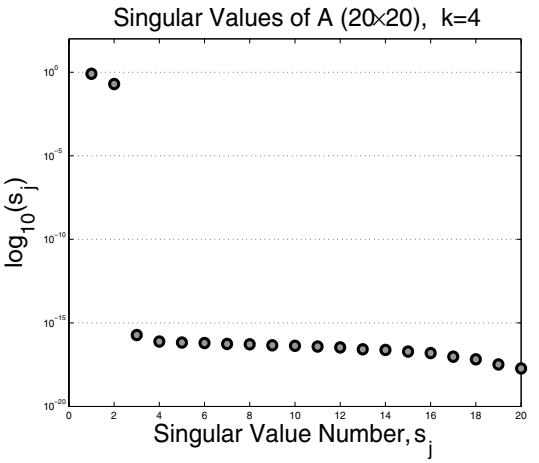


FIG. 4.2. *Case* 1 *(dielectric contrasts only): distribution of the singular values of* A *for* $n = 20$ *illuminations.*

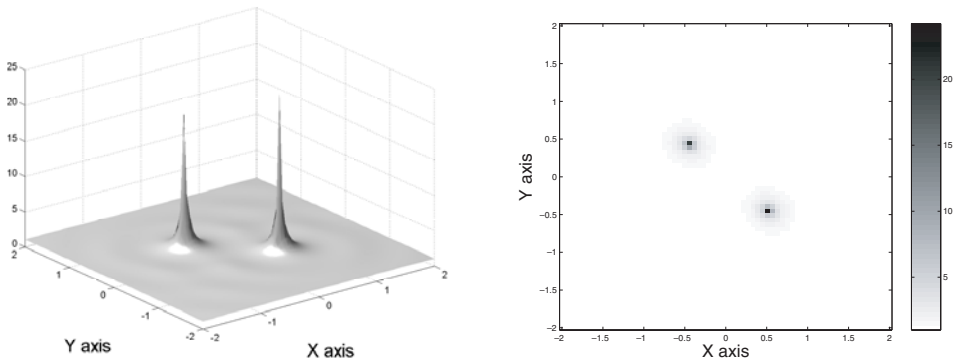


FIG. 4.3. *Case* 1 *(dielectric contrasts only):* 3*D plot and grayscalemap of* $W_a(z)$, $a = 1$, *for all points* z *in* Ω *(refer to Figure* 4.2*).*

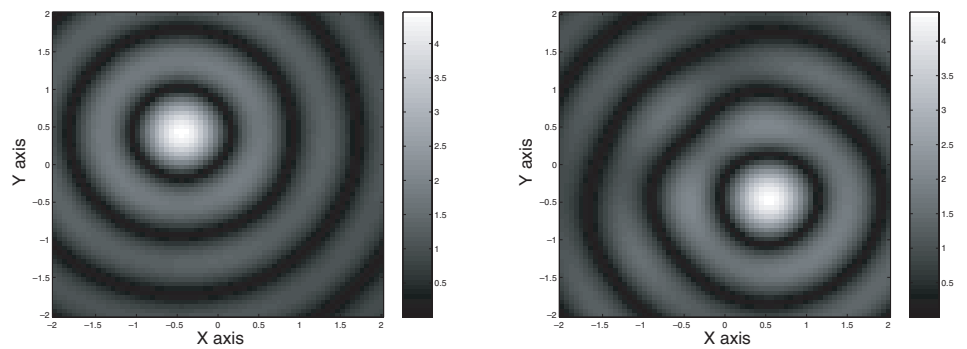


FIG. 4.4. *Case* 1 *(dielectric contrasts only): grayscale maps of the amplitudes of the scalar products of the singular vectors* V_1, V_2 *(ordered from left to right) with* $g_{z,a}$, $a = 1$, *for all points* z *in* Ω *(refer to Figures* 4.2 *and* 4.3*).*

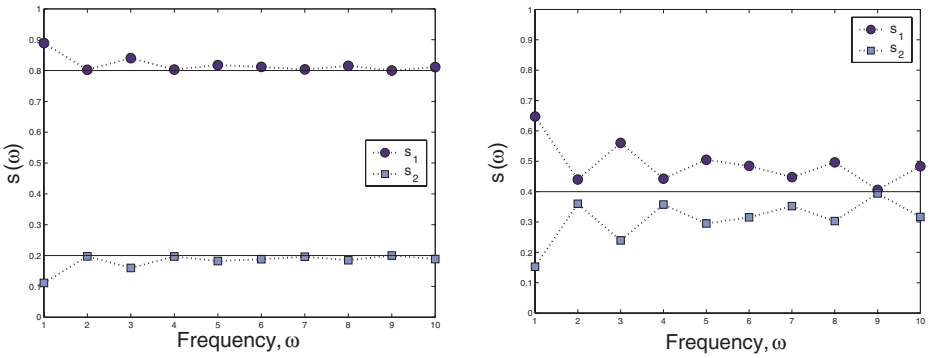


FIG. 4.5. *Case* 1 *(dielectric contrasts only): distribution of the nonzero singular values of* A *as a function of frequency,* $\omega = 1 : 10$*, for different or identical inclusions.* (a) *depicts the first (circles) and second (squares) singular values of* A *for* $\varepsilon^1 = 5$ *and* $\varepsilon^2 = 2$*;* (b) *depicts the first (circle) and second (square) singular values of* A *for* $\varepsilon^1 = 3$ *and* $\varepsilon^2 = 3$*. In both cases the solid line denotes the values taken by* $n\tau_j$*,* $j = 1, 2$*.*

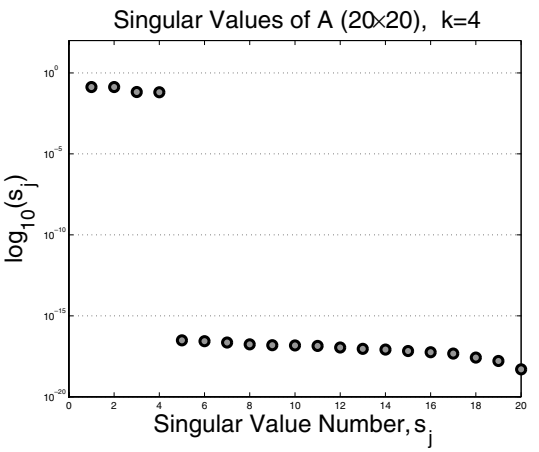


FIG. 4.6. *Case* 2 *(permeability contrasts only): distribution of the singular values of* A *for* $n = 20$ *illuminations.*

singular vectors, once operated upon by $g_{z,a}$, they focus on the inclusions, as expected in the configuration under study. Finally, the evolution as a function of frequency of the singular values confirms that permittivity contrasts that are different enough lead to singular values that are very close to the $n\tau_j$ values on the whole frequency band. This is no longer true, however, with identical contrasts for which the singular values appear to oscillate in opposite phase above or below the common value of $n\tau_j$.

4.1.2. The second case: Permeable inclusions. In this purely magnetic case (in terms of the inclusions), the singular values are displayed in Figure 4.6; the identifier $W_b(z)$ is displayed in Figure 4.7 for $b = (1, 0)$, in Figure 4.8 for $b = (0, 1)$, and in Figure 4.9 for $b = (1, 1)$; and the 2-norm of the matrix

$$\tilde{\mathrm{V}}_j(z) = \begin{bmatrix} \mathrm{V}_{2j-1} \cdot g_{z,(1,0)} & \mathrm{V}_{2j} \cdot g_{z,(1,0)} \\ \mathrm{V}_{2j-1} \cdot g_{z,(0,1)} & \mathrm{V}_{2j} \cdot g_{z,(0,1)} \end{bmatrix}, \quad j = 1, 2, \tag{4.1}$$

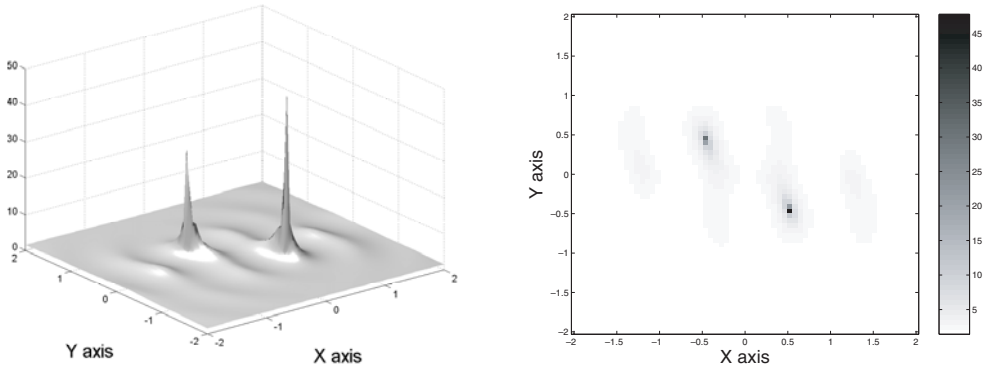


FIG. 4.7. *Case* 2 *(permeability contrasts only):* 3*D plot and grayscale map of* $W_b(z)$, $b = (1, 0)$, *for all points* z *in* Ω *(refer to Figure* 4.6*).*

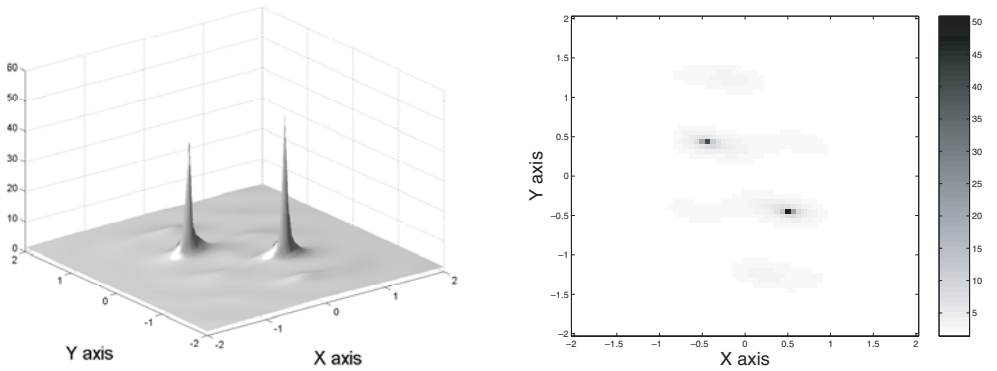


FIG. 4.8. *Case* 2 *(permeability contrasts only):* 3*D plot and grayscale map of* $W_b(z)$, $b = (0, 1)$, *for all points* z *in* Ω *(refer to Figure* 4.6*).*

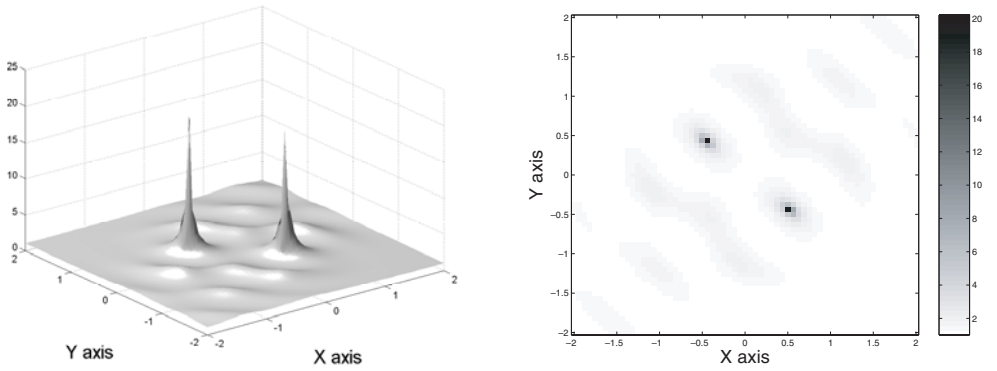


FIG. 4.9. *Case* 2 *(permeability contrasts only):* 3*D plot and grayscale map of* $W_b(z)$, $b = (1, 1)$, *for all points* z *in* Ω *(refer to Figure* 4.6*).*

is displayed in Figure 4.10, each subplot ($j = 1, 2$) being calculated from the combination of two singular vectors associated with the same inclusion. The distribution of the nonzero singular values of A as a function of frequency, $\omega = 1 : 10$, is shown

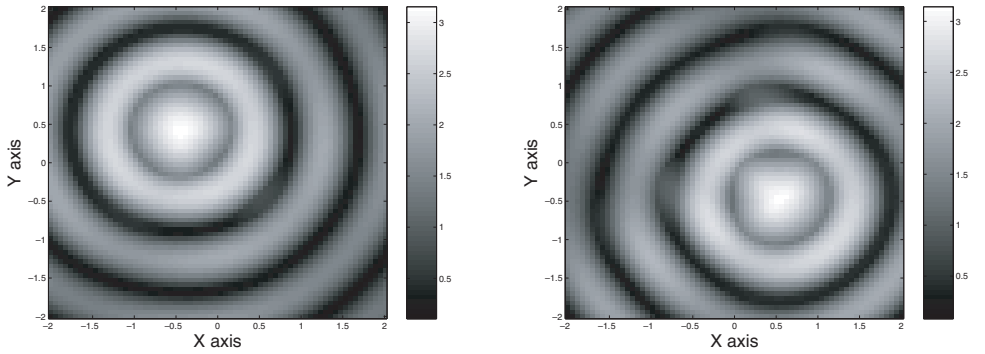


FIG. 4.10. *Case* 2 *(permeability contrasts only): grayscale maps of the* 2*-norm of the matrix* $\tilde{\mathrm{V}}_j(z)$, $j = 1, 2$, *defined by* (4.1), *for all points* z *in* Ω *(refer to Figures* 4.6 *to* 4.9*).*

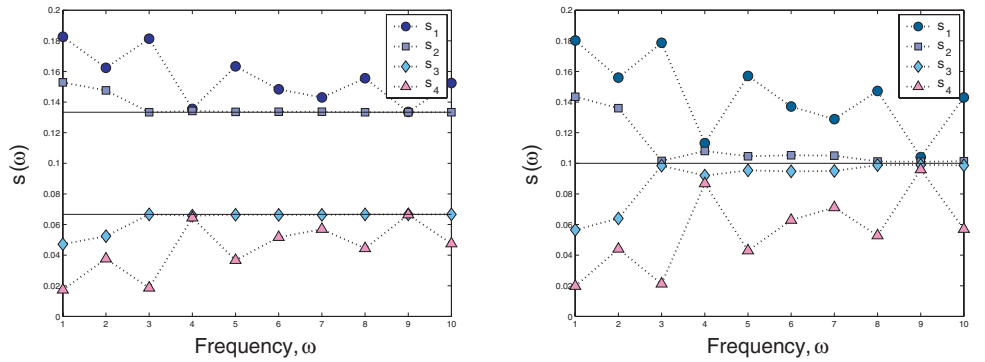


FIG. 4.11. *Case* 2 *(permeability contrasts only): distribution of the nonzero singular values of* A *as a function of frequency,* $\omega = 1 : 10$, *for different or identical inclusions.* (a) *depicts the four nonzero singular values of* A *for* $\mu^1 = 5$, $\mu^2 = 2$*;* (b) *depicts the four nonzero singular values of* A *for* $\mu^1 = 3$, $\mu^2 = 3$. *In both cases the solid line denotes the values taken by* $n\beta_j/2$, $j = 1, 2$.

in Figure 4.11 either for different inclusions ($\mu^1 = 5$, $\mu^2 = 2$) or for identical ones ($\mu^1 = 3$, $\mu^2 = 3$).

As previously, the results obtained are easy to interpret. Only four singular values emerge from noise, each couple (with almost identical value) being associated to one specific inclusion, and those couples range according to the permeability contrast. The inclusions, observed via different $W_b(z)$, are clearly discriminated from the background, the visual aspect depending upon the choice of b. The focusing of the singular vectors as seen from displays of the 2-norm of the vector couples defined in the above, is rather good, noticing that V_1 and V_2 are associated to B_1, and V_3 and V_4 are associated to B_2, i.e., from the most permeable inclusion to the least one. Finally, the evolution as a function of frequency of the singular values—as previously observed in the dielectric case, yet in less obvious a fashion due to the doubling of the number of singular values—exemplifies that permeability contrasts that are different enough lead to singular values that are very close to (here, often almost identical with) the $n\beta_j/2$ values on the whole frequency band. This is no longer true, however, with identical contrasts for which the singular values appear to oscillate in opposite phase

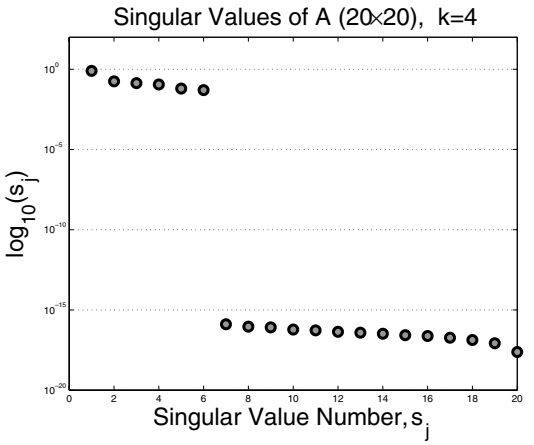


FIG. 4.12. *Case 3 (permittivity and permeability contrasts): distribution of the singular values of* A *for* $n = 20$ *illuminations.*

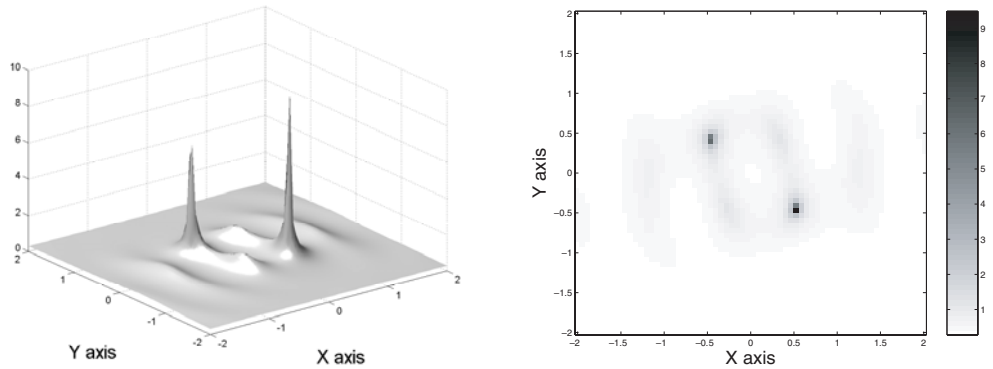


FIG. 4.13. *Case 3 (permittivity and permeability contrasts): 3D plot and grayscale map of* $W_c(z)$, $c = (1, 1, 0)$, *for all points* z *in* Ω *(refer to Figure* 4.12*).*

above or below the common value of $n\beta/2$.

4.1.3. The third case: Dielectric and permeable inclusions. In this full case, the singular values are displayed in Figure 4.12, and the identifier $W_c(z)$ is displayed in Figure 4.13 for $c = (1, 1, 0)$, in Figure 4.14 for $c = (0, 0, 1)$, and in Figure 4.15 for $c = (1, 1, 1)$. The 2-norm of the matrix

$$\tilde{\mathrm{V}}_j(z) = \begin{cases} V_j \cdot g_{z,a=1}, & j = 1, 2, \\ \begin{bmatrix} \mathrm{V}_{2j-3} \cdot g_{z,(1,0)} & \mathrm{V}_{2j-2} \cdot g_{z,(1,0)} \\ \mathrm{V}_{2j-3} \cdot g_{z,(0,1)} & \mathrm{V}_{2j-2} \cdot g_{z,(0,1)} \end{bmatrix}, & j = 3, 4, \end{cases} \tag{4.2}$$

is shown in Figure 4.16 for $j = 1, 2$ and Figure 4.17 for $j = 3, 4$. Each subplot is calculated from the singular vectors (or a combination thereof) associated with the same inclusion and the same property of this inclusion (dielectric or magnetic). As for the distribution of the nonzero singular values of A as a function of frequency considered before, it is left out for simplicity in view of the large number of singular values.

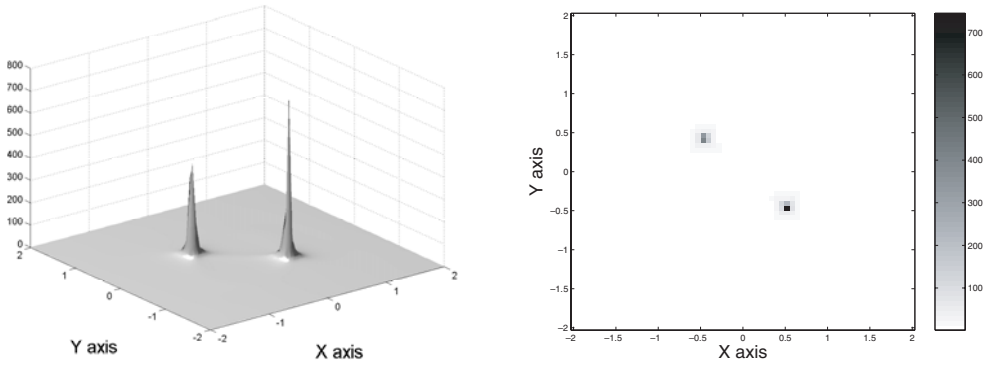


FIG. 4.14. *Case* 3 *(permittivity and permeability contrasts):* 3*D plot and grayscale map of* $W_c(z)$, $c = (0, 0, 1)$, *for all points* z *in* Ω *(refer to Figure* 4.12*).*

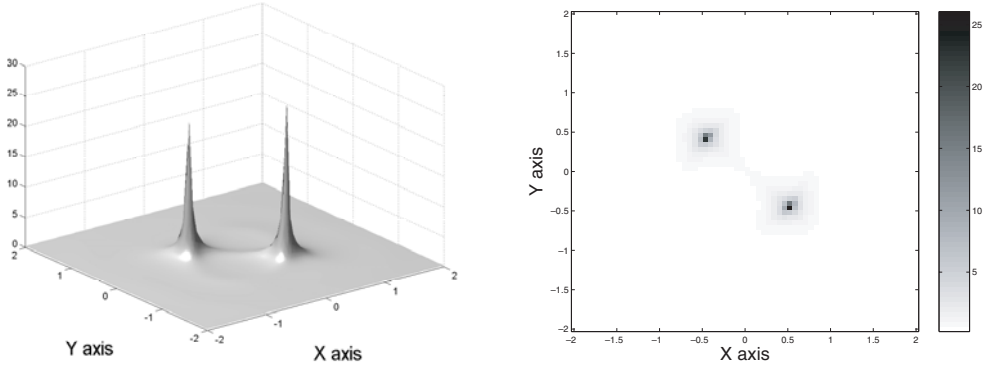


FIG. 4.15. *Case* 3 *(permittivity and permeability contrasts):* 3*D plot and grayscale map of* $W_c(z)$, $c = (1, 1, 1)$, *for all points* z *in* Ω *(refer to Figure* 4.12*).*

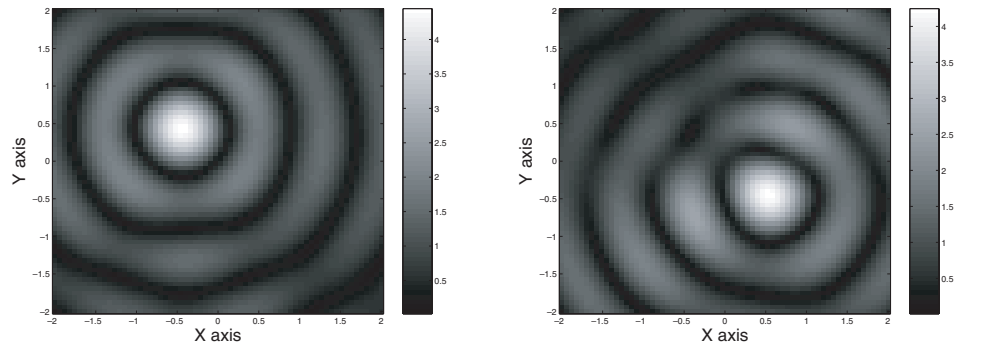


FIG. 4.16. *Case* 3 *(permittivity and permeability contrasts): grayscale maps of the* 2*-norm of the matrix* $\tilde{\mathrm{V}}_j(z)$, $j = 1, 2$, *defined by* (4.2), *for all points* z *in* Ω *(refer to Figures* 4.12 *to* 4.15*).*

The results obtained are evidently not as easy to interpret as before due to the more complicated character of the inclusions. Still, the singular values of the signal subspace emerge from noise—six of them—each triplet being associated to one specific inclusion, but with no obvious ranging observed. The inclusions, again, observed via

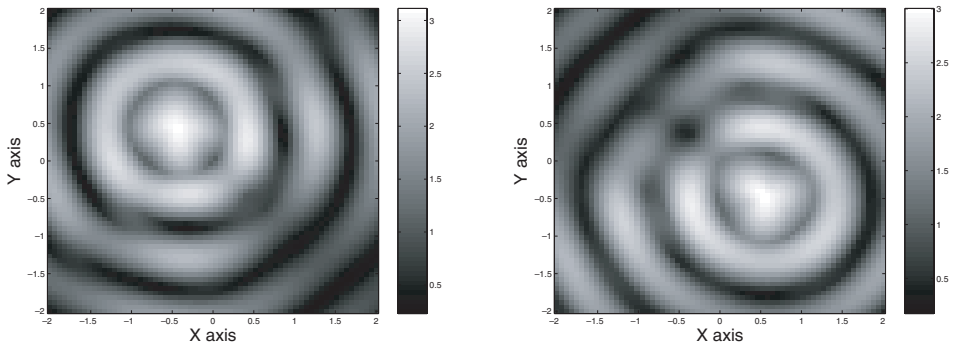


FIG. 4.17. *Case 3 (permittivity and permeability contrasts): grayscale maps of the 2-norm of the matrix* $\tilde{\mathrm{V}}_j(z)$, $j = 3, 4$, *defined by* (4.2), *for all points* z *in* Ω *(refer to Figures* 4.12, 4.13, 4.14, *and* 4.15*).*

different $W_c(z)$, are clearly discriminated from the background, the visual aspect depending upon the choice of c. As for the focusing of the singular vectors as seen from displays of the 2-norms defined in the above, it works for the most contrasted inclusion, B_1, both for the vectors V_3 and V_4 involved with the permeability and those, V_1, involved with the permittivity, but it does less with the least contrasted one, B_2, for which the maps would be difficult to interpret in the absence of any prior knowledge on the inclusions.

4.2. Examples of Fourier reconstructions. In this subsection a number of numerical examples illustrate the reconstruction by the Fourier method. In all such examples, the function $A_\alpha^{(1)}$ is estimated within the square $\Omega(R) = [-\sqrt{2}R, \sqrt{2}R]^2$, letting the spectral radius R be larger than k.

The configuration is the same as it is for the MUSIC algorithm (section 4.1), involving two small homogeneous circular disks of diameter $\alpha = 0.1$ still denoted as B_1 and B_2 and, respectively, centered at $z_1 = (-0.45, 0.43)$ and $z_2 = (0.52, -0.44)$, with $\varepsilon^1 = 3$ and $\varepsilon^2 = 2.5$, and/or $\mu^1 = 3$ and $\mu^2 = 2.5$. The assumption is made that, for the inversion carried out at frequency $\omega = 1$ ($k = 1$), such inclusions lie inside the square $[-6, 6]^2$, for which three-dimensional plots of $|\mathcal{F}^{-1}(A_\alpha^{(1)})|$ calculated by an inverse discrete Fourier transform (DFT) will thus be given, the pertinent contour plots being drawn for visibility only within the smaller square $[-3, 3]^2$. Note that one is using 16 angles θ and ϕ in $[0, 2\pi]$ in order to compute the Fourier coefficients.

The retrieval of purely dielectric inclusions is illustrated in Figures 4.18 and 4.19 via three-dimensional plots and contour plots of $|\mathcal{F}^{-1}(A_\alpha^{(1)})|$ which are obtained by carrying out either zero-padding (Figure 4.18) or analytic continuation (Figure 4.19); note that the number of data (sampling) points N^2 that are needed for an accurate discrete Fourier inversion of $A_\alpha^{(1)}$ in the square $\Omega(R = 5)$ is equal to 28^2.

The results show that the zero-padding approach does not enable us to discriminate between the two inclusions, and the corresponding resolution δ is appraised to be of the order of half a wavelength (1.57). On the contrary, the analytic continuation succeeds in providing this discrimination, associated to a much sharper resolution, of the order of one-seventh of a wavelength and even less (about 0.43 as confirmed by the figure).

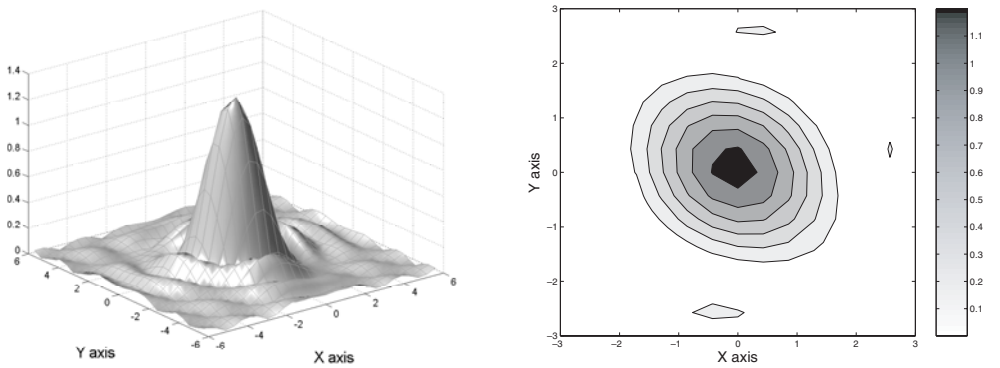


FIG. 4.18. *Retrieval of two purely dielectric inclusions with different contrasts by zero-padding at* $k = 1$ *(the resolution* δ *achieved is* ≈ 1.57*).*

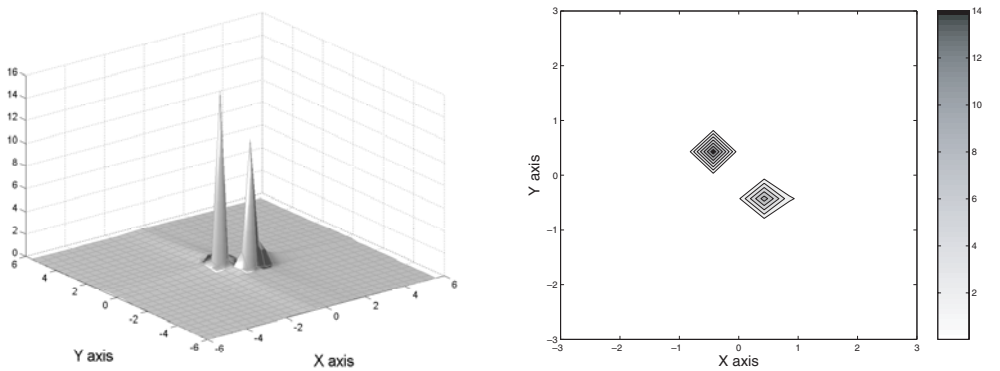


FIG. 4.19. *Retrieval of two purely dielectric inclusions with different contrasts by analytic continuation at* $k = 1$ *(the resolution* δ *achieved is* ≈ 0.43*).*

Correspondingly, the retrieval of purely magnetic inclusions is illustrated in Figures 4.20 and 4.21 via three-dimensional plots and contour plots of $|\mathcal{F}^{-1}(A_\alpha^{(1)})|$ obtained by carrying out either zero-padding (Figure 4.20) or analytic continuation (Figure 4.21) (with a slightly smaller Ω than before, $R = 4.5$); note that the number of data (sampling) points N^2 that are needed for an accurate discrete Fourier inversion of $A_\alpha^{(1)}$ in the square $\Omega(R = 4.5)$ is equal to 24^2.

The same conclusion as with the dielectric inclusions holds: failure of the zero-padding approach (which is yielding a complicated image from which one cannot infer much, except the fact that some object is in the search domain) and efficiency of the analytic continuation, though with a slightly lower resolution (of the order of 0.5), as is easily seen by comparing Figure 4.21 with Figure 4.19, both being computed in close conditions.

Finally, dielectric and permeable inclusions are tackled simultaneously, values of permittivity and permeability being those previously chosen. This is illustrated in Figures 4.22 and 4.23 via three-dimensional plots and contour plots of $|\mathcal{F}^{-1}(A_\alpha^{(1)})|$ obtained by carrying out either zero-padding (Figure 4.22) or analytic continuation (Figure 4.23); note that the number of data (sampling) points N^2 that are needed

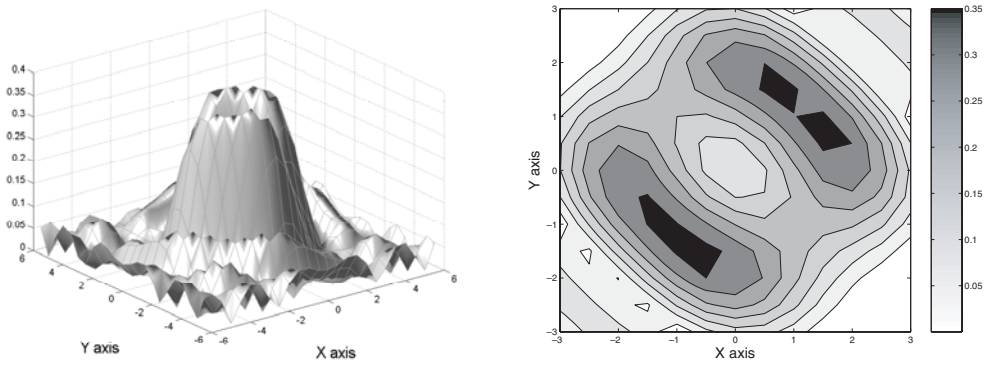


FIG. 4.20. *Retrieval of two purely magnetic inclusions with different contrasts by zero-padding at* $k = 1$ *(the resolution* δ *achieved is* ≈ 1.57*).*

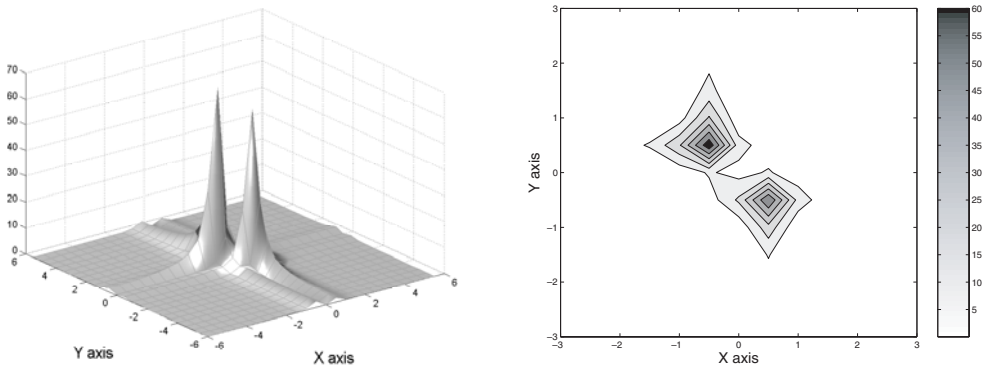


FIG. 4.21. *Retrieval of two purely magnetic inclusions with different contrasts by analytic continuation at* $k = 1$ *(the resolution* δ *achieved is* ≈ 0.5*).*

for an accurate discrete Fourier inversion of $A_\alpha^{(1)}$ in the square $\Omega(R = 4.5)$ is equal to 24^2. Again the analytic continuation is the only successful method retrieving the two inclusions, the resolution (and the visual aspect of the three-dimensional and contour plots) being like the one observed with the purely magnetic inclusions (the poorer result, in effect).

In the second set of numerical examples (for brevity, only purely dielectric inclusions are considered) one still keeps the number of inclusions equal to two, yet one of them is located much farther away from the center of the search domain than before, at $z_1 = (-4.0, 0.43)$, the other being located about as before, at $z_2 = (0.42, -0.44)$. Also, the inclusions are now identical, with $\varepsilon^1 = 3$ and $\varepsilon^2 = 3$. The assumption of the inversion is that they lie within the square $[-12, 12]^2$, contour plots being displayed in the smaller square $[-5.5, 5.5]^2$. The frequency of operation is the same as previously chosen, at $\omega = 1$ $(k = 1)$.

Figures 4.24 and 4.25 show three-dimensional plots and contour plots of $|\mathcal{F}^{-1}(A_\alpha^{(1)})|$ obtained by carrying out either zero-padding (Figure 4.24) or analytic continuation (Figure 4.25); note that the number of data (sampling) points N^2 that are needed for an accurate discrete Fourier inversion of $A_\alpha^{(1)}$ in the square $\Omega(R = 1.8)$ is equal to 20^2.

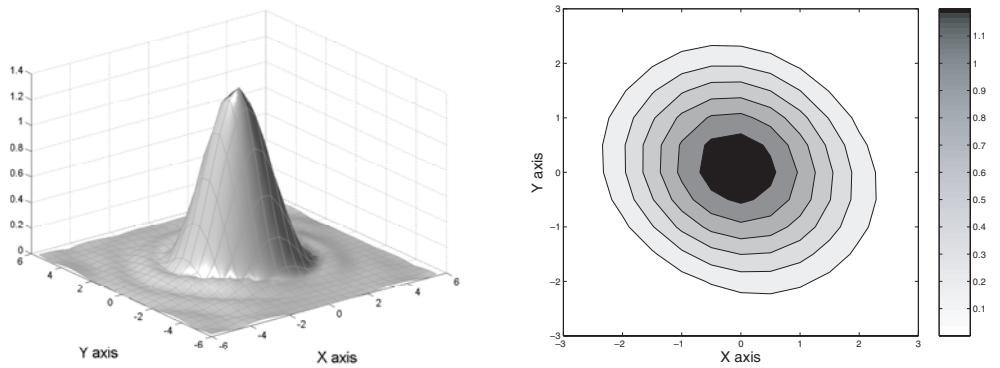


FIG. 4.22. *Retrieval of both dielectric and magnetic inclusions with different contrasts by analytic continuation at* $k = 1$ *(the resolution* δ *achieved is* ≈ 1.57*).*

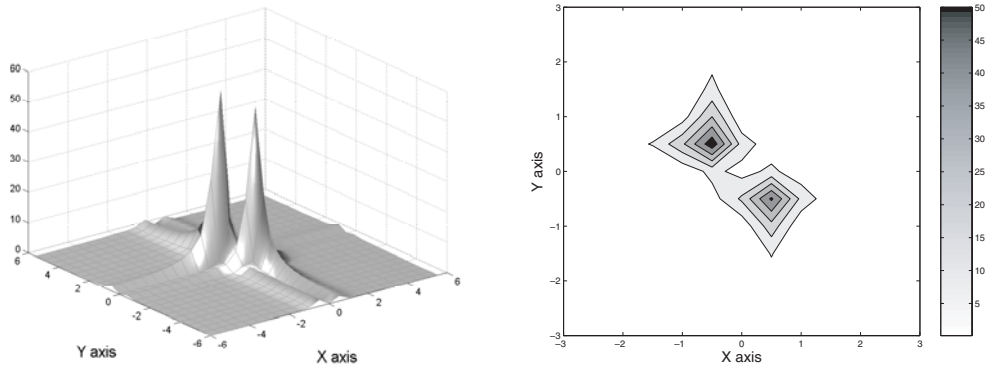


FIG. 4.23. *Retrieval of both dielectric and magnetic inclusions at* $k = 1$ *by analytic continuation (the resolution* δ *achieved is* ≈ 0.5*).*

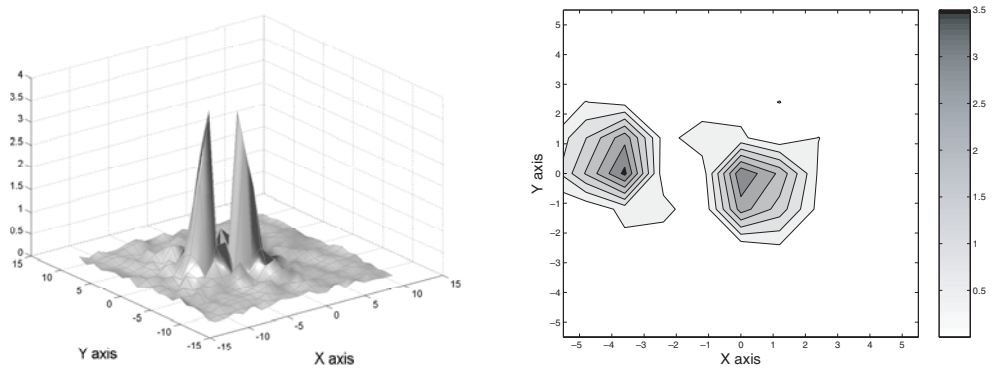


FIG. 4.24. *Retrieval of identical dielectric inclusions, one remote from the center of the search domain, at* $k = 1$ *by zero-padding (the resolution* δ *achieved is* ≈ 1.57*).*

Let us emphasize here that one has been using 32 angles θ and ϕ in $[0, 2\pi]$, because of the eccentricity of the first inclusion, for the angles θ and ϕ in order to suitably calculate the Fourier coefficients, and, as a matter of fact, one is able

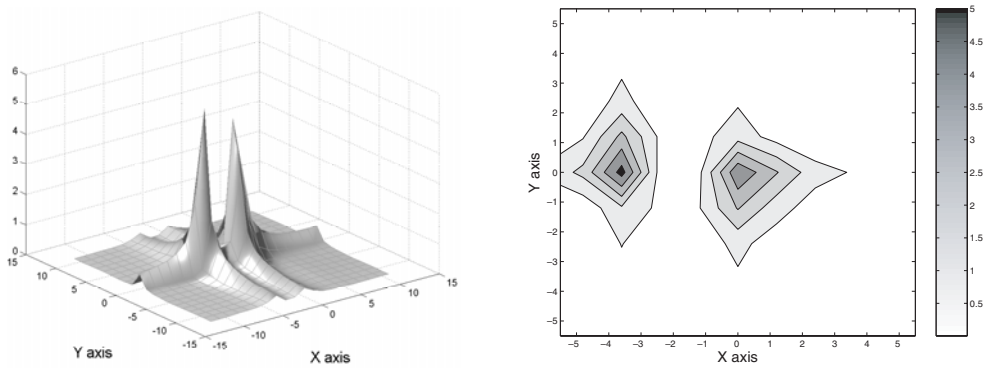


FIG. 4.25. *Retrieval of identical dielectric inclusions, one remote from the center of the search domain, by analytic continuation at* $k = 1$ *(the resolution* δ *achieved is* ≈ 1.2*).*

to extend the function $A_\alpha^{(1)}$ on the square $\Omega(R = 1.8)$—choosing a higher R yields divergent results.

Now, contrary to what was observed before, both approaches (zero-padding and analytic continuation) tend to behave likewise, as evidenced by the similarity of Figures 4.24 and 4.25. That is, there seems no interest in performing the analytic continuation as exemplified by the low maximal value of R, since it is not able to perform this continuation far enough in the spectral plane to improve upon the zero-padding results.

REFERENCES

[1] H. AMMARI AND H. KANG, *Reconstruction of Small Inhomogeneities from Boundary Measurements*, Lecture Notes in Math. 1846, Springer-Verlag, Berlin, 2004.

[2] H. AMMARI, E. IAKOVLEVA, AND D. LESSELIER, *A MUSIC algorithm for locating small inclusions buried in a half-space from the scattering amplitude at a fixed frequency*, Multiscale Model. Simul., 3 (2005), pp. 597–628.

[3] H. AMMARI, E. IAKOVLEVA, AND S. MOSKOW, *Recovery of small inhomogeneities from the scattering amplitude at a fixed frequency*, SIAM J. Math. Anal., 34 (2003), pp. 882–900.

[4] H. AMMARI, S. MOSKOW, AND M. VOGELIUS, *Boundary integral formulae for the reconstruction of electric and electromagnetic inhomogeneities of small volume*, ESAIM Control Optim. Calc. Var., 9 (2003), pp. 49–66.

[5] M. S. BAOUENDI, P. EBENFELT, AND L. P. ROTHSCHILD, *Real Submanifolds in Complex Space and Their Mappings*, Princeton University Press, Princeton, NJ, 1999.

[6] M. BRÜHL, M. HANKE, AND M. S. VOGELIUS, *A direct impedance tomography algorithm for locating small inhomogeneities*, Numer. Math., 93 (2003), pp. 635–654.

[7] D. J. CEDIO-FENGYA, S. MOSKOW, AND M. S. VOGELIUS, *Identification of conductivity imperfections of small diameter by boundary measurements. Continuous dependence and computational reconstruction*, Inverse Problems, 14 (1998), pp. 553–595.

[8] M. CHENEY, *The linear sampling method and the MUSIC algorithm*, Inverse Problems, 17 (2001), pp. 591–595.

[9] M. CHENEY AND D. ISAACSON, *Inverse problems for a perturbed dissipative half-space*, Inverse Problems, 11 (1995), pp. 865–888.

[10] D. COLTON, J. COYLE, AND P. MONK, *Recent developments in inverse acoustic scattering theory*, SIAM Rev., 42 (2000), pp. 369–414.

[11] A. J. DEVANEY, *Super-resolution processing of multi-static data using time reversal and MUSIC*, J. Acoust. Soc. Amer., to appear.

[12] M. FINK AND C. PRADA, *Acoustical time-reversal mirrors*, Inverse Problems, 17 (2001), pp. R1–R38.

[13] J. FRANKLIN, *Analytic continuation by the fast Fourier transform*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 112–122.

[14] C. Hazard and K. Ramdani, *Selective acoustic focusing using time-harmonic reversal mirrors*, SIAM J. Appl. Math., 64 (2004), pp. 1057–1076.
[15] R. A. Horn and C. R. Johnson, *Matrix Analysis,* Cambridge University Press, New York, 1994.
[16] A. Kirsch, *The MUSIC algorithm and the factorisation method in inverse scattering theory for inhomogeneous media*, Inverse Problems, 18 (2002), pp. 1025–1040.
[17] S. K. Lehman and A. J. Devaney, *Transmission mode time-reversal super-resolution imaging*, J. Acoust. Soc. Amer., 113 (2003), pp. 2742–2753.
[18] T. D. Mast, A. I. Nachman, and R. C. Waag, *Focusing and imaging using eigenfunctions of the scattering operator*, J. Acoust. Soc. Amer., 102 (1997), pp. 715–725.
[19] G. Micolau and M. Saillard, *DORT method as applied to electromagnetic subsurface sensing*, Radio Sci., 38 (2003), pp. 1038–1049.
[20] J. C. Mosher and R. M. Leahy, *Source localization using recursively applied and projected (RAP) MUSIC*, IEEE Trans. Signal Processing, 47 (1999), pp. 332–340.
[21] C. Prada and J.-L. Thomas, *Experimental subwavelength localization of scatterers by decomposition of the time reversal operator interpreted as a covariance matrix*, J. Acoust. Soc. Amer., 114 (2003), pp. 235–243.
[22] C. Prada, J.-L. Thomas, and M. Fink, *The iterative time reversal process: Analysis of the convergence*, J. Acoust. Soc. Amer., 97 (1995), pp. 62–71.
[23] C. W. Therrien, *Discrete Random Signals and Statistical Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[24] M. S. Vogelius and D. Volkov, *Asymptotic formulas for perturbations in the electromagnetic fields due to the presence of inhomogeneities*, Math. Model. Numer. Anal., 34 (2000), pp. 723–748.
[25] D. Volkov, *Numerical methods for locating small dielectric inhomogeneities*, Wave Motion, 38 (2003), pp. 189–206.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 159–183
© 2005 Society for Industrial and Applied Mathematics

WHY ARE HIGH-DIMENSIONAL FINANCE PROBLEMS OFTEN OF LOW EFFECTIVE DIMENSION?*

XIAOQUN WANG† AND IAN H. SLOAN‡

Abstract. Many problems in mathematical finance can be formulated as high-dimensional integrals, where the large number of dimensions arises from small time steps in time discretization and/or a large number of state variables. Quasi–Monte Carlo (QMC) methods have been successfully used for approximating such integrals. To understand this success, this paper focuses on investigating the special features of some typical high-dimensional finance problems, namely, option pricing and bond valuation. We provide new insight into the connection between the effective dimension and the efficiency of QMC, and we present methods to analyze the dimension structure of a function. We confirm the observation of Caflisch, Morokoff, and Owen that functions from finance are often of low effective dimension, in the sense that they can be well approximated by their low-order ANOVA (analysis of variance) terms, usually just the order-1 and order-2 terms. We explore *why* the effective dimension is small for many integrals from finance. By deriving explicit forms of the ANOVA terms in simple cases, we find that the importance of each dimension is naturally weighted by certain hidden weights. These weights characterize the relative importance of different variables or groups of variables and limit the importance of the higher-order ANOVA terms. We study the variance ratios captured by low-order ANOVA terms and their asymptotic properties as the dimension tends to infinity, and we show that with the increase of dimension the lower-order terms continue to play a significant role and the higher-order terms tend to be negligible. This provides some insight into high-dimensional problems from finance and explains why QMC algorithms are efficient for problems of this kind.

Key words. quasi–Monte Carlo, effective dimension, option pricing, bond valuation

AMS subject classifications. 65C05, 65D30, 91B28

DOI. 10.1137/S1064827503429429

1. Introduction.

Many problems in mathematical finance can be formulated as high-dimensional integrals, or, after appropriate transformations as integrals over the d-dimensional unit cube,

$$I_d(f) = \int_{[0,1]^d} f(\mathbf{x}) d\mathbf{x}.$$

For large d it is well known that classical methods are not feasible because of the curse of dimension, but it is by now also well known that Monte Carlo (MC) methods (see Boyle, Broadie, and Glasserman [2]) and quasi–Monte Carlo (QMC) methods (see Paskov and Traub [19]) can provide powerful tools for approximating such integrals. The MC estimate in its simplest form is

$$Q_{n,d}(f) = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i), \qquad \mathbf{x}_i \in [0,1]^d,$$

*Received by the editors June 10, 2003; accepted for publication (in revised form) December 15, 2004; published electronically August 22, 2005. This work was supported by the National Science Foundation of China and by the Australian Research Council under its Centre of Excellence program.
http://www.siam.org/journals/sisc/27-1/42942.html

†Department of Mathematical Sciences, Tsinghua University, Beijing 100084, China (xwang@math.tsinghua.edu.cn).

‡School of Mathematics, University of New South Wales, Sydney 2052, Australia (i.sloan@unsw.edu.au).

where the points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are independent and identically distributed (i.i.d.) random samples from the uniform distribution on $[0,1]^d$. The QMC methods take the same form, but now with $\mathbf{x}_1, \ldots, \mathbf{x}_n$ chosen deterministically, so as to yield better uniformity than the random samples.

The apparent success in the 1990s of QMC methods for some high-dimensional problems in finance (Paskov and Traub [19]) was a cause for surprise, given that the error bound given by the Koksma–Hlawka inequality (Niederreiter [14]) is of order $O(n^{-1}(\log n)^d)$ as n increases, which for large d increases with increasing n until we reach astronomically large values of n. Some other research also showed the high efficiency of QMC methods for high-dimensional integrals in finance (see Acworth, Broadie, and Glasserman [1], Joy, Boyle, and Tan [10], and Ninomiya and Tezuka [15]). Sparse grids may also have good performance for finance applications (see Gerstner and Griebel [4]). It is by now well accepted, following the work of Caflisch, Morokoff, and Owen [3], Imai and Tan [9], Owen [17], Paskov [18], and Wang and Fang [28], that the qualitative explanation lies in the fact that the [illegible] is small, even though the nominal dimension is very large. (Two notions of effective dimension, namely, the truncation and superposition dimensions, are defined precisely in section 2.)

The main purpose of this paper is to explore [illegible] the effective dimension is small for many integrals from finance. We do this by studying some simple model problems in finance and exploring what happens (as far as possible by analytic means) as the nominal dimension d approaches ∞. There are two main sources of the high dimensionality of d: one is time discretization (for example, a path-dependent option pricing model can be thought of as arising from discretization of an underlying continuous process), and the other is the existence of a large number of different assets or securities.

In the case of time discretization, the number of time steps (and hence the dimension) increases as the time step shortens, but the problem as it relates to a single time step becomes in a certain sense simpler. For a simplified option pricing situation (geometric Asian call option with zero strike price) we show that the superposition dimension actually has a limit as $d \to \infty$ (see Theorem 5.1). In this situation the high value of d becomes in a sense irrelevant. For nonzero but small strike prices the situation is little changed. For multiasset options the situation can be even more favorable, with the superposition dimension approaching 1 as $d \to \infty$ in the case of zero strike price. It is natural to ask whether the property of low superposition dimension is common for finance problems. We thus study one more problem, namely, the bond valuation (see Theorem 5.2). Since the main focus of this paper is directed towards the possible special properties of finance problems and the reasons for such properties, we limit ourselves to simple models where closed solutions may also exist.

Explaining the surprisingly good performance of QMC methods for some high-dimensional integrals is a challenging problem. A possible theoretical explanation is given in Sloan and Woźniakowski [22] by introducing [illegible] function classes, where the variables are given weights depending on their importance. A number of works have been devoted to studying the (strong) tractability of multivariate integration, as well as constructing algorithms that achieve the corresponding (strong) tractability error bounds (see Hickernell and Wang [7], Wang [27], and Sloan, Kuo, and Joe [20]).

This paper is organized as follows. After introducing the analysis of variance (ANOVA) decomposition and effective dimensions in section 2, we discuss the relationship of effective dimension to the integration error and perform a numerical experiment that illustrates the relationship. In section 3 we discuss methods to ana-

lyze the effective dimensions and variance ratios. In section 4, we consider the pricing of path-dependent and multiasset options and show numerically that these nominally high-dimensional problems are often low-dimensional in the sense that the functions are dominated by their lower-order ANOVA terms (usually just the order-1 and order-2 terms). In section 5 we come to the main results of the paper, where we investigate high-dimensional option pricing and bond valuation problems are often of low superposition dimension: we do this by identifying the inherent in the problems that control the relative importance of different (groups of) variables and by studying the variance ratios and their asymptotic properties (as $d \to \infty$). These weights often become smaller as the nominal dimension d increases, offsetting the increased inherent difficulty caused by the high dimensionality. The problems of nonsmoothness involved in the integrands are also discussed. Concluding remarks are presented in the last section.

2. The effective dimensions of functions.

2.1. The definitions of effective dimensions and variance ratios. Let $\mathcal{A} = \mathcal{A}_d = \{1, 2, \ldots, d\}$. For any subset $u \subseteq \mathcal{A}$, let $|u|$ denote its cardinality and let $\mathcal{A} - u$ denote its complementary set in $\mathcal{A}$. Consider a square-integrable function f defined on $[0,1]^d$. We say that the expansion

$$f(\mathbf{x}) = \sum_{u \subseteq \mathcal{A}} f_u(\mathbf{x}) \tag{1}$$

is an ANOVA decomposition of f if for each f_u with $\emptyset \neq u \subseteq \mathcal{A}$ the following is satisfied:

$$\int_0^1 f_u(\mathbf{x}) dx_j = 0 \ \text{ if } \ j \in u. \tag{2}$$

It is easy to show that $f_\emptyset = I_d(f)$ and that $f_u(\mathbf{x})$ can be determined recursively by

$$f_u(\mathbf{x}) = \int_{[0,1]^{d-|u|}} f(\mathbf{x}) d\mathbf{x}_{\mathcal{A}-u} - \sum_{v \subset u} f_v(\mathbf{x}) \quad \text{for} \quad \emptyset \neq u \subseteq \mathcal{A},$$

where the last sum is over strict subsets v of u. Each term f_u expresses the cooperative contribution of a distinct group of variables to f. It follows from (2) that the ANOVA decomposition (1) is orthogonal: $\int_{[0,1]^d} f_u(\mathbf{x}) f_v(\mathbf{x}) d\mathbf{x} = 0$ whenever $u \neq v$. Therefore,

$$\sigma^2(f) = \sum_{\emptyset \neq u \subseteq \mathcal{A}} \sigma_u^2(f),$$

where $\sigma^2(f) = I_d(f^2) - [I_d(f)]^2$ is the variance of f and $\sigma_u^2(f) = \int_{[0,1]^d} [f_u(\mathbf{x})]^2 d\mathbf{x}$ is the variance of f_u (for $|u| > 0$). The of the ANOVA term f_u is $|u|$, the cardinality of u.

Let u be a nonempty subset of $\mathcal{A}$. The variance corresponding to u and all its subsets is defined as

$$T_u(f) := \sum_{\emptyset \neq v \subseteq u} \sigma_v^2(f). \tag{3}$$

The function f has d_t (called the , or "TD" for short) if

$$T_{\{1,\ldots,d_t\}}(f) \geq p\, \sigma^2(f);$$

it has [illegible] d_s [illegible] (called the [illegible] , or "SD") if

$$\sum_{0<|u|\le d_s} \sigma_u^2(f) \ge p\,\sigma^2(f),$$

where p is a parameter close to 1. In this paper, we choose $p = 0.99$. The notion of effective dimension was introduced by Caflisch, Morokoff, and Owen [3]. The TD is roughly the number of important variables, if the variables are ordered by their importance. The SD does not depend on the order in which the variables are indexed and is the more useful concept when all variables are equally important (e.g., isotropic functions).

The effective dimension depends on p. To avoid this, we introduce two related notions that are independent of p. The [illegible] are defined as the ratios

$$S_u(f) = \sigma_u^2(f)/\sigma^2(f) \quad \text{for} \quad \emptyset \ne u \subseteq \mathcal{A},$$

which measure the relative importance of f_u (see Sobol [24]). The [illegible] captured by all order-ℓ ANOVA terms are defined as

$$R_{(\ell)}(f) = \sum_{|u|=\ell} \sigma_u^2(f)/\sigma^2(f) = \sum_{|u|=\ell} S_u(f),$$

which indicate the relative importance of all order-ℓ ANOVA terms taken together.

2.2. How does QMC error depend on effective dimension? The relation between QMC error and effective dimension can be understood in terms of variants of the Koksma–Hlawka inequality. The classical Koksma–Hlawka inequality (see Niederreiter [14]) is

$$|I_d(f) - Q_{n,d}(f)| < D^*_{n,d}(P_n)V(f),$$

where $P_n := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ is the set of QMC points, $V(f)$ is the variation of f in the sense of Hardy and Krause, and $D^*_{n,d}(P_n)$ is the [illegible] of P_n, which is a geometric measure of the departure from uniformity of P_n. It is the growth of the star discrepancy with n that is the source of the poor behavior when d is large, as mentioned in the introduction.

If f can be well approximated by just a few ANOVA terms, for example,

$$f \approx \sum_{|u|\le d_0,\, u\subseteq\mathcal{A}_d} f_u,$$

where $d_0 = 2$ or 3, then it would seem better to use an error bound of the form

$$|I_d(f) - Q_{n,d}(f)| \le \sum_{\phi\ne u\subseteq\mathcal{A}_d} D_u(P_n)\|f_u\|. \tag{4}$$

A bound of this form was developed by Hickernell [5, 6], with $\|f_u\|$, the norm of f_u in a certain $|u|$-dimensional Hilbert space, given there by

$$\|f_u\|^2 = \int_{[0,1]^{|u|}} \left|\frac{\partial^{|u|} f_u}{\partial x_u}(\mathbf{x}_u)\right|^2 d\mathbf{x}_u,$$

and $D_u(P_n)$ a positive number that depends only on the projection $P_{n,u}$ of P_n obtained by neglecting the components x_{ji} of each point $\mathbf{x}_i \in P_n$ for which $j \notin u$.

To see how QMC error depends on the effective dimension, consider two extreme cases. The first case is that f is a function of the form $f(\mathbf{x}) = \sum_{u \subseteq \{1,\dots,d_0\}} f_u(\mathbf{x})$ for some $d_0 < d$. From (4), the error $I_d(f) - Q_{n,d}(f)$ depends only on $D_u(P)$ with $u \subseteq \{1,\dots,d_0\}$. The second case is that f is a function such that $f(\mathbf{x}) = \sum_{|u| \le d_0} f_u(\mathbf{x})$ for some d_0. Based on (4), the error depends only on $D_u(P)$ with $|u| \le d_0$. For these extreme cases, if d_0 is small, good results can be expected by using QMC.

Of course, it will not usually happen that f is exactly low-dimensional as in the extreme cases, but it is often the case that f can be "close" to low-dimensional. Let

$$h(\mathbf{x}) := \sum_{u \subseteq \{1,\dots,d_0\}} f_u(\mathbf{x}) \quad \text{or} \quad h(\mathbf{x}) := \sum_{|u| \le d_0} f_u(\mathbf{x}),$$

where f_u are the ANOVA terms of f, and d_0 is small. Let $\rho := \sigma^2(h)/\sigma^2(f)$ be the variance ratio captured by h. Then, since $I_d(f-h) = 0$ and $f - h$ is orthogonal to h,

$$||f-h||_2^2 = \int_{[0,1]^d} [f(\mathbf{x}) - h(\mathbf{x})]^2 d\mathbf{x} = \sigma^2(f-h) = (1-\rho)\,\sigma^2(f).$$

If ρ is close to 1, then f can be well approximated in the L_2-norm by h. By splitting f into the sum of two parts, $f = h + (f-h)$, we have

$$|I_d(f) - Q_{n,d}(f)| \le |I_d(h) - Q_{n,d}(h)| + |I_d(f-h) - Q_{n,d}(f-h)|. \tag{5}$$

The analysis for the first term is the same as above. It is reasonable to assume that

$$|I_d(h) - Q_{n,d}(h)| \le a\,\sigma(h)/\sqrt{n} \quad \text{for some } a > 0.$$

The parameter a measures how large the QMC error for h is in comparison to the MC error for h. In general, a is small. In our experience, for common low discrepancy point sets and for n in the thousands, it is common that $a < 1/10$ or even $a < 1/100$.

Now consider the second term on the right-hand side of (5). Since

$$\mathbb{E}[|I_d(f-h) - Q_{n,d}^{\mathrm{MC}}(f-h)|^2] = \sigma^2(f-h)/n,$$

where $Q_{n,d}^{\mathrm{MC}}$ denotes a quadrature with random nodes $\mathbf{x}_1,\dots,\mathbf{x}_n$, Chebyshev's inequality yields

$$\lambda\left(\left\{(\mathbf{x}_1,\dots,\mathbf{x}_n) : |I_d(f-h) - Q_{n,d}^{\mathrm{MC}}(f-h)| \le b\,\sigma(f-h)/\sqrt{n}\right\}\right) \ge 1 - b^{-2}.$$

Here λ is the Lebesgue measure and b is an arbitrary positive number. For instance, with $b = 5$, the probability of the quadratures achieving

$$|I_d(f-h) - Q_{n,d}^{\mathrm{MC}}(f-h)| \le 5\,\sigma(f-h)/\sqrt{n}$$

is at least 0.96. Thus the error of an arbitrary realization of random quadratures would not be much worse than $\sigma(f-h)/\sqrt{n}$. Experience suggests that a QMC rule rarely gives a much worse result than MC. Suppose that for some $b > 0$,

$$|I_d(f-h) - Q_{n,d}(f-h)| \le b\,\sigma(f-h)/\sqrt{n}.$$

From the analysis above, we have

$$|I_d(f) - Q_{n,d}(f)| \le a\sigma(h)/\sqrt{n} + b\sigma(f-h)/\sqrt{n} = \left(a\sqrt{\rho} + b\sqrt{1-\rho}\right)\,\sigma(f)/\sqrt{n}.$$

As a benchmark, we recall that the root mean square of the MC error is $\sigma(f)/\sqrt{n}$. Thus the relative QMC error with respect to MC error depends on the following:

- How well does QMC work for h or $f-h$ with respect to MC? This is characterized by a or b, respectively.
- How well can f be approximated in the L_2-norm by h (or how small is the variance of $f-h$ with respect to that of f)? This is measured by ρ.

To get a feeling of the QMC error with comparing to MC error, consider two cases:

- Normal case: $a=1/100$, $b=1$, $\rho=0.99$; then $|I_d(f)-Q_{n,d}(f)|\le 0.110\,\sigma(f)/\sqrt{n}$.
- Optimistic case: $a=1/1000$, $b=1$, $\rho=0.999$; then $|I_d(f)-Q_{n,d}(f)|\le 0.003\sigma(f)/\sqrt{n}$.

Effective dimensions can be viewed as measures of how well f can be approximated by a sum of lower-dimensional functions. For example, let $h(\mathbf{x}) := \sum_{|u|\le d_s} f_u(\mathbf{x})$, where d_s is the SD; then by definition

$$||f-h||_2^2 \le (1-p)\sigma^2(f) = 0.01\sigma^2(f).$$

If d_s is small (say, $d_s \le 2$), then even if the TD d_t is large, we have good reason to expect superiority of QMC over MC.

It is useful to perform an experiment to see how the QMC error depends on effective dimensions. Consider the functions

$$f(\mathbf{x}) = \prod_{j=1}^{d} \left(1 + a\tau^j\,(x_j - 1/2)\right), \tag{6}$$

where a and τ are parameters which control the effective dimension (a mainly controls the SD, while τ mainly controls the TD). Obviously, $I_d(f)=1$ for all a,τ, and d. The effective dimensions can be easily determined by the method of Wang and Fang [28]. Table 1 presents the effective dimensions and the root mean square errors (RMSEs) of MC and QMC in dimension $d=50$. In MC, the RMSE is computed as $\sigma(f)/\sqrt{n}$. In QMC, the sequence of Sobol [23] is used and the RMSE is computed based on 30 random shifts. The ratios of the RMSE of QMC with respect to that of MC are also given. The dependence of the QMC error on the effective dimensions is clear from the comparisons. In summary, the superiority of QMC is observed for two classes of functions:

- The class of functions with small TD (see the columns with $\tau=0.1$ and $\tau=0.5$ in Table 1). The efficiency of QMC for this class of functions is especially high.
- The class of functions with small SD (especially if the order-1 part plays the major role). In this case the TD can be large (see the row with $a=0.1$ in Table 1). Note that when $\tau=1$, the function f depends equally on the variables, but different orders of ANOVA terms may have quite different contribution to f.

We do not observe the superiority of QMC if both the TD and SD are large (say, $d_t>30$ and $d_s>5$). In fact, in such cases (see the case of $a=10$ and $\tau=1$), both MC and QMC may give unreasonable results (this can be explained by the large variance: $\sigma^2(f)\approx 3.18\times 10^{48}$ for $a=10,\tau=1$, and $d=50$). There is no sense in comparing their RMSEs in this case. Note that it is shown in Owen [16] that low SD is necessary for QMC to be much better than MC with practical sample size n.

3. Computing the effective dimensions.

3.1. Estimating the effective dimensions and variance ratios. In some specific cases, simple formulas are available for computing the effective dimensions

TABLE 1

The TD and SD (d_t, d_s), the RMSE for function (6) in dimension $d = 50, n = 2^{14}$. The ratios of the RMSE of QMC with respect to the RMSE of MC are in the parentheses.

		$\tau = 0.1$	$\tau = 0.5$	$\tau = 0.8$	$\tau = 0.9$	$\tau = 1$
$a = 0.1$	(d_t, d_s)	(2, 1)	(4, 1)	(11, 1)	(22, 1)	(50, 2)
	MC	2.27e-5	1.30e-4	3.01e-4	4.66e-4	1.61e-3
	QMC	1.82e-7	1.38e-6	8.47e-6	4.29e-5	6.71e-4
		(0.008)	(0.011)	(0.028)	(0.092)	(0.416)
$a = 1$	(d_t, d_s)	(2, 1)	(4, 1)	(11, 2)	(23, 3)	(50, 9)
	MC	2.27e-4	1.31e-3	3.10e-3	5.05e-3	5.73e-2
	QMC	1.83e-6	1.64e-5	6.75e-4	4.24e-3	8.00e-2
		(0.008)	(0.013)	(0.218)	(0.840)	(1.400)
$a = 10$	(d_t, d_s)	(2, 1)	(5, 3)	(17, 8)	(39, 15)	(50, 49)
	MC	2.27e-3	1.66e-2	3.12e-1	3.51e+1	1.39e+22
	QMC	1.98e-5	1.21e-3	2.78e-1	2.69e+1	4.27e+20
		(0.009)	(0.073)	(0.891)	(0.769)	− − −

and variance ratios, but in the general case numerical algorithms have to be used. A numerical algorithm to determine the TD of an arbitrary function is given in Wang and Fang [28]. For any fixed set $u \subseteq \mathcal{A}$, write $\mathbf{x} = (\mathbf{x}_u, \mathbf{x}_{\mathcal{A}-u})$ and $\mathbf{y} = (\mathbf{y}_u, \mathbf{y}_{\mathcal{A}-u})$. The variance corresponding to u and all its subsets defined in (3) can be expressed as (see Sobol [24])

$$T_u(f) = \int_{[0,1]^{2d-|u|}} f(\mathbf{x}) f(\mathbf{x}_u, \mathbf{y}_{\mathcal{A}-u}) d\mathbf{x} d\mathbf{y}_{\mathcal{A}-u} - [I_d(f)]^2. \tag{7}$$

The TD can be determined by computing $T_u(f)$ with $u = \{1, \dots, \ell\}$ for $\ell = 1, 2, \dots$.

Determining the SD of an arbitrary function is much more difficult, since we have to compute all the variances $\sigma_u^2(f)$ for subsets u up to some order. In principle, this can be done as follows. For $|u| = 1$, $\sigma_u^2(f) = T_u(f)$, so it can be computed directly by (7). For $|u| = 2$ and, say, $u = \{i, j\}$, $\sigma_{\{i,j\}}^2(f)$ can be estimated from

$$T_{\{i,j\}}(f) = \sigma_{\{i\}}^2(f) + \sigma_{\{j\}}^2(f) + \sigma_{\{i,j\}}^2(f),$$

where $T_{\{i,j\}}(f)$ can be computed by (7). Continuing this process, all the variances $\sigma_u^2(f)$ for subsets u up to some order can be computed. For large d, such an approach is computationally expensive. Moreover, for large $|u|$ (say, $|u| \geq 3$) there is often a loss of accuracy. If f is a symmetric function, the computation can be simplified, since now $\sigma_u^2(f)$ depends on u only through $|u|$.

For large d, one strategy to verify whether a function has low SD is to compute the variance ratios of low-order ANOVA terms, such as the order-1 and order-2 terms. An alternative is to compute the mean dimension in the superposition sense (see Owen [17] and Liu and Owen [11]).

3.2. A special class of functions. Computing $T_u(f)$ is an important step in determining both the TD and SD. The dimension of the resulting integrals for $T_u(f)$ can be very large. We show that such integrals can be reduced to three-dimensional integrals in some cases. In fact, some finance problems are related to the computations of integrals of the form (see the next section for examples)

$$J := \int_{\mathbb{R}^d} \max(0,\ e^{F(\mathbf{z})} - K) p_d(\mathbf{z}) d\mathbf{z} \quad \text{with} \quad F(\mathbf{z}) = \sum_{j=1}^{d} a_j z_j, \tag{8}$$

where a_j, K are constants and

$$p_d(\mathbf{z}) = (2\pi)^{-d/2} \exp\left(-\frac{1}{2}\sum_{j=1}^{d} z_j^2\right) \tag{9}$$

is the density of the standard normal distribution. The integral (8) can be written as a d-dimensional integral over the unit cube $[0,1]^d$:

$$J = \int_{[0,1]^d} f(\mathbf{x})d\mathbf{x} \text{ with } f(\mathbf{x}) = \max(0,\ \exp(F(\Phi^{-1}(x_1), \ldots, \Phi^{-1}(x_d))) - K),$$

where $\Phi(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$ is the normal probability integral and Φ^{-1} is its inverse.

The integral (8) can be reduced to a one-dimensional integral, thereby allowing a useful test of the efficiency of d-dimensional QMC algorithms for such integrals. For the vector $\mathbf{z} = (z_1, \ldots, z_d)^T$ with the density function $p_d(\mathbf{z})$, the variable $\sum_{j=1}^{d} a_j z_j$ is a normal random variable with mean zero and variance $b^2 := \sum_{j=1}^{d} a_j^2$. Put $w := b^{-1} \sum_{j=1}^{d} a_j z_j$, so that w is a standard normal random variable. Then we can write

$$J = \int_{-\infty}^{\infty} \max(0,\ e^{bw} - K) p_1(w) dw = \int_0^1 \max(0, e^{b\Phi^{-1}(x)} - K) dx.$$

A similar method can be used to transform the integral involved in $T_u(f)$ (see (7)). Indeed, for any given nonempty set $u \subseteq \mathcal{A}$, put $b_u^2 := \sum_{j \in u} a_j^2$. According to Sobol's formula (7), we have

$$T_u(f) + [I_d(f)]^2 = \int_0^1 \int_0^1 \int_0^1 F^u(x,y) F^u(x,z) dx dy dz,$$

where

$$F^u(x,y) = \max(0,\ \exp[b_u \Phi^{-1}(x) + b_{\mathcal{A}-u} \Phi^{-1}(y)] - K).$$

Thus the integrals involved in $T_u(f)$ can be reduced to three-dimensional integrals. This reduces the computational cost significantly. This method will be used in the next section.

4. High-dimensional finance problems are often of low effective dimension. In this section we investigate empirically the nature of option pricing problems by estimating the SD. The TD for these problems has been computed in Wang and Fang [28]. It turned out that the TD is almost as large as the nominal dimension d (if the Brownian motion is generated by the standard method), and thus the success of QMC in this case cannot be explained by a small TD.

4.1. Path-dependent options. Consider the pricing of a path-dependent option with payoff $g(S_{t_1}, \ldots, S_{t_d})$, where $S_{t_1}, \ldots, S_{t_d}$ are the prices of the underlying asset at times $t_1, \ldots, t_d$. Suppose the prices are sampled at equally spaced times $t_0 = 0, t_j = t_{j-1} + \Delta t, j = 1, \ldots, d, \Delta t = T/d$, where T is the expiration date. For simplicity of presentation, assume that under [illegible] (i.e., [illegible]) the underlying asset follows geometric Brownian motion:

$$dS_t = rS_t dt + \sigma S_t dB_t, \tag{10}$$

where r is the risk-free interest rate, σ is the volatility, and B_t is the standard Brownian motion. The analytical solution to (10) is

$$S_t = S_0 \exp((r - \sigma^2/2)t + \sigma B_t). \tag{11}$$

Based on the risk-neutral valuation (see Hull [8]), the value of the option at $t = 0$ is

$$\mathbb{E}[e^{-rT} g(S_{t_1}, \ldots, S_{t_d})],$$

where $\mathbb{E}[\cdot]$ is the expectation under the risk-neutral measure.

Let $(y_1, \ldots, y_d)^T := (B_{t_1}, \ldots, B_{t_d})^T$. It is normally distributed with mean zero and covariance matrix $V = (\min(t_i, t_j))_{i,j=1}^d$. Defining $\mu_j = \log S_0 + (r - \sigma^2/2)t_j$, from (11) we have $S_{t_j} = \exp(\mu_j + \sigma y_j)$. The payoff can be written as

$$g(S_{t_1}, \ldots, S_{t_d}) = g(e^{\mu_1 + \sigma y_1}, \ldots, e^{\mu_d + \sigma y_d}) =: H(\mathbf{y}).$$

The value of the option at time $t = 0$ can then be written as

$$\begin{aligned} \mathbb{E}[e^{-rT} g(S_{t_1}, \ldots, S_{t_d})] &= \frac{e^{-rT}}{(2\pi)^{d/2}\sqrt{\det V}} \int_{\mathbb{R}^d} H(\mathbf{y}) e^{-\frac{1}{2}\mathbf{y}^T V^{-1}\mathbf{y}} d\mathbf{y} \\ &= e^{-rT} \int_{\mathbb{R}^d} H(A\mathbf{z}) p_d(\mathbf{z}) d\mathbf{z}, \end{aligned}$$

where a linear transformation $(y_1, \ldots, y_d)^T = A(z_1, \ldots, z_d)^T$ is introduced using an arbitrary $d \times d$ matrix A satisfying $AA^T = V$. This change of variables can be interpreted as a covariance matrix decomposition $V = AA^T$ or as a generation of the Brownian motion:

$$(B_{t_1}, \ldots, B_{t_d})^T = A(z_1, \ldots, z_d)^T, \tag{12}$$

where $z_j \sim N(0,1), j = 1, \ldots, d$, are i.i.d. standard normal variables. The [illegible] (or [illegible]) generates the Brownian motion sequentially in time:

$$B_0 = 0, \;\; B_{t_j} = B_{t_{j-1}} + \sqrt{\Delta t}\, z_j, \;\; z_j \sim N(0,1), \quad j = 1, \ldots, d. \tag{13}$$

Consider Asian call options based on the geometric or arithmetic average of the underlying asset. Their terminal payoffs are, respectively, $\max(0, \prod_{j=1}^d S_{t_j}^{1/d} - K)$ or $\max(0, \frac{1}{d}\sum_{j=1}^d S_{t_j} - K)$, where K is the strike price at T. For the geometric average, we have

$$\prod_{j=1}^d S_{t_j}^{1/d} = \exp\left(m + \frac{\sigma}{d}\sum_{j=1}^d y_j\right) = \exp\left(m + \frac{\sigma}{d}\sum_{k=1}^d A_k z_k\right),$$

where $A_k = \sum_{j=1}^d a_{jk}$, with a_{jk} the elements of A, and

$$m := m_d = \frac{1}{d}\sum_{j=1}^d \mu_j = \log S_0 + \frac{1}{2}\left(r - \frac{\sigma^2}{2}\right)(T + \Delta t). \tag{14}$$

Thus for the geometric average case the resulting integral $\mathbb{E}[e^{-rT} g(S_{t_1}, \ldots, S_{t_d})]$ has the special form (8), excluding a constant factor. The computation of the variance ratios can be significantly simplified, as shown in section 3.2.

TABLE 2

The variance ratios (in percentage) captured by order-1 and order-2 ANOVA terms for geometric Asian options: $S_0 = 100$, $\sigma = 0.2$, $r = 0.1$, $T = 1$ year.

Dimension	$K = 90$		$K = 100$		$K = 110$	
	order-1	order-2	order-1	order-2	order-1	order-2
8	95.58	4.00	85.00	13.74	65.79	29.93
16	95.38	4.40	83.20	15.60	60.76	33.99
32	95.26	4.78	82.22	17.81	57.99	37.71
64	95.12	5.05	81.63	19.40	56.46	40.38

TABLE 3

The same as Table 2, but for arithmetic Asian options, look-back options, and barrier options ($B_a = K - 10$). Look-back options do not depend on the strike price.

Options	Dimension	$K = 90$		$K = 100$		$K = 110$	
		order-1	order-2	order-1	order-2	order-1	order-2
Arithmetic Asian option	4	96.16	3.56	88.52	10.67	74.89	22.90
	8	95.12	4.99	84.53	16.12	65.02	35.17
	16	94.71	5.10	82.22	17.30	60.10	38.42
Look-back option	4	92.68	6.12	92.68	6.12	92.68	6.12
	8	89.49	6.59	89.49	6.59	89.49	6.59
	16	88.56	7.21	88.56	7.21	88.56	7.21
Barrier option	4	92.70	6.17	81.09	16.49	60.94	32.97
	8	91.90	4.87	76.61	19.38	47.94	41.49
	16	90.75	4.04	74.31	21.56	46.37	43.72

We will also consider other path-dependent options look-back and barrier options. For a European look-back call option, the payoff is $g(S_{t_1}, \ldots, S_{t_d}) = S_T - \min\{S_{t_1}, \ldots, S_{t_d}\}$. For a European down-and-out call option, the payoff is (with barrier B_a)

$$g(S_{t_1}, \ldots, S_{t_d}) = \begin{cases} \max(0,\ S_T - K) & \text{if, for all } j = 1, \ldots, d, S_{t_j} > B_a, \\ 0 & \text{if, for some } j = 1, \ldots, d, S_{t_j} \le B_a. \end{cases}$$

In our investigations the Brownian motion is generated by the standard construction (13). All integrals involved in the computations of the variance ratios are approximated by QMC based on Sobol points with $n = 2^{18}$. In Tables 2 and 3 we present the computational results for the variance ratios captured by order-1 and order-2 ANOVA terms. (Due to integration and rounding errors, in some cases $100R_{(1)}(f) + 100R_{(2)}(f) > 100$.) For geometric Asian options (Table 2), we observe that for $K = 90$ and $K = 100$ the order-1 and order-2 terms capture more than 99% of the total variance; for $K = 110$, the percentage is more than 94%. The strike price K has some effects on the variance ratios: when K increases, $R_{(1)}(f)$ decreases, while $R_{(2)}(f)$ increases; however, their sum is always close to 1.

The computations of the variance ratios for the other options are more difficult, since the method in section 3.2 cannot be used. We have to use the method in section 3.1 and must restrict the value of d. We observe a similar phenomenon for these options (Table 3).

4.2. Multiasset options. Consider a European multiasset derivative with terminal payoff $\phi(S_T^1, \ldots, S_T^d)$, where $S_T^1, \ldots, S_T^d$ are the prices of d risky assets at time T. Assume that in a risk-neutral measure the prices $(S_t^1, \ldots, S_t^d)$ of the assets satisfy

$$dS_t^j = rS_t^j dt + \sigma_j S_t^j dB_t^j, \quad j = 1, \ldots, d,$$

where σ_j are volatilities and $B_t^1, \ldots, B_t^d$ are correlated Brownian motions with correlations ρ_{ij}. The solutions to the stochastic differential equations are given by

$$S_t^j = S_0^j \exp((r - \sigma_j^2/2)t + \sigma_j B_t^j), \quad j = 1, \ldots, d. \tag{15}$$

Note that the random vector $(B_T^1, \ldots, B_T^d)^T =: (y_1, \ldots, y_d)^T$ is normally distributed with mean zero and covariance matrix

$$\Sigma = (\rho_{ij} T)_{i,j=1}^d. \tag{16}$$

Put $\nu_j := \log S_0^j + (r - \frac{1}{2}\sigma_j^2)T$. Since $S_T^j = e^{\nu_j + \sigma_j y_j}$, the payoff can be written as $\phi(S_T^1, \ldots, S_T^d) = \phi(e^{\nu_1 + \sigma_1 y_1}, \ldots, e^{\nu_d + \sigma_d y_d}) =: G(\mathbf{y})$. Therefore, the current price of the derivative security can be expressed as

$$\mathbb{E}[e^{-rT}\phi(S_T^1, \ldots, S_T^d)] = e^{-rT} \int_{\mathbb{R}^d} G(A\mathbf{z}) p_d(\mathbf{z}) d\mathbf{z},$$

where A is any real $d \times d$ matrix satisfying $AA^T = \Sigma$. This matrix A corresponds to a method of generating the correlated Brownian motions:

$$(B_T^1, \ldots, B_T^d)^T = A\,(z_1, \ldots, z_d)^T.$$

The standard method corresponds to the Cholesky decomposition of the matrix $\Sigma = AA^T$. This is the choice we use for our analysis and numerical computations.

For a European call option on the geometric average over the d assets, we have

$$\phi(S_T^1, \ldots, S_T^d) = \max\left(0,\ \exp\left(\frac{1}{d}\sum_{j=1}^d (\nu_j + \sigma_j y_j)\right) - K\right). \tag{17}$$

Just the same as for the geometric Asian option, the corresponding integral also has the special form (8), allowing a faster computation of the variance ratios. For a call option on the arithmetic average over the d assets, the payoff is $\phi(S_T^1, \ldots, S_T^d) = \max(0,\ \frac{1}{d}\sum_{j=1}^d S_T^j - K)$.

In Tables 4 and 5 we present the computational results. The conclusion is similar to the cases of path-dependent options. For $K = 90$ and $K = 100$, the order-1 and order-2 terms capture over 99% of the total variance; for $K = 110$, the percentage is larger than 97%. A new feature is that as the nominal dimension increases, the sum of the variance ratios of the order-1 and order-2 terms may increase (for example, for the cases of $K = 90$ and 100). We also observe that regarding the variance ratios, there is not much difference between the geometric and arithmetic Asian options (though their analytical tractability is different).

5. Why are high-dimensional finance problems often of low effective dimension? Very high-dimensional problems often occur in finance because of small time increments and/or a large number of assets or other state variables. In the previous section we found numerically that these nominally high-dimensional problems are often of low SD (even if dimension reduction techniques are not used). Here we try to investigate theoretically the reason for this phenomenon, i.e., why the effective dimension is small. It is important to know what controls the complexity of the problems, and how the quantities relating to dimension (e.g., variance ratios and effective dimensions) change as the number of time steps and/or the number of risky

TABLE 4
The variance ratios (in percentage) captured by the order-1 and order-2 ANOVA terms for geometric multiasset options: $S_0^j = 100, \sigma_j = 0.2, \rho_{ij} = 0.3 (i \neq j), r = 0.1, T = 1$.

Dimension	$K = 90$		$K = 100$		$K = 110$	
	order-1	order-2	order-1	order-2	order-1	order-2
8	97.34	2.45	90.03	9.11	74.38	23.07
16	97.63	2.51	90.13	9.23	72.84	24.68
32	97.74	2.80	90.22	9.43	72.04	26.82
64	97.72	2.22	90.22	9.71	71.56	28.60

TABLE 5
The same as Table 4, but for arithmetic multiasset options.

Dimension	$K = 90$		$K = 100$		$K = 110$	
	order-1	order-2	order-1	order-2	order-1	order-2
4	98.19	1.36	92.96	5.95	81.94	16.11
8	97.52	3.36	91.91	7.99	77.62	22.19
16	97.23	3.64	91.12	8.63	75.57	23.38

assets increases (or even tends to infinity). We also explore how the model parameters affect the dimension structure of the integrands. We consider not only option pricing problems but also bond valuation problems. Note that our purpose is not to solve the pricing or valuation problems themselves under complicated models. We are trying to understand the reason for the special property (e.g., low SD) under simplified models or situations where closed solutions may also exist.

5.1. Option pricing. Consider first the case of a geometric Asian option. In the standard construction of Brownian motion (13), the generating matrix A in (12) is just the [illegible] of the covariance matrix $V = (\min(t_i, t_j))_{i,j=1}^d$:

$$A = \sqrt{\Delta t} \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Let A_j be the sum of the jth column of A. From the discussion in section 4.1 the price of the geometric Asian call option can be written as

$$e^{-rT} \int_{[0,1]^d} \max \left(0,\ \exp \left(m + \sum_{j=1}^{d} a_j \Phi^{-1}(x_j) \right) - K \right) d\mathbf{x}, \tag{18}$$

where

$$a_j = \frac{\sigma}{d} A_j = \frac{d-j+1}{d}\, \sigma \sqrt{\Delta t} = \frac{d-j+1}{d\sqrt{d}}\, \sigma \sqrt{T}, \quad j = 1, \ldots, d. \tag{19}$$

For a given T, the nominal dimension d is inversely proportional to the time step Δt (since $d = T/\Delta t$): the smaller the time step Δt, the larger the dimension d. The integrand in (18) depends on the parameters a_j (the "weights"), which in turn depend on the time step Δt: as Δt becomes smaller, the weights a_j decrease (the weights also depend on the model parameters σ and T). We shall show that these weights control the relative importance of different (groups of) variables and play an essential role

in characterizing the complexity of the problem. These weights are not introduced artificially, but are determined by the nature of the problem and by the generation of the Brownian motion. The weights are small when the step length Δt is small, i.e., when the dimension d is large.

To see the potential influence of the parameters a_j, it is convenient to consider first the extreme case $K = 0$. The general case $K \neq 0$ will be studied in the next subsection. For $K = 0$, the integrand in (18) can be written as (excluding a constant factor)

$$f(\mathbf{x}) := \prod_{j=1}^{d} g_j(x_j) \quad \text{with} \quad g_j(x_j) = \exp\left(a_j \Phi^{-1}(x_j)\right). \tag{20}$$

(An arbitrary construction of Brownian motion (12) leads to the same form of function, but with different parameters a_j.) By direct calculation we then have

$$m_j := \int_0^1 g_j(x)dx = e^{\frac{1}{2}a_j^2} \quad \text{and} \quad \lambda_j^2 := \int_0^1 (g_j(x) - m_j)^2 dx = e^{a_j^2}(e^{a_j^2} - 1).$$

It follows that $I_d(f) = \prod_{j=1}^d m_j$ and

$$\sigma^2(f) = \prod_{j=1}^{d} \left(m_j^2 + \lambda_j^2\right) - \prod_{j=1}^{d} m_j^2 = [I_d(f)]^2 \left(e^{\sum_{j=1}^d a_j^2} - 1\right). \tag{21}$$

The ANOVA terms of f and the corresponding variances (for $u \neq \emptyset$) are

$$f_u = \prod_{j \in u} (g_j(x_j) - m_j) \cdot \prod_{j \notin u} m_j,$$

and

$$\sigma_u^2(f) = \prod_{j \in u} \lambda_j^2 \prod_{j \notin u} m_j^2 = [I_d(f)]^2 \prod_{j \in u} \left(e^{a_j^2} - 1\right). \tag{22}$$

The relative importance of the term f_u to f can be measured by the sensitivity index,

$$S_u(f) = \frac{\sigma_u^2(f)}{\sigma^2(f)} = \frac{\prod_{j \in u} \left(e^{a_j^2} - 1\right)}{e^{\sum_{j=1}^d a_j^2} - 1},$$

indicating how the parameters a_j control the relative importance of different variables. The relative importance of all order-ℓ ANOVA terms can be measured by the variance ratio (see section 2.1):

$$R_{(\ell)}(f) = \frac{\sum_{|u|=\ell} \sigma_u^2(f)}{\sigma^2(f)} = \frac{\sum_{|u|=\ell} \prod_{j \in u} \left(e^{a_j^2} - 1\right)}{e^{\sum_{j=1}^d a_j^2} - 1}. \tag{23}$$

Before studying the relative size of $R_{(\ell)}(f)$ and their asymptotic properties as $d \to \infty$, we mention several important features of the parameters a_j given in (19). First, from (19) we have, for fixed T,

$$a_j = O(\sqrt{\Delta t}) = O(d^{-1/2}) \ \text{ as } \ \Delta t \to 0, \ \ j = 1, 2, \ldots. \tag{24}$$

It follows that

$$e^{a_j^2} - 1 = a_j^2 + a_j^4/2 + \cdots = a_j^2 + O(d^{-2}). \tag{25}$$

Second, according to (19),

$$\sum_{j=1}^{d} a_j^2 = \frac{\sigma^2 \Delta t}{d^2} \sum_{j=1}^{d} j^2 = \frac{\sigma^2 T}{3} + O(\Delta t). \tag{26}$$

Therefore, $\Gamma := \lim_{d\to\infty} \sum_{j=1}^{d} a_j^2 = \frac{\sigma^2 T}{3}$. Based on these properties, we see from (21) that the variance of f is stable with respect to d, $\lim_{d\to\infty} \sigma^2(f) = e^{\Gamma}(e^{\Gamma} - 1)$, implying that the nominal dimension becomes in a sense irrelevant at least for MC. Moreover, from (25) we have $e^{a_j^2} - 1 = O(\Delta t)$. Therefore, from (22) we have, for $\ell = 1, 2, \ldots,$

$$\sigma_u^2(f) = O((\Delta t)^{\ell}) = O(d^{-\ell}) \ \text{ for } \ |u| = \ell.$$

This means that a small time step Δt leads to effects that are increasingly dominated by the lower-order terms.

We are particularly interested in the variance ratios captured by low-order ANOVA terms and their asymptotic properties. The variance ratio captured by order-1 terms is

$$\begin{aligned} R_{(1)}(f) = \frac{\sum_{j=1}^{d} \sigma_{\{j\}}^2(f)}{\sigma^2(f)} &= \frac{\sum_{j=1}^{d} (e^{a_j^2} - 1)}{e^{\sum_{j=1}^{d} a_j^2} - 1} \\ &= \frac{\sum_{j=1}^{d} a_j^2 + O(\Delta t)}{e^{\sum_{j=1}^{d} a_j^2} - 1} = \frac{\Gamma}{e^{\Gamma} - 1} + O(\Delta t), \end{aligned} \tag{27}$$

where we have used the relations (25) and (26). The leading term is close to 1 for usual values of the volatility σ and T (see Table 6). Similarly, the variance ratio captured by order-2 ANOVA terms is

$$\begin{aligned} R_{(2)}(f) = \frac{\sum_{i<j} \sigma_{\{i,j\}}^2(f)}{\sigma^2(f)} &= \frac{\sum_{i<j} (e^{a_i^2} - 1)(e^{a_j^2} - 1)}{e^{\sum_{j=1}^{d} a_j^2} - 1} \\ &= \frac{\sum_{i<j} (a_i^2 + O(d^{-2}))\ (a_j^2 + O(d^{-2}))}{e^{\sum_{j=1}^{d} a_j^2} - 1} \\ &= \frac{\frac{1}{2} \sum_{i=1}^{d} a_i^2 \ \sum_{j=1}^{d} a_j^2 - \frac{1}{2} \sum_{j=1}^{d} a_j^4 + O(\Delta t)}{e^{\sum_{j=1}^{d} a_j^2} - 1} \\ &= \frac{\Gamma^2/2}{e^{\Gamma} - 1} + O(\Delta t). \end{aligned}$$

The leading term is smaller than that for $R_{(1)}(f)$ for usual values of σ and T. In general, for any fixed $\ell = 1, 2, \ldots,$ a simple formula exists for $\lim_{d\to\infty} R_{(\ell)}(f)$.

THEOREM 5.1. [illegible] $f_d(\mathbf{x}) := f(\mathbf{x})$ [illegible] (20) [illegible] a_j [illegible] (19) [illegible]

$$\Gamma := \lim_{d\to\infty} \sum_{j=1}^{d} a_j^2 = \sigma^2 T/3.$$

$\ell = 1, 2, \ldots,$

$$\lim_{d\to\infty} R_{(\ell)}(f_d) = \lim_{d\to\infty} \frac{\sum_{i_1<i_2<\cdots<i_\ell} \prod_{k=1}^{\ell} \left(e^{a_{i_k}^2} - 1\right)}{e^{\sum_{j=1}^{d} a_j^2} - 1} = \frac{\Gamma^\ell/\ell!}{(e^\Gamma - 1)}.$$

The easy proof is analogous to that for $R_{(2)}(f)$ above and is based on the special properties (24)–(26) of the parameters a_j. Note that the numerator in the formula for $\lim_{d\to\infty} R_{(\ell)}(f_d)$ is just the ℓth term of the Taylor expansion of $e^\Gamma - 1$:

$$e^\Gamma - 1 = \sum_{\ell=1}^{\infty} \frac{\Gamma^\ell}{\ell!}.$$

The asymptotics above also show how the model parameters affect the variance ratios: the variance ratios depend on σ and T through $\sigma^2 T$. The quantity $\sigma^2 T$ is typically small for real option pricing problems (in the range, say, $0.01 \sim 0.2$), so $R_{(1)}(f)$ is close to 1, $R_{(2)}(f)$ is small, and all higher-order variance ratios $R_{(\ell)}(f)$ for $\ell > 2$ are very small. Therefore, the functions are dominated by order-1 and order-2 terms (especially the order-1 terms). As the volatility increases and/or as the time to expiration lengthens, $R_{(1)}(f)$ decreases slightly, while others increase. Numerical comparisons of the limiting variance ratios make their relative size clear. For example, for $\sigma = 0.2, T = 1$, the values of $\lim_{d\to\infty} R_{(\ell)}(f)$ are $9.933\times10^{-1}, 6.622\times10^{-3}, 2.943\times10^{-5}$, and 9.811×10^{-8} for $\ell = 1, \ldots, 4$, respectively.

1. If we change the value of the volatility to $\sigma = 10$, which is unrealistic but of academic interest, then Theorem 5.1 is still true. However, the order-1 and order-2 terms no longer dominate the functions. In fact, for $\sigma = 10$ and $T = 1$, $\lim_{d\to\infty} R_{(1)}(f_d) = 1.113\times10^{-13}$ and $\lim_{d\to\infty} R_{(2)}(f_d) = 1.855\times10^{-12}$. Fortunately, for finance problems, σ is often between 0.1 and 0.5. This is one reason for the low SD of real finance problems.

2. Consider an arbitrary construction of Brownian motion (12), i.e., a decomposition for the covariance matrix V. Let A_j be the sum of the jth column of the associated matrix A and let $a_j = \frac{\sigma}{d} A_j$. The quantity $\sum_{j=1}^{d} a_j^2$ is independent of the construction, since it is the variance of $\log(\prod_{j=1}^{d} S_{t_j}^{1/d})$ and must be the same for all constructions, and so is the limit $\Gamma := \lim_{d\to\infty} \sum_{j=1}^{d} a_j^2 = \sigma^2 T/3$. Under any construction the variance ratio captured by order-1 terms for the corresponding function f as in (20) becomes

$$R_{(1)}(f) = \frac{\sum_{j=1}^{d}(e^{a_j^2} - 1)}{e^{\sum_{j=1}^{d} a_j^2} - 1} = \frac{\sum_{j=1}^{d} a_j^2 + \frac{1}{2}\sum_{j=1}^{d} a_j^4 + \cdots}{e^{\sum_{j=1}^{d} a_j^2} - 1} \ge \frac{\Gamma}{e^\Gamma - 1} + O(\Delta t). \tag{28}$$

Thus any construction leads asymptotically to no smaller variance ratio $R_{(1)}(f)$ (or no weaker property of the function f) than the standard construction does; see (27).

Now consider the arithmetic Asian option. With $K = 0$, the payoff function is

$$g_A(S_{t_1}, \ldots, S_{t_d}) = \frac{1}{d}\sum_{j=1}^{d} S_{t_j}.$$

Under the standard construction of the Brownian motion (13), from (11) we have

$$S_{t_j} = \exp\left(\mu_j + \sigma\sqrt{\Delta t}(\Phi^{-1}(x_1) + \cdots + \Phi^{-1}(x_j))\right) =: e^{\mu_j} F_j(\mathbf{x}),$$

with $x_1, \ldots, x_d$ being i.i.d. uniformly distributed random variables on $[0,1]$. Thus the function g_A is the arithmetic mean of some functions of the form (20), excluding constant factors:

$$g_A(S_{t_1}, \ldots, S_{t_d}) = \frac{1}{d} \sum_{j=1}^{d} e^{\mu_j} F_j(\mathbf{x}).$$

Take the function $F_d(\mathbf{x})$ as an example:

$$F_d(\mathbf{x}) = \exp\left(\sigma\sqrt{\Delta t}(\Phi^{-1}(x_1) + \cdots + \Phi^{-1}(x_d))\right).$$

Such a function has exactly the same form as (20), but with different weights:

$$a_j = \sigma\sqrt{\Delta t} = O(\sqrt{\Delta t}) \ \text{ as } \ \Delta t \to 0, \ \ j = 1, \ldots, d. \tag{29}$$

The time step Δt enters to the parameters a_j in a similar way to that seen earlier. Moreover, in this case $\Gamma := \lim_{d\to\infty} \sum_{j=1}^{d} a_j^2 = \sigma^2 T$. With this Γ, Theorem 5.1 is still valid, and thus we have similar conclusions to those above: the variance ratio captured by order-ℓ ANOVA terms satisfies

$$\lim_{d\to\infty} R_{(\ell)}(F_d) = \frac{\Gamma^\ell/\ell!}{(e^\Gamma - 1)}.$$

Again, the lower-order terms capture most of the variance. The comparisons are given in Table 6.

For the multiasset options studied in subsection 4.2 the source of the high dimensionality is different: now the dimensionality comes from the number of risky assets rather than from time discretization. Since our main purpose is to see how the number of risky assets affects the integrand, we restrict ourselves to the case that $\rho_{ij} = 0$ for $i \neq j$ (choosing a high degree of correlation would be similar to reducing the dimension of the problem). Consider the geometric multiasset option. The Cholesky matrix of the covariance matrix in (16) is $A = \sqrt{T}\ \mathbf{I}$, where $\mathbf{I}$ is the $d \times d$ identity matrix; thus $B_T^j = \sqrt{T}\ \Phi^{-1}(x_j)$. From (15) we have

$$S_T^j = S_0^j \exp\left(\left(r - \frac{1}{2}\sigma_j^2\right) T + \sigma_j B_T^j\right) = S_0^j \exp\left(\left(r - \frac{1}{2}\sigma_j^2\right) T + \sigma_j\sqrt{T}\Phi^{-1}(x_j)\right),$$

which depends only on one variable x_j. With $K = 0$, from (17) the corresponding integrand is of the form (excluding a constant factor)

$$f(\mathbf{x}) := \exp\left(\sum_{j=1}^{d} \frac{\sigma_j\sqrt{T}}{d}\Phi^{-1}(x_j)\right).$$

Once more, such a function is of the same form as (20), now with the weights

$$a_j = \sigma_j\sqrt{T}/d = O(d^{-1}) \ \text{ as } \ d \to \infty, \ \ j = 1, \ldots, d. \tag{30}$$

With the number of risky assets increasing, the parameters a_j decrease (they decrease even faster than for the cases of Asian options). For the parameters in (30), assume that $\sigma_j = \sigma$ for all j. We have

$$\sum_{j=1}^{d} a_j^2 = \sigma^2 T/d.$$

TABLE 6

The variance (in percentage) captured by the order-1 and order-2 ANOVA terms for the functions (20) *with three choices of a_j, which corresponds to several kinds of option pricing problems ($\sigma = 0.2, T = 1, K = 0$). The last row gives the limiting values.*

Dimension	$a_j = \frac{d-j+1}{d}\,\sigma\sqrt{T/d}$		$a_j = \sigma\sqrt{T/d}$		$a_j = \sigma\sqrt{T}/d$	
	order-1	order-2	order-1	order-2	order-1	order-2
8	99.372	0.626	98.259	1.724	99.781	0.218
16	99.350	0.647	98.136	1.842	99.883	0.117
64	99.338	0.659	98.044	1.931	99.969	0.031
256	99.336	0.661	98.021	1.953	99.992	0.008
1024	99.335	0.662	98.015	1.958	99.998	0.002
4096	99.335	0.662	98.014	1.960	99.9999	0.00001
∞	99.335	0.662	98.013	1.960	100.00	0.0

Thus the variance ratio captured by order-1 ANOVA terms is

$$R_{(1)}(f) = \frac{\sum_{j=1}^d \sigma^2_{\{j\}}(f)}{\sigma^2(f)} = \frac{\sum_{j=1}^d a_j^2 + O(d^{-3})}{e^{\sum_{j=1}^d a_j^2} - 1} = \frac{\sigma^2/d + O(d^{-3})}{\sigma^2/d + O(d^{-2})} = 1 + O(d^{-1}).$$

Similarly, the variance ratio captured by order-2 ANOVA terms is

$$R_{(2)}(f) = \frac{\sum_{i<j} \sigma^2_{\{i,j\}}(f)}{\sigma^2(f)} = \frac{\sum_{i<j}(e^{a_i^2} - 1)(e^{a_j^2} - 1)}{\sigma^2(f)} = \frac{O(d^{-2})}{O(d^{-1})} = O(d^{-1}).$$

The variance ratios captured by higher-order ANOVA terms are of order $o(d^{-1})$. So even the order-1 part of the integrand captures almost all of the variance. As the number of assets increases, the integrand becomes even closer (in the L_2-norm) to the sum of its order-1 terms (this is confirmed by the computational results in Table 6).

For the arithmetic multiasset options, the corresponding integrands are even simpler. Indeed, under the assumption of $\rho_{ij} = 0$ for $i \neq j$, as shown above we have

$$S_T^j = S_0^j \exp\left(\left(r - \frac{1}{2}\sigma_j^2\right)T + \sigma_j\sqrt{T}\Phi^{-1}(x_j)\right), \quad j = 1, \ldots, d.$$

Thus with $K = 0$ the payoff function, $\phi(S_T^1, \ldots, S_T^d) = \frac{1}{d}\sum_{j=1}^d S_T^j$, is completely a sum of one-dimensional functions.

Table 6 presents the variance ratios of order-1 and order-2 terms for functions of the form (20) with the parameters a_j given by (19), or (29) or (30), respectively, which correspond to several kinds of option pricing problems (geometric Asian option, arithmetic Asian option, and geometric multiasset option). These ratios are computed by (23) for each given d. We see that in all cases for dimensions up to thousands the variance percentages of order-1 and order-2 terms are close to 100%, and they do not change much when the nominal dimension increases. The limiting values of $R_{(1)}(f)$ and $R_{(2)}(f)$ give us confidence to conclude that no matter how large the nominal dimension is, the functions remain low-dimensional in the superposition sense.

For a multiasset path-dependent option (Asian basket option), the dimension is the number of time steps times the number of assets. Similar method can be used to analyze the dimension structure in this case. We omit the details.

5.2. The impact of the strike price and the nonsmoothness. The whole section 4 has been devoted to the empirical investigation for strike prices $K \neq 0$. By

comparing the empirical results there with the results in section 5.1 for $K = 0$, we find that their associated functions have similar dimension structure if K/S_0 is relatively small: the low-order terms dominate the functions. For relatively large K/S_0, the difference can be significant. For $K \neq 0$, the theoretical ANOVA analysis is more difficult and is more involved (in general, the theoretical ANOVA analysis is much more difficult than the pricing problem itself). As a partial solution, we make some attempts to understand (a) the impact of the strike price on the total variance; (b) the impact of the nonsmoothness involved in the integrands.

We concentrate on the geometric Asian call option. The underlying integral is shown in (18). Define

$$F(\mathbf{x}) := e^{-rT}\left[e^m f(\mathbf{x}) - K\right] \quad \text{with} \quad f(\mathbf{x}) = \prod_{j=1}^{d} \exp\left(a_j \Phi^{-1}(x_j)\right), \tag{31}$$

where m is defined in (14). The underlying function in (18) is just the function

$$F^+(\mathbf{x}) := \mathbf{1}_{\{F(\mathbf{x})\geq 0\}}\, F(\mathbf{x}),$$

where $\mathbf{1}_{\{.\}}$ is the indicator function. The function $f(\mathbf{x})$ in (31) is the one we studied earlier for $K = 0$, and its dimension structure is clear. Obviously, the functions $F(\mathbf{x})$ and $f(\mathbf{x})$ have the same variance ratios and the same effective dimensions (since a constant factor and a constant term do not affect these), and thus $F(\mathbf{x})$ also has the same low SD as $f(\mathbf{x})$ does.

(a) The impact of the strike price on the total variance. The strike price K determines how large a part of the function $F(\mathbf{x})$ is cut away by the indicator function $\mathbf{1}_{\{F(\mathbf{x})\geq 0\}}$. The impact of K on the total variance is an indicator of its importance. Analytical formulas are available for the variances $\sigma^2(F)$ and $\sigma^2(F^+)$. First,

$$\sigma^2(F) = \exp(-2rT + 2m)\, \sigma^2(f),$$

where $\sigma^2(f)$ is given by (21). Second, for the function $F^+(\mathbf{x})$, a change of variables (similar to the one used in subsection 3.2) leads to

$$I_d(F^+) = \int_{[0,1]^d} F^+(\mathbf{x}) d\mathbf{x} = e^{-rT}\left[e^{m+s_d^2/2}\Phi(d_1) - K\Phi(d_2)\right]$$

(this is the analytical formula for the geometric Asian call option) and

$$I_d((F^+)^2) = e^{-2rT}\left[e^{2m+2s_d^2}\Phi(d_0) - 2Ke^{m+s_d^2/2}\Phi(d_1) + K^2\Phi(d_2)\right],$$

where

$$s_d^2 = \sum_{j=1}^{d} a_j^2, \quad d_1 = \frac{m - \log K + s_d^2}{s_d}, \quad d_2 = d_1 - s_d, \quad d_0 = d_1 + s_d.$$

The variance of F^+ is $\sigma^2(F^+) = I_d((F^+)^2) - [I_d(F^+)]^2$. Note that when $d \to \infty$, all the parameters m, s_d, d_0, d_1, d_2 have limits; thus $\lim_{d\to\infty} \sigma^2(F)$ and $\lim_{d\to\infty} \sigma^2(F^+)$ exist (this implies that the increase of d has no essential influence on the variance and thus on the MC error). The values of the ratios $\sigma^2(F)/\sigma^2(F^+)$ and their limit are

TABLE 7

The values of the ratio $\sigma^2(F)/\sigma^2(F^+)$ for $\sigma = 0.2$, $r = 0.1$, $T = 1$. The last row gives the limit as $d \to \infty$. The function $F(\mathbf{x})$ is defined in (31).

Dimension	$K/S_0 = 0.8$	$K/S_0 = 0.9$	$K/S_0 = 1.0$	$K/S_0 = 1.1$	$K/S_0 = 1.2$
16	1.016	1.154	1.765	4.050	13.587
64	1.013	1.148	1.778	4.266	15.419
256	1.013	1.146	1.782	4.325	15.944
1024	1.013	1.146	1.782	4.340	16.080
4096	1.013	1.146	1.783	4.344	16.114
∞	1.013	1.146	1.783	4.345	16.126

given in Table 7. We observe that for relatively small K/S_0 (say, $K/S_0 \le 0.9$), the variance of F^+ is close to that of F, i.e., the influence of K on the total variance is not significant; thus we have good reason to expect that the function F^+ has dimension structure similar to F. This is confirmed by the empirical investigation in section 4 (see Table 2). If K/S_0 is relatively large (say $K/S_0 \ge 1.1$), the difference between F and F^+ can be significant. However, the total variance of F^+ is small, implying the problem is relatively easy for MC.

(b) Nonsmoothness or discontinuity. Note that for $\mathbf{x}$ satisfying $F(\mathbf{x}) \neq K$,

$$\frac{\partial F^+(\mathbf{x})}{\partial x_j} = \mathbf{1}_{\{F(\mathbf{x})\ge 0\}} \frac{\partial F(\mathbf{x})}{\partial x_j}.$$

Due to the characteristic function $\mathbf{1}_{\{F(\mathbf{x})\ge 0\}}$, the function $F^+(\mathbf{x})$ has no partial derivatives at $\mathbf{x}$ satisfying $F(\mathbf{x}) = K$ unless $\frac{\partial F(\mathbf{x})}{\partial x_j} = 0$ when $F(\mathbf{x}) = K$. Nevertheless, most ANOVA terms are smooth: using a method similar to that of Liu and Owen [11], we conclude that for the integrand $F^+(\mathbf{x})$, the only nonsmooth ANOVA term is the highest-order one $F^+_{\mathcal{A}}(\mathbf{x})$, which is the difference of F^+ and all its lower-order terms. In fact, letting $h(x_2, \dots, x_d) = m + \sum_{j=2}^d a_j \Phi^{-1}(x_j)$, we have

$$\begin{aligned} H(x_2, \dots, x_d) &:= \int_0^1 F^+(\mathbf{x}) dx_1 \\ &= e^{-rT} \left[e^{h + a_1^2/2} \Phi\left(\frac{h - \log K + a_1^2}{a_1} \right) - K \Phi\left(\frac{h - \log K}{a_1} \right) \right]. \end{aligned}$$

Thus integrating $F^+(\mathbf{x})$ with respect to one variable results in a function that is differentiable with respect to all the others. Further integration with respect to other variables gives still smoother functions. The smaller $|u|$ is, the smoother F^+_u is.

The importance of the term $F^+_{\mathcal{A}}$ is measured by the sensitivity index $S(F^+_{\mathcal{A}})$, i.e., the ratio $\sigma^2(F^+_{\mathcal{A}})/\sigma^2(F^+)$. A conservative estimate of the variance of $F^+_{\mathcal{A}}$ can be obtained by

$$\sigma^2(F^+_{\mathcal{A}}) \le \sigma^2(F^+) - T_{\{d\}}(F^+) - T_{\{1,\dots,d-1\}}(F^+),$$

where $T_{\{d\}}(F^+)$ and $T_{\{1,\dots,d-1\}}(F^+)$ can be easily computed by the method in subsection 3.2. We find, for the geometric Asian option with the model parameters in Table 2 and for $d = 64$, that the ratio $\sigma^2(F^+_{\mathcal{A}})/\sigma^2(F^+)$ is bounded by 0.00016, 0.00019, or 0.00035 for $K = 90, 100, 110$, respectively. Therefore, the nonsmooth part of the function F^+ is negligible, and the smooth part dominates. Some other options also have nonsmooth or discontinuous payoff, allowing similar methods to be used.

An alternative approach to dealing with the nonsmoothness is to use smoothing techniques. Smoothing techniques may lead to improved QMC results, since the accuracy of QMC relates to the smoothness of the integrands (see Moskowitz and Caflisch [12] and Wang [26]). Here we use conditional MC. Since $\mathbb{E}(F^+|x_2,\dots,x_d) = H(x_2,\dots,x_d)$, we have

$$\mathbb{E}(F^+) = \mathbb{E}(\mathbb{E}(F^+|x_2,\dots,x_d)) = \mathbb{E}(H(x_2,\dots,x_d)).$$

Here we treat $x_1,\dots,x_d$ as being the i.i.d. uniformly distributed random variables on $[0,1]$. We believe that smoothing techniques may also change the ANOVA structure of the underlying function and make the problem more suited to QMC. The efficiency of smoothing techniques for QMC in finance deserves extensive study.

5.3. Bond valuation for a stochastic interest rate. Is the inherent feature (e.g., small SD) of option pricing common in finance problems? Consider now a different kind of problem: finding the fair price of a T-year zero-coupon bond with a face price of 1 (see Ninomiya and Tezuka [15] and Morokoff [13]). We assume that the interest rate follows a mean reverting process (Vasicek [25]):

$$dr = a(b-r)dt + \sigma dB_t, \quad r(0) = r_0, \tag{32}$$

where a, b are constants (the reversion speed and reversion level, respectively), σ is the volatility, and B_t is the standard Brownian motion. According to the arbitrage pricing theory (Hull [8]), the price of a discount bond at time 0 with maturity T is given by

$$P(T) = \mathbb{E}\left(\exp\left(-\int_0^T r(s)ds\right)\right) \tag{33}$$

under the risk-neutral measure. It can be solved analytically (Vasicek [25]). Still, it is a good example to study the dimension structure of the underlying function. A simulation-based procedure is to discretize the time and approximate the model by Euler approximation:

$$r_j - r_{j-1} = a(b-r_{j-1})\Delta t + \sigma\sqrt{\Delta t}\, z_j, \quad z_j \sim N(0,1), \quad j = 1,\dots,d-1, \tag{34}$$

where $\Delta t = T/d$ and r_j is the interest rate at $t_j = j\Delta t$. The discrete-time version of (33) is

$$P_d(T) = \mathbb{E}\left(\exp\left(-\Delta t\sum_{j=0}^{d-1} r_j\right)\right). \tag{35}$$

The resulting integral is $(d-1)$-dimensional. To reduce the time discretization error to an acceptable level, one should make the discretization fine enough. As a result, the dimension can be thousands (Ninomiya and Tezuka [15]). Now we analyze the integrand in (35). Let $\beta = 1 - a\Delta t$; then (34) can be written as

$$r_j - b = (r_{j-1} - b)\beta + \sigma\sqrt{\Delta t}\, z_j, \quad j = 1,\dots,d-1.$$

This leads in matrix notation to

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_{d-1} \end{pmatrix} = \sigma\sqrt{\Delta t} \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ \beta & 1 & 0 & \cdots & 0 \\ \beta^2 & \beta & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \beta^{d-2} & \beta^{d-3} & \beta^{d-4} & \cdots & 1 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_{d-1} \end{pmatrix} + \mathbf{H}, \tag{36}$$

where $\mathbf{H}$ is a vector with the jth entry $H_j = b + (r_0 - b)\beta^j$. It follows that

$$r_j = \sigma\sqrt{\Delta t}\,\left(\beta^{j-1}z_1 + \beta^{j-2}z_2 + \cdots + z_j\right) + H_j, \quad j = 1, \ldots, d-1.$$

Thus

$$\exp\left(-\Delta t \sum_{j=0}^{d-1} r_j\right) = \exp\left(\gamma_1 z_1 + \cdots + \gamma_{d-1} z_{d-1} - G\right), \tag{37}$$

where $G = \Delta t(r_0 + \sum_{j=1}^{d-1} H_j)$, and the parameters γ_j are given by

$$\gamma_j = -\sigma\Delta t^{3/2}(1 + \beta + \cdots + \beta^{d-j-1}), \quad j = 1, \ldots, d-1.$$

We are interested in the case $a \neq 0$ (if $a = 0$, then the Vasicek model reduces to Brownian motion, which is unsuitable for interest rate processes). For $a \neq 0$, we have $\beta \neq 1$, and

$$\gamma_j = -\sigma\Delta t^{3/2}\frac{1-\beta^{d-j}}{1-\beta} = -\frac{\sigma\sqrt{\Delta t}}{a}(1-\beta^{d-j}). \tag{38}$$

From (37) the integral (35) can then be written as

$$P_d(T) = \int_{\mathbb{R}^{d-1}} \exp\left(\sum_{j=1}^{d-1}\gamma_j z_j - G\right) p_{d-1}(z_1, \ldots, z_{d-1}) dz_1 \cdots dz_{d-1},$$

where p_{d-1} is given in (9). Using a change of variables $z_j = \Phi^{-1}(x_j), j = 1, \ldots, d-1$, we obtain an integral over $[0,1]^{d-1}$, with the integrand in the form (excluding a constant factor)

$$f(\mathbf{x}) := \prod_{j=1}^{d-1} g_j(x_j) \quad \text{with} \quad g_j(x_j) = \exp\left(\gamma_j \Phi^{-1}(x_j)\right), \tag{39}$$

This is of exactly the same form as the one in (20); only the parameters γ_j are different. The parameters in (38) have a property similar to the ones in (19). For example, we have $\gamma_j = O(\sqrt{\Delta t})$ as $\Delta t \to 0$ for each j. By direct calculation, we obtain

$$\Gamma := \lim_{d\to\infty}\sum_{j=1}^{d-1}\gamma_j^2 = \frac{\sigma^2}{2a^3}(4e^{-aT} - e^{-2aT} + 2aT - 3) = \sigma^2 T^3\left(\frac{1}{3} - \frac{a}{4}T + \frac{7a^2}{60}T^2 + \cdots\right). \tag{40}$$

Analogous to Theorem 5.1 for the case of Asian options, we have the following.

THEOREM 5.2. (32)

(i) (39) γ_j (38) $R_{(\ell)}(f)$ (23) a_j γ_j

(ii) ℓ

TABLE 8

The variance ratios (in percentage) captured by low-order ANOVA terms for bond valuation with $a = 0.1817303, \sigma = 0.02$. The last row gives the limiting values. The effective dimensions (d_t and d_s) are in the parentheses. The paths are constructed by the standard method (36)*. Note that the results are independent of the parameter b in the Vasicek model.*

$d-1$	$T=5$		$T=20$		$T=30$	
	order-1 (d_t)	order-2 (d_s)	order-1 (d_t)	order-2 (d_s)	order-1 (d_t)	order-2 (d_s)
16-1	99.615 (13)	0.384 (1)	93.303 (14)	6.420 (2)	87.968 (15)	11.121 (2)
64-1	99.570 (52)	0.428 (1)	92.994 (56)	6.682 (2)	87.534 (58)	11.426 (3)
256-1	99.559 (210)	0.439 (1)	92.915 (225)	6.748 (2)	87.421 (231)	11.506 (3)
1024-1	99.557 (838)	0.442 (1)	92.896 (901)	6.764 (2)	87.392 (924)	11.526 (3)
4096-1	99.556 (3352)	0.443 (1)	92.891 (3603)	6.768 (2)	87.385 (3694)	11.531 (3)
∞	99.556	0.443	92.889	6.770	87.382	11.532

$$\lim_{d\to\infty} R_{(\ell)}(f) = \frac{\Gamma^\ell/\ell!}{(e^\Gamma - 1)}, \quad \ell = 1, 2, \ldots,$$

[illegible] Γ [illegible] (40)

The variance ratios and their limits for bond valuation are similar to those for option pricing. From Theorem 5.2 and from the formula (40) for Γ, it can be seen that the variance ratios are mainly determined by the quantity $\sigma^2 T^3$. A difference from option pricing is that now the time length T may be quite long, say, $T > 10$ years. Thus the bond valuation problems have a more sensitive dependence on σ. If σ is relatively small, then the conclusions are similar to those for option pricing (even if T is large). Table 8 presents the variance ratios for $\sigma = 0.02$. The last row presents the limiting values. We also compute the effective dimensions using the method in Wang and Fang [28]. We observe that the SD is small and is insensitive to the nominal dimension ($d_s \le 2$ for $T = 20$ and $d_s \le 3$ for $T = 30$), but the TD is almost as large as the nominal dimension.

For larger σ (say, $\sigma = 0.1$), the situation can be quite different. If T is also large (say, $T \ge 20$), the variance ratios of low-order terms are very small. The effective dimension (not only the TD, but also the SD) can be large (see Table 9). The associated problem can be hard for QMC, since now the high-order ANOVA terms are important, and it is difficult to make the high-order projections of a QMC point set more uniform than those of a random point set for practical sample size (see Wang and Sloan [30]).

However, the situation changes completely if the interest rate paths are generated by [illegible] (PCA). The idea of PCA remains the same as in Acworth, Broadie, and Glassman [1]. The comparisons are given in Table 9. We observe that "low-dimensional" integrands are regained by using PCA: the integrands are low-dimensional, [illegible]. For example, for $T = 30$ the TD and SD in PCA are no larger than 6 and 3, respectively (for nominal dimension up to thousands)! Both the TD and SD are reduced significantly; especially, the variance ratios of the order-1 terms are increased dramatically. Moreover, in PCA construction both the TD and SD are very [illegible] to the nominal dimension, indicating the possibility of breaking the curse of dimension-

TABLE 9

The same as Table 8*, but with* $a = 0.1817303$, $\sigma = 0.1$ *(the volatility is much larger than the one used in Table* 8*). Two methods of path constructions are considered: the standard construction* (36) *and the construction based on PCA.*

$d-1$	$T=20$ (Standard)		$T=20$ (PCA)		$T=30$ (Standard)		$T=30$ (PCA)	
	order-1 (d_t)	order-2 (d_s)	order-1 (d_t)	order-2 (d_s)	order-1 (d_t)	order-2 (d_s)	order-1 (d_t)	order-2 (d_s)
16-1	10.790 (15)	21.465 (7)	90.835 (4)	8.932 (2)	1.021 (15)	4.147 (10)	75.994 (6)	22.781 (3)
64-1	10.029 (59)	18.653 (8)	91.170 (4)	8.599 (2)	0.930 (61)	3.221 (12)	78.472 (6)	20.407 (3)
256-1	9.845 (236)	18.028 (9)	91.276 (4)	8.497 (2)	0.907 (243)	3.030 (13)	79.032 (6)	19.886 (3)
1024-1	9.799 (946)	17.877 (9)	91.304 (4)	8.470 (2)	0.902 (971)	2.984 (13)	79.169 (6)	19.759 (3)
4096-1	9.788 (3782)	17.839 (9)	91.311 (4)	8.463 (2)	0.900 (3885)	2.973 (13)	79.203 (6)	19.727 (3)

ality. More research is needed to understand theoretically how dimension reduction techniques could change the dimension structure and change the (strong) tractability properties of the underlying problems. Moreover, dimension reduction techniques which can take into account the characteristics of a given problem are desirable.

6. Concluding remarks. The potential sources of dimensionality of finance problems are the small time increments and/or the large number of the state variables. Is there really a curse of dimensionality for such problems? It is found that these problems are often of low effective dimension in the superposition sense. In other words, an ANOVA expansion up to the second order can often provide a quite satisfactory approximation to the function. This property does not seem to change as the nominal dimension increases. We believe that these properties are closely related to the concentration of measure.

The observations are justified by theoretical analysis. By carefully analyzing simplified models of some typical finance problems, we found that the underlying functions are controlled by some weights, which are determined by the nature of the problems. These weights limit the importance of the higher-dimensional structures and offset the effect of high dimensionality. By studying the variance ratios and their asymptotic behavior as $d \to \infty$, we found that these quantities depend critically on the weights. For option pricing problems the low-order ANOVA terms (usually the order-1 and order-2 terms) always play the major role, while the higher-order terms tend to be negligible. It can even happen that as the dimension increases the problem becomes easier (in the sense of smaller SD)! For bond valuation problems with small value of σ or T, the conclusions are similar. However, large values of σ and T may lead to quite different situation: the SD can be not so small and the problems can be truly hard for QMC. However, "low-dimensional" integrands, not only in the superposition sense, but also in the truncation sense, are regained by using PCA. This demonstrates the power of PCA. Further study of the effect of dimension reduction techniques is needed. We observed that in all examples the SD (in standard or PCA construction) is very insensitive to the nominal dimension. This justifies the use of "finite-order" weights in Sloan, Wang, and Woźniakowski [21] (where it was shown that the computational complexity depends exponentially on the highest order of the ANOVA terms that need to be taken into account).

We have analyzed some finance problems under relatively basic models and using

basic discretization schemes. More complex models, problems, and more accuracy discretization schemes are worth investigating (however, theoretical ANOVA analysis under more complicated models can be much more difficult). Stochastic differential equations have been widely used to describe the dynamics of the financial assets. In numerical practice it is common to approximate a continuous process with a time-discretized one with a large number of time steps. We believe that "low SD" property (in ANOVA decomposition) could be a common feature for the integrals arising from time discretization. For approximating such integrals the uniformity of the low-order projections of the point set is crucial. Fortunately, as shown in Wang and Sloan [30], common low discrepancy point sets have smaller superposition discrepancy of lower order (such as order-1) than random point sets do, even for large d and small n. The inherent features of low discrepancy point sets, in combination with the special properties of the finance problems (low effective dimension in superposition or even in truncation sense) result in the effectiveness of QMC methods for nominally high-dimensional finance problems.

The investigations enable us to better understand the special features of finance problems and indicate ways to construct more efficient QMC algorithms and ways to design powerful efficiency improvement techniques. For example, based on such information, we were able in [29] to find suitable weights for use in the algorithmic construction of efficient lattice rules (in that paper we also explained why classical lattice rules are not suitable for high-dimensional finance problems). Moreover, the special properties of finance problems indicate that in order to construct more efficient QMC point sets for such problems, we need mainly to concentrate on the uniformity of low-order projections, especially the order-1 and order-2 projections. The lattice rules constructed based on such an idea are potentially useful. For the construction of digital nets (see Niederreiter [14]), it would be also desirable to use new quality parameters (instead of the traditional t-value) to reflect those characteristics. Such quality parameters would describe the quality of the distribution in the low-order projections of the point sets. The common property of low SD or even nearly additive feature for finance problems may enable us to develop general techniques to improve the efficiency for such problems.

Acknowledgment. The authors would like to thank the editor and the referees for their valuable comments.

REFERENCES

[1] P. Acworth, M. Broadie, and P. Glasserman, *A comparison of some Monte Carlo and quasi-Monte Carlo techniques for option pricing*, in Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, P. Hellekalek and H. Niederreiter, eds., Springer-Verlag, New York, 1997, pp. 1–18.

[2] P. Boyle, M. Broadie, and P. Glasserman, *Monte Carlo methods for security pricing*, J. Econom. Dynam. Control, 21 (1997), pp. 1267–1321.

[3] R. Caflisch, W. Morokoff, and A. Owen, *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*, J. Comp. Finance, 1 (1997), pp. 27–46.

[4] T. Gerstner and M. Griebel, *Dimension-adaptive tensor-product quadrature*, Computing, 71 (2003), pp. 65–87.

[5] F. J. Hickernell, *Quadrature error bounds with applications to lattice rules*, SIAM J. Numer. Anal., 33 (1996), pp. 1995–2016; *Erratum*, SIAM J. Numer. Anal., 34 (1997), pp. 853–866.

[6] F. J. Hickernell, *A generalized discrepancy and quadrature error bound*, Math. Comp., 67 (1998), pp. 299–322.

[7] F. J. Hickernell and X. Wang, *The error bounds and tractability of quasi-Monte Carlo algorithms in infinite dimension*, Math. Comp., 71 (2002), pp. 1641–1661.

[8] J. Hull, *Options, Futures, and Other Derivatives*, 5th ed., Prentice-Hall, Upper Saddle River, NJ, 2003.
[9] J. Imai and K. S. Tan, *Minimizing effective dimension using linear transformation*, in Monte Carlo and Quasi-Monte Carlo Methods 2002, H. Niederreiter, ed., Springer-Verlag, Berlin, 2004, pp. 275–292.
[10] C. Joy, P. Boyle, and K. Tan, *Quasi-Monte Carlo methods in numerical finance*, Management Sci., 46 (1996), pp. 926–938.
[11] R. Liu and A. B. Owen, *Estimating Mean Dimensionality*, Stanford University, Stanford, CA, 2003.
[12] B. Moskowitz and R. E. Caflisch, *Smoothness and dimension reduction in quasi-Monte Carlo methods*, Math. Comput. Modelling, 23 (1996), pp. 37–54.
[13] W. J. Morokoff, *Generating quasi-random paths for stochastic processes*, SIAM Rev., 40 (1998), pp. 765–788.
[14] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.
[15] S. Ninomiya and S. Tezuka, *Toward real time pricing of complex financial derivatives*, Appl. Math. Finance, 3 (1996), pp. 1–20.
[16] A. Owen, *The Necessity of Low Effective Dimension*, Stanford University, Stanford, CA, 2002.
[17] A. Owen, *The dimension distribution, and quadrature test functions*, Statist. Sinica, 13 (2003), pp. 1–17.
[18] S. Paskov, *New methodologies for valuing derivatives*, in Mathematics of Derivative Securities, S. Pliska and M. Dempster, eds., Cambridge University Press, Cambridge, UK, 1997, pp. 545–582.
[19] S. Paskov and J. Traub, *Faster valuation of financial derivatives*, J. Portfolio Management, 22 (1995), pp. 113–120.
[20] I. H. Sloan, F. Y. Kuo, and S. Joe, *Constructing randomly shifted lattice rules in weighted Sobolev spaces*, SIAM J. Numer. Anal., 40 (2002), pp. 1650–1665.
[21] I. H. Sloan, X. Wang, and H. Woźniakowski, *Finite-order weights imply tractability of multivariate integration*, J. Complexity, 20 (2004), pp. 46–74.
[22] I. H. Sloan and H. Woźniakowski, *When are quasi-Monte Carlo algorithms efficient for high dimensional integrals?*, J. Complexity, 14 (1998), pp. 1–33.
[23] I. Sobol, *On the distribution of points in a cube and the approximate evaluation of integrals*, Zh. Vychisli. Mat. i Mat. Fiz., 7 (1967), pp. 784–802.
[24] I. Sobol, *Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates*, Math. Comput. Simulation, 55 (2001), pp. 271–280.
[25] O. Vasicek, *An equilibrium characterization of the term structure*, J. Financial Economics, 5 (1977), pp. 177–188.
[26] X. Wang, *Improving the rejection sampling method in quasi-Monte Carlo methods*, J. Comput. Appl. Math., 114 (2000), pp. 231–246.
[27] X. Wang, *Strong tractability of multivariate integration using quasi-Monte Carlo algorithms*, Math. Comp., 72 (2003), pp. 823–838.
[28] X. Wang and K.-T. Fang, *Effective dimension and quasi-Monte Carlo integration*, J. Complexity, 19 (2003), pp. 101–124.
[29] X. Wang and I. H. Sloan, *Efficient weighted lattice rules with applications to finance*, SIAM J. Sci. Comput., submitted.
[30] X. Wang and I. H. Sloan, *Low discrepancy sequences in high dimensions: How well are their projections distributed?*, in preparation.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 184–207
© 2005 Society for Industrial and Applied Mathematics

COMPUTATION OF MEAN DRAG FOR BLUFF BODY PROBLEMS USING ADAPTIVE DNS/LES*

JOHAN HOFFMAN†

Abstract. We compute the time average of the drag in two benchmark bluff body problems: a surface mounted cube at Reynolds number 40000, and a square cylinder at Reynolds number 22000, using adaptive DNS/LES. In adaptive DNS/LES the Galerkin least-squares finite element method is used, with adaptive mesh refinement until a given stopping criterion is satisfied. Both the mesh refinement criterion and the stopping criterion are based on a posteriori error estimates of a given output of interest, in the form of a space-time integral of a computable residual multiplied by a dual weight, where the dual weight is obtained from solving an associated dual problem computationally, with the data of the dual problem coupling to the output of interest. No filtering is used, and in particular no Reynolds stresses are introduced. We thus circumvent the problem of closure, and instead we estimate the error contribution from subgrid modeling a posteriori, which we find to be small. We are able to predict the mean drag with an estimated tolerance of a few percent using about 10^5 mesh points in space, with the computational power of a PC.

Key words. adaptive DNS/LES, adaptive finite element method, duality, a posteriori error estimate, turbulence, large eddy simulation, direct numerical simulation, bluff body problem, surface mounted cube, square cylinder

AMS subject classifications. 65M60, 76F65

DOI. 10.1137/040614463

1. Introduction.

The Navier–Stokes equations form the basic mathematical model in fluid mechanics and describe a large variety of phenomena of fluid flow occurring in hydro- and aerodynamics, the processing industry, biology, oceanography, meteorology, geophysics, and astrophysics. The [illegible] $Re = \frac{UL}{\nu}$, where U is a characteristic flow velocity, L a characteristic length scale, and ν the viscosity of the fluid, is used to characterize different flow regimes. If Re is relatively small ($Re \leq 10-100$), then the flow is viscous and the flow field is ordered and smooth or [illegible], while for larger Re, the flow will at least partly be [illegible] with time-dependent nonordered features on a range of length scales down to a smallest scale of size $Re^{-3/4}$, assuming $L = 1$. In many applications of scientific and industrial importance Re is very large, of the order 10^6 or larger, and the flow shows a combination of laminar and turbulent features. To resolve all the features of a turbulent flow at $Re = 10^6$ in a [illegible] would require of the order $Re^3 = 10^{18}$ mesh points in space time, and thus would be impossible on any foreseeable computer.

To overcome the impossibility of DNS for higher Reynolds numbers one seeks to reduce the problem by some sort of [illegible]. In a traditional approach to CTM, such as large eddy simulation (LES), one seeks to computationally approximate an averaged solution of the Navier–Stokes equations. This is typically done by averaging (or filtering) the Navier–Stokes equations to derive a new set of modified equations, for the averaged variables, to solve. By averaging the nonlinear term in the Navier–Stokes equations one is left with the problem of [illegible] — to model the so-called [illegible] in a [illegible], which is the major open

*Received by the editors September 4, 2004; accepted for publication (in revised form) October 1, 2004; published electronically August 22, 2005. This research was supported by DOE grant DE-FG02-88ER25053.

http://www.siam.org/journals/sisc/27-1/61446.html

†NADA, KTH, SE-10044 Stockholm, Sweden (jhoffman@nada.kth.se).

problem of today. There is an extensive amount of work on LES, in particular regarding the closure problem and the construction of various subgrid models, and we refer the reader to [23] and the references therein for details.

Another important aspect of turbulence simulation is in what sense a solution is accurate, that is, what output from a solution can be considered to be well determined. The computational version of this question is, This connects to questions of weak uniqueness for the Navier–Stokes equations, investigated in [14, 17, 18]. In particular, full resolution does not automatically give pointwise accuracy, and thus also in the case of DNS, questions of computability and predictability are crucial. For example, mean value output is typically less expensive to compute than more pointwise output; see [12, 13, 14, 17, 18].

In this paper we compute the mean drag of two bluff bodies in turbulent flow, using , a new approach to CTM. In adaptive DNS/LES we do not use any filtering, instead we use the Galerkin/least-squares space-time finite element method, here referred to as the G2, to compute approximate weak solutions to the exact Navier–Stokes equations directly, adaptively refining the computational mesh until a given tolerance is reached with respect to the error in a certain output of interest. Parts of the computational domain are resolved in a DNS, and other parts are left unresolved in an LES with the least-squares stabilization in G2 acting as a dissipative subgrid model. In particular, the adaptive algorithm in itself determines the degree of resolution needed in different parts of the domain to satisfy the stopping criterion. For the computational examples in this paper, the flow is typically fully resolved in a DNS in laminar parts of the domain, whereas the smallest mesh size in the turbulent wake is much larger than the theoretically estimated smallest scale of the turbulent flow, and thus we regard the wake as being left unresolved in an LES. The adaptive mesh refinement is based on in the form of a space-time integral of a residual times a dual weight, where the dual weight is obtained by solving an associated linear dual problem with data connecting to the output of interest. The stability of the dual weight under the mesh refinement is crucial, and is controlled computationally through the adaptive algorithm, assuring reliability of the a posteriori error estimates.

Adaptive DNS/LES thus provides an adaptive method for the computation of chosen output, based on quantitative error control, with a key ingredient being the output sensitivity information obtained from the dual problem. No Reynolds stresses are introduced, and we thus circumvent the problem of closure.

The idea of using duality arguments in a posteriori error estimation goes back to Babuška and Miller [2] in the context of postprocessing "quantities of physical interest" in elliptic model problems. A framework for more general situations has since then been systematically developed. For a more detailed account of the development of this framework, including references, we refer the reader to the review papers [6, 4, 15, 9]. For incompressible flow, applications of adaptive finite element methods based on this framework have been increasingly advanced with computation of functionals such as the drag force for two-dimensional (2d) stationary benchmark problems in [3, 8], drag and lift forces and pressure differences for three-dimensional (3d) stationary benchmark problems in [11], and time-dependent problems in 3d in [13]. In [12], this framework is extended to turbulent flow using LES, with a posteriori error estimates with respect to the averaged solution, in terms of the Reynolds stresses that need to be modelled. These a posteriori error estimates are then used in an adaptive algorithm

to compute the drag coefficient for a surface mounted cube in [10].

In [16], the generalization to adaptive DNS/LES is presented as a general methodology to handle laminar and turbulent flows. Adaptive DNS/LES may be thought of as a straightforward application of the framework for adaptive error control in [6] to the Navier–Stokes equations, based on a posteriori error estimates, with an error contribution from discretization of the exact Navier–Stokes equations, and a modeling error contribution from the stabilization in G2.

To study the efficiency of adaptive DNS/LES, we here present results for two benchmark bluff body problems, described, e.g., in [22]: (i) a surface mounted cube at Reynolds number 40000, for which some results using adaptive DNS/LES are included also in [16], and (ii) a square cylinder at Reynolds number 22000. The results in this paper indicate that we are able to predict the drag coefficient in (i)–(ii) with an estimated tolerance of a few percent, using about 10^5 mesh points in space, with the computational power of a PC.

An outline of this paper is as follows. In section 2 we recall the Navier–Stokes equations as a model for viscous incompressible flow, and we state the corresponding discretization using cG(1)cG(1), a variant of G2. In section 3 we present adaptive DNS/LES cG(1)cG(1), including a posteriori error estimates for drag, and in section 4 we present computational results for the problems (i)–(ii) using adaptive DNS/LES cG(1)cG(1).

2. The Navier–Stokes equations. The incompressible Navier–Stokes equations expressing conservation of momentum and incompressibility of a unit density Newtonian fluid, with constant kinematic viscosity $\nu > 0$ enclosed in a volume Ω in $\mathbb{R}^3$ (where we assume that Ω is a polygonal domain) with homogeneous Dirichlet boundary conditions, take the form: Find (u, p) such that

$$\begin{aligned} \dot{u} + (u \cdot \nabla)u - \nu\Delta u + \nabla p &= f \quad \text{in } \Omega \times I, \\ \nabla \cdot u &= 0 \quad \text{in } \Omega \times I, \\ u &= 0 \quad \text{on } \partial\Omega \times I, \\ u(\cdot, 0) &= u_0 \quad \text{in } \Omega, \end{aligned} \tag{2.1}$$

where $u(x,t) = (u_i(x,t))$ is the [illegible] vector and $p(x,t)$ is the [illegible] of the fluid at (x,t), and f, u_0, $I = (0,T)$ are a given driving force, initial data, and time interval, respectively. The quantity $\nu\Delta u - \nabla p$ represents the total fluid force, and may alternatively be expressed as

$$\nu\Delta u - \nabla p = \operatorname{div}\sigma(u,p), \tag{2.2}$$

where $\sigma(u,p) = (\sigma_{ij}(u,p))$ is the [illegible], with components $\sigma_{ij}(u,p) = 2\nu\epsilon_{ij}(u) - p\delta_{ij}$, composed of the [illegible] $2\nu\epsilon_{ij}(u)$ with zero trace and an isotropic pressure: here $\epsilon_{ij}(u) = (u_{i,j} + u_{j,i})/2$ is the [illegible], with $u_{i,j} = \partial u_i/\partial x_j$, and δ_{ij} is the usual Kronecker delta, the indices i and j ranging from 1 to 3. We assume that (2.1) is normalized so that the reference velocity and typical length scale are both equal to one. The Reynolds number Re is then equal to ν^{-1}.

2.1. Discretization: cG(1)cG(1). The cG(1)cG(1) method is a variant of the G2 method [20, 13] using the continuous Galerkin method cG(1) in time instead of a discontinuous Galerkin method. With cG(1) in time the trial functions are continuous piecewise linear and the test functions are piecewise constant. cG(1) in space corresponds to both test functions and trial functions being continuous piecewise linear. Let $0 = t_0 < t_1 < \cdots < t_N = T$ be a sequence of discrete time steps with

associated time intervals $I_n = (t_{n-1}, t_n]$ of length $k_n = t_n - t_{n-1}$ and space-time slabs $S_n = \Omega \times I_n$, and let $W^n \subset H^1(\Omega)$ be a finite element space consisting of continuous piecewise linear functions on a mesh $\mathcal{T}_n = \{K\}$ of mesh size $h_n(x)$ with W_w^n the functions $v \in W^n$ satisfying the Dirichlet boundary condition $v|_{\Gamma_D} = w$.

We seek functions (U_h, P_h), continuous piecewise linear in space and time, and the cG(1)cG(1) method for the Navier–Stokes equations (2.1), with homogeneous Dirichlet boundary conditions, reads as follows:

For $n = 1, \ldots, N$, find $(U_h^n, P_h^n) \equiv (U_h(t_n), P_h(t_n))$ with $U_h^n \in V_0^n \equiv [W_0^n]^3$ and $P_h^n \in W^n$, such that

$$\begin{aligned}(2.3)\qquad &((U_h^n - U_h^{n-1})k_n^{-1} + \hat{U}_h^n \cdot \nabla \hat{U}_h^n, v) + (2\nu\epsilon(\hat{U}_h^n), \epsilon(v)) - (P_h^n, \nabla \cdot v)\\ &\qquad + (\nabla \cdot \hat{U}_h^n, q) + SD_\delta(\hat{U}_h^n, P_h^n; v, q) = (f, v) \quad \forall (v, q) \in V_0^n \times W^n,\end{aligned}$$

where $\hat{U}_h^n = \frac{1}{2}(U_h^n + U_h^{n-1})$, with the stabilizing term

$$SD_\delta(\hat{U}_h^n, P_h^n; v, q) \equiv (\delta_1(\hat{U}_h^n \cdot \nabla \hat{U}_h^n + \nabla P_h^n - f), \hat{U}_h^n \cdot \nabla v + \nabla q) + (\delta_2 \nabla \cdot \hat{U}_h^n, \nabla \cdot v),$$

and $\delta_1 = \frac{1}{2}(k_n^{-2} + |U|^2 h_n^{-2})^{-1/2}$ in the convection-dominated case $\nu < \hat{U}_h^n h_n$ and $\delta_1 = \kappa_1 h^2$ otherwise, $\delta_2 = \kappa_2 h$ if $\nu < \hat{U}_h^n h_n$ and $\delta_2 = \kappa_2 h^2$ otherwise, with κ_1 and κ_2 positive constants of unit size (in this paper we have $\kappa_1 = \kappa_2 = 1$), and

$$\begin{aligned}(v, w) &= \sum_{K \in \mathcal{T}_n} \int_K v \cdot w \, dx,\\ (\epsilon(v), \epsilon(w)) &= \sum_{i,j=1}^{3} (\epsilon_{ij}(v), \epsilon_{ij}(w)).\end{aligned}$$

We note that the viscous term $(2\nu\epsilon(U_h), \epsilon(v))$ may alternatively occur in the form $(\nu \nabla U_h, \nabla v) = \sum_{i=1}^3 (\nu \nabla (U_h)_i, \nabla v_i)$. In the case of Dirichlet boundary conditions the corresponding variational formulations are equivalent, but not so in the case of Neumann boundary conditions. If we have Neumann boundary conditions, we use the standard technique to apply these boundary conditions weakly.

3. Adaptive DNS/LES cG(1)cG1(1). In adaptive DNS/LES cG(1)cG(1) we do not apply any filter to the Navier–Stokes equations, and thus no Reynolds stresses are introduced. Instead we use cG(1)cG(1) to adaptively compute approximate solutions with the goal of satisfying a given tolerance with respect to the error in a specified output of interest.

Error indicators for mesh refinement and stopping criterions are both based on a posteriori error estimates, in the form of a space-time integral of a computable residual times a dual weight, where the dual weight is obtained from a computational approximation of the solution to an associated dual problem with data coupled to the output of interest.

3.1. Adaptive computation of the drag of a bluff body. We want to compute a mean value in time of the drag of a bluff body in a channel subject to a time-dependent turbulent flow:

$$(3.1)\qquad N(\sigma(u,p)) \equiv \frac{1}{|I|} \int_I \int_{\Gamma_0} \sum_{i,j=1}^{3} \sigma_{ij}(u,p) n_j \phi_i \, ds \, dt,$$

where (u,p) solves (2.1) in the fluid volume Ω surrounding the body (using suitable boundary conditions as specified below), Γ_0 is the surface of the body in contact with the fluid, and ϕ is a unit vector along the channel in the direction of the flow. We first use a standard technique to derive an alternative expression for the drag, which naturally fits with a Galerkin formulation, by extending ϕ to a function Φ defined in the fluid volume Ω and being zero on the remaining boundary Γ_1 of the fluid volume. Multiplying the momentum equation in (2.1) by Φ and integrating by parts, we get, using the zero boundary condition on Γ_1,

$$\begin{aligned} N(\sigma(u,p)) = \frac{1}{|I|}\int_I (\dot{u} + u\cdot\nabla u - f, \Phi) - (p, \nabla\cdot\Phi) \\ + (2\nu\epsilon(u), \epsilon(\Phi)) + (\nabla\cdot u, \Theta)\ dt, \end{aligned} \tag{3.2}$$

where we also added the integral of $\nabla\cdot u = 0$ multiplied by a function Θ. Obviously, the representation does not depend on the particular extension Φ of ϕ, or Θ.

We are thus led to compute an approximation of the drag $N(\sigma(u,p))$ from a computed (U_h, P_h) using the formula

$$\begin{aligned} N^h(\sigma(U_h,P_h)) = \frac{1}{|I|}\int_I (\dot{U}_h + U_h\cdot\nabla U_h - f, \Phi) - (P_h, \nabla\cdot\Phi) + (2\nu\epsilon(U_h), \epsilon(\Phi)) \\ + SD_\delta(U_h, P_h; \Phi, \Theta) + (\nabla\cdot U_h, \Theta)\ dt, \end{aligned} \tag{3.3}$$

where now Φ and Θ are finite element functions (with as before $\Phi = \phi$ on Γ_0 and $\Phi = 0$ on Γ_1), and where $\dot{U}_h = (U_h^n - U_h^{n-1})/k_n$ on I_n. We note the presence of the stabilizing term SD_δ in (3.3), compared to (3.2). This term is added in order to obtain the independence of $N^h(\sigma(U_h,P_h))$ from the choice of (Φ,Θ), which follows from (2.3).

3.2. The dual problem. We introduce the following dual problem: Find (φ,θ) with $\varphi = \phi$ on Γ_0 and $\varphi = 0$ on Γ_1, such that

$$\begin{aligned} -\dot{\varphi} - (u\cdot\nabla)\varphi + \nabla U_h\cdot\varphi - \nu\Delta\varphi + \nabla\theta = 0 \quad &\text{in } \Omega\times I, \\ \operatorname{div}\varphi = 0 \quad &\text{in } \Omega\times I, \\ \varphi(\cdot,T) = 0 \quad &\text{in } \Omega, \end{aligned} \tag{3.4}$$

where $(\nabla U_h\cdot\varphi)_j = (U_h)_{,j}\cdot\varphi$. The dual problem is a linear convection-diffusion-reaction problem where the convection acts backward in time and in the opposite direction of the exact flow velocity u. We note that the coefficient ∇U_h of the reaction term locally is large in turbulent regions, and thus potentially generating rapid exponential growth. However, ∇U_h is fluctuating and the net effect of the reaction term for the problems in this paper turns out to generate slower growth, as we learn from computing approximations of the dual solution.

3.3. An a posteriori error estimate. To derive a representation of the error $N(\sigma(u,p)) - N^h(\sigma(U_h,P_h))$, we subtract (3.3) from (3.2) with (Φ,Θ) finite element functions to get

$$\begin{aligned} N(\sigma(u,p)) - N^h(\sigma(U_h,P_h)) = \frac{1}{|I|}\int_I (\dot{u} + u\cdot\nabla u, \Phi) - (p, \nabla\cdot\Phi) \\ + (2\nu\epsilon(u), \epsilon(\Phi)) + (\nabla\cdot u, \Theta) - ((\dot{U}_h + U_h\cdot\nabla U_h, \Phi) - (P_h, \nabla\cdot\Phi) \\ + (2\nu\epsilon(U_h), \epsilon(\Phi)) + (\nabla\cdot U_h, \Theta) + SD_\delta(U_h, P_h; \Phi, \Theta))\ dt. \end{aligned} \tag{3.5}$$

With (φ,θ) the solution to the dual problem (3.4), we also have that

$$\begin{aligned}
&\frac{1}{|I|}\int_I (\dot u + u\cdot\nabla u,\varphi) - (p,\nabla\cdot\varphi) + (2\nu\epsilon(u),\epsilon(\varphi)) + (\nabla\cdot u,\theta)\\
&\quad - ((\dot U_h + U_h\cdot\nabla U_h,\varphi) - (P_h,\nabla\cdot\varphi) + (2\nu\epsilon(U_h),\epsilon(\varphi)) + (\nabla\cdot U_h,\theta))\ dt\\
&\quad = \frac{1}{|I|}\int_I -(\dot\varphi,e) + (u\cdot\nabla e + e\cdot\nabla U_h,\varphi) - (p-P_h,\nabla\cdot\varphi)\\
&\quad\quad + (2\nu\epsilon(e),\epsilon(\varphi)) + (\nabla\cdot e,\theta)\ dt = 0,
\end{aligned}\tag{3.6}$$

using partial integration with $\varphi(T) = e(0) = 0$, where $e = u - U_h$, and that $(u\cdot\nabla)u - (U_h\cdot\nabla)U_h = (u\cdot\nabla)e + (e\cdot\nabla)U_h$. By (3.5) and (3.6), we thus have that

$$\begin{aligned}
&N(\sigma(u,p)) - N^h(\sigma(U_h,P_h)) = \frac{1}{|I|}\int_I (\dot U_h + U_h\cdot\nabla U_h,\varphi-\Phi)\\
&- (P_h,\nabla\cdot(\varphi-\Phi)) + (\nabla\cdot U_h,\theta-\Theta) + (2\nu\epsilon(U_h),\nabla(\varphi-\Phi)) + SD_\delta(U_h,P_h;\Phi,\Theta)\\
&- ((\dot u + u\cdot\nabla u,\varphi-\Phi) - (p,\nabla\cdot(\varphi-\Phi)) + (\nabla\cdot u,\theta-\Theta) + (2\nu\epsilon(u),\nabla(\varphi-\Phi)))\ dt\\
&= \frac{1}{|I|}\int_I (\dot U_h + U_h\cdot\nabla U_h - f,\varphi-\Phi) - (P_h,\nabla\cdot(\varphi-\Phi))\\
&+ (\nabla\cdot U_h,\theta-\Theta) + (2\nu\epsilon(U_h),\nabla(\varphi-\Phi)) + SD_\delta(U_h,P_h;\Phi,\Theta)\ dt,
\end{aligned}$$

using partial integration with $\varphi = \Phi = \phi$ on Γ_0 and $\varphi = \Phi = 0$ on Γ_1. We have now proved the following error representation (Theorem 3.1), where we express the total error as a sum of error contributions from the different elements K in space (assuming here for simplicity that the space mesh is constant in time), and we use the subindex K to denote integration over element K so that $(\cdot,\cdot)_K$ denotes the appropriate $L_2(K)$ inner product.

THEOREM 3.1. [illegible] (u,p) [illegible] (U_h,P_h) [illegible] $cG(1)$ $cG(1)$ [illegible] (φ,θ) [illegible] (3.4) [illegible]

$$|N(\sigma(u,p)) - N^h(\sigma(U_h,P_h))| = \left|\sum_{K\in\mathcal{T}_n}\mathcal{E}_K\right|,$$

[illegible] $\mathcal{E}_K = e_D^K + e_M^K$ [illegible]

$$\begin{aligned}
e_D^K &= \frac{1}{|I|}\int_I \big((\dot U_h + U_h\cdot\nabla U_h - f,\varphi-\Phi)_K - (P_h,\nabla\cdot(\varphi-\Phi))_K\\
&\quad + (\nabla\cdot U_h,\theta-\Theta)_K + (2\nu\epsilon(U_h),\epsilon(\varphi-\Phi))_K\big)\ dt,\\
e_M^K &= \frac{1}{|I|}\int_I SD_\delta(U_h,P_h;\Phi,\Theta)_K\ dt.
\end{aligned}$$

We may view e_D^K as the error contribution from the discretization on element K, and e_M^K as the contribution from the subgrid model (stabilization) on element K.

From the error representation in Theorem 3.1 there are various possibilities to construct error indicators and stopping criterions in an adaptive algorithm. Using standard interpolation estimates, with (Φ,Θ) a finite element interpolant of (φ,θ), we may estimate the contribution e_D^K from discretization as follows (cf. [12]):

$$\begin{aligned}
e_D^K \le \frac{1}{|I|}\int_I \big((|R_1(U_h,P_h)|_K + |R_2(U_h,P_h)|_K)\cdot(C_h h^2|D^2\varphi|_K + C_k k|\dot\varphi|_K)\\
+ \|R_3(U_h)\|_K(C_h h^2\|D^2\theta\|_K + C_k k\|\dot\theta\|_K)\big)\ dt,
\end{aligned}$$

where the residuals R_i are defined by

$$\begin{aligned} R_1(U_h,P_h) &= |\dot{U}_h + U_h\cdot\nabla U_h + \nabla P_h - f - \nu\Delta U_h|,\\ R_2(U_h,P_h) &= \nu D_2(U_h),\\ R_3(U_h,P_h) &= |\operatorname{div} U_h|, \end{aligned} \tag{3.7}$$

with

$$D_2(U_h)(x,t) = \max_{y\in\partial K}(h_n(x))^{-1}\left|\left[\frac{\partial U_h}{\partial n}(y,t)\right]\right| \tag{3.8}$$

for $x\in K$, with $[\cdot]$ the jump across the element edge ∂K. D^2 denotes second-order spatial derivatives, and we write $|w|_K \equiv (\|w_1\|_K, \|w_2\|_K, \|w_3\|_K)$, with $\|w\|_K = (w,w)_K^{1/2}$, and let the dot denote the scalar product in $\mathbb{R}^3$.

Note that $R_1(U_h,P_h)$ is defined elementwise and that with piecewise linear basis functions in space the Laplacian ΔU_h is zero, and that $R_1(U_h,P_h)+R_2(U_h,P_h)$ bounds the residual of the momentum equation, with the Laplacian term bounded by the second-order difference quotient $D_2(U_h)$ arising from the jumps of normal derivatives across element boundaries.

Replacing the exact dual solution (φ,θ) by a computed approximation (φ_h,θ_h), we are led to the following a posteriori error estimate:

$$|N(\sigma(u,p)) - N^h(\sigma(U_h,P_h))| \approx \left|\sum_{K\in\mathcal{T}_n}\mathcal{E}_{K,h}\right|, \tag{3.9}$$

where $\mathcal{E}_{K,h} = e^K_{D,h} + e^K_{M,h}$ with

$$\begin{aligned} e^K_{D,h} &= \frac{1}{|I|}\int_I \big((|R_1(U_h,P_h)|_K + |R_2(U_h,P_h)|_K)\cdot(C_h h^2|D^2\varphi_h|_K + C_k k|\dot{\varphi}_h|_K)\\ &\quad + \|R_3(U_h)\|_K\cdot(C_h h^2\|D^2\theta_h\|_K + C_k k\|\dot{\theta}_h\|_K)\big)\,dt,\\ e^K_{M,h} &= \frac{1}{|I|}\int_I SD_\delta(U_h,P_h;\varphi_h,\theta_h)_K\,dt, \end{aligned}$$

where we have replaced the interpolant (Φ,Θ) by (φ_h,θ_h). Again we may view $e^K_{D,h}$ as the error contribution from the discretization on element K, and $e^K_{M,h}$ as the contribution from the subgrid model (stabilization) on element K.

Remark 3.2. Non-Dirichlet boundary conditions, such as slip conditions at lateral boundaries and transparant outflow conditions, introduce additional boundary terms in the error representation in Theorem 3.1. Since the dual solutions for the examples in this paper are small at such non-Dirichlet boundaries, we neglect the corresponding boundary terms in the computations.

Remark 3.3. In the computations we use $C_k = 1/2$ and $C_h = 1/8$ as constant approximations of the interpolation constants in Theorem 3.1. These values are motivated by simple analysis on reference elements.

Remark 3.4. Note that for Theorem 3.1 to be valid, we only need the solution (u,p) to satisfy a weak formulation of the Navier–Stokes equations, with sufficient regularity for (3.2) to make sense.

3.4. An adaptive algorithm. For simplicity, we have the space mesh and time step k being constant in time, with the time step being equal to the smallest element diameter in the space mesh. In the computations we use an algorithm for adaptive mesh refinement in space based on the a posteriori error estimate (3.9), of the form: [illegible] $\mathcal{T}^0$ [illegible] $k = 0$ [illegible]

(1) [illegible] $\mathcal{T}^k$
(2) [illegible] $\mathcal{T}^k$
(3) [illegible] $\left|\sum_{K\in\mathcal{T}_k}\mathcal{E}^k_{K,h}\right| <$ [illegible]
(4) [illegible] $\mathcal{T}^k$ [illegible] $\mathcal{E}^k_{K,h} \to \mathcal{T}^{k+1}$
(5) [illegible] $k = k + 1$ [illegible] (1)

By refining the mesh size h, also the time step k and the subgrid model from stabilization SD_δ are refined implicitly, since both k and δ depend on h. The subgrid model may thus be thought of as being implicitly modified as we refine the mesh. But the error estimate (3.9) also allows for a direct adaptive modification of both the time step k and the subgrid model SD_δ, by locally refining k and δ, respectively. In this paper, we have not investigated the potential gain in using direct local refinement of k and δ, or using a space mesh varying in time.

We note that the structure of the error estimate (3.9) allows for a general interpretation of the subgrid model SD_δ; we may think of SD_δ as being the part of the numerical method not being a standard Galerkin (here cG(1)cG(1)) discretization of the Navier–Stokes equations.

4. Numerical results. We now seek to compute the [illegible] c_D, a mean value in time of a normalized drag force. Due to computational cost we have to truncate the time interval over which we average, and thus we approximate c_D by $\bar{c}_D$, a normalized drag force averaged over a given time interval I, defined by

$$\bar{c}_D = \frac{2}{\bar{U}_h^2 A} \times N(\sigma(u,p)), \tag{4.1}$$

where $\bar{U}_h$ is a bulk inflow velocity, A is an area, and $N(\sigma(u,p))$ is defined by (3.1). In computations, we approximate $\bar{c}_D$ by $\bar{c}_D^h$, defined by

$$\bar{c}_D^h = \frac{2}{\bar{U}_h^2 A} \times N^h(\sigma(U_h,P_h)), \tag{4.2}$$

with $N^h(\sigma(U_h,P_h))$ defined by (3.3). Thus we may use a scaled version of the a posteriori error estimate (3.9) to estimate the error $|\bar{c}_D - \bar{c}_D^h|$.

Below we will show that in the case of two bluff body benchmark problems, it is possible to use adaptive DNS/LES to compute approximations $\bar{c}_D^h$, such that the error $|\bar{c}_D - \bar{c}_D^h|$ is less than a few percent, using the computational power of a PC.

4.1. Flow around a surface mounted cube. We first consider a basic benchmark problem of CFD of computing the drag of a surface mounted cube at Reynolds number 40000. The incoming flow is laminar time-independent with a laminar boundary layer on the front surface of the body, which separates and develops a turbulent time-dependent wake attaching to the rear of the body. The flow in this problem is thus very complex with a combination of laminar and turbulent features including boundary layers and a large turbulent wake; see Figure 4.1.

The cube side length is $H = 0.1$, and the cube is centrally mounted on the floor of a rectangular channel of length $15H$, height $2H$, and width $7H$, at a distance of $3.5H$ from the inlet. The cube is subject to a Newtonian flow (u,p) governed by the Navier–Stokes equations (2.1) with kinematic viscosity $\nu = 2.5\times10^{-6}$ and a unit inlet bulk velocity corresponding to a Reynolds number of 40000, using the dimension of the cube as characteristic dimension. The inlet velocity profile is interpolated from

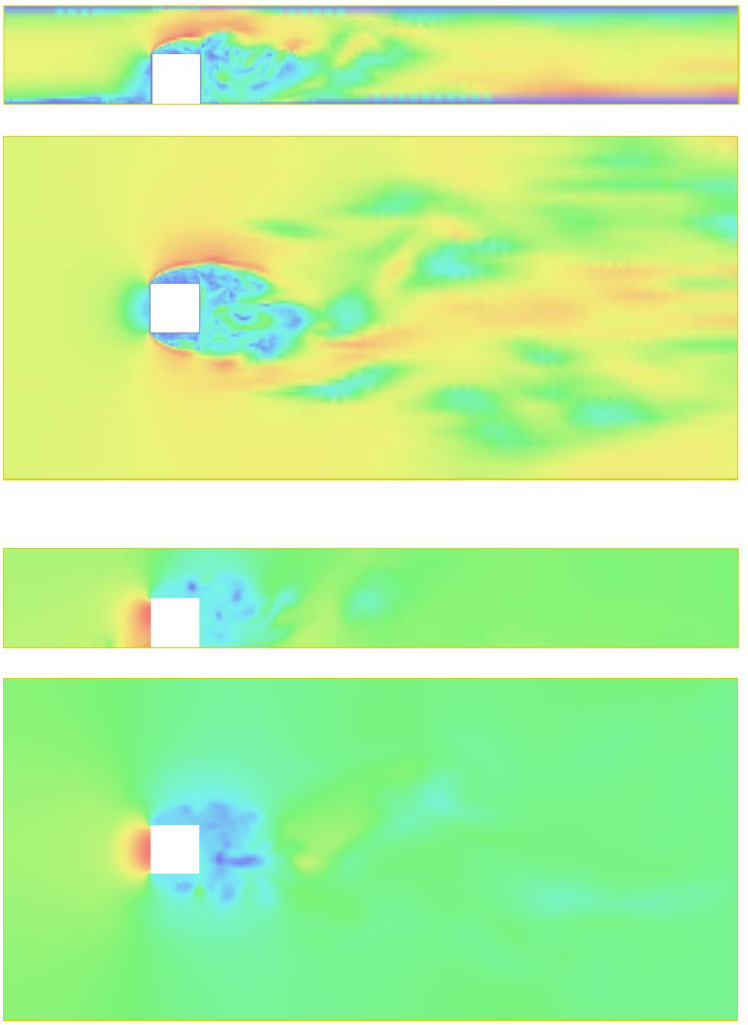


FIG. 4.1. *Surface mounted cube: Velocity* $|u|$ *(upper) and pressure* p *(lower), after* 16 *adaptive mesh refinements, in the* x_1x_2*-plane at* $x_3 = 3.5H$ *and in the* x_1x_3*-plane at* $x_2 = 0.5H$.

experiments, and we use no slip boundary conditions on the cube and the vertical channel boundaries, slip boundary conditions on the lateral channel boundaries, and a transparent outflow boundary condition.

We seek to approximate the drag coefficient $\bar{c}_D$, given by (4.1), averaged over a time interval $I = [0, 40H/\bar{U}_h]$ at fully developed flow. We set $\bar{U}_h = 1$ based on the bulk inflow velocity, and the area $A = H \times H = H^2$. This is a standard benchmark problem at the [illegible] [1], where also the inlet velocity profile is available for download.

In Figure 4.2 we plot the approximations $\bar{c}_D^h$ as we refine the mesh. We find that for the finer meshes $\bar{c}_D^h \approx 1.5$, a value that is well captured already using less than 10^5 mesh points. Without the contribution from the stabilizing terms in (3.3), we get a somewhat lower value for $\bar{c}_D^h$, where the difference is large on coarser meshes but

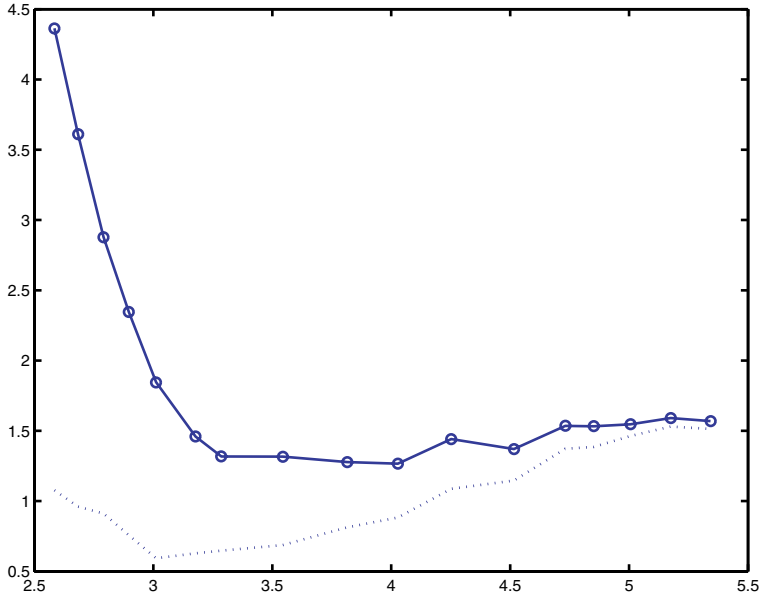


FIG. 4.2. *Surface mounted cube: Approximations of the drag coefficient* $\bar{c}_D^h$ *"∘" and the corresponding approximations without the contribution from the stabilizing term ":" in* (3.3)*, as functions of the* 10*-logarithm of the number of mesh points.*

small (less than 5%) on the finer meshes.

We know of no experimental reference values of $\bar{c}_D$, but in [5] a DNS is performed using about 70×10^6 degrees of freedom, with the same data as in this paper, giving $\bar{c}_D \approx 1.42$, where the stabilizing terms in (3.3) are not used for the evaluation of $\bar{c}_D$. The results in [5] should thus be compared to the somewhat lower values in Figure 4.2, resulting in a good agreement.

In [21], $\bar{c}_D$ is approximated using LES. The computational setup is similar to the one in this paper except for the numerical method, a different length of the time interval, and that in this paper we use a channel of length $15H$ compared to a channel of length $10H$ in [21]. Using LES with different meshes and subgrid models, approximations of $\bar{c}_D$ in the interval $[1.14, 1.24]$ are presented.

4.1.1. The dual solution. A snapshot of the dual solution is shown in Figure 4.3. We note that the dual solution, with boundary data on the cube, is of moderate size, and in particular is not exploding as pessimistic worst case analytical estimates may suggest, but rather seems to behave as if the net effect of the crucial reaction term (with large oscillating coefficient ∇U_h) is only a moderate growth. We also note that (φ_h, θ_h) is very concentrated in space, thus significantly influencing the adaptive mesh refinement. The resulting computational mesh after 15 adaptive mesh refinements is shown in Figure 4.3. The initial space mesh is uniform and very coarse, 384 mesh points, and without the dual weights in the a posteriori error estimate the mesh would come out quite differently. We notice in particular that the adaptive method automatically captures the turbulent wake, which seems to be essential for accurately computing drag.

4.1.2. A posteriori error estimates. In Figure 4.4 we plot the (normalized) a posteriori error estimates $e_{D,h}$ and $e_{M,h}$ from (3.9), as well as the true error based

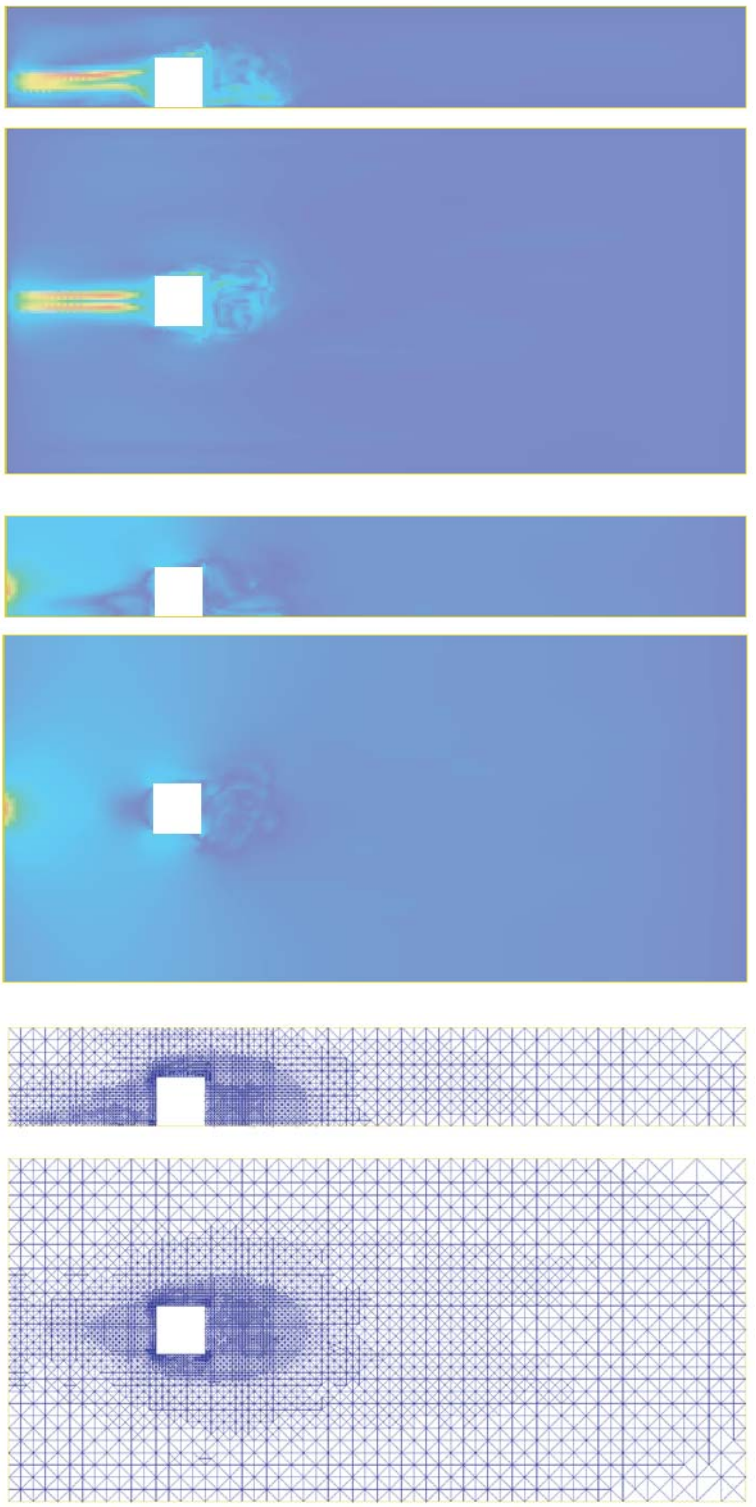


FIG. 4.3. *Surface mounted cube: Dual velocity* $|\varphi|$ *(upper), dual pressure* $|\theta|$ *(middle), and the resulting computational mesh (lower), in the* x_1x_2*-plane at* $x_3 = 3.5H$ *and in the* x_1x_3*-plane at* $x_2 = 0.5H$.

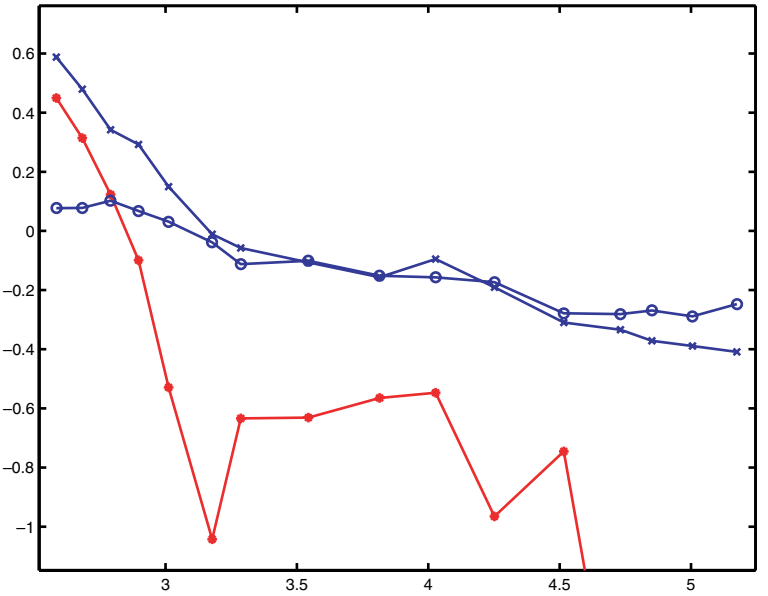


FIG. 4.4. *Surface mounted cube:* $\log_{10}$-$\log_{10}$ *plot of the (normalized) a posteriori error estimates* $e_{D,h}$ "$\circ$" *and* $e_{M,h}$ "$\times$" *and the true error* "$*$" *based on* $\bar{c}_D = 1.55$*, as functions of the number of mesh points in space.*

on the computational approximation on the finest mesh. The modeling error $e_{M,h}$ consists of sums in space and time of integrals over the space-time elements, and we may want to use a more conservative estimate of this term by taking the absolute values inside any or both of these sums. In the evaluation of $e_{M,h}$ in Figure 4.4, we have set the absolute values inside the sums in space and time.

We find that once the value for $\bar{c}_D^h$ has stabilized, the a posteriori error estimates indicate that it may be hard to further increase the precision.

4.2. Flow around a square cylinder. We now consider the benchmark problem of a square cylinder at Reynolds number 22000, based on the cylinder diameter $D = 0.1$ and the inflow velocity. The computational domain is a channel of size $21D \times 14D \times 4D$ in the x_1-direction with the cylinder directed in the x_3-direction and centered at $x_1 = 5D$ and $x_2 = 7D$. We have a unit inflow velocity, no slip boundary conditions on the cylinder, slip boundary conditions on the lateral boundaries, and a transparant outflow boundary condition.

As described, e.g., in [22, 7], characteristics of this flow are a turbulent wake of approximate diameter $1D$ attached to the trailing face of the cylinder, and two opposite shear layers periodically shedding vortices; see Figure 4.5. In addition, many authors have reported on a cycle-to-cycle variation, so-called [illegible], due to turbulence and 3d instabilities in the shear layers. In Figure 4.6, we show a time series of the vorticity for a computation on a fine mesh illustrating the transition in the shear layers.

The presence of phase jitter leads to complications when evaluating time averages, since the time averages are highly dependent of the length and location of the averaging interval, which has led to alternative ways to represent averages. One approach is to consider the so-called [illegible] [19], where a number of shedding cycles are chosen as "typical" for the flow, over which mean values are computed.

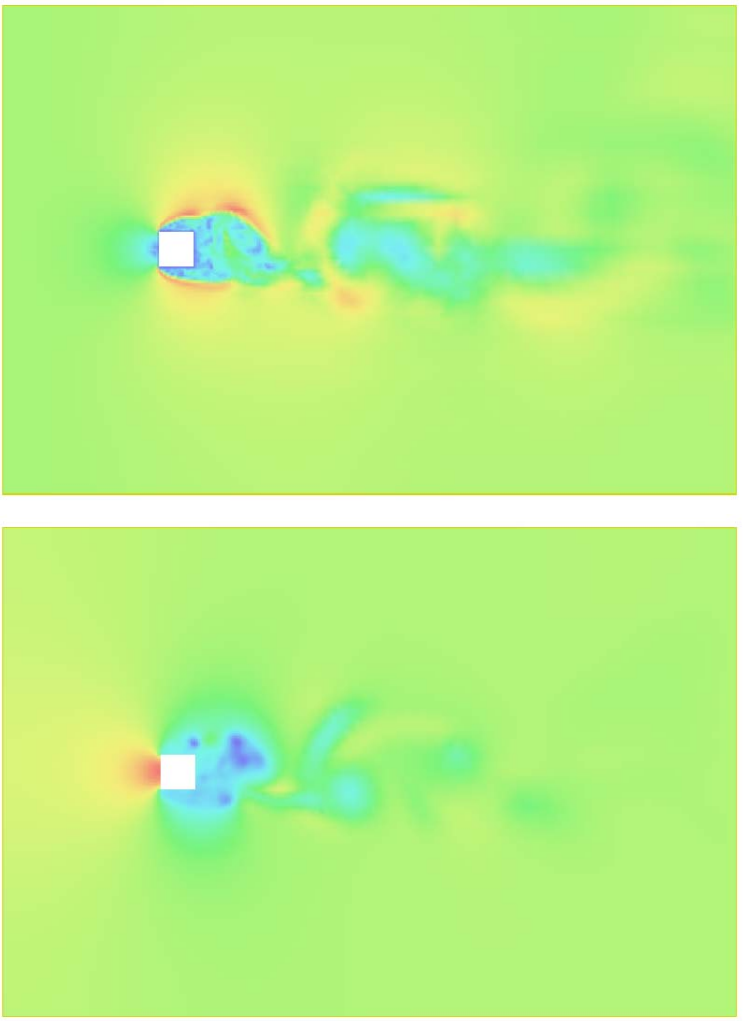


FIG. 4.5. *Square cylinder: Velocity* $|u|$ *(upper) and pressure* p *(lower), after nine adaptive mesh refinements, in the* x_1x_2*-plane at* $x_3 = 2D$.

We seek to approximate the drag coefficient $\bar{c}_D$, given by (4.1), averaged over a time interval $I = [0, 100D/\bar{U}_h]$ at fully developed flow, where we set $\bar{U}_h = 1$ based on the bulk inflow velocity, and the area $A = 4D \times D = 4D^2$.

In Figure 4.7 we plot the computed values of $\bar{c}_D^h$, as we refine the mesh. In Figure 4.7 we have also included the value for $\bar{c}_D^h$ without the stabilizing term in (3.3). For the finer meshes we get a $\bar{c}_D^h$ in the interval 2.0–2.4, and a value about 5% lower for the formulation without the stabilizing term. The variation in $\bar{c}_D^h$ could, to a certain degree, be explained by the effects of phase jitter, as noted above. In Figure 4.7 we plot the trajectory of the normalized drag force for the finest mesh, where we note variations in amplitude and local mean of the drag.

In [22], various reference values for this problem, including mean drag, are reported. Experimental reference values for $\bar{c}_D$ are reported to be in the interval 1.9–2.1, where the experiments are carried out with slightly different parameters, such as

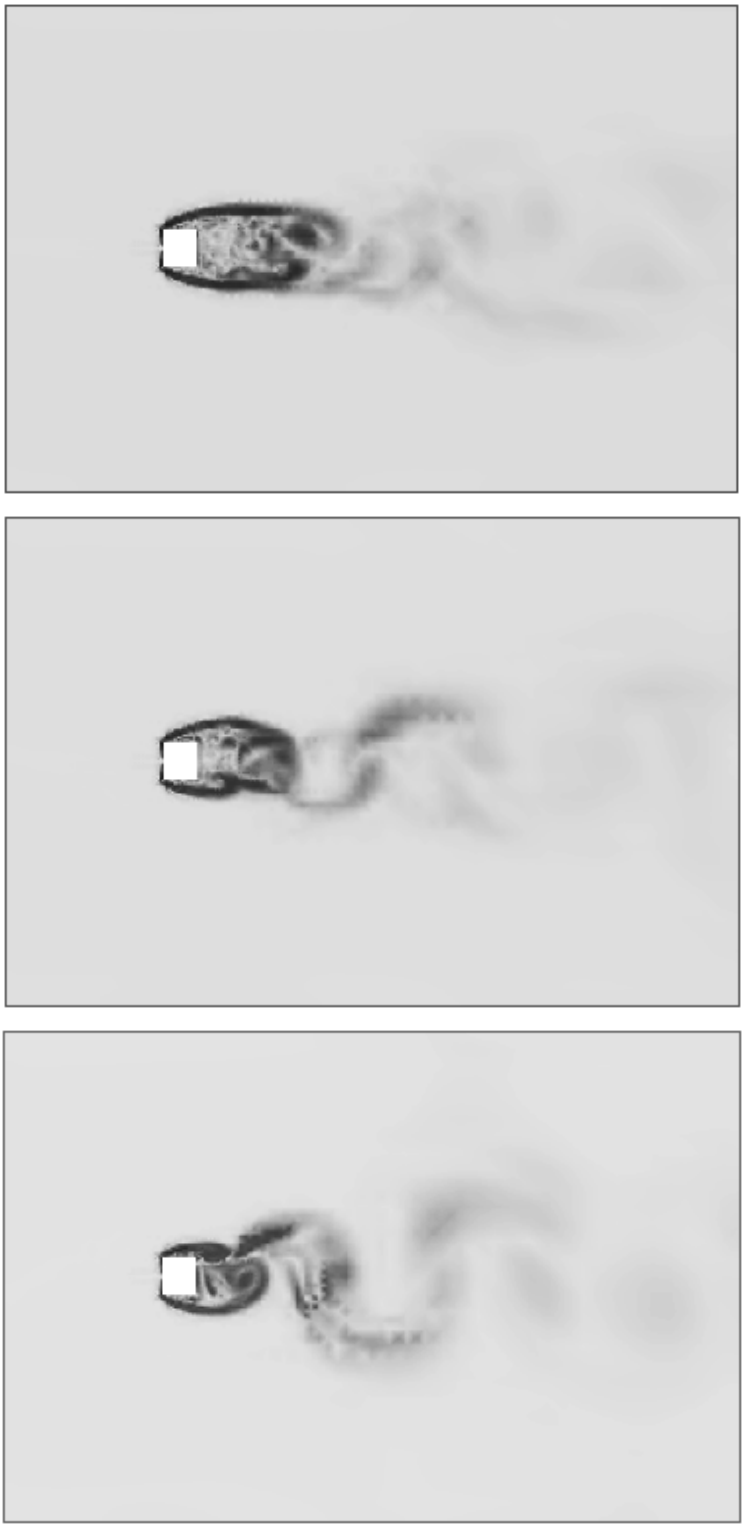


FIG. 4.6. *Square cylinder: Time evolution of the vorticity* $|\nabla \times u|$.

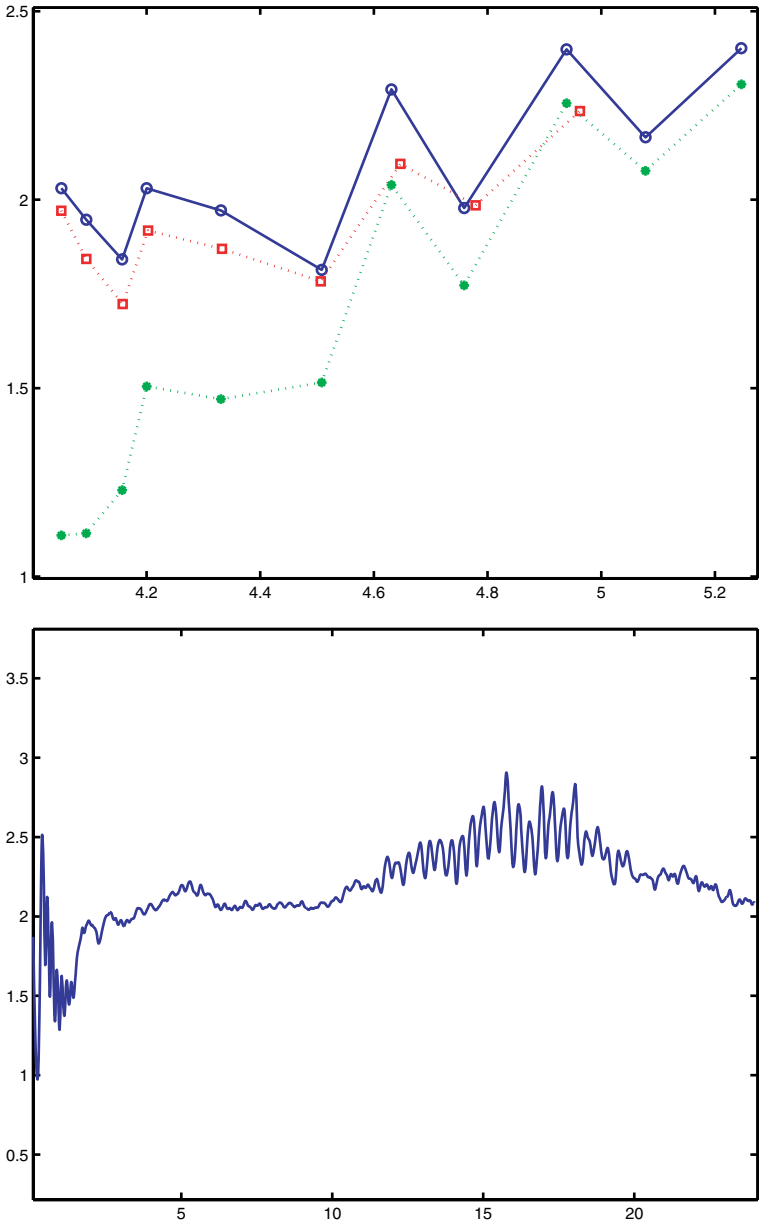


FIG. 4.7. *Square cylinder. Upper: Approximations of the drag coefficient* $\bar{c}_D^h$ *"∘," the corresponding approximations without the contribution from the stabilizing term "∗" in* (3.3)*, and the corresponding approximations with* 2% *white noise in inflow velocity "□" as functions of the* 10*-logarithm of the number of mesh points in space. Lower: normalized drag force as a function of time.*

a slightly lower Reynolds number, a longer cylinder, and a turbulent level of 2% in the inflow velocity. LES results are reported in the interval 1.66–2.77, and Reynolds averaged Naview–Stokes equations (RANS) results in the interval 1.6–2.0. To test the sensitivity in inflow data, we compare with a similar computation with 2% white noise added to the inflow velocity. The results are plotted in Figure 4.7, giving similar values for $\bar{c}_D^h$, although possibly somewhat lower, closer to the experimental results.

TABLE 4.1

Square cylinder. Reference values from [22]: Drag and lift ($\bar{c}_D$ and $\bar{c}_L$), and corresponding rms-values ($c_{D,rms}$ and $c_{L,rms}$), and Strouhal number St. The values in the parentheses correspond to evaluation of drag without contribution from the stabilizing term.

Output	Experiment	LES	RANS	Adaptive DNS/LES
$\bar{c}_D$	1.9–2.1	1.66–2.77	1.637–2.004	2.40 (2.31)
$c_{D,rms}$	0.1–0.2	0.10–0.27	—	0.17 (0.17)
$\bar{c}_L$	—	−0.3–0.007	—	−0.02
$c_{L,rms}$	0.1–1.4	0.38–1.79	—	0.84
St	0.132	0.066–0.15	0.134–0.143	0.14

As for the rest of the output data presented in [22], our results show a very good agreement, which we present in Table 4.1, where we give our results with adaptive DNS/LES for the finest mesh with about 1.2×10^5 mesh points in space, after nine adaptive mesh refinements.

Again we note the sensitivity in mean drag, where we, from Figure 4.7, can see that translating the averaging interval results in different $\bar{c}_D^h$. The computation of $\bar{c}_D^h$ in Table 4.1 corresponds to the interval $[10, 20]$ in Figure 4.7, and we see that translating this interval suitably would result in a lower $\bar{c}_D^h$, closer to the experimental reference values.

4.2.1. Dual solution. A snapshot of the dual solution is shown in Figure 4.8. We note that the dual solution, as in the case of the surface mounted cube, is of moderate size, and is concentrated in space. The resulting computational mesh after nine adaptive mesh refinements is shown in Figure 4.9. We notice that again the adaptive method automatically captures the turbulent wake, which is essential for accurately computing drag.

4.2.2. A posteriori error estimates. In Figure 4.10 we plot the (normalized) a posteriori error estimates $e_{D,h}$ and $e_{M,h}$ from (3.9), as well as an estimate of the true error based on the computational approximations on the finest meshes, suggesting that 2.2 may be a good candidate for a representative value of $\bar{c}_D$. In Figure 4.10, we set the absolute values inside the sums in space and time of $e_{D,h}$ and $e_{M,h}$. Thus error cancellation is not possible, leading to conservative error estimates.

4.3. Reliability and efficiency of the adaptive method. We now focus on two key points relating to the [illegible] and [illegible] of the adaptive method based on the a posteriori error estimate (3.9), which directly couples to whether this estimate indeed gives a reasonably sharp bound of the true error or not. The two key points are (i) replacement of u by a computed velocity U_h in the dual problem, and (ii) replacement of the dual solution (φ, θ) by a computed dual solution (φ_h, θ_h). We may view both these points to relate to a [illegible] of the dual solution under perturbations of (i) the convection coefficient and (ii) numerical computation.

We first consider the discretization error term $e_{D,h}$. We obtain a rough estimate of this term using Cauchy's inequality in space and time as follows (taking only space discretization coupled to φ into account and neglecting the small ν-term):

$$e_{D,h} \leq C_h \|hR_1(U_h, P_h)\| \|hD^2\varphi_h\|,$$

where $C_h \approx 0.1$, and by the least-squares stabilization in cG(1)cG(1) we have that $\|\sqrt{h}R_1(U_h, P_h)\|$ is bounded (recalling that $\delta_1 \sim h$, and neglecting the time derivative). Here $\|\cdot\| = \|\cdot\|_{L_2(I;L_2(\Omega))}$ denotes an L_2 norm in space time. Thus, very

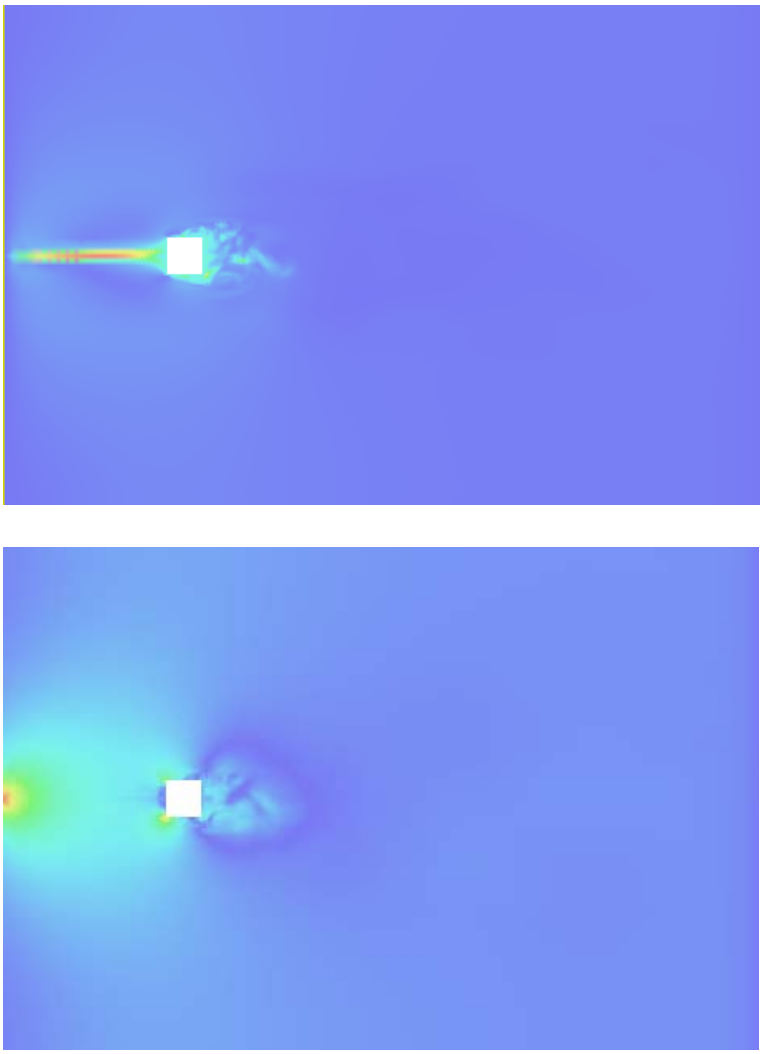


FIG. 4.8. *Square cylinder: Dual velocity* $|\varphi|$ *(upper) and dual pressure* $|\theta|$ *(lower), after nine adaptive mesh refinements with respect to mean drag, in the* x_1x_3*-plane at* $x_2 = 7D$ *and in the* x_1x_2*-plane at* $x_3 = 2D$*, with the darker color representing high values.*

roughly, we would expect to have

$$e_{D,h} \leq C_h\sqrt{h}\|hD^2\varphi_h\|.$$

In Figure 4.11 we display the variation of $\|hD^2\varphi_h\|$ as a function of the number of mesh points in space for the bluff body problems, and we find that after some initial refinements the dual solution shows a stability of $\|hD^2\varphi_h\|$ under the mesh refinement.

Next, the error contribution from subgrid modeling $e_{M,h} = SD_\delta(U_h, P_h; \varphi_h, \theta_h)$ may be estimated roughly as follows, using the basic energy estimate to bound $SD_\delta(U_h, P_h; U_h, P_h)$, Cauchy's inequality, and recalling that $\delta_1 \sim h$, to get

$$e_{M,h} \leq \sqrt{h}\|\nabla\varphi_h\|,$$

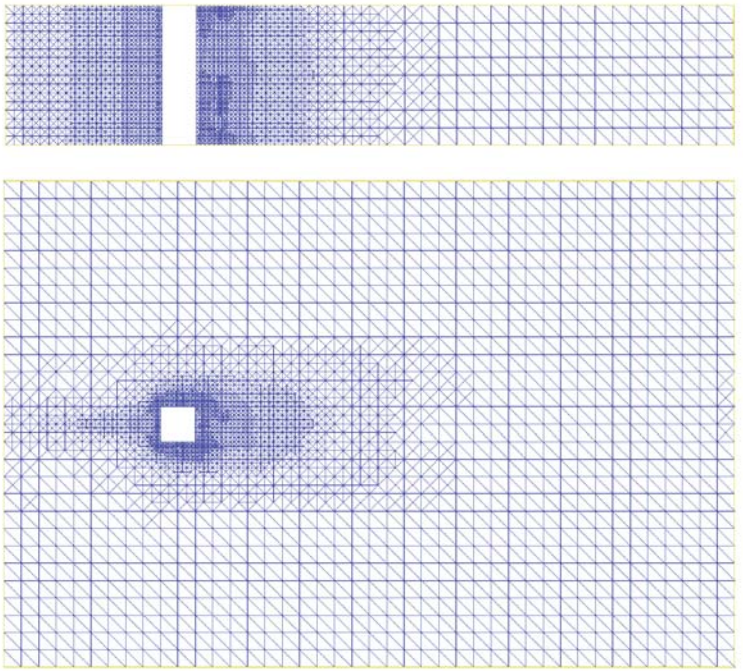


FIG. 4.9. *Square cylinder: Computational mesh after nine adaptive mesh refinements with respect to mean drag, in the x_1x_3-plane at $x_2 = 7D$ and in the x_1x_2-plane at $x_3 = 2D$.*

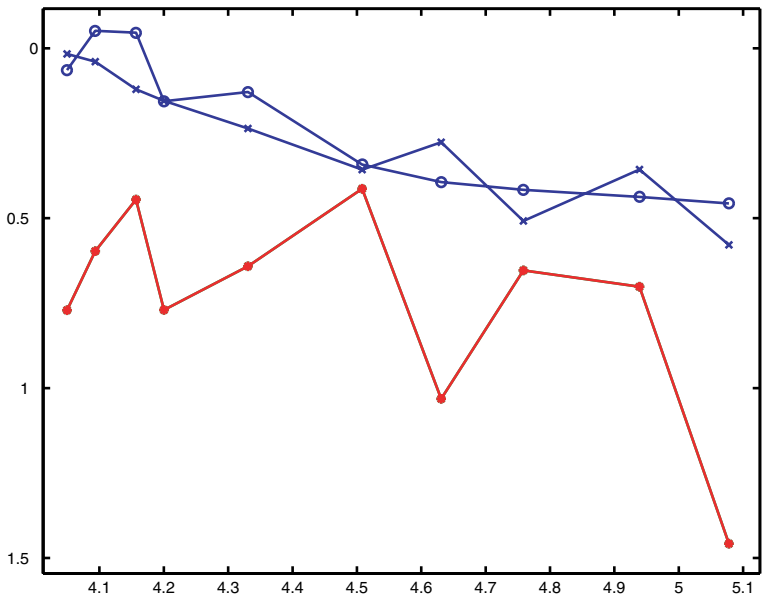


FIG. 4.10. *Square cylinder:* $\log_{10}$-$\log_{10}$ *plot of the (normalized) a posteriori error estimates* $e_{D,h}$ "$\circ$" *and* $e_{M,h}$ "$\times$" *and the true error "$*$" based on* $\bar{c}_D = 2.2$*, as functions of the number of mesh points in space.*

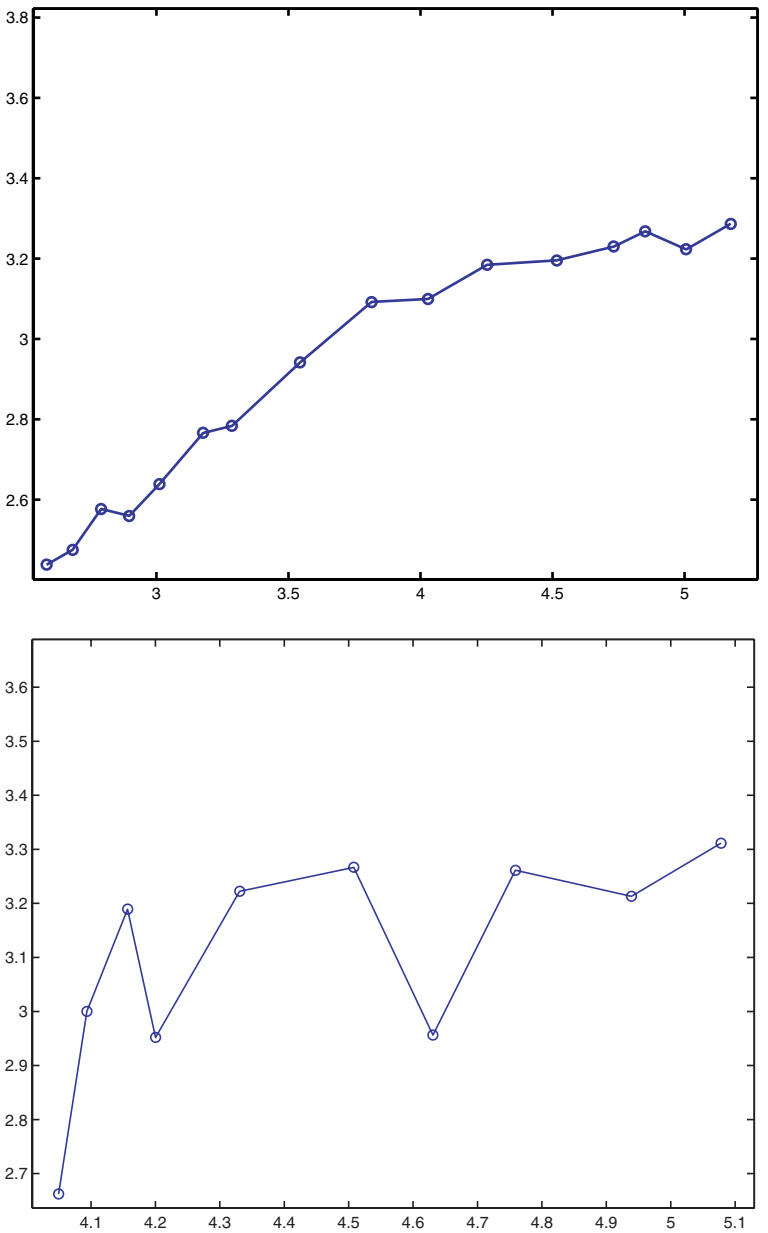


FIG. 4.11. Log_{10}-$\log_{10}$ *plot of the stability factor* $\|hD^2\varphi_h\|_{L_2(I;L_2(\Omega))}$ *for the surface mounted cube* (*upper*) *and the square cylinder* (*lower*), *versus number of mesh points in space.*

where we accounted only for the φ_h term. We notice in Figure 4.12 that $\|\nabla\varphi_h\|$ shows a stability under the mesh refinement suggesting that indeed $e_{M,h}$ may get below a moderate tolerance under refinement of the mesh without reaching a DNS.

Thus we conclude that the crucial computed dual weights show a stability under mesh refinement which indicates that the a posteriori error estimate (3.9) may be reliable and also reasonably efficient. In particular, we have given evidence that the net effect on the dual weights from replacing u by U_h in the dual problem may

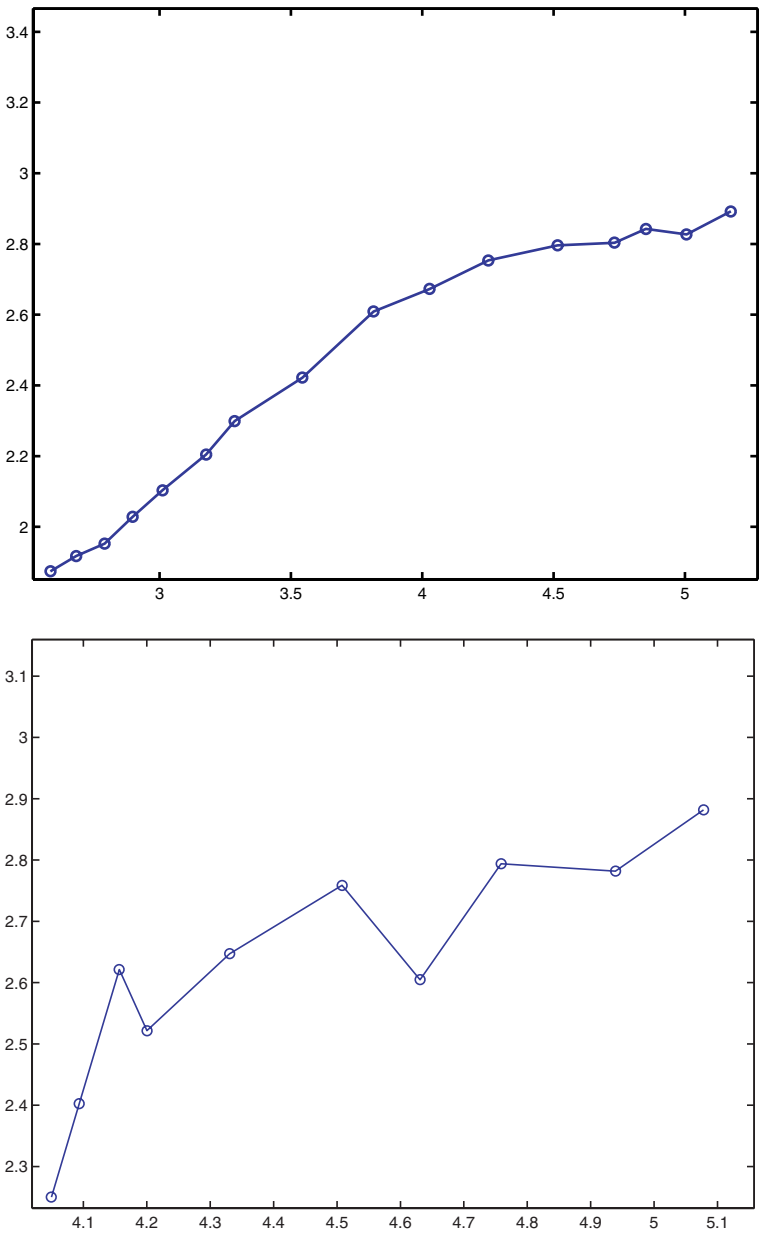


FIG. 4.12. Log_{10}-$\log_{10}$ *plot of the stability factor* $\|\nabla\varphi_h\|_{L_2(I;L_2(\Omega))}$ *for the surface mounted cube* (*upper*) *and the square cylinder* (*lower*), *versus number of mesh points in space.*

be small, even though this may correspond locally to a large perturbation since U_h cannot be expected to approximate u pointwise.

4.4. Turbulent dissipation. For accurate approximation of the drag in bluff body problems we need to capture the correct global dissipation, which follows from the basic global energy balances for the Navier–Stokes equations (2.1) and cG(1)cG(1) (2.3), obtained by multiplication by (u, p) and choosing $(v, q) = (U_h, P_h)$, respectively,

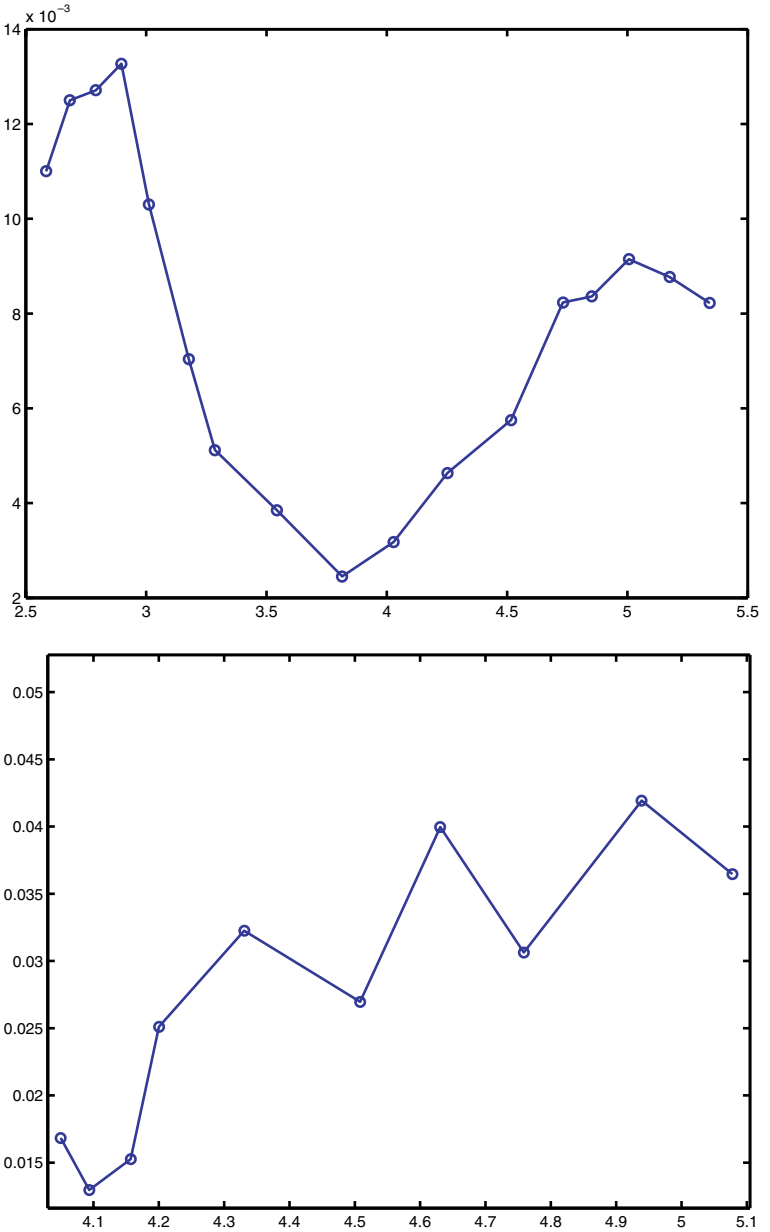


FIG. 4.13. *Volume of the turbulent wake, defined as the part of the domain with mean dissipation intensity of* $D(U_h, P_h) > 0.2$, *versus* $\log_{10}$ *of the number of mesh points, for the surface mounted cube* (*upper*) *and the square cylinder* (*lower*).

to get

$$\frac{d}{dt}\|u\|^2 \approx \int_{\Gamma_{\rm in}} pu \cdot n \; ds - \int_{\Gamma_{\rm out}} pu \cdot n \; ds - \nu\|\epsilon(u)\|^2,$$
$$\frac{d}{dt}\|U_h\|^2 \approx \int_{\Gamma_{\rm in}} P_h U_h \cdot n \; ds - \int_{\Gamma_{\rm out}} P_h U_h \cdot n \; ds - (\nu\|\epsilon(U_h)\|^2 + SD_\delta(U_h, P_h; U_h, P_h)),$$

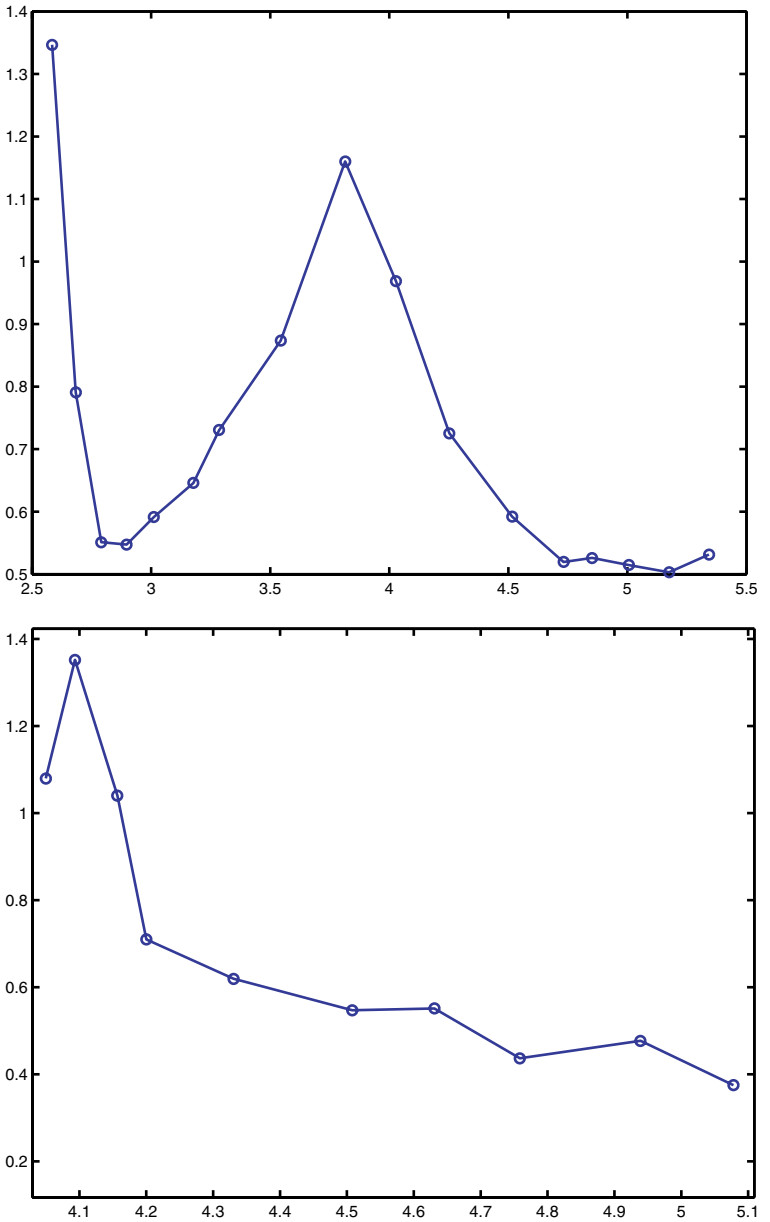


FIG. 4.14. *The mean value of the dissipation intensity of* $D(U_h, P_h)$ *as the sum of the intensity of* $D_\nu(U_h)$ *and the intensity of the stabilization term* $SD_\delta(U_h, P_h; U_h, P_h)$ *in the turbulent wake, versus* $\log_{10}$ *of the number of mesh points, for the surface mounted cube* (*upper*) *and the square cylinder* (*lower*).

using partial integration, dropping the small boundary terms containing ν, and denoting by Γ_{in} and Γ_{out} the inflow and outflow boundaries, respectively. Here $\|\cdot\|$ denotes the $L_2(\Omega)$ norm, and $D_\nu(u) = \nu\|\epsilon(u)\|^2$ represents the exact global dissipation (rate), and $D(U_h, P_h) = D_\nu(U_h) + SD_\delta(U_h, P_h; U_h, P_h)$ the corresponding cG(1)cG(1) approximate global dissipation (rate). We notice that the difference of the two boundary integrals represents the pressure drop from inflow to outflow, which

roughly should correspond to the drag. Since in the bluff body problems we have $\frac{d}{dt}\|u\|^2 \sim \frac{d}{dt}\|U_h\|^2 \sim 0$, we thus have that the pressure drop $\sim$ global dissipation, and thus we expect that the drag $\sim$ global dissipation.

In Figure 4.13 we show the development under the mesh refinement process of the volume of the turbulent wake. We note that the volume increases as the early refinement proceeds. The initial large values for the volume on the coarsest meshes are related to large numerical dissipation from the stabilization on these under resolved meshes. We may view the refinement as increasing the effective Reynolds number in the computation, and thus we may expect the expansion of the turbulent wake to parallel an expansion of the wake as the Reynolds number increases.

In Figure 4.14 we show (the mean value of) the computed intensity of the global dissipation $D(U_h, P_h)$ in the turbulent wake, being the sum of the intensity of $D_\nu(U_h)$, which is small, and the intensity of $SD_\delta(U_h, P_h; U_h, P_h)$ corresponding to the stabilization. We hope (the mean value of) $D(U_h, P_h)$ to be an approximation of (the mean value of) the dissipation rate $D_\nu(u)$, which we expect to be most significant in the turbulent wake. We observe that (the mean value of) the intensity of $D(U_h, P_h)$ is very nearly constant during the finer part of the refinement process, which we may take as evidence that indeed (the mean of) the intensity of the mesh dissipation $D(U_h, P_h)$ may approximate (the mean value of) the true fine scale dissipation intensity $D_\nu(u)$.

5. Summary and future directions. We have considered two bluff body benchmark problems, where we have shown that we are able to compute the drag coefficient to an estimated tolerance of a few percent, using about 10^5 mesh points in space, with the computational power of a PC.

The mesh refinement criterion and the stopping criterion of the adaptive algorithm are based on a posteriori error estimates of an output of interest, in the form of a space-time integral of a residual times a dual weight. The dual weight is obtained from the computational approximation of an associated dual problem with data coupled to the output of interest. We have presented computational evidence that crucial properties of the dual solution are stable under the adaptive mesh refinement.

In adaptive DNS/LES no filtering is used, but instead output from the exact Navier–Stokes equations is approximated directly, using cG(1)cG(1) with the least-squares stabilization acting as a dissipative subgrid model. In particular, we circumvent the problem of closure as no modeling of Reynolds stresses is necessary. Instead we estimate the error contribution from subgrid modeling a posteriori, which we find to be small.

Encouraged by the results in this paper, we plan to extend the study of adaptive DNS/LES to more benchmark problems of turbulent flow. In addition, there are many potential optimizations of the adaptive algorithm to investigate. For example, one would like to consider the possible gains of using dynamically changing meshes, adaptively refined time steps, and adaptive choice of different time steps in different parts of the domain. Also, one would like to adaptively choose δ in the stabilization SD_δ, or more generally, to adaptively choose the form of SD_δ.

REFERENCES

[1] Cde forum, http://www.phi.chalmers.se/cdeforum/.

[2] I. Babuška and A. D. Miller, *The post-processing approach in the finite element method,* iii: *A posteriori error estimation and adaptive mesh selection*, Int. J. Numer. Methods Engrg., 20 (1984), pp. 2311–2324.

[3] R. Becker and R. Rannacher, *A feed-back approach to error control in adaptive finite element*

methods: Basic analysis and examples, East-West J. Numer. Math., 4 (1996), pp. 237–264.
[4] R. Becker and R. Rannacher, *A posteriori error estimation in finite element methods*, Acta Numer., 10 (2001), pp. 1–102.
[5] S. Buijssen and S. Turek, *private communication*, University of Dortmund, Dortmund, Germany, 2004.
[6] K. Eriksson, D. Estep, P. Hansbo, and C. Johnson, *Introduction to adaptive methods for differential equations*, in Acta Numerica 1995, Acta Numer. 4, Cambridge University Press, Cambridge, UK, 1995, pp. 105–158.
[7] S. Fohanno and R. Martinuzzi, *Coherent motion and droplet dispersion in the near-wake of a square cylinder*, in Proceedings of FLUCOME 2000, Sherbooke, Canada, 2000.
[8] M. Giles, M. Larson, M. Levenstam, and E. Süli, *Adaptive Error Control for Finite Element Approximations of the Lift and Drag Coefficients in Viscous Flow*, Technical report NA-76/06, Oxford University Computing Laboratory, Oxford, UK, 1997.
[9] M. Giles and E. Süli, *Adjoint methods for PDEs: A posteriori error analysis and postprocessing by duality*, in Acta Numerica 2002, Acta Numer. 11, Cambridge University Press, Cambridge, UK, 2002, pp. 145–236.
[10] J. Hoffman, *Adaptive finite element methods for LES: Computation of the drag coefficient in a turbulent flow around a surface mounted cube*, SIAM J. Sci. Comput., submitted.
[11] J. Hoffman, *Computation of functionals in 3d incompressible flow for stationary benchmark problems using adaptive finite element methods*, DOE report 03-003, Courant Mathematics and Computing Laboratory, Courant Institute of Mathematical Sciences, New York University, New York, NY, 2003.
[12] J. Hoffman, *On duality based a posteriori error estimation in various norms and linear functionals for LES*, SIAM J. Sci. Comput., 26 (2004), pp. 178–195.
[13] J. Hoffman and C. Johnson, *Adaptive finite element methods for incompressible fluid flow*, in Error Estimation and Solution Adaptive Discretization in Computational Fluid Dynamics, Lecture Notes in Comput. Sci. and Engrg., T. J. Barth and H. Deconinck, eds., Springer-Verlag, Heidelberg, 2002.
[14] J. Hoffman and C. Johnson, *Applied Mathematics Body and Soul: Vol* 5: *Fluid Dynamics A: Incompressible Flow*, Springer-Verlag, Berlin, 2005.
[15] J. Hoffman and C. Johnson, *Computability and Adaptivity in CFD*, Encyclopedia of Computational Mechanics, E. Stein, R. de Borst, and T. J. R. Hughes, eds., John Wiley and Sons, New York, 2004.
[16] J. Hoffman and C. Johnson, *A new approach to computational turbulence modeling*, Comput. Methods Appl. Mech. Engrg., to appear.
[17] J. Hoffman and C. Johnson, *On the Uniqueness of Weak Solutions of Navier-Stokes Equations: Remarks on a Clay Mathematics Institute Prize Problem*, Chalmers Finite Element Center Preprint 2004–07, Chalmers University of Technology, Göteborg, Sweden, 2004.
[18] J. Hoffman, C. Johnson, and A. Logg, *Dreams of Calculus: Perspectives on Mathematics Education*, Springer-Verlag, Berlin, 2004.
[19] A. Hussein, *Coherent structures—reality and myths*, Phys. Fluids, 26 (1983), pp. 2816–2850.
[20] C. Johnson, *Adaptive Finite Element Methods for Conservation Laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Springer Lecture Notes in Mathematics, Springer-Verlag, Berlin, 1998, pp. 269–323.
[21] S. Krajnović and L. Davidson, *Large-eddy simulation of the flow around a bluff body*, AIAA J., 40 (2002), pp. 927–936.
[22] W. Rodi, J. H. Ferziger, M. Breuer, and M. Pourquié, *Status of large eddy simulation: Results of a workshop*, ASME J. Fluids Engrg, 119 (1997), pp. 248–262.
[23] P. Sagaut, *Large Eddy Simulation for Incompressible Flows*, Springer-Verlag, Berlin, Heidelberg, New York, 2001.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 208–230
© 2005 Society for Industrial and Applied Mathematics

THE EXPLICIT SIMPLIFIED INTERFACE METHOD FOR COMPRESSIBLE MULTICOMPONENT FLOWS*

BRUNO LOMBARD† AND ROSA DONAT‡

Abstract. This paper concerns the numerical approximation of the Euler equations for multicomponent flows. A numerical method is proposed to reduce spurious oscillations that classically occur around material interfaces. It is based on the explicit simplified interface method (ESIM), previously developed in the linear case of acoustics with stationary interfaces [J. Piraux and B. Lombard, *J. Comput. Phys.*, 168 (2001), pp. 227–248]. This technique amounts to a higher-order extension of the ghost fluid method introduced in Euler multicomponent flows [R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *J. Comput. Phys.*, 152 (1999), pp. 457–492]. The ESIM is coupled to sophisticated shock-capturing schemes for time-marching, and to level-sets for tracking material interfaces. Jump conditions satisfied by the exact solution and by its spatial derivative are incorporated in numerical schemes, ensuring a subcell resolution of material interfaces inside the meshing. Numerical experiments show the efficiency of the method for rich-structured flows.

Key words. Euler equations, multicomponent flows, jump conditions, interface methods, ghost fluid method, ENO-WENO, level-set

AMS subject classifications. 65M06, 65M99, 76T05

DOI. 10.1137/030601041

1. Introduction.

Let us consider multicomponent flows composed of pure inviscid fluids separated by material interfaces. These flows arise in a wide range of physical situations, from water-steam to bubbly flows, liquid suspensions, or even high-speed impacts on solids. They may be modeled by the Euler equations, augmented by additional equations describing the fluid composition. The numerical simulation of such configurations leads to major difficulties.

Indeed, even state-of-the-art numerical schemes for single-component flows cannot be applied directly in multicomponent flow simulations. These schemes give rise to oscillations and other computational inaccuracies near material interfaces. For example, any Godunov-type shock-capturing scheme which conserves the mass of individual species fails to maintain pressure equilibrium at a material interface [1]. The unphysical pressure oscillations stem from the fact that in a shock-capturing scheme, the transition across a material interface is governed by the numerical viscosity of the scheme. When two fluids are involved, intermediate states generated in the numerical transition layer (corresponding to the material interface) are not physically consistent with any component of the mixture: updating the pressure field via one particular equation of state (EOS) generates erroneous pressure fluctuations. Since material interfaces lack the compressive mechanisms associated to shocks, errors generated within the diffused interface escape and contaminate all flow variables.

*Received by the editors September 30, 2003; accepted for publication (in revised form) September 21, 2004; published electronically August 22, 2005.
http://www.siam.org/journals/sisc/27-1/60104.html

†Laboratoire de Mécanique et d'Acoustique, 31 chemin Joseph Aiguier, 13402 Marseille, France (lombard@lma.cnrs-mrs.fr). The research of this author was done during his postdoctoral position at the University of Valencia; it was supported by the European network HYKE, contract HPRN-CT-2002-00282.

‡Departament de Matematica Aplicada, Universitat de Valencia, 46100 Burjassot, Spain (donat@uv.es). The research of this author was partially supported by the Spanish MCYT-BFM-2001-2814.

Many numerical methods have been proposed to avoid these unphysical oscillations. See, e.g., [2] for a concise survey of up-to-date multicomponent methods. Remaining within the [illegible] (as opposed to [illegible]) computational framework, we distinguish two main approaches.

[illegible]. Material interfaces are approximated by diffused fronts. An artificial EOS (or mixture model) is defined, based on thermodynamical arguments [3, 17, 23], which is considered valid for the entire fluid mixture and is used to update the pressure from conserved variables. These models have been analyzed in [1, 2, 15, 16, 31]. The elimination of unphysical oscillations generally involves sacrificing strict conservation to some degree, except in [23] where a conservative algorithm maintains oscillations under a computationally acceptable level.

[illegible] (as in the present paper). A level-set function is used to track material interfaces in an Eulerian manner, i.e., without explicitly computing their location. Depending on the sign of the level-set, one applies the corresponding EOS. Thermodynamic properties of the fluid change discontinuously across material interfaces, which are preserved as sharp discontinuities. A simple approach was first used in [24], but it suffers from severe computational inaccuracies in multicomponent flow computations [16].

The [illegible] (GFM) [8] is the best-known method belonging to this second group. On each side of a material interface, two fluids are considered: the "real fluid" (i.e., the fluid that really exists on this side) and a "ghost fluid" (i.e., the fluid with the same pressure and velocity as the real fluid on that side, but the entropy of the real fluid on the other side). Near a material interface, classical single-component schemes are then simply applied both on "real fluid" values and on "ghost fluid" values. Extensions to multidimensional problems and to other physical situations have been tackled by the GFM [7, 9]. However, the GFM suffers from some inaccuracies which are not appreciated when material interfaces separate uniform states. These numerical artifacts are mainly due to the zeroth-order extrapolations used to define the ghost fluid. Simple minded first-order extrapolations do not work [8].

The goal of the present paper is to remove these drawbacks of the GFM by correctly increasing the precision of the computation across a material interface. We propose to implement a carefully designed first-order extrapolation procedure to obtain the "ghost fluid" values. To do so, we adapt the [illegible] (ESIM), previously developed for the linear hyperbolic systems of acoustics with stationary interfaces [21, 22, 27]. Note that another linear extrapolation in the context of the GFM has been proposed in [4] in order to couple Eulerian and Lagrangian computations. In the present paper, however, the goal is to enforce first-order jump conditions at material interfaces.

Interface methods have been widely used in the numerical treatment of boundaries and interfaces in PDEs; see, e.g., the [illegible] (IIM), applied to elliptic equations [18] and linear hyperbolic systems [34]. See [20, 27] for a concise survey of up-to-date interface methods. To our knowledge, the present paper is the first attempt to apply interface methods to the nonlinear hyperbolic system of the Euler equations. The computational complexity of the resulting algorithm is essentially the same as that of the GFM, and it can easily be coupled to a wide class of high-order shock-capturing schemes.

The paper is organized as follows. Section 2 recalls the level-set framework to model compressible multicomponent flows. Section 3 describes the numerical schemes. The interface method is detailed in section 4. Numerical tests are proposed in section

5. Conclusions and future works are drawn in section 6.

2. The level-set framework for multicomponent flows.

2.1. The Euler equations. We focus on multicomponent flows consisting of pure fluids separated by material interfaces. Assuming that all components can be described by a single velocity and pressure function, the flow can be modeled by the compressible Euler equations expressing conservation of mass, momentum, and energy of the fluid mixture. Let ρ be the density of the fluid mixture, u the velocity, p the pressure, and $e = \varepsilon + \frac{1}{2}u^2$ the specific total energy, with ε the specific internal energy. Then, we have

$$\frac{\partial}{\partial t}\boldsymbol{U} + \frac{\partial}{\partial x}\boldsymbol{f}(\boldsymbol{U}) = \boldsymbol{0}, \tag{2.1}$$

where the vector of conserved quantities $\boldsymbol{U}$ and the flux function $\boldsymbol{f}$ are

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho\, u \\ \rho\, e \end{pmatrix}, \qquad \boldsymbol{f}(\boldsymbol{U}) = \begin{pmatrix} \rho\, u \\ \rho\, u^2 + p \\ u\,(\rho\, e + p) \end{pmatrix}. \tag{2.2}$$

To close the system (2.1), we need to specify the EOS. In this paper, we consider the [illegible] EOS

$$p = (\gamma - 1)\,\rho\,\varepsilon - \gamma\, p_\infty, \tag{2.3}$$

where p_∞ is the stiffness parameter. The basic [illegible] case is recovered for $p_\infty = 0$; in this case, γ represents the ratio of specific heats. The EOS (2.3) is a reasonable approximation for gases, liquids, and even solids under huge pressure conditions [29]. The sound speed c is given by

$$c^2 = \frac{\gamma\,(p + p_\infty)}{\rho}. \tag{2.4}$$

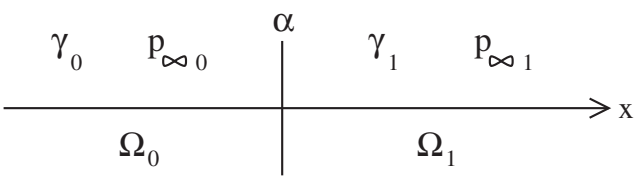


FIG. 2.1. *Two components Ω_0 and Ω_1 separated by a material interface.*

2.2. The level-set equation. The flow description is completed by an additional equation describing the fluid composition. In the level-set framework, a scalar function ϕ is used to track material interfaces. For the sake of simplicity, we consider only two components Ω_0 and Ω_1, separated by a material interface at $\alpha(t)$ (see Figure 2.1). We suppose that each fluid satisfies (2.3) and that physical parameters of (2.3) may be discontinuous at α:

$$(\gamma, p_\infty) = \begin{cases} (\gamma_0, p_{\infty\,0}) & \text{if } x \le \alpha, \\ (\gamma_1, p_{\infty\,1}) & \text{if } x > \alpha. \end{cases} \tag{2.5}$$

The marker variable ϕ is initialized as the signed distance function to $\alpha(0)$, which is known initially. Hence its zero level-set defines the material interface at $t = 0$, while its sign determines the region occupied by each fluid.

Since material interfaces propagate with the fluid velocity, the zero level-set of ϕ identifies the material interface for all $t > 0$ if $\phi(x, t)$ satisfies the advection initial value problem

$$\begin{cases} \dfrac{\partial \phi}{\partial t} + u \dfrac{\partial \phi}{\partial x} = 0, \\ \\ \phi(x, 0) = x - \alpha(0). \end{cases} \tag{2.6}$$

The evolution equation in (2.6) can be recast in conservation form by combining it with the conservation of mass,

$$\frac{\partial \rho \phi}{\partial t} + \frac{\partial u \rho \phi}{\partial x} = 0. \tag{2.7}$$

Equations (2.1), (2.3), and (2.6) (or (2.7)) form an "inmiscible" model in which the fluid "mixture" consists of either fluid Ω_0 or Ω_1, and where thermodynamic properties change discontinuously across the material interface.

3. Numerical schemes. To integrate the Euler equations and the level-set equation for multicomponent flows, two strategies are available. The first one is to discretize the conservative system composed by the Euler equations (2.1) together with the level-set equation in conservation form (2.7). This discretization can be performed by applying the classical methodology of shock-capturing schemes for systems of conservation laws. It has been used in other papers (see, e.g., [24]), but the addition of the level-set equation enlarges the Jacobian matrix, which increases the complexity of characteristic decompositions. This feature may become especially cumbersome in multidimensional calculations.

The second strategy—simpler, and followed here—is considered in [8], where the Euler equations (2.1) are discretized independently from the level-set equation in nonconservative form (2.6). Following [11], the Euler system is identified as the [illegible]; hence it can be discretized with any standard conservative scheme, while the level-set equation is independently discretized following its advection form (2.6). The numerical flux function involved in the Euler equations will therefore not depend directly on ϕ, as in a single-component flow. The Lax–Wendroff theorem can still be applied [11]; hence the obtained numerical solution converges to a weak solution of the full conservative system.

The discrete set-up to solve the system (2.1)–(2.6) is as follows: we consider a lattice of points in the x-plane $x_i = i\,\Delta x$, where Δx is a uniform spacing parameter. Numerical values $\boldsymbol{U}_i^n$ and ϕ_i^n are, respectively, considered as approximations to $\boldsymbol{U}$ and ϕ at x_i and time t_n [30]. To decouple the spatial and time integrations, we follow a [illegible] approach [19] for both the Euler system and the level-set equation. The semidiscrete approximation of the nonlinear hyperbolic system (2.1) is written

$$\frac{d}{d\,t}\,\boldsymbol{U}_i = \boldsymbol{L}_\Omega(\boldsymbol{U}, i), \tag{3.1}$$

whereas the semidiscrete approximation of the advection equation (2.6) is

$$\frac{d}{d\,t}\,\phi_i = G(\phi, i). \tag{3.2}$$

The discrete operator $\boldsymbol{L}_\Omega$ in (3.1) is specified in subsection 3.1. We say that a grid point is [illegible] if $\boldsymbol{L}_\Omega$ uses numerical values belonging only to one fluid component. Otherwise, a grid point is called [illegible], and the expression of $\boldsymbol{L}_\Omega$ is modified by the interface method detailed in section 4. The discrete operator G in (3.2) is specified in subsection 3.2.

We emphasize the fact that the numerical schemes proposed in the next two subsections are by no means fundamental for the interface method. The reader's favorite single-component solvers can be adapted easily to the forthcoming discussion.

3.1. Discretization of the Euler equations. When x_i is a regular point, all the values involved in the flux computations belong to one of the fluid components, say, Ω_l. Then, the spatial discrete operator $\boldsymbol{L}_\Omega = \boldsymbol{L}_{\Omega_l}$ in (3.1) is written in the customary conservation form

$$\boldsymbol{L}_{\Omega_l}(\boldsymbol{U}, i) = -\frac{1}{\Delta x}\left(\boldsymbol{F}_{\Omega_l}\left(\boldsymbol{U}_{i-s+1}, \ldots, \boldsymbol{U}_{i+s}\right) - \boldsymbol{F}_{\Omega_l}\left(\boldsymbol{U}_{i-s}, \ldots, \boldsymbol{U}_{i+s-1}\right)\right). \tag{3.3}$$

The numerical flux function $\boldsymbol{F}_\Omega$ in (3.3) is the trademark of the scheme; it is defined by a reconstruction procedure, and by a solver. In numerical experiments, we use ENO or WENO reconstructions [30, 13]. The width of the stencil s in (3.3) is related to the theoretical order of accuracy of the spatial reconstruction ($s = 3$ for ENO-3 or WENO-5).

As a solver, we consider a flux-splitting construction [6, 10]. This choice has been considered in [23] within the mass fraction model for two-ideal gas flows. This flux-splitting requires two spectral decompositions of the Jacobian at each cell interface, which serve to perform upwind reconstructions of characteristic variables and fluxes. The additional cost is counterbalanced by the robust behavior of the scheme in pathology-prone situations. In addition, and as opposed to a Roe-type numerical flux function, no average state needs to be computed at a cell interface, which is particularly useful for real-gas simulations. We need only to know the spectral decomposition of the Jacobian matrix: for the EOS (2.3), this is given, e.g., in [32].

The time integration of (3.1) is performed by the standard third-order TVD Runge–Kutta [30], even when the fifth-order WENO-5 reconstruction is used. Δt follows from Δx and from the classical CFL condition of stability

$$\mathrm{CFL} = \max_{i=0,\ldots,N_x}\left(|u_i^n| + c_i^n\right)\frac{\Delta t}{\Delta x} \leq 1, \tag{3.4}$$

where $N_x + 1$ is the number of grid points.

3.2. Discretization of the level-set equation. We follow the same method of lines as for the Euler system. The spatial discretization is carried out using the Hamilton–Jacobi framework for the numerical approximation of the derivative terms [14, 26]. The upwind direction at x_i is determined by the sign of u_i. Full details are in Appendix A.1 of [8].

Theoretically, solving (2.6) is sufficient to track the material interface. However, some extra care must be taken for numerical purposes. Since u is nonuniform along the flow, the numerical representation of $\phi(x, t)$ may become distorted [28], leading to a poor estimation of the position of α. To keep ϕ approximately equal to the distance function near the material interface, we follow a classical procedure proposed in [33] and called [illegible] of the level-set. This procedure can be carried out in a

number of different ways; here we follow [8] and solve to steady state the Hamilton–Jacobi equation

$$\begin{cases} \dfrac{\partial \tilde{\phi}}{\partial t} + S(\tilde{\phi}) \left(\left| \dfrac{\partial \tilde{\phi}}{\partial x} \right| - 1 \right) = 0, \\ \\ \tilde{\phi}(x,0) = \phi(x,t_n), \end{cases} \tag{3.5}$$

where S is a smeared sign function given by

$$S(\tilde{\phi}) = \frac{\tilde{\phi}}{\sqrt{\tilde{\phi}^2 + \Delta x^2}}.$$

The reinitialization equation is solved in fictitious time after each fully complete time-step for the Euler equations, with a method of lines approach. For the spatial integration of (3.5), we use a modification of Godunov's method detailed in Appendix A.3 of [8]. For the time integration, we use the same TVD Runge–Kutta scheme as for the Euler equations. As a time-step, we take $\Delta \tau = \Delta x$. After five integrations, we obtain a steady distance function, $\tilde{\phi}$, which is then exchanged with ϕ.

4. The ESIM for the Euler equations.

4.1. The numerical interface treatment. The ESIM is a numerical treatment to be applied at irregular points, i.e., at points for which the flux computations in the right-hand side of (3.1) involve more than one fluid component. This treatment is carried out at each time integration (such as a Runge–Kutta substep). Let us fix an instant in time (to simplify the notation, the time variable is omitted from now on). Then, the basic strategy of the ESIM can be schematically described as follows.

Consider the material interface at $x = \alpha$ separating the two media Ω_0 and Ω_1 (Figure 2.1). We define smooth extensions $\boldsymbol{U}^*(x)$ of the exact solution $\boldsymbol{U}(x)$; here, we consider two-term Taylor-like expansions past the interface, written

$$\begin{aligned} &\text{for } x > \alpha, \qquad \boldsymbol{U}^*(x) = \boldsymbol{U}(\alpha^-) + (x-\alpha) \frac{\partial}{\partial x} \boldsymbol{U}(\alpha^-), \\ &\text{for } x \le \alpha, \qquad \boldsymbol{U}^*(x) = \boldsymbol{U}(\alpha^+) + (x-\alpha) \frac{\partial}{\partial x} \boldsymbol{U}(\alpha^+). \end{aligned} \tag{4.1}$$

Then, the interface treatment is divided in two parts.

First, numerical estimations $\boldsymbol{U}_i^*$ of $\boldsymbol{U}^*(x_i)$ in (4.1), called [illegible] (or [illegible], in the GFM framework), are sought at grid points surrounding α (see Figure 4.1). To do so, one must estimate $\boldsymbol{U}(\alpha^\pm)$ and $\frac{\partial}{\partial x}\boldsymbol{U}(\alpha^\pm)$. These last estimations, based on jump conditions presented in subsection 4.2, are detailed in subsection 4.3.

Second, at each irregular point x_i belonging to a particular fluid component, say, Ω_l, the spatial operator $L_\Omega(\boldsymbol{U}, i)$ is constructed by considering only thermodynamically similar state values, that is, numerical values for grid locations in Ω_l and modified values at grid points on the other fluid component. This subject is detailed in subsection 4.4.

4.2. Jump conditions at material interfaces. Across α, the primitive variables satisfy the classical zeroth-order jump conditions

$$[u] = 0, \qquad [p] = 0, \tag{4.2}$$

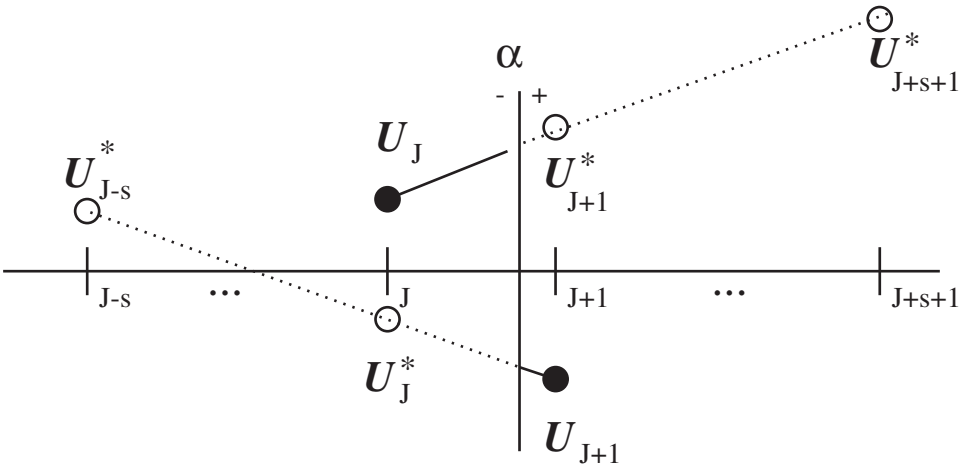


FIG. 4.1. *Numerical values* $_i$ *and modified values* $^*_i$.

where, for any function $f(x,t)$,

$$[f] = \lim_{x\to\alpha^+} f(x,t) - \lim_{x\to\alpha^-} f(x,t).$$

Since u and p do not jump across the interface, then their material—or Lagrangian—derivatives do not either. This fact immediately leads to the less classical first-order jump conditions

$$\left[\frac{1}{\rho}\frac{\partial p}{\partial x}\right] = 0, \qquad \left[\rho c^2 \frac{\partial u}{\partial x}\right] = 0. \tag{4.3}$$

One can also find second-order (and higher) jump conditions. However, and unlike the acoustics case [27, 20], these jump conditions (not presented here) involve nonlinear combinations of p, u, and their successive spatial derivatives, making the computations much more intricate.

For polytropic gases, $\rho c^2 = \gamma p$; since $[p] = 0$, the second equation in (4.3) can be simplified in $[\gamma \frac{\partial u}{\partial x}] = 0$. Note also that nothing is said about $[\rho]$ and $[\frac{\partial \rho}{\partial x}]$.

4.3. Numerical estimation of one-sided quantities. In a level-set model, the location of the material interface is given by a sign change in the marker variable ϕ. Suppose that

$$\phi_J \times \phi_{J+1} < 0. \tag{4.4}$$

As a consequence, the interface lies somewhere between x_J and x_{J+1}. The subcell location of α is specified by the parameter

$$\tilde{\theta} = \frac{\alpha - x_J}{\Delta x} \in [0,1[, \tag{4.5}$$

which is readily estimated to second order by a simple linear interpolation involving the smooth level-set function ϕ. Indeed, defining

$$\theta = \frac{|\phi_J|}{|\phi_J| + |\phi_{J+1}|}, \tag{4.6}$$

we have $\theta = \tilde{\theta} + O(\Delta x^2)$. Since the jump conditions (4.2) and (4.3) concern the primitive variables $\boldsymbol{W} = (\rho, u, p)^T$, it is easier to first look for estimations of $\boldsymbol{W}(\alpha^{\pm})$ and $\frac{\partial}{\partial x}\boldsymbol{W}(\alpha^{\pm})$, and then go back to the conserved variables $\boldsymbol{U}(\alpha^{\pm})$ and $\frac{\partial}{\partial x}\boldsymbol{U}(\alpha^{\pm})$. In the following discussion and for any function f, the numerical estimation of $f(\alpha^{\pm})$ or $\frac{\partial}{\partial x} f(\alpha^{\pm})$ is denoted by $f^{\pm}$ or $f_x^{\pm}$.

. The density is not subject to any constraint at a material interface; therefore, numerical estimations $\rho^{\pm}$ and $\rho_x^{\pm}$ are performed via one-sided interpolations. On the left of α, elementary interpolations lead to

$$\begin{aligned} \rho^- &= -\theta\,\rho_{J-1} + (1+\theta)\,\rho_J, \\ \rho_x^- &= \frac{1}{\Delta x}\,(\rho_J - \rho_{J-1})\,, \end{aligned} \tag{4.7}$$

while on the right of α, we get

$$\begin{aligned} \rho^+ &= (2-\theta)\,\rho_{J+1} - (1-\theta)\,\rho_{J+2}, \\ \rho_x^+ &= \frac{1}{\Delta x}\,(\rho_{J+2} - \rho_{J+1})\,. \end{aligned} \tag{4.8}$$

If $\rho(x)$ is a piecewise C^1 function, we obviously have

$$\rho(\alpha^{\pm}) = \rho^{\pm} + O(\Delta x^2), \qquad \frac{\partial \rho}{\partial x}(\alpha^{\pm}) = \rho_x^{\pm} + O(\Delta x).$$

1. The density estimations obtained above must remain within the physical range, i.e., $\rho^{\pm} > 0$. In some cases, negative values can be numerically obtained (e.g., in regions close to vacuum). Then, a different approximation procedure with constraints would be necessary (which is not considered here).

. The pressure and its spatial derivative satisfy (4.2) and (4.3):

$$\begin{aligned} p(\alpha^+) &= p(\alpha^-), \\ \frac{1}{\rho(\alpha^+)}\,\frac{\partial p}{\partial x}(\alpha^+) &= \frac{1}{\rho(\alpha^-)}\,\frac{\partial p}{\partial x}(\alpha^-). \end{aligned} \tag{4.9}$$

From the definition of $p^{\pm}$ and $p_x^{\pm}$, we have

$$\begin{aligned} p_J &= p^- + (x_J - \alpha)\,p_x^-, \\ p_{J+1} &= p^+ + (x_{J+1} - \alpha)\,p_x^+. \end{aligned} \tag{4.10}$$

We impose that numerical estimations satisfy exactly the same jump conditions as exact values. Combining (4.10) with (4.9) leads to the 2×2 system

$$\begin{pmatrix} p_J \\ p_{J+1} \end{pmatrix} = \begin{pmatrix} 1 & -\theta\,\Delta x \\ 1 & (1-\theta)\,\dfrac{\rho^+}{\rho^-}\,\Delta x \end{pmatrix} \begin{pmatrix} p^- \\ p_x^- \end{pmatrix}. \tag{4.11}$$

The above matrix is always invertible, provided that $\rho^{\pm} > 0$. Inverting the above system gives the numerical values at α^-:

$$
\begin{aligned}
p^- &= \frac{1}{(1-\theta)\,\dfrac{\rho^+}{\rho^-} + \theta} \left((1-\theta)\,\frac{\rho^+}{\rho^-}\,p_J + \theta\,p_{J+1} \right), \\
p_x^- &= \frac{1}{\left((1-\theta)\,\dfrac{\rho^+}{\rho^-} + \theta \right) \Delta x}\,(p_{J+1} - p_J)\,.
\end{aligned}
\tag{4.12}
$$

To get estimations at α^+, one uses (4.12) and jump conditions (4.9). Note that p^+ (or p^-) is a convex combination with positive coefficients of p_J and p_{J+1}, and hence $p^{\pm} > 0$. As in the previous case, standard results allow us to conclude that if both $p(x)$ and $\rho(x)$ are C^1 away from the material interface, then

$$
p(\alpha^{\pm}) = p^{\pm} + O(\Delta x^2), \qquad \frac{\partial p}{\partial x}(\alpha^{\pm}) = p_x^{\pm} + O(\Delta x).
$$

. Zeroth-order and first-order jump conditions for the velocity are

$$
\begin{aligned}
u(\alpha^+) &= u(\alpha^-), \\
\rho(\alpha^+)\,c^2(\alpha^+)\,\frac{\partial u}{\partial x}(\alpha^+) &= \rho(\alpha^-)\,c^2(\alpha^-)\,\frac{\partial u}{\partial x}(\alpha^-).
\end{aligned}
\tag{4.13}
$$

Applying the same procedure as for the pressure, we obtain a system which is invertible as long as $(1-\theta)\gamma_0\,(p^- + p_{\infty\,0})\,/\gamma_1\,(p^+ + p_{\infty\,1}) + \theta > 0$. This is always the case since $p^{\pm} > 0$. For stiffened gas EOS (2.3), defining

$$
\xi = \frac{\gamma_0\,(p^- + p_{\infty\,0})}{\gamma_1\,(p^- + p_{\infty\,1})}
\tag{4.14}
$$

(for the ideal polytropic case $\xi = \gamma_0/\gamma_1$) with p^- given by (4.12), we obtain

$$
\begin{aligned}
u^- &= \frac{1}{(1-\theta)\,\xi + \theta}\,\left((1-\theta)\,\xi\,u_J + \theta\,u_{J+1} \right), \\
u_x^- &= \frac{1}{\left((1-\theta)\,\xi + \theta \right)\,\Delta x}\,(u_{J+1} - u_J)\,.
\end{aligned}
\tag{4.15}
$$

To get estimations at α^+, one uses (4.15) and the jump conditions (4.13). As before, under the appropriate smoothness assumptions, one has

$$
u(\alpha^{\pm}) = u^{\pm} + O(\Delta x^2), \qquad \frac{\partial u}{\partial x}(\alpha^{\pm}) = u_x^{\pm} + O(\Delta x).
\tag{4.16}
$$

. Once sided estimations $\boldsymbol{W}^{\pm}$ and $\boldsymbol{W}_x^{\pm}$ are computed, we deduce $\boldsymbol{U}^{\pm}$ and $\boldsymbol{U}_x^{\pm}$ from (2.3). These values are second-order approximations to the exact values under appropriate smoothness assumptions; hence

$$
\hat{\boldsymbol{U}}(x) = \begin{cases} \boldsymbol{U}^+ + (x-\alpha)\,\boldsymbol{U}_x^+ & \text{for } x \le \alpha, \\ \boldsymbol{U}^- + (x-\alpha)\,\boldsymbol{U}_x^- & \text{for } x > \alpha \end{cases}
\tag{4.17}
$$

satisfies $\boldsymbol{U}^*(x) = \hat{\boldsymbol{U}}(x) + O(\Delta x^2)$ near the interface.

4.4. Spatial integration at irregular points. From (3.3) and (4.4), one deduces that the irregular points are

$$x_{J-s+1}, \ldots, x_{J+s}. \tag{4.18}$$

Modified values must also be computed at x_{J-s} and x_{J+s+1} because the material interface can move between t_n and t_{n+1}, crossing one grid point (but no more, because of the CFL condition); as a consequence, these two regular points can become irregular points during one Runge–Kutta substep. The modified values are computed by substituting the appropriate grid point in the expresion (4.17). Using (4.6) as a second-order approximation to (4.5), we obtain

$$\begin{aligned} &i = J+1, \ldots, J+s+1, && \boldsymbol{U}_i^* = \boldsymbol{U}^- + (i - J - \theta)\, \Delta x\, \boldsymbol{U}_x^-, \\ &i = J-s, \ldots, J, && \boldsymbol{U}_i^* = \boldsymbol{U}^+ + (i - J - \theta)\, \Delta x\, \boldsymbol{U}_x^+. \end{aligned} \tag{4.19}$$

At each irregular point x_i, $\boldsymbol{L}_\Omega$ is now applied to numerical values on the same side as x_i, and to modified values on the other side; hence

$$\begin{aligned} &i = J-s+1, \ldots, J, \\ &\boldsymbol{L}_\Omega(\boldsymbol{U}, i) = -\frac{1}{\Delta x} \left(\boldsymbol{F}_{\Omega\, 0} \left(\boldsymbol{U}_{i-s+1}, \ldots, \boldsymbol{U}_J, \boldsymbol{U}_{J+1}^*, \ldots, \boldsymbol{U}_{i+s}^* \right) \right. \\ &\qquad \left. - \boldsymbol{F}_{\Omega\, 0} \left(\boldsymbol{U}_{i-s}, \ldots, \boldsymbol{U}_J, \boldsymbol{U}_{J+1}^*, \ldots, \boldsymbol{U}_{i+s-1}^* \right) \right), \\ &i = J, \ldots, J+s, \\ &\boldsymbol{L}_\Omega(\boldsymbol{U}, i) = -\frac{1}{\Delta x} \left(\boldsymbol{F}_{\Omega\, 1} \left(\boldsymbol{U}_{i-s+1}^*, \ldots, \boldsymbol{U}_J^*, \boldsymbol{U}_{J+1}, \ldots, \boldsymbol{U}_{i+s} \right) \right. \\ &\qquad \left. - \boldsymbol{F}_{\Omega\, 1} \left(\boldsymbol{U}_{i-s}^*, \ldots, \boldsymbol{U}_J^*, \boldsymbol{U}_{J+1}, \ldots, \boldsymbol{U}_{i+s-1} \right) \right). \end{aligned} \tag{4.20}$$

4.5. Summary of the implementation. Suppose that Runge–Kutta integrations (denoted by RK) have been performed up to the $(m-1)$th substep. Then, the time-marching for both the Euler equations and the level-set equation can be summed up in eight steps.

- , 1 (location of the interface). Compute J (4.4) and θ (4.5).
- , 2 (construction of modified values). Compute (4.19).
- , 3 (construction of temporary values). Build two sets $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$:

$$\begin{aligned} \boldsymbol{A}_i^{(m-1)} &= \begin{cases} \boldsymbol{U}_i^{(m-1)},\ i = 0, \ldots, J, \\ \boldsymbol{U}_i^*,\ i = J+1, \ldots, J+s+1, \end{cases} \\ \boldsymbol{B}_i^{(m-1)} &= \begin{cases} \boldsymbol{U}_i^*,\ i = J-s, \ldots, J, \\ \boldsymbol{U}_i^{(m-1)},\ i = J+1, \ldots, N_x. \end{cases} \end{aligned} \tag{4.21}$$

- , 4 (update $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$). Compute one Runge–Kutta substep of (3.1):

$$\begin{aligned} &i = s, \ldots, J+1, && \boldsymbol{A}_i^{(m)} = \mathrm{RK}\left(\boldsymbol{A}_i^{(k)}, \boldsymbol{L}_{\Omega_0} \right), \quad k \le m, \\ &i = J, \ldots, N_x - s, && \boldsymbol{B}_i^{(m)} = \mathrm{RK}\left(\boldsymbol{B}_i^{(k)}, \boldsymbol{L}_{\Omega_1} \right), \quad k \le m. \end{aligned} \tag{4.22}$$

Step 5 (update ϕ). Compute one Runge–Kutta substep of (2.6) via $u^{(m-1)}$.

Step 6 (location of the interface). Compute J (4.4) from $\phi^{(m)}$.

Step 7 (update $\boldsymbol{U}$). From (4.22) and J, select

$$\boldsymbol{U}_i^{(m)} = \begin{cases} \boldsymbol{A}_i^{(m)} \text{ for } i = 0, \dots, J, \\ \\ \boldsymbol{B}_i^{(m)} \text{ for } i = J+1, \dots, N_x. \end{cases} \tag{4.23}$$

Step 8 (reinitialization of ϕ). If $m = 3$ (i.e., $t = t_{n+1}$), integrate (3.5).

4.6. Some remarks.

Simplicity. The algorithm presented in section 4.5 is simple. The computation of modified values $\boldsymbol{U}_i^*$ does not depend on the discrete spatial operator. No analytical results—such as the solution of a Riemann problem—are required. As deduced from (4.22), one does not need to write a new solver: the adaptation of known single-component solvers to the multicomponent case is direct. The only difficulty is switching precisely modified values in the appropriate solver, depending on the sign of the level-set function, which is in fact an easy task.

Computational cost. In comparison with single-component simulations, our approach leads to a +25% additional cost, both from a memory and a computational time point of view. This cost is almost completely due to the level-set function ϕ. Such a cost is inherent to level-set formulations in order to know the composition of the fluid at each grid point. Eulerian methods that do not use level-sets require, in counterpart, modification of the Euler eigenstructure, leading to a similar additional cost.

Thermodynamic consistency. As for the GFM [2], the ESIM works because values used for time-stepping in (4.22) are thermodynamically similar: $\boldsymbol{A}_i^*$ (respectively $\boldsymbol{B}_i^*$) in (4.21) satisfies the same EOS. The algorithm amounts to considering separately two single-component flows, where no oscillations exist.

Limit case. In the limit case $\gamma_0 = \gamma_1$, $p_{\infty\,0} = p_{\infty\,1}$, the flow is single-component, and material interfaces amount to classical contact discontinuities. The ESIM behaves equally well, sharpening contact discontinuities.

Nonconservativity. The ESIM is formally nonconservative locally, since

$$\begin{aligned} &\boldsymbol{F}_{\Omega_0}\left(\boldsymbol{U}_{J-s+1}, \dots, \boldsymbol{U}_J, \boldsymbol{U}_{J+1}^*, \dots, \boldsymbol{U}_{J+s}^*\right) \\ &\quad \neq \boldsymbol{F}_{\Omega_1}\left(\boldsymbol{U}_{J-s+1}^*, \dots, \boldsymbol{U}_J^*, \boldsymbol{U}_{J+1}, \dots, \boldsymbol{U}_{J+s}\right), \end{aligned} \tag{4.24}$$

like in [15] or [8]. However, the lack of conservation is only introduced at one cell boundary on the entire domain. In practice, this feature does not seem to spoil convergence to the correct solution.

Note that a fully conservative version of the GFM has been developed [25]. Such an approach could probably be adapted to the ESIM, but we do not look further in the present paper; unlike in [25] where inert shocks and detonation waves are addressed, we focus only on the material interfaces (where conservation errors are not crucial).

5. Numerical experiments.

5.1. Configurations. Four numerical experiments are proposed. Tests 1 and 2 illustrate the interaction of shock waves with one and two material interfaces. Test

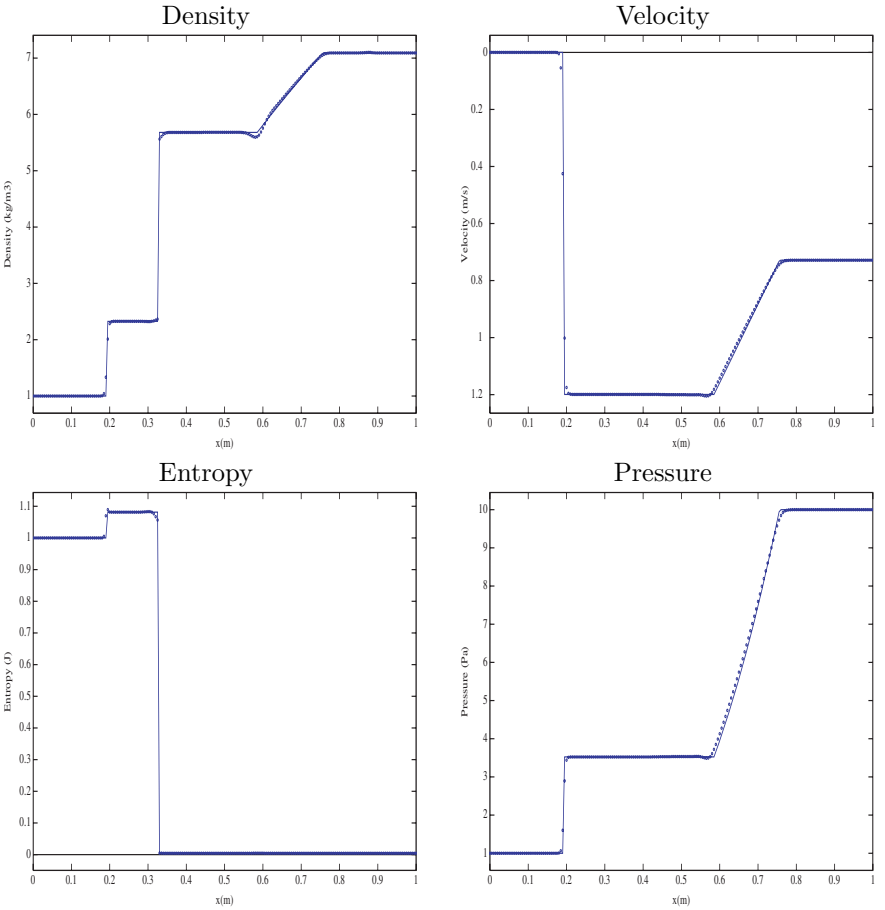


FIG. 5.1. *Test* 1-a: *Mach* 1.95 *shock wave interacting with a material interface. WENO*-5 *coupled with the ESIM. (Exact values: Solid line. Numerical values: Points.)*

3 is a pure advection problem, with smooth structures. Test 4 concerns nonlinear acoustics. Analytical values and numerical values are, respectively, shown in solid lines and dotted lines, and we take CFL = 0.66.

All tests have initially isolated material interfaces. Riemann problems are not our concern, since jump conditions (4.2)—which are a building block of the ESIM—are then generally not satisfied. Note that the GFM has the same limitation. In practice, the GFM often works for Riemann problems, but some failures (see, e.g., data of Test 4 in [2]) can be explained by the fact that basic assumptions are not satisfied.

5.2. Test 1: Shock-interface interaction. First, we consider a shock-interface interaction problem, previously studied in [31]. On a 1 m long domain, the initial configuration consists of a stationary material interface at $\alpha_0 = 0.5$ m, separating two fluids with different EOSs: a polytropic gas on the left, and a stiffened gas on the right. A left-going Mach 1.95 shock wave is initially set at $\alpha_1 = 0.7$ m. Physical

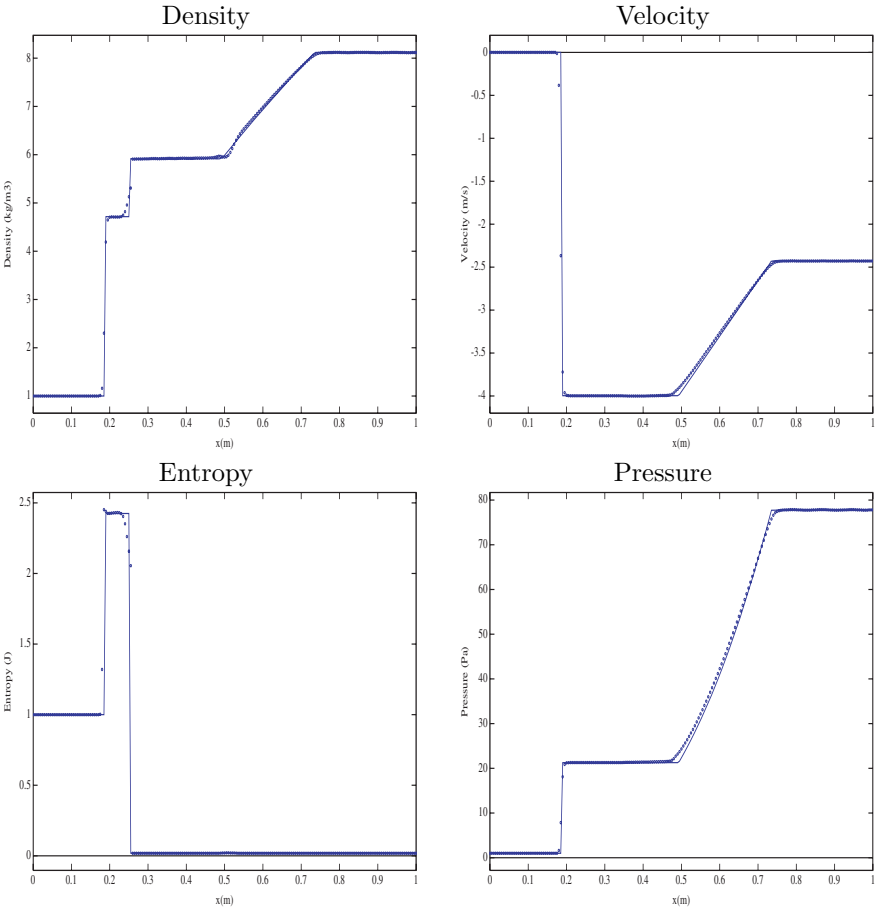


FIG. 5.2. *Test* 1-b: *Mach* 5 *shock wave interacting with a material interface. WENO*-5 *coupled with the ESIM. (Exact values: Solid line. Numerical values: Points.)*

parameters are

(5.1)
$$\begin{cases} \rho_0 = 1.000 \text{ kg/m}^3,\ p_0 = 1 \text{ Pa}\ ,\ u_0 = 0 \text{ m/s}\ ,\gamma_0 = 1.4,\ p_{\infty\,0} = 0 \text{ Pa}, \\ \rho_1 = 5.000 \text{ kg/m}^3,\ p_1 = 1 \text{ Pa}\ ,\ u_1 = 0 \text{ m/s}\ ,\gamma_1 = 4,\ p_{\infty\,1} = 1 \text{ Pa}, \\ \rho_2 = 7.093 \text{ kg/m}^3,\ p_2 = 10 \text{ Pa}\ ,u_2 = -0.7288 \text{ m/s}\ ,\gamma_2 = 4.0,\ p_{\infty\,2} = 1 \text{ Pa}. \end{cases}$$

At $t = 4.69\,10^{-2}$ s, the shock wave collides with the material interface, leading to a left-going shock, a left-going material interface, and a right-going rarefaction fan (see Figure 4 in [31]).

Figure 5.1 shows exact values and numerical values of ρ, u, p, and entropy S at $t = 0.202$ s (after 200 time-steps). The computations are performed on $N_x = 200$ grid points with WENO-5 coupled with the ESIM. The agreement between analytical and numerical values is good. The main interest of this example is to show that the ESIM is robust and behaves well, even when a shock wave is in the vicinity of the material

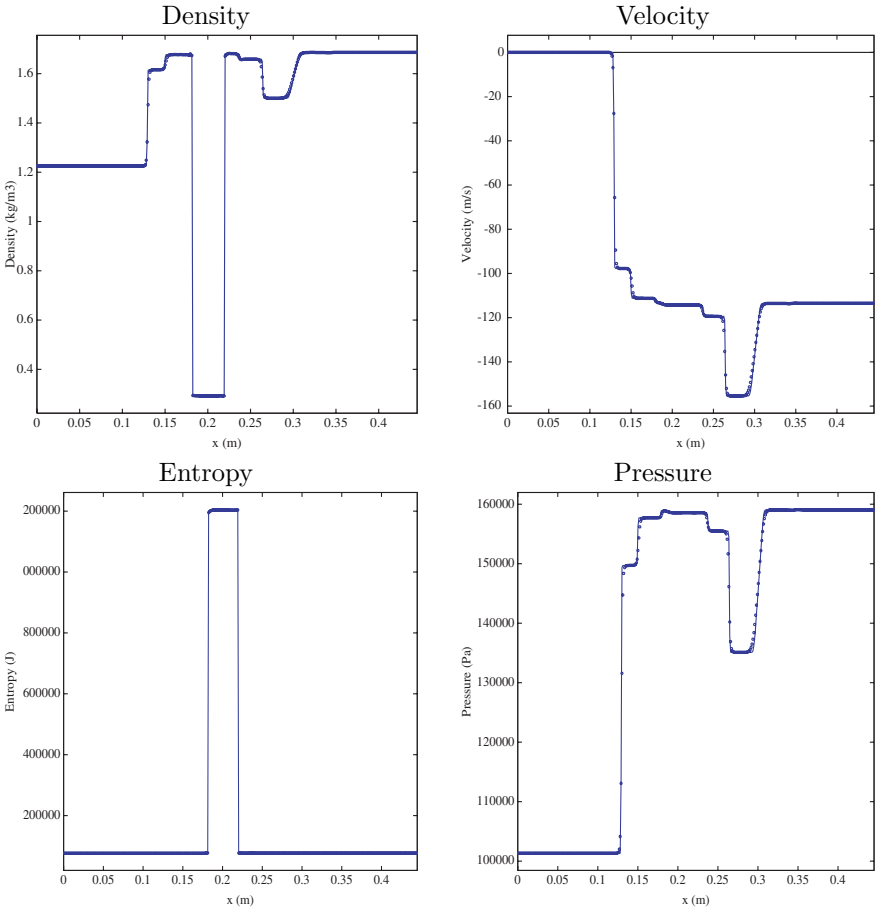


FIG. 5.3. *Test* 2: *Shock-bubble interaction. WENO-*5 *coupled with the ESIM. (Fine grid solution: Solid line. Numerical values: Points.)*

interface. No spurious oscillations induced by Taylor expansions are seen in u and p around the material interface (near $x = 0.32$ m).

Second, we consider the case of a stronger shock wave. The physical parameters around the material interface are the same as in (5.1); only the parameters in the post-shock fluid are modified:

$$\rho_2 = 8.116 \text{ kg/m}^3, \quad p_2 = 77.80 \text{ Pa}, \quad u_2 = -2.428 \text{ m/s}, \quad \gamma_2 = 1.4, \quad p_{\infty 2} = 1 \text{ Pa},$$

which amounts to a Mach 5 left-going shock wave. The wave phenomena (after the collision between the shock wave and the material interface) are the same as in the previous example. The computations are performed with WENO-5 coupled with the ESIM.

Figure 5.2 shows exact values and numerical values of ρ, u, p, and entropy S at $t = 0.112$ s (after 300 time-steps). The agreement between numerical and analytical values is good, even for this strong shock test. No oscillations are visible in u and p around the material interface (near $x = 0.26$ m).

5.3. Test 2: Shock-bubble interaction. We consider the one-dimensional version of the classical shock-bubble interaction problem, numerically addressed, e.g., in [23] with a conservative algorithm. On a 0.445 m long domain, a helium domain, initially delimitated by $\alpha_0 = 0.2$ m and $\alpha_1 = 0.25$ m, is at rest in air. A left-going Mach 1.22 shock wave is initially set at $\alpha_2 = 0.275$ m. Physical parameters are

$$\begin{cases} \rho_0 = 1225 \text{ kg/m}^3,\ p_0 = 1.01325\,10^5 \text{ Pa},\ u_0 = 0.0 \text{ m/s}, \gamma_0 = 1.4,\ p_{\infty\,0} = 0 \text{ Pa}, \\ \rho_1 = 0.2228 \text{ kg/m}^3,\ p_1 = 1.01325\,10^5 \text{ Pa},\ u_1 = 0.0 \text{ m/s}, \gamma_1 = 1.648,\ p_{\infty\,1} = 0 \text{ Pa}, \\ \rho_2 = 1225 \text{ kg/m}^3,\ p_2 = 1.01325\,10^5 \text{ Pa},\ u_2 = 0.0 \text{ m/s}, \gamma_2 = 1.4,\ p_{\infty\,2} = 0 \text{ Pa}, \\ \rho_3 = 1686 \text{ kg/m}^3,\ p_3 = 1.59059\,10^5 \text{ Pa},\ u_3 = -3.59 \text{ m/s}, \gamma_3 = 1.4,\ p_{\infty\,3} = 0 \text{ Pa}. \end{cases}$$

See [23] for a description of wave phenomena. Since no analytic solution is available, we compute the solution with WENO-5 coupled with ESIM on a fine grid with 3200 mesh points; we refer to this approximation as the "exact solution" for comparison purposes.

Figure 5.3 shows "exact values" and numerical values of ρ, u, p, and S at $t = 2.86\,10^{-3}$ s (after 400 time-steps). The computations are performed on $N_x = 400$ grid points with WENO-5 coupled to the ESIM. The agreement between "exact values" and numerical values is very good. The resolution is better than in [23]: compare the density inside the helium in Figure 5.3 with the density in Figure 2 of [23].

To conclude Tests 1 and 2, let us note that similar results could be obtained with the GFM treatment (not shown here). Indeed, when only flat profiles are involved, the GFM and the ESIM have similar behavior. When a shock wave collides with a material interface, both treatments produce "sloping" behavior in density and entropy very close to the "overheating" phenomenon found in shock-reflection problems [6]. Increasing the Mach number seems to accentuate this "overheating," producing in addition slightly perturbed post-shock values: small acoustic perturbations can be observed downstream in Figure 5.2. More testing (not shown) has been performed with Mach numbers up to 9, which confirms these observations. No crash due to negative pressure values has been observed on these tests.

The goal of the next two tests is to show the advantage of using the ESIM (compared to the GFM) for rich-structured flows.

5.4. Test 3: Pure advection. We consider a 1 m long domain with two material interfaces initially at $\alpha_0 = 0.160$ m and $\alpha_1 = 0.526$ m. The pressure and the velocity are initially constant: $u(x,0) = 300$ m/s, $p(x,0) = 10^5$ Pa. The density is initially

$$\rho(x,0) = \begin{cases} 1 + 0.3 \sin(50\,(x - \alpha_0)) \text{ kg/m}^3 \text{ if } \alpha_0 \le x < \alpha_1, \\ 1 \text{ kg/m}^3 \text{ else.} \end{cases}$$

Physical parameters are

$$(\gamma, p_\infty) = \begin{cases} \gamma_0 = 1.40,\ p_{\infty\,0} = 10^4 \text{ Pa} & \text{if } x \le \alpha_0, \\ \gamma_1 = 1.67,\ p_{\infty\,1} = 10^5 \text{ Pa} & \text{if } \alpha_0 < x \le \alpha_1, \\ \gamma_2 = 1.40,\ p_{\infty\,2} = 10^4 \text{ Pa} & \text{if } x > \alpha_1. \end{cases}$$

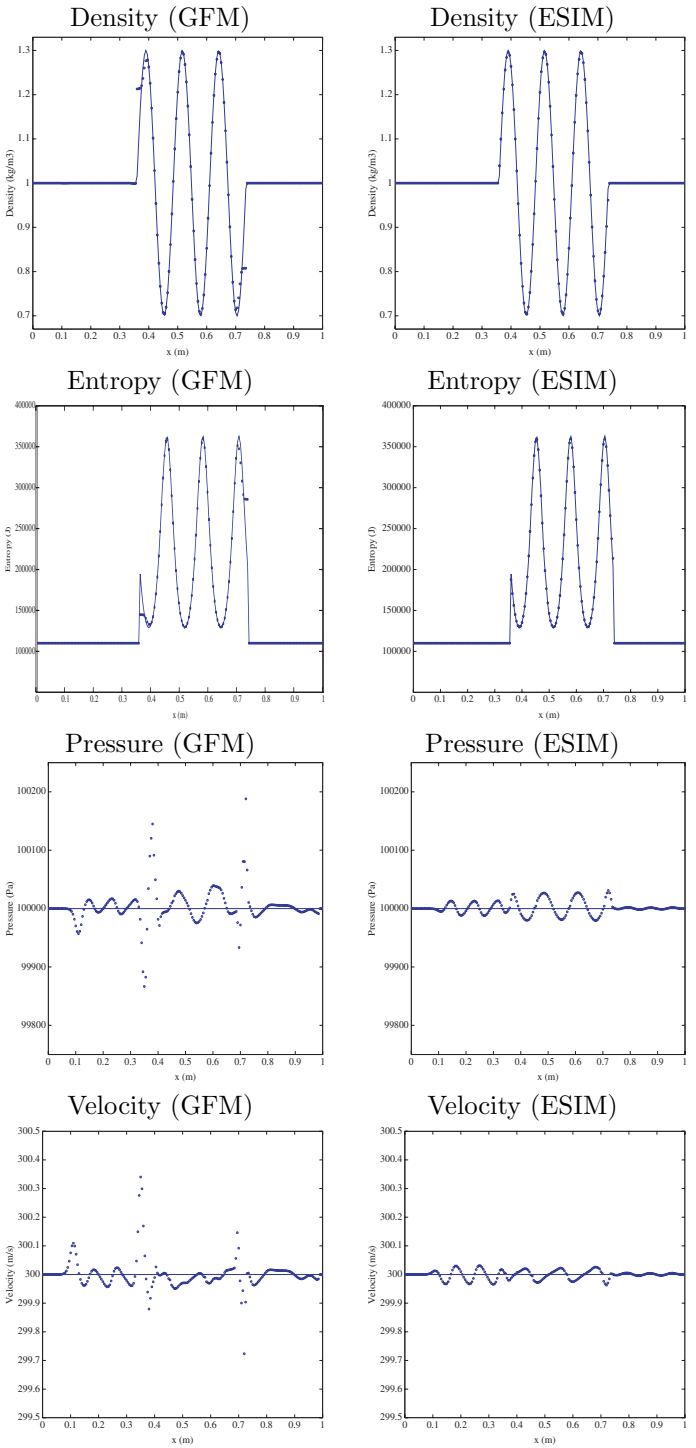


FIG. 5.4. *Test* 3-a: 200 *grid points. ENO*-3 *coupled with the GFM (left column) and with the ESIM (right column). Exact values: Solid line. Numerical values: Points (note that the scales for* p *and* u *are magnified).*

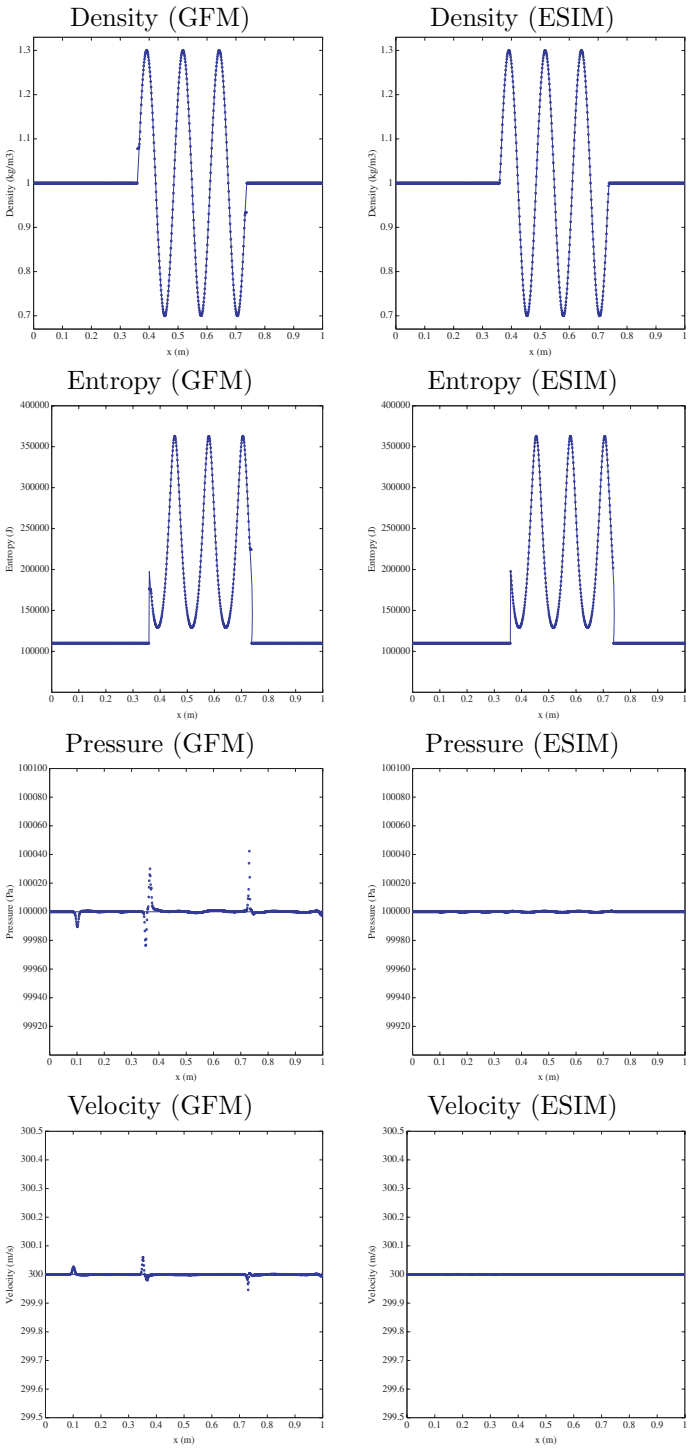


FIG. 5.5. *Test* 3-b: 800 *grid points. ENO-*3 *coupled with the GFM (left column) and with the ESIM (right column). Exact values: Solid line. Numerical values: Points (note that the scales for* p *and* u *are magnified).*

TABLE 5.1
Conservation errors in Test 3.

Method	N_x	$\Delta(\rho)$	Order	$\Delta(\rho u)$	Order	$\Delta(\rho e)$	Order
ENO-3 + GFM	100	7.25e-1	-	2.21e+2	-	3.22e+4	-
	200	3.82e-1	**0.92**	1.14e+2	**0.95**	1.72e+4	**0.90**
	400	2.21e-1	**0.79**	6.65e+1	**0.77**	9.97e+3	**0.78**
	800	1.27e-1	**0.80**	3.83e+1	**0.79**	5.74e+3	**0.79**
	1600	7.10e-2	**0.84**	2.13e+1	**0.84**	3.19e+3	**0.84**
	3200	3.61e-2	**0.97**	1.08e+1	**0.98**	1.62e+3	**0.97**
ENO-3 + ESIM	100	1.98e-1	-	5.96e+1	-	8.94e+3	-
	200	4.69e-2	**2.08**	1.40e+1	**2.09**	2.11e+3	**2.08**
	400	1.07e-2	**2.13**	3.23e+0	**2.11**	4.85e+2	**2.12**
	800	2.75e-3	**1.96**	8.27e-1	**1.96**	1.24e+2	**1.96**
	1600	6.82e-4	**2.01**	2.37e-1	**1.80**	3.41e+1	**1.86**
	3200	1.54e-4	**2.14**	6.72e-2	**1.81**	9.36e+0	**1.86**

This configuration amounts to an advection equation for ρ; material interfaces are advected at the velocity u; p and u remain theoretically constant. Because of the discontinuous physical parameters, the entropy S is discontinuous at α_0 and at α_1. Numerical experiments are performed with ENO-3, coupled with the GFM or with the ESIM.

Figures 5.4 and 5.5 show exact values and numerical values of ρ, S, p, and u at $t = 6.62\,10^{-4}$ s, respectively, for $N_x = 200$ grid points (hence 25 grid points by wavelength) and $N_x = 800$ grid points (hence 100 grid points by wavelength). At material interfaces, the zeroth-order extrapolations of S used by the GFM result in jumps of ρ that are advected with the flow; these glitches are also transferred to u, and p, near $x = 0.35$ m and $x = 0.75$ m, and act as sources of acoustic noise. These glitches still exist with 800 grid points (Figure 5.5) with the GFM. On the other hand, no entropy or density glitches are observed when the ESIM is used instead.

We must mention that ENO reconstructions of a sinusoidal profile might produce oscillations on the level of the truncation error. These small spurious oscillations are seen in p and u, which should have flat profiles, and can be observed even without material interfaces. When these oscillations interact with the GFM-produced glitches at material interfaces, they are amplified and lead also to spurious acoustic waves. These oscillations are entirely reconstruction-dependent, and, since they are of the order of the truncation error (see also our convergence measures), they are essentially not visible on 800 grid points when the ESIM is used (Figure 5.5).

Measures of convergence are provided in Table 5.1. Coupled with the ESIM, the ENO-3 scheme maintains third-order convergence. Coupled with the GFM, the ENO-3 scheme loses accuracy and shows a 1.6 order of convergence. Table 5.2 shows measures of conservation errors induced by the GFM and by the ESIM, for various values of N_x (see subsection 4.6). To do so, we compute

$$\Delta \boldsymbol{U}(T, N_x) = \max_{n=0,\ldots,N_t} \left| \sum_{i=i_0}^{i_1} \left(\boldsymbol{U}_i^n - \boldsymbol{U}_i^0 \right) + n \frac{\Delta t}{\Delta x} \left(\boldsymbol{f}\left(\boldsymbol{U}_{i_1}^n \right) - \boldsymbol{f}\left(\boldsymbol{U}_{i_0}^n \right) \right) \right| \tag{5.2}$$

at $T = 1.05\,10^{-3}$ s, where $\boldsymbol{f}$ is the flux function (2.2), and $N_t = \text{Trunc}\,(T/\Delta t)$. The

TABLE 5.2
Measures of convergence in Test 3.

Method	N_x	L_1 error	L_1 order	Method	N_x	L_1 error	L_1 order
	100	6.44e-3	-		100	3.76e-3	-
ENO-3	200	1.62e-3	**1.99**	ENO-3	200	4.74e-4	**2.98**
+	400	6.86e-4	**1.24**	+	400	6.42e-5	**2.88**
GFM	800	2.40e-4	**1.51**	ESIM	800	8.62e-6	**2.89**
	1600	6.44e-5	**1.89**		1600	9.50e-7	**3.18**
	3200	2.05e-5	**1.65**		3200	1.21e-7	**2.98**

domain of measure is bounded by $i_0 = 10$ and $i_1 = N_x - 10$. Getting reliable and meaningful measures requires some care. Indeed, errors of conservativity vary a lot with the position of α inside the meshing, and hence with t. The "max" in (5.2) ensures almost steady values of $\Delta \boldsymbol{U}(T, N_x)$ and reliable measures.

A fully conservative scheme would satisfy $\Delta \boldsymbol{U}(T, N_x) = 0$ for all values of T and N_x (see, e.g., (12.33) in [19]). Here, errors are nonnull, but they decrease with Δx. Conservation errors induced by the ESIM are much smaller than those induced by the GFM. Table 5.2 indicates first-order conservation errors for the GFM and second-order conservation errors for the ESIM.

Measures of convergence and conservativity have been done also with the WENO-5 scheme coupled with the GFM or with the ESIM. The results are essentially the same as in Tables 5.1 and 5.2. To get a fifth-order accurate scheme (as in [27]) and a higher order of conservation would require developing a higher-order ESIM, which is not investigated in the present study.

5.5. Test 4: Nonlinear acoustics. As a last example, we consider a rich-structured wave interacting with a stationary material interface. This example is a generalization of Test 1 studied in [27]: for small amplitude initial data, the simple linear equations of acoustics used in [27] are valid, and the wave remains smooth. Higher amplitudes of initial data lead to a nonlinear problem, where shocks can develop.

To illustrate both cases, we use exactly the same physical parameters as in [27]. On a 300 m long domain, a material interface separates piecewise constant values, initially motionless

$$\begin{cases} \rho_0 = 1000 \text{ kg/m}^3,\ p_0 = 10^5 \text{ Pa},\ u_0 = 0.0 \text{ m/s}, \gamma_0 = 3,\ p_{\infty\,0} = 7.50\,10^8 \text{ Pa}, \\ \\ \rho_1 = 1200 \text{ kg/m}^3,\ p_1 = 10^5 \text{ Pa},\ u_1 = 0.0 \text{ m/s}, \gamma_1 = 4,\ p_{\infty\,1} = 2.35\,10^9 \text{ Pa}. \end{cases}$$

From (2.4), one deduces sound speeds: $c_0 = 1500$ m/s, and $c_1 = 2800$ m/s. These physical parameters correspond, respectively, to water and Plexiglass under atmospheric pressure. A perturbation $\Delta \boldsymbol{W}_0(x)$ is added to initial data on medium Ω_0, which is an exact solution of acoustics and leads to a right-going wave

$$\Delta \boldsymbol{W}_0(x) = \varepsilon\, f\left(t_0 - \frac{x}{c_0}\right)^T \left(-\frac{p_0}{c_0^2}, -\frac{p_0}{\rho_0\, c_0}, -p_0\right). \tag{5.3}$$

The function f is a C^5 spatially bounded sinusoid

$$f(\xi) = \begin{cases} \displaystyle\sum_{k=1}^{q} a_k \, \sin(\beta_k \, \omega_c \, \xi) & \text{if } 0 < \xi < \dfrac{1}{f_c}, \\ \\ 0 \text{ else}, \end{cases} \tag{5.4}$$

with $\beta_k = 2^{k-1}$, $\omega_c = 2\pi \, f_c$; the coefficients a_k are $a_1 = 1$, $a_2 = -21/32$, $a_3 = 63/768$, and $a_4 = -1/512$. The central frequency is $f_c = 50$ Hz. Elementary calculations show that

$$\frac{\Delta \, \rho_0}{\rho_0} \sim \frac{\Delta \, u_0}{c_0} \sim \frac{\Delta \, p_0}{p_0} \sim \varepsilon. \tag{5.5}$$

For $\varepsilon = 0$, the solution remains a stationary material interface. For small values of ε (typically $\varepsilon = 10^{-3}$), the acoustics limit is valid and one obtains a pure right-going wave. For higher values of ε (typically $\varepsilon = 0.1$), the perturbation (5.3) does not satisfy exactly the nonlinear Euler equations: the wave separates into a weak left-going wave and a right-going wave. In both cases, the right-going wave is reflected and transmitted by the material interface. The analytical solution is not detailed in the linear case [20]. No analytical solution is available in the nonlinear case: one computes an "exact solution" on a fine grid of 3200 grid points. Numerical experiments are performed with $N_x = 400$ grid points, which is 40 points in the wavelength $\lambda_c = c_0/f_c$, and $t_0 = 5.1 \, 10^{-2}$ s. Initial values of the p at t_0 are shown in Figure 5.6 for $\varepsilon = 10^{-3}$.

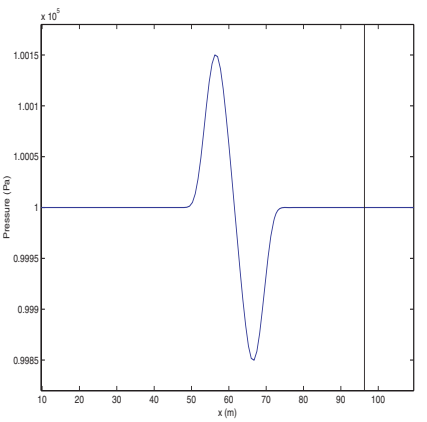


FIG. 5.6. *Initial values of the pressure p at $t_0 = 5.1 \, 10^{-2}$ s in Test 4.*

Figure 5.7 shows results at $t_1 = 8.61 \, 10^{-2}$ s, after 200 time-steps. Left and right columns concern, respectively, $\varepsilon = 10^{-3}$ and $\varepsilon = 0.1$. Figures 5.7(a) and (b) show numerical values and exact values of p computed by WENO-5 coupled to the ESIM. Logically, (a) is similar to Figure 3(d) of [27]. In the nonlinear case (b), the reflected and transmitted waves have developed a shock. In both cases, the agreement between exact values and numerical values is excellent.

To see clearly differences between the ESIM and GFM treatments, we display errors for both methods, with the ESIM in (c) and (d), and with the GFM in (e) and (f). In the nonlinear cases (d) and (f), errors are displayed from $x = 85$ m to

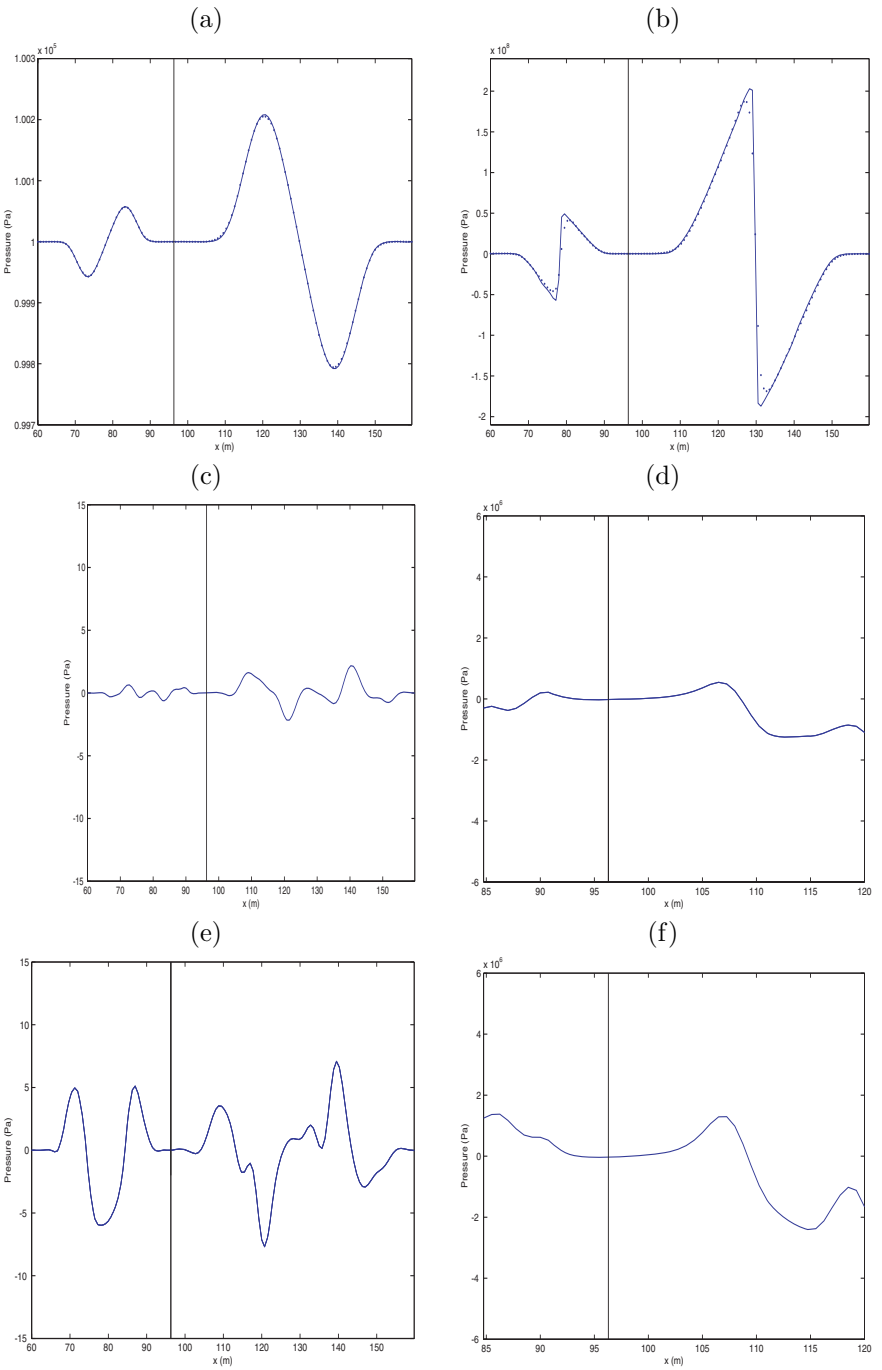


FIG. 5.7. *Test* 4. $\varepsilon = 10^{-3}$ *(left column),* $\varepsilon = 1$ *(right column) at* $t_1 = 8.61\,10^{-2}$ s. *Snapshots of fine grid solution (solid line) and numerical values (dotted line) of* p *with the ESIM:* (a), (b). *Errors with the ESIM:* (c), (d). *Errors with the GFM:* (e), (f).

$x = 120$ m to avoid the shock area. In both cases, the graphs confirm that the ESIM is more accurate than the GFM.

Unlike in [27], the material interface is allowed to move with the flow. For $\varepsilon = 10^{-3}$, the measured movement is lower than 10^{-7} m, so the approximation of stationary material interfaces is justified. For $\varepsilon = 0.1$, the material interface moves from 96.3 m to 96.114 m, and then it comes back to the initial position 96.3 m; it is logical, since the velocity is symetric with respect to zero.

6. Conclusion. We have proposed an extension of the explicit simplified interface method (ESIM), previously developed in acoustics [21, 22, 27], to treat material interfaces in one-dimensional multicomponent Euler flows.

The method enforces the numerical solution to satisfy zeroth-order and first-order jump conditions at the material interface and can be coupled with the user's favorite high-order shock-capturing scheme for single-component flows. It behaves as robustly as the GFM [8] in numerical simulations involving flat states, while it displays a superior performance in flows with rich structures.

Our numerical simulations show that the interface treatment we propose guarantees a numerical solution with no unphysical numerical artifacts due to the material interface. The behavior observed for the numerical errors is similar to that observed for the same underlying method when applied to a flow without material interfaces.

This paper focused on one-dimensional flows; applying the ESIM to two-dimensional cases is a challenging project and will be the subject of future works. Some key tools for the two-dimensional implementation have been validated already in the simple case of two-dimensional linear acoustics with stationary interfaces [22]. Other ingredients (such as multidimensional level-set-based extrapolations) may be found in [5, 12].

Acknowledgments. The authors would like to thank the anonymous referees for their comments and for pointing out several useful references.

REFERENCES

[1] R. Abgrall, *How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach*, J. Comput. Phys., 125 (1996), pp. 150–160.

[2] R. Abgrall and S. Karni, *Computations of compressible multifluids*, J. Comput. Phys., 169 (2001), pp. 594–623.

[3] G. Allaire, S. Clerc, and S. Kokh, *A five-equation model for the simulation of interfaces between compressible flows*, J. Comput. Phys., 181 (2002), pp. 577–616.

[4] M. Arienti, P. Hung, E. Morano, and J. E. Shepherd, *A level-set approach to Eulerian-Lagrangian coupling*, J. Comput. Phys., 185 (2003), pp. 213–251.

[5] T. D. Aslam, *A partial differential equation approach to multidimensional extrapolation*, J. Comput. Phys., 193 (2003), pp. 349–355.

[6] R. Donat and A. Marquina, *Capturing shock reflections: An improved flux formula*, J. Comput. Phys., 125 (1996), pp. 42–58.

[7] R. P. Fedkiw, *Coupling an Eulerian fluid calculation to a Lagrangian solid calculation with the ghost fluid method*, J. Comput. Phys., 175 (2002), pp. 200–224.

[8] R. P. Fedkiw, T. Aslam, B. Merriman, and S. Osher, *A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method)*, J. Comput. Phys., 152 (1999), pp. 457–492.

[9] R. P. Fedkiw, T. Aslam, and S. Xu, *The ghost fluid method for deflagration and detonation discontinuities*, J. Comput. Phys., 154 (1999), pp. 393–427.

[10] R. P. Fedkiw, B. Merriman, R. Donat, and S. Osher, *The Penultimate Scheme for Systems of Conservation Laws: Finite-Difference ENO with Marquina's Flux Splitting*, UCLA CAM Report 96-18, UCLA, Los Angeles, CA, 1996.

[11] R. P. Fedkiw, B. Merriman, and S. Osher, *Simplified discretization of systems of hyperbolic conservation laws containing advection equations*, J. Comput. Phys., 157 (2000), pp. 302–326.

[12] T. Y. Hou, Z. Li, S. Osher, and H. Zhao, *A hybrid method for moving interface problems with application to Hele-Shaw flow*, J. Comput. Phys., 134 (1997), pp. 236–252.
[13] G.-S. Jiang and C. W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.
[14] G.-S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.
[15] S. Karni, *Multicomponent flow calculations by a consistent primitive algorithm*, J. Comput. Phys., 112 (1994), pp. 31–43.
[16] S. Karni, *Hybrid multifluid algorithms*, SIAM J. Sci. Comput., 17 (1996), pp. 1019–1039.
[17] B. Larrouturou, *How to preserve the mass fractions positivity when computing compressible multi-component flows*, J. Comput. Phys., 95 (1991), pp. 59–84.
[18] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.
[19] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser, Basel, 1990.
[20] B. Lombard, *Modélisation numérique de la propagation des ondes acoustiques et élastiques en présence d'interfaces*, Ph.D. thesis, University of Aix-Marseille II, Marseille, France, 2002.
[21] B. Lombard and J. Piraux, *How to incorporate the spring-mass conditions in finite-difference schemes*, SIAM J. Sci. Comput., 24 (2003), pp. 1379–1407.
[22] B. Lombard and J. Piraux, *Numerical treatment of two-dimensional interfaces for acoustic and elastic waves*, J. Comput. Phys., 195 (2004), pp. 90–116.
[23] A. Marquina and P. Mulet, *A flux-split algorithm applied to conservative models for multicomponent compressible flows*, J. Comput. Phys., 185 (2003), pp. 120–139.
[24] W. Mulder, S. Osher, and J. A. Sethian, *Computing interface motion in compressible gas dynamics*, J. Comput. Phys., 100 (1992), pp. 209–228.
[25] D. Nguyen, F. Gibou, and R. P. Fedkiw, *A fully conservative ghost fluid method and stiff detonation waves*, in 12th International Detonation Symposium, San Diego, CA, 2002.
[26] S. Osher and C.-W. Shu, *High-order nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[27] J. Piraux and B. Lombard, *A new interface method for hyperbolic problems with discontinuous coefficients: One-dimensional acoustic example*, J. Comput. Phys., 168 (2001), pp. 227–248.
[28] G. Russo and P. Smereka, *A remark on computing distance functions*, J. Comput. Phys., 163 (2000), pp. 51–67.
[29] R. Saurel and R. Abgrall, *A simple method for compressible multifluid flows*, SIAM J. Sci. Comput., 21 (1999), pp. 1115–1145.
[30] C.-W. Shu and S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes.* II, J. Comput. Phys., 83 (1989), pp. 32–78.
[31] K. M. Shyue, *An efficient shock-capturing algorithm for compressible multicomponent problems*, J. Comput. Phys., 142 (1998), pp. 208–242.
[32] Y. Stiriba, A. Marquina, and R. Donat, *Equilibrium real gas computations using Marquina's scheme*, Internat. J. Numer. Methods Fluids, 141 (2003), pp. 275–301.
[33] M. Sussman, P. Smereka, and S. Osher, *A level-set approach for computing solutions to incompressible two-phase flows*, J. Comput. Phys., 114 (1994), pp. 146–159.
[34] C. Zhang and R. J. LeVeque, *The immersed interface method for acoustic wave equations with discontinuous coefficients*, Wave Motion, 25 (1997), pp. 237–263.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 231–252
© 2005 Society for Industrial and Applied Mathematics

NUMERICAL CONTINUATION OF BIFURCATIONS OF LIMIT CYCLES IN MATLAB*

WILLY GOVAERTS†, YURI A. KUZNETSOV‡, AND ANNICK DHOOGE†

Abstract. CL_MATCONT and MATCONT are MATLAB continuation packages for the interactive numerical study of a range of parameterized nonlinear dynamical systems, in particular ODEs. MATCONT is an interactive graphical package and CL_MATCONT is a commandline version. Both packages allow us to compute curves of equilibria, limit points, Hopf points, limit cycles, flip, fold, and torus bifurcation points of limit cycles. We discuss computational details of the continuation of limit cycles and flip, fold, and torus bifurcations of limit cycles in MATCONT and CL_MATCONT using orthogonal collocation. Instead of the more commonly used fully extended systems we use minimally extended systems. We further describe the use of the MATLAB sparse matrix routines and the initialization and adaptation of the bordering vectors that are essential in minimally extended systems. Finally, we compare the use of the minimally and the fully extended systems in the MATLAB environment.

Key words. limit cycle, fold, flip, torus, MATLAB

AMS subject classification. 65P30

DOI. 10.1137/030600746

1. Introduction.

Numerical continuation is a well-understood subject; see, e.g., [1], [8], [9], [4], [17], [20], [22]. The idea is as follows. Consider a smooth function $F : \mathbb{R}^{n+1} \to \mathbb{R}^n$. We want to compute a solution curve of the equation

$$F(x) = 0. \tag{1.1}$$

Numerical continuation is a technique to compute a sequence of points which approximate the desired solution branch.

In particular, we consider a dynamical system of the form

$$\frac{dx}{dt} = f(x, \alpha) \tag{1.2}$$

with $x \in \mathbb{R}^n$, $f(x, \alpha) \in \mathbb{R}^n$, and α a vector of parameters where equilibria, limit points, limit cycles, etc. can be computed.

The existing software packages such as AUTO [10] and CONTENT [21] require the user to rewrite his/her models in a specific form and use special internal formats to store results; this complicates the export of the results, graphical representation, etc.

The aim of MATCONT and CL_MATCONT is to provide a continuation toolbox which is compatible with the standard MATLAB ODE representation of differential equations. General descriptions of CL_MATCONT and MATCONT are in [7] and [6], respectively. The current version of the package is freely available for download at http://www.matcont.ugent.be. CL_MATCONT requires MATLAB 5.3, whereas

*Received by the editors August 26, 2003; accepted for publication (in revised form) October 14, 2004; published electronically August 22, 2005.
http://www.siam.org/journals/sisc/27-1/60074.html

†Department of Applied Mathematics and Computer Science, Ghent University, Krijgslaan 281-S9,B-9000 Gent, Belgium (Willy.Govaerts@UGent.be, Annick.Dhooge@UGent.be).
‡Mathematisch Instituut, Universiteit Utrecht, Budapestlaan 6, 3854 CD Utrecht, The Netherlands (Yu.A.Kuznetsov@math.uu.nl).

MATCONT requires MATLAB 6* or higher. In the present paper we concentrate on the implementation in CL_MATCONT and MATCONT of the continuation of the flip (period doubling (PD)), fold (limit point of cycles (LPC)), and torus (Neimark–Sacker (NS)) bifurcations of limit cycles, using minimally extended systems and orthogonal collocation to discretize appearing boundary value problems (BVPs). The only existing software to perform these continuations is AUTO [10] which uses fully extended systems and their orthogonal collocation. Several algorithms for the continuation of limit cycles and fold bifurcations of limit cycles are studied in [18]. They are also based on MATLAB and are minimally extended systems but are otherwise very different from ours since they use a combination of multiple shooting and automatic differentiation, using coarse meshes and Taylor series expansions. Accuracy and robustness are the main objectives of [18] and these are compared to the AUTO results; since no code is available we cannot compare them directly to our results.

We note that the above-mentioned software, like ours, considers only parameterized two-point BVPs. Bifurcation software for parameterized partial differential equations might require different numerical methods (though the underlying mathematics of the bifurcation problems would be identical).

The theory of the bordering method for the numerical continuation of bifurcations of limit cycles was developed in [12] and is summarized in a slightly generalized form in section 2. We note that the convergence properties of the discretized solutions of the PD, LPC, and NS equations can be proved by the method of reformulation of BVPs as described in [3] and applied in [9] to include the period of the orbit, a phase condition, and parameters in the BVP formulation. In particular, the convergence of the systems for PD, LPC, and NS is of order h^m, where h is the maximum length of the mesh intervals and m is the number of collocation points in each mesh interval. Also, there is superconvergence (of order h^{2m}) in the endpoints of the mesh intervals and for the scalar equations added to the LC equations.

In this paper we deal with the numerical and computational aspects of the implementation in MATCONT and CL_MATCONT. Note that this makes MATCONT the first fully interactive software that supports the continuation of the limit cycle bifurcations using orthogonal collocation. In section 3 we recall the basic methods for the continuation of limit cycles. In sections 4, 5, and 6 we discuss the continuation of the PD, LPC, and NS bifurcations, respectively. Section 7 discusses further details on the initialization and adaptation of the continuation which are largely common to the three cases. In section 8 we consider the computation of multipliers along the three curve types. In section 9 we provide an example with the LP and NS continuations in MATCONT. In section 10 we make a comparison between the use of minimally and fully extended systems in the case of PD cycles.

2. Mathematical background on limit cycles and bifurcations of limit cycles. In this section we summarize the main results of [12]; in the flip and torus cases we actually present an easy generalization.

2.1. Limit cycles. A [illegible] is a closed orbit corresponding to a periodic solution of (1.2) with period T, i.e., $x(0) = x(T)$. By definition, in a neighborhood of a [illegible] there are no other cycles.

Since T is not known in advance, it is customary to use an equivalent system defined on the fixed interval $[0, 1]$ by rescaling time [10], [21]:

$$\begin{cases} \dfrac{dx}{dt} - Tf(x, \alpha) = 0, \\ x(0) = x(1). \end{cases} \tag{2.1}$$

A phase shifted function $\phi(t) = x(t+s)$ is also a solution of (2.1) for any value of s. To obtain a unique solution an extra constraint is needed. The following integral constraint is often used [10], [21]:

$$\int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt = 0, \tag{2.2}$$

where $\dot{x}_{old}(t)$ is the tangent vector of a previously calculated limit cycle and is therefore known; $\langle x, v\rangle$ is just a different notation for $x^T v$. This condition tries to select the solution which has the smallest phase difference with respect to the previous solution x_{old}.

Thus, the complete BVP defining a limit cycle consists of (2.1) and (2.2).

2.2. Flip bifurcation of limit cycles. A flip bifurcation of limit cycles (PD) generically indicates a PD of the periodic solution; i.e., there are nearby periodic solutions of approximately double period. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2), where G is the flip test function specified below. The complete BVP defining a PD point using the minimally extended system is

$$\begin{cases} \dfrac{dx}{dt} - Tf(x,\alpha) & = 0, \\ x(0) - x(1) & = 0, \\ \int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt & = 0, \\ G[x, T, \alpha] & = 0, \end{cases} \tag{2.3}$$

where G is defined by requiring

$$N_1 \begin{pmatrix} v \\ G \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.4}$$

Here v is a function, G is a scalar, and

$$N_1 = \begin{bmatrix} D - Tf_x(x(\cdot), \alpha) & w_{01} \\ \delta_0 + \delta_1 & w_{02} \\ \mathrm{Int}_{v_{01}} & 0 \end{bmatrix}, \tag{2.5}$$

where the bordering functions v_{01}, w_{01}, vector w_{02} are chosen so that N_1 is nonsingular [12]; δ_0, δ_1 are the Dirac operators defined by $\delta_i(f) = f(i)$ for $f \in \mathcal{C}^1([0,1], \mathbb{R}^n)$; $\mathrm{Int}_{v_{01}}$ is the operator defined by $\mathrm{Int}_{v_{01}} f = \int_0^1 v_{01}^T(t) f(t) dt$ for $f \in \mathcal{C}^1([0,1], \mathbb{R}^n)$. We note that in [12] the entry corresponding to w_{02} is zero; the generalization to (2.5) is easy.

2.3. Fold bifurcation of limit cycles. A fold bifurcation of limit cycles (LPC) generically corresponds to a turning point of a curve of limit cycles. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2), where G is the fold test function specified below. The complete BVP defining an LPC point using the minimally extended system is

$$\begin{cases} \dfrac{dx}{dt} - Tf(x,\alpha) & = 0, \\ x(0) - x(1) & = 0, \\ \int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt & = 0, \\ G[x, T, \alpha] & = 0, \end{cases} \tag{2.6}$$

where G is defined by requiring

$$N_2 \begin{pmatrix} v \\ S \\ G \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}. \tag{2.7}$$

Here v is a function, S and G are scalars, and

$$N_2 = \begin{bmatrix} D - Tf_x(x(\cdot),\alpha) & -f(x(\cdot),\alpha) & w_{01} \\ \delta_1 - \delta_0 & 0 & w_{02} \\ \mathrm{Int}_{f(x(\cdot),\alpha)} & 0 & w_{03} \\ \mathrm{Int}_{v_{01}} & v_{02} & 0 \end{bmatrix}, \tag{2.8}$$

where the bordering functions v_{01}, w_{01}, vector w_{02}, and scalars v_{02} and w_{03} are chosen so that N_2 is nonsingular [12].

2.4. Torus bifurcation of limit cycles. A torus bifurcation of limit cycles (NS) generically corresponds to a bifurcation to an invariant torus, on which the flow contains periodic or quasi-periodic motions. It can be characterized by adding an extra constraint $G = 0$ to (2.1), (2.2), where G is the torus test function which has four components. The complete BVP defining an NS point using the minimally extended system is

$$\begin{cases} \dfrac{dx}{dt} - Tf(x,\alpha) & = 0, \\ x(0) - x(1) & = 0, \\ \int_0^1 \langle x(t), \dot{x}_{old}(t)\rangle dt & = 0, \\ G[x,T,\alpha] & = 0, \end{cases} \tag{2.9}$$

where

$$G = \begin{pmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix}$$

is defined by requiring

$$N_3 \begin{pmatrix} v^1 & v^2 \\ G_{11} & G_{12} \\ G_{21} & G_{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{2.10}$$

Here v^1 and v^2 are functions, G_{11}, G_{12}, G_{21}, and G_{22} are scalars, and

$$N_3 = \begin{bmatrix} D - Tf_x(x(\cdot),\alpha) & w_{11} & w_{12} \\ \delta_0 - 2\kappa\delta_1 + \delta_2 & w_{21} & w_{22} \\ \mathrm{Int}_{v_{01}} & 0 & 0 \\ \mathrm{Int}_{v_{02}} & 0 & 0 \end{bmatrix}, \tag{2.11}$$

where the bordering functions $v_{01}, v_{02}, w_{11}, w_{12}$ and vectors $w_{21}$and w_{22} are chosen so that N_3 is nonsingular. We note that in [12] the entries corresponding to w_{21} and w_{22} are zero; the generalization to (2.11) is easy.

3. Numerical continuation of limit cycles. For the numerical continuation of a limit cycle with respect to a parameter we need to discretize the system consisting of (2.1) and (2.2); to use a Newton-like method the Jacobian matrix of the discretized system is also needed. This is a sparse matrix and we exploit the sparsity by using the MATLAB routines for sparse matrices. The same applies to the Jacobians of the equations that define bifurcations of limit cycles in section 4, 5, and 6, which have similar sparsity patterns.

3.1. Discrete representation of the solution function. The method used to discretize the BVP is the same as in COLSYS [2], AUTO [10], and CONTENT [21] and is called the [illegible] [5]. First the interval [0 1] is subdivided into N smaller [illegible]:

$$0 = \tau_0 < \tau_1 < \cdots < \tau_N = 1.$$

On each of these intervals the solution $x(\tau)$ is approximated by an order m vector valued polynomial $x^{(i)}(\tau)$. This is done by defining $m+1$ equidistant [illegible] on each test interval

$$\tau_{i,j} = \tau_i + \frac{j}{m}(\tau_{i+1} - \tau_i) \ (i = 0, 1, \ldots, N-1)(j = 0, 1, \ldots, m)$$

and defining the polynomials $x^{(i)}(\tau)$ as

$$x^{(i)}(\tau) = \sum_{j=0}^{m} x^{i,j} l_{i,j}(\tau).$$

Here $x^{i,j}$ is the approximation of $x(\tau)$ at $\tau = \tau_{i,j}$ (we note that $x^{i,m} = x^{i+1,0}$ for $i = 0, 1, \ldots, N-1$), and the $l_{i,j}(\tau)$'s are the Lagrange basis polynomials

$$l_{i,j}(\tau) = \prod_{k=0, k\neq j}^{m} \frac{\tau - \tau_{i,k}}{\tau_{i,j} - \tau_{i,k}}.$$

On each test interval $[\tau_i, \tau_{i+1}]$ we require that the polynomials $x^{(i)}(\tau)$ satisfy the BVP exactly in m collocation points $\zeta_{i,j}$ $(j = 1, \ldots, m)$ (which are different from the mesh points $\tau_{i,j}$). It can be proved that the best choice for the collocation points are the Gauss points [5]. These are the roots of the Legendre polynomial relative to the interval $[\tau_i, \tau_{i+1}]$. With this choice of collocation points the error in the approximation is extremely small,

$$||x(\tau_{i,j}) - x^{i,j}|| = \mathcal{O}(h^m),$$

where $h = \max\{|t_i| : i = 1, 2, \ldots, N\}$, $t_i = \tau_i - \tau_{i-1}$, and for the mesh points τ_i it is even better:

$$||x(\tau_i) - x^{i,0}|| = \mathcal{O}(h^{2m}).$$

3.2. Numerical evaluation of integrals. In (2.6) and several other places we need to compute integrals over $[0, 1]$ using the discretization discussed in section 3.1. For $N = 3$ test intervals and $m = 2$ collocation points, the following data are

associated with the discretized interval [0 1]:

$$\begin{array}{ccccccc}
\tau_0 & & \tau_1 & & \tau_2 & & \tau_3 \\
\circ & \circ & \bullet & \circ & \bullet & \circ & \circ \\
\tau_{0,0} & \tau_{0,1} & \tau_{0,2} & & \tau_{2,0} & \tau_{2,1} & \tau_{2,2} \\
 & & \tau_{1,0} & \tau_{1,1} & \tau_{1,2} & & \tau_{3,0} \\
t_1w_1 & t_1w_2 & t_1w_3+t_2w_1 & t_2w_2 & t_2w_3+t_3w_1 & t_3w_2 & t_3w_3 \\
\sigma_{0,0} & \sigma_{0,1} & \sigma_{1,0} & \sigma_{1,1} & \sigma_{2,0} & \sigma_{2,1} & \sigma_{3,0}
\end{array}$$

The total number of mesh points (*tps*) is $N \times m + 1$, the total number of variables (*ncoords*) is $tps \times \dim(x)$. Each mesh point $\tau_{i,j}$ in a test interval $[\tau_i, \tau_{i+1}]$ has a particular weight w_{j+1}, the Gauss–Lagrange quadrature coefficient. Some mesh points (the black bullets) belong to two test intervals. We set $t_i = \tau_i - \tau_{i-1}$ $(i = 1, \ldots, N)$. The integration weight $\sigma_{i,j}$ of $\tau_{i,j}$ is given by $w_{j+1}t_{i+1}$ for $0 \le i \le N-1$ and $0 < j < m$. For $i = 0, \ldots, N-2$ the integration weight of $\tau_{i,m} = \tau_{i+1,0}$ is given by $\sigma_{i,m} = w_{m+1}t_{i+1} + w_1 t_{i+2}$, and the integration weights of τ_0 and τ_N are given by $w_1 t_1$ and $w_{m+1}t_N$, respectively. The integral $\int_0^1 f(t)dt$ is, therefore, approximated by $\sum_{i=0}^{N-1}\sum_{j=0}^{m-1} f(\tau_{i,j})\sigma_{i,j} + f(1)\sigma_{N,0}$.

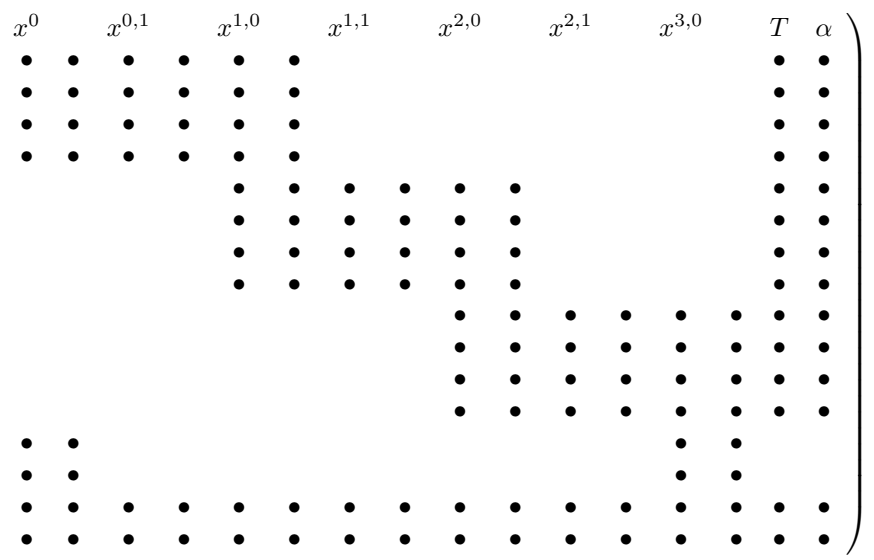
3.3. Discretization of the BVP. Using the discretization described in section 3.1, we obtain the discretized BVP

$$\begin{cases}
\left(\sum_{j=0}^{m} x^{i,j} l'_{i,j}(\zeta_{i,k})\right) - Tf(\sum_{j=0}^{m} x^{i,j} l_{i,j}(\zeta_{i,k}), \alpha) = 0 \\
x^{0,0} - x^{N-1,m} = 0 \\
\sum_{i=0}^{N-1}\sum_{j=0}^{m-1} \sigma_{i,j}\langle x^{i,j}, \dot{x}^{i,j}_{old}\rangle + \sigma_{N,0}\langle x^{N,0}, \dot{x}^{N,0}_{old}\rangle = 0
\end{cases}$$

The first equation in fact consists of Nm equations, one for each combination of $i = 0, 1, 2, \ldots, N-1$ and $k = 1, 2, \ldots, m$.

3.4. The Jacobian of the discretized equations. The Jacobian of the discretized system is sparse. In the Newton iterations during the continuation process a system consisting of this Jacobian matrix and an extra row (the tangent vector) is solved. For $N = 3$ test intervals, $m = 2$ collocation points, and $\dim(x) = 2$ this matrix has the following sparsity structure (•'s are a priori nonzeros):

(3.1)



The columns of (3.1) label the unknowns of the discretized problem. The first $\dim(x)$ rows correspond to the first collocation point, etc. In (2.1) and (2.2) there are three unknowns: x, the period T, and a parameter α. So the part of the Jacobian corresponding with the first equation of (2.1) has the following form:

$$[D - Tf_x(x,\alpha) \quad -f(x,\alpha) \quad -Tf_\alpha(x,\alpha)].$$

In (3.1) $D-Tf_x(x,\alpha)$ corresponds to $N=3$ blocks with dimension 4×6 $(=(\dim(x)*m)\times(\dim(x)\times(m+1)))$. The part in (3.1) that defines the boundary conditions for limit cycles has the form

$$[I_{\dim(x)} \quad 0_{\dim(x)\times(Nm-1)\dim(x)} \quad -I_{\dim(x)} \quad 0_{\dim(x)}].$$

These are in (3.1) the $\dim(x)=2$ rows following the 4×6 blocks. These rows contain two nonzero parts corresponding with $x^{0,0}$ and $x^{N,0}(\pm 2\times 2$ identity matrix). The last but one row in (3.1) is the derivative of the discretization of (2.2). The last row is added by the continuation code.

4. Continuation of PD cycles.

4.1. Discretization of the PD equations. The last equation in (2.3) expresses the condition that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) \\ \delta_0 + \delta_1 \end{bmatrix} \tag{4.1}$$

that appears in (2.5) is rank deficient [12]. In the implementation in MATCONT and CL_MATCONT we replace this by the condition that the discretized operator (matrix) of (4.1) has rank defect 1. To this end we solve

$$N_1^d \begin{pmatrix} v_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \tag{4.2}$$

where

$$N_1^d = \begin{bmatrix} [\ D \ - \ Tf_x(x(\cdot),\alpha)\ ]_d & & w_d^1 \\ I_{\dim(x)} \ 0_{\dim(x)\times(Nm-1)\dim(x)} \ I_{\dim(x)} & & w_d^2 \\ v_d^1 & & 0 \end{bmatrix} \tag{4.3}$$

and the bordering vectors v_d^1, w_d^1, and w_d^2 are chosen so that (4.3) has full rank. Here and elsewhere the subscript d denotes discretization using orthogonal collocation. The structure is similar to that of (3.1); however, the two last rows and columns of (3.1) are replaced by the single last row and column of (4.3).

4.2. The Jacobian of the discretized PD equations. To continue the discretized equations of (2.3) the Jacobian matrix of the system is needed, which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

Taking derivatives of (4.2) with respect to z (being a component of $x^{i,j}, T$, or α) we obtain

$$N_1^d \begin{pmatrix} v_{dz} \\ G_{dz} \end{pmatrix} + \begin{bmatrix} ([-Tf_x(x(\cdot),\alpha)]_d)_z & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} v_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

or, equivalently,

$$N_1^d \begin{pmatrix} v_{dz} \\ G_{dz} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot),\alpha)]_{dz} v_d \\ 0 \\ 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$(w_1^*, w_2^*, w_3^*) N_1^d = (0, 1), \tag{4.4}$$

where w_1 is a $\dim(x) * N * m$ vector, w_2 is a $\dim(x)$ vector, and w_3 is a scalar. Combining (4.2) and (4.4), we find

$$G_{dz} = -w_1^* [Tf_x(x(\cdot),\alpha)]_{dz} v_d. \tag{4.5}$$

So in each iteration step we solve two systems with the structure of (4.3) or its transpose.

5. Continuation of LPC cycles.

5.1. Discretization of the LPC equations. The last equation in (2.6) expresses that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) & -f(x(\cdot),\alpha) \\ \delta_1 - \delta_0 & 0 \\ \mathrm{Int}_{f(x(\cdot),\alpha)} & 0 \end{bmatrix} \tag{5.1}$$

that appears in (2.8) is rank deficient. In the numerical implementation in MATCONT and CL_MATCONT we replace this by the condition that the discretized operator of (5.1) is rank deficient. We solve

$$N_2^d \begin{pmatrix} v_d \\ S_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}, \tag{5.2}$$

where

$$N_2^d = \begin{bmatrix} [\ D\ -\ Tf_x(x(\cdot),\alpha)\ ]_d & [-f(x(\cdot),\alpha)]_d & w_d^1 \\ I_{\dim(x)}\ 0_{n\times(Nm-1)\dim(x)}\ -I_{\dim(x)} & 0 & w_d^2 \\ \mathrm{Int}_{f(x(\cdot),\alpha)_d} & 0 & w_d^3 \\ v_d^1 & v_d^2 & 0 \end{bmatrix}, \tag{5.3}$$

where the bordering vectors v_d^1, w_d^1, and w_d^2 and scalars v_d^2 and w_d^3 are chosen so that N_2^d is nonsingular. The structure is similar to that of (3.1); however, the last two rows and columns have a different meaning. The last but one row corresponds with $\mathrm{Int}_{[f(x(\cdot),\alpha)]_d}$ and the last but one column corresponds with $[-f(x(\cdot),\alpha)]_d$.

5.2. The Jacobian of the discretized LPC equations. To continue the discretized equations of (2.6) the Jacobian matrix of the system is needed, which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

The derivative with respect to z (being a component of $x^{i,j}, T$ or α) is

$$N_2^d \begin{pmatrix} v_{dz} \\ S_{dz} \\ G_{dz} \end{pmatrix} + \begin{bmatrix} ([-Tf_x(x(\cdot),\alpha)]_d)_z & ([-f(x(\cdot),\alpha)]_d)_z & 0 \\ 0 & 0 & 0 \\ (\mathrm{Int}_{[f(x(\cdot),\alpha)]_d})_z & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{pmatrix} v_d \\ S_d \\ G_d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Simplifying gives

$$N_2^d \begin{pmatrix} v_{dz} \\ S_{dz} \\ G_{dz} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot),\alpha)]_{dz} v_d + [f(x(\cdot),\alpha)]_{dz} S_d \\ 0 \\ -\mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d \\ 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$(w_1^*, w_2^*, w_3, w_4) N_2^d = (0, 0, 1), \tag{5.4}$$

where w_1 is a $\dim(x) * N * m$ vector, w_2 is a $\dim(x)$ vector, and w_3 and w_4 are scalars. Combining (5.2) and (5.4), we find

$$G_{dz} = w_1^*([Tf_x(x(\cdot),\alpha)]_{dz} v_d + [f(x(\cdot),\alpha)]_{dz} S_d) - w_3 \mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d. \tag{5.5}$$

So in each iteration step we solve two systems with the structure of (3.1) or its transpose.

5.3. Details of the computation of G . We restrict ourselves to the computation of the contribution of the term $-w_3 \mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d$ in (5.5), the computation of the other terms being similar. We first introduce the vector

$$\Sigma = (\sigma_{0,0}, \ldots, \sigma_{0,0}, \sigma_{0,1}, \ldots, \sigma_{0,1}, \ldots, \sigma_{1,0}, \ldots, \sigma_{1,0}, \ldots, \sigma_{N,0}, \ldots, \sigma_{N,0})^T,$$

where each weight $\sigma_{i,j}$ is repeated $\dim(x)$ times.

Let $(\Sigma. * v_d)$ be the element-by-element product of the vectors Σ and v_d. Then

$$\mathrm{Int}_{[f(x(\cdot),\alpha)]_d} v_d = (\Sigma. * v_d)^T F((x^{i,j}), \alpha),$$

where $F((x^{i,j}), \alpha)$ is the column vector consisting of the $f(x^{i,j}, \alpha)$ for $i = 0, \ldots, N - 1,\ j = 0, \ldots, m - 1$, and $f(x^{N,0}, \alpha)$. So

$$\mathrm{Int}_{[f(x(\cdot),\alpha)]_{dz}} v_d = (\Sigma. * v_d)^T F((x^{i,j}), \alpha)_z.$$

Since T does not appear in this expression, there is no contribution to G_{dT}. The contribution to $G_{dx^{i,j}}$ is

$$-w_3 (\Sigma. * v_d)^T_{(i+mj)\dim(x)+1,\ldots,(i+mj+1)\dim(x)} f_x(x^{i,j}, \alpha);$$

the contribution to $G_{d\alpha}$ is

$$-w_3 (\Sigma. * v_d)^T_{(i+mj)\dim(x)+1,\ldots,(i+mj+1)\dim(x)} f_\alpha(x^{i,j}, \alpha).$$

6. Continuation of NS cycles.

6.1. Discretization of the NS equations. The last equation in (2.9) expresses that the operator

$$\begin{bmatrix} D - Tf_x(x(\cdot),\alpha) \\ \delta_0 - 2\kappa\delta_1 + \delta_2 \end{bmatrix} \tag{6.1}$$

which appears in (2.11) has rank defect 2. In the numerical implementation in MATCONT and CL_MATCONT we replace this by the requirement that the discretized equation of (6.1) has rank defect 2. We solve

$$N_3^d \begin{pmatrix} v_d^1 & v_d^2 \\ G_d^{11} & G_d^{12} \\ G_d^{21} & G_d^{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \tag{6.2}$$

where

$$N_3^d = \begin{bmatrix} [D - Tf_x(x(\cdot),\alpha)]_d & w_d^{11} & w_d^{12} \\ (\delta_0 - 2\kappa\delta_1 + \delta_2)_d & w_d^{21} & w_d^{22} \\ v_d^{01} & 0 & 0 \\ v_d^{02} & 0 & 0 \end{bmatrix} \tag{6.3}$$

and

$$[(\delta_0 - 2\kappa\delta_1 + \delta_2)_d]$$
$$= [I_{\dim(x)} \;\; 0_{\dim(x)\times(Nm-1)\dim(x)} \;\; -2\kappa I_{\dim(x)} \;\; 0_{\dim(x)\times(Nm-1)\dim(x)} \;\; I_{\dim(x)}],$$

and the bordering vectors $v_d^{01}, v_d^{02}, w_d^{11}, w_d^{12}, w_d^{21}$, and w_d^{22} are chosen so that (6.3) has full rank. Here the subscript d denotes discretization using orthogonal collocation over the interval [0 2]. For $N = 2$ test intervals and $m = 2$ collocation points the following data are associated with the discretized interval [0 2]:

0				1				2
τ_0		τ_1		τ_2		τ_3		τ_4
$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$	$\bullet$	$\circ$	$\circ$
$\tau_{0,0}$	$\tau_{0,1}$	$\tau_{0,2}$		$\tau_{2,0}$	$\tau_{2,1}$	$\tau_{2,2}$		$\tau_{4,0}$
		$\tau_{1,0}$	$\tau_{1,1}$	$\tau_{1,2}$		$\tau_{3,0}$	$\tau_{3,1}$	$\tau_{3,2}$
t_1w_1	t_1w_2	$t_1w_3 + t_2w_1$	t_2w_2	$t_2w_3 + t_1w_1$	t_1w_2	$t_1w_3 + t_2w_1$	t_2w_2	t_2w_3
$\sigma_{0,0}$	$\sigma_{0,1}$	$\sigma_{1,0}$	$\sigma_{1,1}$	$\sigma_{2,0}$	$\sigma_{2,1}$	$\sigma_{3,0}$	$\sigma_{3,1}$	$\sigma_{4,0}$

The total number of mesh points (*tps*) is now $2N \times m + 1$ and the total number of points (*ncoords*) is *tps* $\times \dim(x)$.

The structure is quite similar to that of (3.1); the first part ($2\dim(x)Nm \times$ *ncoords*) is now over [0 2] in (6.3), so it is duplicated (the dimension is approximately doubled). The two last rows and columns are also different. They correspond with the borders.

6.2. The Jacobian of the discretized NS equations. To continue the discretized equations of (2.9) the Jacobian matrix of the system is needed, which means that the derivatives of G_d with respect to the unknowns of the system, i.e., with respect to $x^{i,j}, T, \alpha$, have to be calculated.

The derivative with respect to z (being a component of $x^{i,j}, T$, or α) is

$$N_3^d \begin{pmatrix} v_{dz}^1 & v_{dz}^2 \\ G_{dz}^{11} & G_{dz}^{12} \\ G_{dz}^{21} & G_{dz}^{22} \end{pmatrix} = \begin{pmatrix} [Tf_x(x(\cdot),\alpha)]_{dz} v_{dz}^1 & [Tf_x(x(\cdot),\alpha)]_{dz} v_{dz}^2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

Instead of solving this for every z we solve the transposed equations

$$\begin{pmatrix} w_1^{1*} & w_1^{2*} & G_{dz}^{11} & G_{dz}^{12} \\ w_2^{1*} & w_2^{2*} & G_{dz}^{21} & G_{dz}^{22} \end{pmatrix} N_3^d = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{6.4}$$

where w_1^{1*}, w_2^{1*} are $\dim(x) * 2N * m$ vectors and w_1^{2*}, w_2^{2*} are $\dim(x)$ vectors. Combining (6.2) and (6.4), we find

$$G_{dz}^{ij} = w_i^{1*}([Tf_x(x(\cdot),\alpha)]_d)_z v_d^j. \tag{6.5}$$

So in each iteration step we solve two systems with the almost doubled structure of (3.1) or its transpose.

One also needs the derivatives with respect to κ; for this we find

$$M_d^1 \begin{pmatrix} v_{d\kappa}^1 & v_{d\kappa}^2 \\ G_{d\kappa}^{11} & G_{d\kappa}^{12} \\ G_{d\kappa}^{21} & G_{d\kappa}^{22} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 2v_d^1(1) & 2v_d^2(1) \\ 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

So for κ we find

$$G_{d\kappa}^{ij} = w_i^{2*} v_d^j(1), \tag{6.6}$$

where $i, j \in \{1, 2\}$.

7. Initialization and adaptation of the borders. The bordering vectors in (4.3), (5.3), and (6.3) must be such that the matrices $N_{1,2,3}^d$ are nonsingular. We actually choose them in such a way that the corresponding matrices $N_{1,2,3}^d$ are as well conditioned as possible. This involves an initialization of the borders when the continuation is started and a subsequent adaptation during the continuation.

7.1. The flip and fold cases. During the initialization the borders must be chosen so that the extensions N_1^d of

$$O_1 = \begin{bmatrix} [\quad D \quad - \quad Tf_x(x(\cdot),\alpha) \quad]_d \\ I_{\dim(x)} \quad 0_{\dim(x)\times(Nm-1)\dim(x)} \quad I_{\dim(x)} \end{bmatrix}$$

in the flip case and N_2^d of

$$O_2 = \begin{bmatrix} [\ D \quad - \quad Tf_x(x(\cdot),\alpha) \ ]_d & & [-f(x(\cdot),\alpha)]_d \\ I_{\dim(x)} \quad 0_{\dim(x)\times(Nm-1)\dim(x)} & -I_{\dim(x)} & 0 \\ \mathrm{Int}_{f(x(\cdot),\alpha)_d} & & 0 \end{bmatrix}$$

in the fold case have full rank. We first perform a QR orthogonal-triangular decomposition with column pivoting.

The MATLAB command $[Q, R, E] = QR(full(O_{1,2}))$ produces a permutation matrix E, an upper triangular matrix R of the same dimension as $O_{1,2}$, and a unitary

matrix Q so that $O_{1,2} * E = Q * R$. The column pivoting guarantees that the QR pivoting is rank revealing and in particular that $abs(\mathrm{diag}(R))$ is decreasing; cf. [15, section 5.4]. Since $O_{1,2}$ has rank defect 1, the last element on the diagonal of R should be zero (up to approximation). The borders v_d^1 in (4.3) in the flip case and v_d^1 in (5.3) in the fold case are chosen as the right null vector p of, respectively, O_1 and O_2. It is well known that this makes the bordered matrices nonsingular. This is sometimes called Keller's lemma; cf. [19]. A detailed study of the rank of bordered matrices is given in [17, Proposition 3.2.2]. From $O_{1,2}p = 0$ follows that $RE^Tp = 0$. Setting the bottom right element of R to zero, we obtain

$$\left(\begin{array}{ccccc|c} * & * & * & \dots & * & * \\ 0 & * & * & \dots & * & * \\ 0 & 0 & * & \dots & * & * \\ & & & \dots & & \dots \\ 0 & 0 & 0 & \dots & * & * \\ \hline 0 & 0 & 0 & \dots & 0 & 0 \end{array}\right) E^Tp = \left(\begin{array}{c} 0 \\ \\ \hline 0 \end{array}\right). \tag{7.1}$$

This defines E^Tp up to a scalar multiple. Since its last entry must be nonzero we represent E^Tp as $E^Tp = \binom{p_1}{1}$. Therefore (7.1) has the form

$$\left(\begin{array}{c|c} R_1 & b \\ \hline 0 & 0 \end{array}\right)\left(\begin{array}{c} p_1 \\ \hline 1 \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \end{array}\right)$$

with R_1 a nonsingular upper triangular matrix. This is equivalent to

$$R_1p_1 = -b.$$

So the right border is the normalization of $E*[(R_1\backslash -b); 1]$ (in MATLAB notation). The computation of the left null vector q, which corresponds to w_d^1 and w_d^2 in (4.3) in the flip case and to w_d^1, w_d^2, and w_d^3 in (5.3) in the fold case, is easier. In fact q is the last column q_L of Q because $q_L^T * O_{1,2} = q_L^TQRE^T = (0, \dots, 0, 1)RE^T = 0E^T = 0$.

It is an attractive idea to use the sparsity of $O_{1,2}$ to do a Q-less QR decomposition, $R = QR(O_{1,2})$ as provided in MATLAB, which returns only R. This is enough to find the right singular vector of $O_{1,2}$. The left null vector would then be computed as the normalized right null vector of the transposed $O_{1,2}$. However, we found that this method is not robust and that a QR factorization with pivoting is needed. We tested this by comparing a sparse Q-less QR decomposition, a full QR decompositon without column pivoting, and a full QR decomposition with column pivoting for O_2^T, for the system

$$\left\{\begin{array}{lcl} \dot{u}_1 &=& -u_1 + p_1(1-u_1)e^{u_3}, \\ \dot{u}_2 &=& -u_2 + p_1(1-u_1-p_5u_2)e^{u_3}, \\ \dot{u}_3 &=& -u_3 - p_3u_3 + p_1p_4(1-u_1+p_2*p_5*u_2)e^{u_3}, \end{array}\right.$$

where u_1, u_2, u_3 are state variables and p_1, p_2, p_3, p_4, p_5 are parameters. This is the classical model of the $A \to B \to C$ chemical reaction in a stirred tank [11]; we recall that $1 - u_1$ and u_2 denote the concentrations of A and B, respectively, and u_3 is the temperature. The parameters $p_1, \dots, p_5$ have a physical meaning; e.g., p_1 is the Damkohler number and p_3 is the heat transfer coefficient. There is a Hopf point for

$p_1 = 0.19547, p_2 = 1, p_3 = 1.5, p_4 = 8, p_5 = 0.04$ and $u_1 = 0.57456, u_2 = 0.54511, u_3 = 1.9328$. Starting an LC continuation with p_1 as a free parameter, we find an LPC point at $p_1 = 0.19545$. Starting an LPC continuation from this LPC point with p_1 and p_3 both free parameters for $N = 30$ and $m = 4$, we obtain O_2 as a matrix of dimension 364×364. The matrix

$$R_s = \begin{pmatrix} 0 & 0.000101297163 \\ 0 & 0 \end{pmatrix}$$

is the bottom right 4×4 block of the Q-less decomposition of O_2^T. The matrix

$$R_f = \begin{pmatrix} -0.000000000001 & 0.000088855185 \\ 0 & 0.000048640223 \end{pmatrix}$$

similarly corresponds to the QR decomposition without pivoting of O_2^T as a full matrix and

$$R_{cp} = \begin{pmatrix} 0.009688518843 & 0.000000271536 \\ 0 & 0.000000423924 \end{pmatrix}$$

corresponds to the QR decomposition with column pivoting of O_2^T as a full matrix. Remarkably, there is a clear difference between the bottom right 2×2 blocks of R_s and R_f. The source code of the Q-less QR decomposition is not available, but one easily guesses that small numbers are rounded to zero to increase the sparsity. But in a situation like ours, the presence of a zero as the first entry on the diagonal of R_s leads to a division by zero in the computation of the left null vector q, where q is the result of the analogue of $E*[(R_1 \backslash -b); 1]$. We further note that the QR decomposition as a full matrix without column pivoting is not satisfactory because the third element on the diagonal of R_f is the smallest one and so it does not define the rank. With column pivoting the fourth element on the diagonal of R_{cp} is the smallest and the diagonal elements indeed define the rank.

During the computation of a curve of limit cycle bifurcations the borders v_d^1 in (4.3) in the flip case and v_d^1 and v_d^2 in (5.3) in the fold case can be adapted by replacing them, respectively, by the normalized v_d in (4.2) and by the normalized v_d and S in (5.2). The borders w_d^1, w_d^2 in (4.3) and w_d^1, w_d^2, and w_d^3 in (5.3) are adapted by solving the transposed equations and replacing them, respectively, by the normalized w_1^*, w_2^* in (4.4) and by w_1^*, w_2^*, and w_3 in (5.4). Only one border is adapted at the same time. An adaptation is performed after each number of k computed continuation points where k is a user-chosen number; $k = 0$ means no adaptation at all.

7.2. The torus case. To initialize the borders in the case of a torus bifurcation, the borders are chosen so that the extension

$$O_3 = \begin{bmatrix} [\quad D \quad - \quad Tf_x(x(\cdot), \alpha) \quad]_d \\ I_{\dim(x)} \; 0_{\dim(x)\times(Nm-1)\dim(x)} \; -2\kappa I_{\dim(x)} \; 0_{\dim(x)\times(Nm-1)\dim(x)} \; I_{\dim(x)} \end{bmatrix}$$

of N_3^d has full rank. We implement this by using a QR orthogonal-triangular decomposition with column pivoting, $[Q, R, E] = QR(full(O_3))$. Here R is an upper triangular matrix whose bottom right 2×2 block consists of zeros up to approximation. Setting

this block to zero, we obtain

$$\left(\begin{array}{ccccc|cc} * & * & * & \dots & * & * & * \\ 0 & * & * & \dots & * & * & * \\ 0 & 0 & * & \dots & * & * & * \\ & & & \dots & & \dots & * \\ 0 & 0 & 0 & \dots & * & * & * \\ \hline 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 \end{array}\right) E^T p = \left(\begin{array}{cc} & \\ 0 & 0 \\ & \\ \hline 0 & 0 \end{array}\right)$$

if p is a two-column matrix whose columns span the right null space of O_3. By imposing some structure on $E^T p$ we get

$$\left(\begin{array}{c|cc} R_1 & b_1 & b_2 \\ \hline 0 & 0 & 0 \\ 0 & 0 & 0 \end{array}\right) \left(\begin{array}{c|c} p_1 & p_2 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right) = \left(\begin{array}{cc} & \\ 0 & 0 \\ & \\ \hline 0 & 0 \end{array}\right)$$

or

$$R_1(p_1\ p_2) = -(b_1\ b_2),$$

where R_1 is a nonsingular square upper triangular matrix. So the two right bordering vectors in (6.3) are initially chosen as the normalization and orthogonalization of $E * [(R_1 \backslash [-b_1, -b_2]); eye(2)]$ where $eye(2)$ is the 2×2 identity matrix. The left null matrix consists of the two last columns of Q. Indeed, if we denote this part of Q as Q_L, then

$$Q_L^T * O_3 = Q_L^T Q R E^T = \left(\begin{array}{c} 0, \dots 0, 1, 0 \\ 0, \dots 0, 0, 1 \end{array}\right) R E^T = 0 E^T = 0.$$

From (2.9) and (2.10) we get four equations $G_{ij} = 0$ ($(i, j) \in \{1, 2\}$) in the case of a torus bifurcation. The continuation algorithm needs only two of them. To select those two we start with the full QR decomposition, $[Q_1, R_1] = QR((jac\ jacT\ jacp\ 0)')$, where $(jac\ jacT\ jacp)$ is the Jacobi matrix of the limit cycle equations with respect to the state variables, period, and parameters, respectively. Extending the equality $(jac\ jacT\ jacp\ 0)Q_1 = R_1^T$ by some simple computations we obtain a decomposition

$$\left(\begin{array}{cccc} jac & jacT & jacp & 0 \\ G_{11x} & G_{11T} & G_{11p} & G_{11\kappa} \\ G_{12x} & G_{12T} & G_{12p} & G_{12\kappa} \\ G_{21x} & G_{21T} & G_{21p} & G_{21\kappa} \\ G_{22x} & G_{22T} & G_{22p} & G_{22\kappa} \end{array}\right) Q_1 = \left(\begin{array}{ccccc|cc} * & 0 & \dots & 0 & 0 & 0 & 0 \\ * & * & \dots & 0 & 0 & 0 & 0 \\ & & \dots & & & \dots & \\ * & * & \dots & * & 0 & 0 & 0 \\ \hline ** & * & \dots & * & & J'_{res} & \end{array}\right) \tag{7.2}$$

where J_{res} is a 3×4 matrix with rank 2. We now want to choose two among the last four rows of the right-hand side of (7.2) to make the right-hand side as well conditioned as possible. We perform a QR decomposition with pivoting $[Q_2, R_2, E_2] = QR(full(J_{res}))$. So

$$J_{res} E_2 = (q_1\ \ q_2\ \ q_3) \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & 0 & 0 \end{array}\right)$$

$$= (\ q_1\ \ q_2\ ) \left(\begin{array}{cccc} * & * & * & * \\ 0 & * & * & * \end{array}\right),$$

TABLE 9.1

Variable	Value	Parameter	Value
A	31.78997	k_1	0.1631021
B	1.45468	k_2	1250
X	0.01524586	k_3	0.046875
Y	0.1776113	k_4	20
		k_5	1.104
		k_6	0.001
		k_7	4.235322
		k_8	0.5
		k_{-7}	0.1175

from which it follows that the first two columns of $J_{res}E_2$ are linearly independent. This also means that the columns of J_{res} we need to use (equivalently, which G_{ij} we need to choose) are those where the first or second column of E_2 contains an entry equal to 1.

The borders v_d^{01} and v_d^{02} in (6.3) are adapted by replacing them, respectively, by the normalized and orthogonalized v_d^1 and v_d^2 in (6.2). The borders w_d^{11}, w_d^{22} and w_d^{12}, w_d^{22} in (6.3) are adapted by solving the transposed equations and replacing them, respectively, by the normalized and orthogonalized w_1^{1*}, w_1^{2*} and w_2^{1*}, w_2^{2*} in (6.4). The new indices are computed in the same way as in the initialization. The general strategy for adaptation during the computation of a curve of NS cycles is the same as in the fold and flip cases.

8. Multipliers. Multipliers are (optionally) computed in MATCONT as in AUTO [10] and CONTENT [21] by making a special decomposition (condensation of parameters) in (3.1) (see section 2.3 in [9]). A periodic solution always has a multiplier equal to 1. At a fold, the multiplier 1 has algebraic multiplicity 2 and geometric multiplicity 1, at a flip there is a simple multiplier equal to -1, and at a torus bifurcation there is a simple conjugate pair of complex eigenvalues with modulus 1. This can be used to check the correctness of the continuation. We note that the computation of multipliers is expensive because the sparse matrix is not handled with the sparse MATLAB routines. As in AUTO, it is good practice to switch off the computation of multipliers when they are not really needed.

9. Example: Fold and torus bifurcations in a chemical model. The following model of the peroxidase-oxidase reaction was studied by Steinmetz and Larter [23] where A, B, X, Y are state variables and k_1, k_2, k_3, k_4, k_5, k_6, k_7, k_8, k_{-7} are parameters:

$$\left\{\begin{array}{rcl} \dot{A} &=& -k_1ABX - k_3ABY + k_7 - k_{-7}A, \\ \dot{B} &=& -k_1ABX - k_3ABY + k_8, \\ \dot{X} &=& k_1ABX - 2k_2X^2 + 2k_3ABY - k_4X + k_6, \\ \dot{Y} &=& -k_3ABY + 2k_2X^2 - k_5Y. \end{array}\right. \tag{9.1}$$

In [23] Hopf bifurcations, torus bifurcations, and the onset of chaos are studied. It has been shown recently that (9.1) also possesses generalized Hopf points and associated LPCs (see [16]).

The values in Table 9.1 correspond to an unstable equilibrium in (9.1).

First, we continue this equilibrium with increasing k_7 while keeping all other parameters fixed. The computed equilibrium branch is a nearly horizontal curve

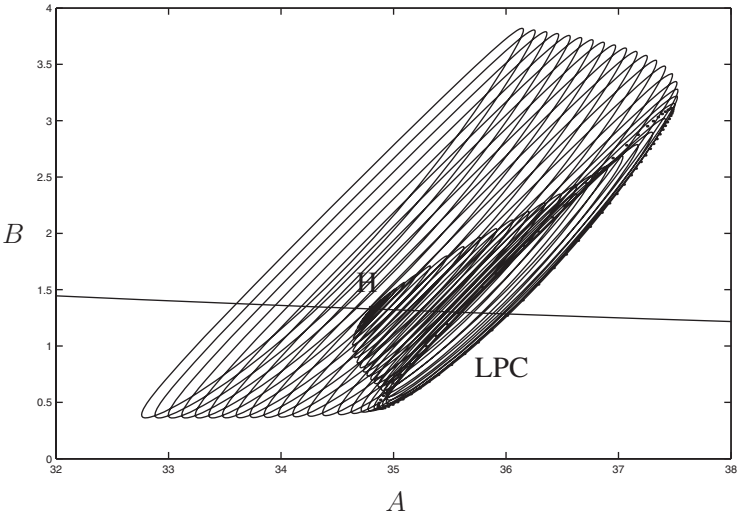


FIG. 9.1. *LPC in the Steinmetz–Larter model.*

in Figure 9.1. At $k_7 = 4.59004\ldots$ a subcritical Hopf bifurcation is located (point H). Indeed, there are two complex eigenvalues of the equilibrium with Re $\lambda_{1,2} \approx 0$ at this parameter, while the first Lyapunov coefficient l_1 is positive. Thus, there should exist an unstable limit cycle bifurcating from this equilibrium. Selecting the found Hopf point as the new initial point, we can continue in MATCONT a branch of the bifurcating limit cycles and obtain the rest of Figure 9.1. There is an LPC at $k_7 = 4.74839\ldots$, where two cycles collide and disappear. The critical cycle has a double multiplier $\mu = 1$ (counting the trivial one). The continuation algorithm automatically follows the second (stable) cycle branch after the LPC point.

Computing the original equilibrium curve in the opposite direction, we get another Hopf point at $k_7 = 0.712475\ldots$, where the first Lyapunov coefficient is negative. This means that a stable limit cycle bifurcates from the equilibrium when it loses stability. Starting from this Hopf point, we obtain the family of stable limit cycles bifurcating from it. At $k_7 = 0.716434\ldots$ a message indicates that a [illegible] (NS) bifurcation occurs. Indeed, there are two complex multipliers with (approximately) $|\mu| = 1$. After the bifurcation point the limit cycle becomes unstable (with two multipliers satisfying $|\mu| > 1$) but regains stability at $k_7 = 0.818566\ldots$ through another NS bifurcation.

The found Hopf, LPC, and NS points can be used as starting points for the two-parameter continuation of the corresponding codim 1 bifurcations, using k_7 and k_8 as two control parameters. Figure 9.2 gives a graphical impression of a typical MATCONT session. This figure shows the MATCONT main window, the starter and continuer windows for the NS continuation, and also graphical output in a two-dimensional plot window; in the latter A is plotted versus B. For more information on the MATCONT windows we refer to the MATCONT website.

The bifurcation curves computed with MATCONT are presented in Figure 9.3, where a (partial) bifurcation diagram in the (k_7, k_8)-plane is shown together with the enlargement of an interesting region. There are two [illegible] in

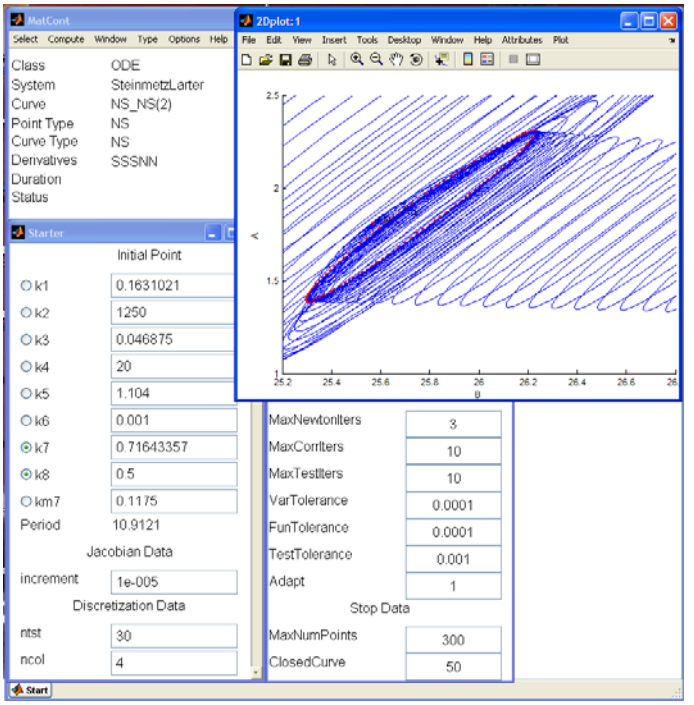


FIG. 9.2. *NS continuation in the Steinmetz–Larter model.*

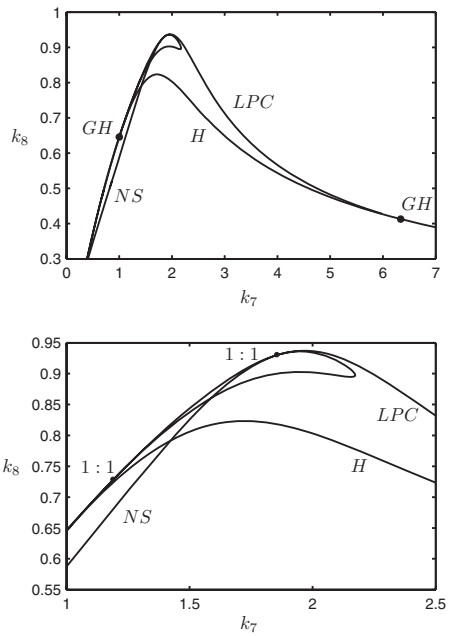


FIG. 9.3. *Bifurcation curves in Steinmetz–Larter model. H:Hopf bifurcation with two generalized Hopf points (labeled by GH); LPC: limit point of cycles; NS: Neimark–Sacker (torus) bifurcation curve with two* $1:1$ *strong resonances.*

the Hopf bifurcation curve (labeled by GH), where the associated first Lyapunov coefficient l_1 vanishes [16]. As theory predicts (see, for example, [20]), from each GH point emanates an LPC curve. Actually, there is just one LPC curve connecting the GH points; while approaching these points, the critical nonhyperbolic cycle shrinks to the equilibrium point. The shape of the NS curve is more complicated: there are two more codim 2 points on it, where a triple multiplier $\mu = 1$ is present (counting the trivial one). These are 1:1 [illegible] points [20], where the LPC and NS curves meet tangentially. Between the 1:1 points, the NS curve is a [illegible] curve. Near such codim 2 points complicated homoclinic structures exist.

It should be noted that the algorithms implemented in MATCONT for the LPC and NS continuation are robust enough to pass through the 1:1 resonance points and to closely approach the GH points (within the 10^{-3}-range in the parameters).

10. Fully versus minimally extended system. The methods described in this paper to continue bifurcation points of limit cycles are different from the traditional ones. Classical software like AUTO uses fully extended systems where additional vector functions are added to the system for the limit cycle to form a big system of approximately double the size in the fold and flip cases and triple the size in the torus case. In the PD case this leads to

$$\begin{cases} \dfrac{dx}{dt} - Tf(x,\alpha) & = 0, \\ x(0) - x(1) & = 0, \\ \int_0^1 \langle x(\cdot), \dot{x}_{old}(t)\rangle dt & = 0, \\ \dot{v}(t) - Tf_x(x,\alpha)v(t) & = 0, \\ v(0) + v(1) & = 0, \\ \int_0^1 \langle v_{old}(t), v(t)\rangle dt - 1 & = 0, \end{cases}$$

where x, v, T, α are the unknowns and v is a rescaled version of the variable with the same name in (2.4).

This system is easier to code since there is no need to find and adapt the bordering vectors. However, it has a big disadvantage: it has approximately twice the size of the minimally extended system. On the other hand, for the minimally extended system the additional system (2.4) has to be solved. But solving two systems of size $n \times n$ takes in general less time than solving one system of size $2n \times 2n$ (in the case of full matrices about a quarter). This suggests that using the big system will be slower than using the minimally extended system.

We did some comparisons in the case of the PD bifurcations in a feedback control system, described in [13], [14] and further used in [20, Example 5.4, p. 178]:

$$\begin{cases} \dot{x} & = & y, \\ \dot{y} & = & z, \\ \dot{z} & = & -\alpha z - \beta y - x + x^2. \end{cases}$$

Due to the special structure of this system, a good approximation to the PD curve can be found by the harmonic balance method; cf. [24], [25].

We compute numerically a branch of PD cycles as described in the CL_MATCONT manual and [7]. To avoid the overhead of the graphical user interface (GUI) all computations were done in CL_MATCONT. Both the minimally and the fully extended systems were implemented and tested for several numbers of test and collocation

TABLE 10.1
Time comparison between two methods for 300 *PD cycles.*

	N	m	Minimally extended system	Fully extended system
(a)	10	4	183,2 s	198,3 s
(b)	10	5	230,2 s	251,7 s
(c)	20	4	360,7 s	409,5 s
(d)	20	5	458,0 s	534,1 s
(e)	30	4	540,6 s	642,0 s
(f)	30	5	683,6 s	857,8 s
(g)	50	4	885,6 s	1184,8 s
(h)	50	5	1150,6 s	1616,5 s

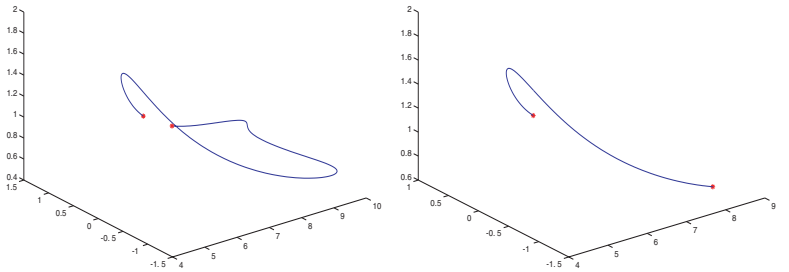


FIG. 10.1. *PD curves computed in Table* 10.1(e).

points. In each case we compute 300 continuation points with the same continuation parameters.

From Table 10.1 it appears that the minimally extended system is faster; moreover, it computes a longer stretch of the PD branch. We illustrate this in Figure 10.1 in the case (e) ($N = 30, m = 4$), but it holds in all cases.

This comparison is crude because the continuation variables are in different spaces and therefore thresholds and stepsizes have a different meaning in the two cases. To describe the more refined comparisons we note that in MATCONT and CL_MATCONT the convergence of Newton iterations in the continuation of a branch of solutions to (1.1) is declared if two conditions are satisfied, namely,

$$\|F(x)\| \leq F_t, \|\delta x\| \leq V_t,$$

where δx is the Newton correction and F_t (function tolerance) and V_t (variable tolerance) are user-chosen threshold values. Also, the code uses an integer parameter N_a which indicates that the settings of the continuation are adapted after each N_a computed points. Adaptation is a curve-type-dependent process. In the continuation of limit cycles it always includes the adaptation of the mesh. In the minimally extended systems it also includes updating the borders as described in section 7. Setting $N_a = 0$ means no adaptation at all. This cannot be recommended, but it can be studied usefully as the limit case for large N_a.

In the other experiments we choose situations where both methods compute the same stretch of curve with the same number of points but with F_t, V_t adapted to the dimension of the space of continuation variables. In all cases we have $N = 50, m = 5$, $\dim(x) = 3$. The object function of the minimally extended system (F in (1.1)) has dimension $Nm\dim(x) + \dim(x) + 2 = 755$. The dimension of the

TABLE 10.2
Profile report of PD curve continuation with minimally extended system.

Filename	Calls	Total time
`cont`	1	111.390 s
`perioddoubling`	177	107.093 s
`newtcorr`	25	105.654 s
`contjac`	50	84.091 s

TABLE 10.3
Profile report of PD curve continuation with fully extended system.

Filename	Calls	Total time
`cont`	1	118.430 s
`perioddoubling2`	178	107.243 s
`newtcorr`	25	110.169 s
`contjac`	50	90.071 s

space of independent variables (x in (1.1)) is therefore 756. For the fully extended system these dimensions are, respectively, $2Nm\dim(x) + 2\dim(x) + 2 = 1508$ and 1509. In the case of the minimally extended system we choose $F_t = V_t = 10^{-4}$. If we ignore scaling problems, then corresponding bounds for the fully extended systems are $10^{-4}\sqrt{\frac{1508}{755}} = 1.4133e-4$ and $10^{-4}\sqrt{\frac{1509}{756}} = 1.4128e-4$, respectively.

Both algorithms were forced to compute 150 points in the given stretch by varying the maximal stepsizes. With $N_a = 0$ the minimally extended system took 561.0 seconds, the fully extended system 617.2 seconds. With $N_a = 1$ the minimally extended system took 586.3 seconds; the fully extended system took 687.4 seconds.

In a similar experiment with only 25 computed points and $N_a = 0$ the minimally and fully extended systems took 108.56 and 111.70 seconds, respectively. With $N_a = 1$ the time spans were 111.39 and 118.43 seconds, respectively.

Since scaling is a difficult issue, one might wonder if setting F_t, V_t in the above way really corresponds to the same accuracy in the computed solutions. We therefore did an additional test in the last mentioned experiment (25 points, $N_a = 1$) by computing the norms of the (x, T, α) parts in the last Newton correction in each continuation point. For the minimally extended system the average value of these norms was $5.086e-8$, and for the fully extended system it was $1.111e-6$. This clearly indicates that the solutions obtained by the minimally extended system are an order of magnitude more precise. Also, it shows that the computations are actually much more accurate than the modest bounds for F_t, V_t suggest.

To obtain more detailed information we used the MATLAB Profiler. The lines where the most time was spent are given in Tables 10.2 and 10.3, respectively.

The most obvious conclusion is that in both cases the `newtcorr` procedure (the process of Newton iterates from a predicted to a declared continuation point) consumes nearly all the time spent in the continuation process; therefore a separate compilation of this algorithm seems highly desirable.

The second observation is that in `newtcorr` most of the time is spent in `contjac`, i.e., in the evaluation of the Jacobian matrix of the system, rather than in the solvers. In Tables 10.4 and 10.5 we provide more detailed information. An application of the sparse solver in the first system takes only 0.0487 seconds; in the second system it takes 0.1381 seconds, a clear advantage for the minimally extended system. The total time spent during solves is 5.9790 s versus 10.3550 s; we note that the 5.9790 s include not only the times in the 4th and 5th rows of Table 10.4 but also a part of the 3rd row since the evaluation of the object function of the minimally extended system involves

TABLE 10.4
Detailed profile report of `newtcorr` *in Table* 10.2.

Code	Calls	Total time
A = `contjac`(x)	25	42.081 s
A=`contjac`(x)	25	42.040 s
Q =[`feval`(`cds.curve`,x);...	49	17.756 s
D=b \[Q R']	49	2.484 s
D=([A;v'] \R'	25	1.123 s
All other lines		0.170 s
Totals		105.654 s

TABLE 10.5
Detailed profile report of `newtcorr` *in Table* 10.3.

Code	Calls	Total time
A = `contjac`(x)	25	45.064 s
A=`contjac`(x)	25	45.007 s
Q =[`feval`(`cds.curve`,x);...	50	9.443 s
D=b \[Q R']	50	7.141 s
D=([A;v'] \R'	25	3.214 s
All other lines		0.300 s
Totals		110.169 s

solving the linear system (2.4). These more detailed data were also provided by the MATLAB Profiler.

For the evaluation of the Jacobians the times are 1.6824 and 1.8014 seconds, respectively, a small advantage for the minimally extended system.

The superior performance of the minimally extended system is even more apparent if a higher precision is required. In another experiment we put $F_t = V_t = 10^{-6}$ for the minimally extended system and correspondingly $F_t = 10^{-6}\sqrt{\frac{1508}{755}} = 1.4133e-6$, $V_t = 10^{-6}\sqrt{\frac{1509}{756}} = 1.4128e-6$ for the fully extended system. We computed 25 points with $N_a = 1$; the minimally extended system needed 117.57 seconds, and the fully extended system needed 176.21 seconds.

Our conclusion is that the minimally extended system outperforms the fully extended system. However, the gain is modest because most of the computing time is not spent in the solves (these are performed by the compiled built-in solvers of MATLAB) but in constructing the Jacobian matrices. We can therefore expect that future developments (compiling parts of the code or better handling of for-loops in MATLAB) will make the advantages of the minimally extended system even greater.

REFERENCES

[1] E. L. ALLGOWER AND K. GEORG, *Numerical Continuation Methods: An Introduction*, Springer-Verlag, Berlin, 1990.

[2] U. ASCHER, J. CHRISTIANSEN, AND R. D. RUSSELL, *Collocation software for boundary value ODE's*, ACM Trans. Math. Software, 7 (1981), pp. 209–222.

[3] U. ASCHER AND R. D. RUSSELL, *Reformulation of boundary value problems into "standard" form*, SIAM Rev., 23 (1981), pp. 238–254.

[4] W. J. BEYN, A. CHAMPNEYS, E. DOEDEL, W. GOVAERTS, YU. A. KUZNETSOV, AND B. SANDSTEDE, *Numerical continuation and computation of normal forms*, in Handbook of Dynamical Systems, Vol. 2, B. Fiedler, G. Iooss, and N. Kopell, eds., North–Holland, Amsterdam, 2002, pp. 149–219.

[5] C. DE BOOR AND B. SWARTZ, *Collocation at Gaussian points*, SIAM J. Numer. Anal., 10 (1973), pp. 582–606.

[6] A. Dhooge, W. Govaerts, and Yu. A. Kuznetsov, *MATCONT: A MATLAB package for numerical bifurcation analysis of ODEs*, ACM Trans. Math. Software, 29 (2003), pp. 141–164.

[7] A. Dhooge, W. Govaerts, Yu. A. Kuznetsov, W. Mestrom, and A. M. Riet, CL_MATCONT*: A continuation toolbox in MATLAB*, Proceedings of the 2003 ACM Symposium on Applied Computing (Melbourne, FL), ACM, New York, 2003, pp. 161–166.

[8] E. J. Doedel, H. B. Keller, and J. P. Kernevez, *Numerical analysis and control of bifurcation problems.* I. *Bifurcation in finite dimensions*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 1 (1991), pp. 493–520.

[9] E. J. Doedel, H. B. Keller, and J. P. Kernevez, *Numerical analysis and control of bifurcation problems.* II. *Bifurcation in infinite dimensions*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 1 (1991), pp. 745–772.

[10] E. J. Doedel, A. R. Champneys, T. F. Fairgrieve, Yu. A. Kuznetsov, B. Sandstede, and X. J. Wang, AUTO97–AUTO2000*: Continuation and Bifurcation Software for Ordinary Differential Equations (with HomCont)*, User's Guide, Concordia University, Montreal, Canada, 1997–2000; http://indy.cs.concordia.ca.

[11] E. J. Doedel and R. F. Heinemann, *Numerical computation of periodic solution branches and oscillatory dynamics of the stirred tank reactor with $A \to B \to C$ reactions*, Chem. Engrg. Sci., 38 (1983), pp. 1493–1499.

[12] E. J. Doedel, W. Govaerts, and Yu. A. Kuznetsov, *Computation of periodic solution bifurcations in ODEs using bordered systems*, SIAM J. Numer. Anal., 41 (2003), pp. 401–435.

[13] R. Genesio and A. Tesi, *Harmonic balance methods for the analysis of chaotic dynamics in nonlinear systems*, Automatica J. IFAC, 28 (1992), pp. 531–548.

[14] R. Genesio, A. Tesi, and F. Villoresi, *Models of complex dynamics in nonlinear systems*, Systems Control Lett., 25 (1995), pp. 185–192.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., John Hopkins University Press, Baltimore, 1996.

[16] W. Govaerts, Yu. A. Kuznetsov, and B. Sijnave, *Numerical methods for the generalized Hopf bifurcation*, SIAM J. Numer. Anal., 38 (2000), pp. 329–346.

[17] W. Govaerts, *Numerical Methods for Bifurcations of Dynamical Equilibria*, SIAM, Philadelphia, 2000.

[18] J. Guckenheimer and B. Meloon, *Computing periodic orbits and their bifurcations with automatic differentiation*, SIAM J. Sci. Comput., 22 (2000), pp. 951–985.

[19] H. B. Keller, *Numerical Methods in Bifurcation Problems*, Springer-Verlag, Berlin, New York, 1987.

[20] Yu. A. Kuznetsov, *Elements of Applied Bifurcation Theory*, 2nd ed., Springer-Verlag, New York, 1998.

[21] Yu. A. Kuznetsov and V. V. Levitin, *Integrated environment for analysis of dynamical systems*, CWI, Amsterdam, 1997. Available via anonymous ftp from ftp://ftp.cwi.nl/pub/CONTENT.

[22] W. Rheinboldt, D. Roose, and R. Seydel, *Aspects of continuation software*, in Continuation and Bifurcations: Numerical Techniques and Applications, D. Roose, B. De Dier, and A. Spence, eds., NATO ASI Series, Series C 313, Kluwer, Dordrecht, The Netherlands, 1990, pp. 261–268.

[23] C. Steinmetz and R. Larter, *The quasiperiodic route to chaos in a model of the peroxidase-oxidase reaction*, J. Chem. Phys. 74 (1991), pp. 1388–1396.

[24] A. Tesi, E. H. Abed, R. Genesio, and H. O. Wang, *Harmonic balance analysis of period-doubling bifurcations with implications for control of nonlinear dynamics*, Automatica J. IFAC, 32 (1996), pp. 1255–1271.

[25] G. Torrini, R. Genesio, and A. Tesi, *On the computation of characteristic multipliers for predicting limit cycle bifurcations*, Chaos Solitons Fractals, 9 (1998), pp. 121–133.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 253–277
© 2005 Society for Industrial and Applied Mathematics

MULTIPLE EXPLICITLY RESTARTED ARNOLDI METHOD FOR SOLVING LARGE EIGENPROBLEMS*

NAHID EMAD†, SERGE PETITON‡, AND GUY EDJLALI§

Abstract. In this paper we propose a new approach for calculating some eigenpairs of large sparse non-Hermitian matrices. This method, called Multiple Explicitly Restarted Arnoldi (MERAM), is particularly well suited for environments that combine different parallel programming paradigms. This technique is based on a multiple use of the Explicitly Restarted Arnoldi method (ERAM) and improves its convergence.

This technique is implemented and tested on a distributed environment consisting of two interconnected parallel machines. The MERAM technique is compared with ERAM, and one can notice that the convergence is improved. In some cases, more than a twofold improvement can be seen in MERAM results.

We also implemented MERAM on a cluster of workstations. According to our experiments, MERAM converges better than the Explicitly Restarted Block Arnoldi method and, for some matrices, more quickly than the PARPACK package, which implements the Implicitly Restarted Arnoldi method.

Key words. large eigenproblem, Arnoldi method, explicit restarting, parallel programming, asynchronous communication, heterogeneous environment

AMS subject classifications. 65F15, 65F50, 65Y05

DOI. 10.1137/S1064827500366082

1. Introduction.

Hybrid methods were proposed during the last decade to accelerate the convergence and/or improve the accuracy of the solution of some linear algebra problems. These methods combine several different numerical methods to solve these problems efficiently. For example, both convergence acceleration techniques and preconditioning methods could be used to develop a hybrid method. Hybrid methods were used successfully to solve linear systems, such as the method introduced by Brézinski and Redivo-Zagila [4], some eigenproblems with the Arnoldi–Chebyshev method proposed by Saad [20, 19, 3], and Jacobi–Davidson algorithms from Sleijpen and Van der Vorst [30, 29].

In this paper, we propose a new approach for calculating some of the eigenpairs in the context of large sparse non-Hermitian matrices. This new technique, called the Multiple Explicitly Restarted Arnoldi method (MERAM), is a hybrid method and is based on the Explicitly Restarted Arnoldi method (ERAM). It involves multiple invocations of ERAM and is closely linked to the Explicitly Restarted Block Arnoldi method (ERBAM). Each ERAM invocation is done with a different set of parameter values. This approach also may be applied to some other restarted projection methods.

We propose an asynchronous parallel algorithm for MERAM and show that some of its properties such as asynchronous communications between its coarse grain subtasks, fault tolerance, and dynamic load balancing make this method well adapted

*Received by the editors January 13, 2000; accepted for publication (in revised form) October 26, 2004; published electronically September 12, 2005. The two first authors were supported in part by French ACI-GRID ANCG and CGP2P Projects.

http://www.siam.org/journals/sisc/27-1/36608.html

†Laboratoire PRiSM, Université Versailles St-Quentin, 45, av. des États-Unis, 78035 Versailles, France (emad@prism.uvsq.fr).

‡Laboratoire d'Informatique Fondamentale de Lille, Université des Sciences et Technologies de Lille, Cité Scientifique, 59655 Villeneuve d'Ascq, France (petiton@lifl.fr).

§Google, 1600 Amphitheater Parkway, Mountain View, CA 94043 (edjlali@google.com).

to the GRID computational environments. Our experiments ran on two different platforms—a network of parallel machines and a cluster of workstations—and showed that MERAM converges quicker than ERAM and ERBAM. Moreover, according to our experiments, for some matrices, one has a slow convergence with the Implicitly Restarted Arnoldi method (IRAM) and a faster convergence with MERAM.

Our asynchronous MERAM algorithm can be easily implemented on a cluster of heterogeneous machines as it exhibits a coarse grain parallelism. These machines can be sequential, vector, or parallel. The number of iterations to convergence of the main process is reduced by combining results from other processes at run time. For example, a dedicated parallel machine can take advantage of the availability of a cluster of nondedicated processors to speed up its convergence. We implemented our approach on a Connection Machine 5 (CM5) and a small cluster of workstations [7] and were able to show improvement on the number of iterations before convergence by just combining two Sun Sparc machines with our 32-node CM5 parallel machine. In this paper, we present results when multiple parallel machines are available for computation. In particular, we show that in some cases we can achieve more than twofold improvement in the number of iterations before convergence by combining the results from a faster and a slower parallel machine. We also will present certain results of comparison of MERAM with the Explicitly Restarted Block Arnoldi and Implicitly Restarted Arnoldi methods on a cluster of workstations.

Before defining MERAM in the context of the projection methods, a brief presentation of these methods is given in the next section. An overview of the Arnoldi method and some of its variants is provided in section 3. MERAM and an asynchronous algorithm of this method are presented in section 4. It also includes the restarting strategy selected in MERAM, the stopping criterion of the restarts in ERAM/MERAM, and the relationship between MERAM and ERBAM. Parallel versions of these methods are discussed in section 5. Section 6 is devoted to numerical experiments. Finally, possible applications of the concept are discussed along with the conclusion in section 7.

2. Notation. Let A be a complex non-Hermitian matrix of dimension $n \times n$ and let $\mathbb{K}$ be a subspace of $\mathbb{C}^n$. A Krylov subspace method allows an approximation of an eigenpair (λ, u) of A by a Ritz pair ($\lambda^{(m)} \in \mathbb{C}$, $u^{(m)} \in \mathbb{K}$), where the subspace $\mathbb{K}$ is defined by

$$\mathbb{K}_{m,X} = \mathrm{Span}(X, AX, \ldots, A^{m-1}X) \tag{2.1}$$

and X is spanned by a set of μ vectors. The point (resp., block) Krylov subspace methods are characterized by choosing $\mu = 1$ (resp., $\mu > 1$).

A Krylov subspace method approximates s eigenpairs of A by those of a matrix of order m, where $s \leq m \ll n$. This matrix is obtained by orthogonal projection of A onto an m-dimensional subspace $\mathbb{K}_{m,v}$ with $X = \mathrm{Span}(v)$. Let W_m be the matrix whose columns $w_1, \ldots, w_m$ constitute an orthogonal basis of $\mathbb{K}_{m,v}$. The problem is to find $\lambda^{(m)} \in \mathbb{C}$ and $y^{(m)} \in \mathbb{C}^m$ such that

$$(H_m - \lambda^{(m)} I) y^{(m)} = 0, \tag{2.2}$$

where the matrix H_m of dimension $m \times m$ is defined by $H_m = W_m^H A W_m$. Note that W_m^H is the transpose conjugate of W_m and $u^{(m)} = W_m y^{(m)}$. Therefore, some eigenvalues of A can be approximated by the eigenvalues of the matrix H_m. These eigenvalues can be found by building an orthogonal basis of $\mathbb{K}_{m,v}$ and by solving the

problem (2.2). There are different ways of building such a basis, and the most used process is the Arnoldi orthogonalization.

3. Arnoldi method and some of its variants.

3.1. Arnoldi method. Let the initial guess w_1 be equal to $v/\|v\|_2$. The well-known Arnoldi process generates an orthogonal basis $w_1, \ldots, w_m$ of Krylov subspace $\mathbb{K}_{m,v}$ by using the Gram–Schmidt orthogonalization process:

`Arnoldi Reduction :` $\texttt{AR}(input : A, m, v; output : H_m, W_m)$.

For $j = 1, 2, \ldots, m$ do:

- $h_{i,j} = (Aw_j, w_i)$, for $i = 1, 2, \ldots, j$
- $\bar{w}_j = Aw_j - \sum_{i=1}^{j} h_{i,j} w_i$
- $h_{j+1,j} = \|\bar{w}_j\|_2$
- $w_{j+1} = \bar{w}_j / h_{j+1,j}$

The above algorithm may break down if $w_j = 0$ for some j. This may happen if the minimal polynomial of v is of degree j. In this case, the subspace $\mathbb{K}_{m,v}$ is invariant and the Ritz elements are exact [19].

This method was introduced by Arnoldi [1] in 1951 to reduce a matrix to a Hessenberg form. The reduced matrix $H_m = (h_{i,j})$ is a Hessenberg representation of A in the orthonormal basis W_m of $\mathbb{K}_{m,v}$ when $m = n$. Arnoldi hinted that the process could give good approximations to some eigenvalues if stopped before completion [1], i.e., when $m < n$.

The matrices H_m and W_m issued from the `AR` algorithm and the matrix A satisfy the equation

$$AW_m = W_m H_m + f_m e_m^H, \tag{3.1}$$

where $f_m = h_{m+1,m} w_{m+1}$ and e_m is the mth vector of the canonical basis of $\mathbb{C}^m$. Note that the `AR` algorithm is the standard Gram–Schmidt orthogonalization process and assumes exact arithmetics. Therefore, the modified Gram–Schmidt or the Householder variants of `AR` are more interesting as they both deal with inaccuracies introduced by the hardware. The cost analysis and comparison of each version are given by Saad in [19].

Once the choice of the orthogonalization process is fixed, the s desired Ritz values (with largest/smallest real part or largest/smallest magnitude) $\Lambda_m = (\lambda_1^{(m)}, \ldots, \lambda_s^{(m)})$ and the corresponding Ritz vectors $U_m = (u_1^{(m)}, \ldots, u_s^{(m)})$ can be calculated as follows:[1]

`Basic Arnoldi Algorithm :` $\texttt{BAA}(input : A, s, m, v; output : r_s, \Lambda_m, U_m)$.

---

1. Compute an $\texttt{AR}(input : A, m, v; output : H_m, W_m)$ step.
2. Compute the eigenpairs of H_m and select the s desired ones.
3. Compute the s associate Ritz vectors $u_i^{(m)} = W_m y_i^{(m)}$.
4. Compute $r_s = (\rho_1, \ldots, \rho_s)$ with $\rho_i = \|(A - \lambda_i^{(m)} I) u_i^{(m)}\|_2$.

---

[1] We suppose that the eigenvalues and corresponding eigenvectors of H_m are reindexed so that the first s Ritz pairs are the desired ones.

If the accuracy of the computed Ritz elements is not good enough, the projection can be restarted to generate a new $\mathbb{K}_{m,v}$.

The new $\mathbb{K}_{m,v}$ can be defined with [illegible] initial vector v and [illegible] m value. It is clear that, according to the hypothesis that v does not belong to any desired invariant subspace, m has to be as large as possible. An important, well-known shortcoming of this version of the Arnoldi method is its large storage space requirement and computational cost for large values of m.

3.2. Explicitly Restarted Arnoldi method. In this version the projection subspace size is [illegible]. Starting with an initial vector v, it computes a BAA (Basic Arnoldi Algorithm). The starting vector is updated, and a BAA process is restarted again until the accuracy of the approximated solution is satisfactory (using appropriate methods on the computed Ritz vectors). This update is designed to force the vector to be in the desired invariant subspace. This goal can be reached by some polynomial restarting strategies proposed in [19] and discussed in section 4.2. This method is called the Explicitly Restarted Arnoldi method (ERAM) and its algorithm is given below:

ERAM$(input : A, s, m, v, tol; output : r_s, \Lambda_m, U_m)$.

---

1. `Start.` Choose a parameter m and an initial vector v.
2. `Iterate.` Compute a BAA$(input : A, s, m, v; output : r_s, \Lambda_m, U_m)$ step.
3. `Restart.` If $g(r_s) > tol$, then use Λ_m and U_m to update the starting vector v and go to 2.

In the previous algorithm, *tol* is a threshold value and the function g defines the stopping criterion of iterations. We will see some possible definition of this function in the next section.

3.3. Implicitly Restarted Arnoldi method. Another variant of the Arnoldi method based on a restarting technique is proposed by Sorensen [23]. This refined version, called the Implicitly Restarted Arnoldi method (IRAM), is a technique that combines the implicitly shifted QR with an Arnoldi factorization and can be viewed as a truncated form of the implicitly shifted QR technique. This technique involves an [illegible] application of a polynomial in A to the starting vector. It has been shown that IRAM leads to good approximations of Ritz elements of large sparse matrices [23, 11].

Several other versions of the Arnoldi method based on the augmented Krylov methods exist, including variants proposed by Wu and Simon [31], Stewart [25], Morgan [12], and Morgan and Zeng [13].

4. Multiple Explicitly Restarted Arnoldi method. The method we propose is characterized by a new restarting technique for the Arnoldi method. Our restarting technique combines the two explicit algorithms described above. In this version, we neither fix the parameter m nor the initial vector v. To overcome the storage-dependent shortcoming of the first variant of the Arnoldi method, we introduce a constraint on the subspace size m. More precisely, we suppose that m belongs to a discrete interval $I_m = [m_{min}, m_{max}]$. The lower bound m_{min} and upper bound m_{max} may be chosen as a function of the computation and storage resources. In addition, the constraint $m_{min} \leq m_{max} \ll n$ has to be fulfilled. Let $M = (m_1, \ldots, m_\ell)$ be a set of subspace sizes with $m_i \leq \cdots \leq m_\ell$ and $m_i \in I_m$ $(1 \leq i \leq \ell)$, and let $V^\ell = [v^1, \ldots, v^\ell]$ be the matrix of ℓ starting vectors. An algorithm of this method for

calculating s ($s \leq m_1$) desired Ritz elements of A is as follows:

MERAM($input : A, s, M, V^\ell, tol; output : r_s^m, \Lambda_m, U_m$).

---

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \ldots, m_\ell)$.
2. `Iterate.` For $i = 1, \ldots, \ell$ do:
   (a) Compute a BAA($input : A, s, m_i, v^i; output : r_s^i, \Lambda_{m_i}, U_{m_i}$) step.
   (b) If $g(r_s^i) \leq tol$, then stop all processes.
3. `Restart.` Update the vectors $v^1, \ldots, v^\ell$ and go to 2.

Note that r_s^i is the vector of the residual norms at the ith iteration.
We suppose that

$$(\lambda_j^{(m_p)}, u_j^{(m_p)}) \text{ is better than } (\lambda_j^{(m_q)}, u_j^{(m_q)}) \text{ if } \rho_j^p \leq \rho_j^q. \tag{4.1}$$

Then an interesting updating strategy would be the choice of v^i as a function f^i of "the best" Ritz vectors:

$$v^i = f^i(U^{best}), \tag{4.2}$$

where $U^{best} = (u_1^{best}, \ldots, u_s^{best})$ and u_j^{best} is "the best" jth Ritz vector. The function f^i has to be chosen in order to force the vector v^i to be in the desired invariant subspace. A simple choice of f^i would be a linear combination of $u_1^{best}, \ldots, u_s^{best}$ vectors. In order to ensure that the BAA processes used in the MERAM algorithm produce ℓ different Krylov subspaces, $(f^i)_{i=1,\ldots,\ell}$ have to be chosen such that $f^i \neq f^j$ for $i \neq j$. Nevertheless, the vectors v^i and v^j (for $i \neq j$) of step 3 can be defined using the same restarting strategy. In other words, (4.2) may become

$$v^i = f(U^{best}), \tag{4.3}$$

where $f = f^i$ (for $i = 1, \ldots, \ell$). The vectors $v^1, \ldots, v^\ell$ would become identical and the Krylov subspaces of different processes would look extremely close to one another after a few iterations. Meanwhile, we will see in the next section that this restarting strategy can be useful for the asynchronous version of MERAM.

Clearly, the MERAM algorithm is equivalent to a particular use of several ERAMs. It allows the restarting vector v^i of an ERAM to be updated by taking into account interesting eigeninformation obtained by the other ERAMs. Another advantage of this algorithm is that it can be parallelized easily, as it contains coarse grain parallelism. In fact, each iteration of step 2 can be executed as an independent process. Suppose that we have ℓ processes and a process (i) is assigned to the iteration (i) of the inner loop of the algorithm. Thereafter, the processes have to synchronize at step 3. The only drawback of this parallelization is that the execution time of the algorithm is dominated by the execution time of the slowest process. The algorithm defined in the following section is a remedy for this problem.

4.1. Asynchronous MERAM. We suppose that after each BAA step, the process sends its output of interest to all other processes. The computation of a restarting vector v^i on process (i) can then be done by combining "the best" computed Ritz elements within that process. As a consequence, this parallel algorithm precisely a parallel version of the above MERAM algorithm where BAA processes are synchronized in step 3.

We refer to a dynamic algorithm as an algorithm whose one or more components are dynamic and to a dynamic component as a component whose run time behavior can depend on outside events. The parallel asynchronous MERAM can be considered a dynamic algorithm. To each run of this version corresponds one parallel static MERAM version in which the matrices U^{best} in the updating strategy (4.2) are different for every ERAM process. That means that the initial vector v^i can be thus defined by $v^i = f^i(U^{best(i)})$, where $U^{best(i)}$ denotes the matrix U^{best} of the ERAM process (i). Consequently, to ensure that MERAM does not produce the same Krylov subspaces, one must have $f^i \neq f^j$ and/or $U^{best(i)} \neq U^{best(j)}$ (for $i \neq j$). In other words, this way of asynchronizing communications between the BAA processes allows MERAM to produce ℓ [illegible] Krylov subspaces even with the restarting strategy (4.3).[2]

The asynchronous MERAM consists of ℓ "independent" ERAM processes which cooperate to update their restarting vectors. Let `Send_Eigen_Info` represents the task of sending eigeninformation of interest from an ERAM process to all other ERAM processes. Let `Receiv_Eigen_Info` be the task of receiving eigeninformation of interest from one or more ERAM processes by the current ERAM process. Finally, let `Rcv_Eigen_Info` be a boolean variable that is true if the current ERAM process has received eigeninformation from the other ERAM processes. An asynchronous parallel version of MERAM is explained in the following:

`Asynchronous MERAM Algorithm.`

---

1. `Start.` Choose a starting matrix V^ℓ and a set of subspace sizes $M = (m_1, \ldots, m_\ell)$.
2. `Iterate.` For $i = 1, \ldots, \ell$ do in parallel (`ERAM` process):
   - Computation process
     - `(a)` Compute a `BAA`$(input : A, s, m_i, v^i; output : r_s, \Lambda_{m_i}, U_{m_i})$ step.
     - `(b)` If $g(r_s^i) \leq tol$, stop all processes.
     - `(c)` If (`Rcv_Eigen_Info`), then `hybrid restart` else `simple restart`
   - Communication process
     - `(d)` `Send_Eigen_Info`
     - `(e)` `Receiv_Eigen_Info`

In this algorithm, each ERAM process has either a simple or a hybrid restarting strategy. A `simple restart` is a classical restarting strategy taking into account the eigeninformation computed by the current ERAM process. A `hybrid restart` is a restarting strategy taking into account the eigeninformation received by the other ERAM processes also. A hybrid restart includes data from other processes when eigeninformation has been received, while the simple restart uses just the eigeninformation computed by the current process. Let $u_j^{(m_k)}$ be the jth Ritz vector computed by the kth process (corresponding to the one with the subspace size m_k) and received by the current process. The matrix U^{best} of (4.2) is then defined by $(u_1^{best}, \ldots, u_s^{best})$, where u_j^{best} is "the best" of $u_j^{(m_k)}$ and the jth Ritz vector computed by the current process, where $k \in F \subset [1, \ell]$, cardinal $(F) = np$, and np is the number of processes that sent their eigeninformation to the current process. When a new and "better" Ritz pair is received from a process, its older value is overwritten by the new value. Moreover,

[2]With $m_i \neq m_j$ (for $i \neq j$) on homogeneous resources.

when the initial vector is updated by the hybrid restart strategy (i.e., step 2(c)) the current ERAM process changes the flag `Rcv_Eigen_Info` to false. Once a new Ritz pair is received from a process, this flag changes to true by the current ERAM process. We notice that in addition to the coarse grain parallelism between ℓ ERAM processes, one can also overlap communication steps with computations. More details about the parallelization of this algorithm are given in section 5.

A consequence of the dynamicity of this algorithm is its large sensitivity to the variations of the system on which it runs. Indeed, the results of this algorithm are a function of architectural parameters such as the load of the interconnection network or that of the processors which constitute the system. One important property of the asynchronous MERAM algorithm is its fault tolerance. That means that the loss of an ERAM process at run time doesn't impede the other ERAM processes from continuing to run and finish the algorithm. Another important property of this algorithm is its capacity of dynamic load balancing. Indeed, the time and space complexities of an iteration of an ERAM process of MERAM is a function of its subspace size. Because these complexities are known, we can dynamically allocate ERAM processes according to the system work load. All of these properties make the asynchronous MERAM a well-suited algorithm to the GRID computational environments.

4.2. Restarting strategies—convergence. The restarting strategy is a critical part of both ERAM and MERAM. Saad [20] proposed to restart the iteration of ERAM with a preconditioning vector to force it to be in the desired invariant subspace. It concerns a polynomial preconditioning applied to the starting vector of ERAM. The aim of the preconditioning is to make sure that the components of the restarting vector are nonzero in the desired invariant subspace and zero in the unwanted invariant subspace:

$$v(k) = p(A)v, \tag{4.4}$$

where $v(k)$ is the kth update of the starting vector v of ERAM, $v = v(0)$ is the starting vector, and p is a member of the space of polynomials of degree $< m$. One can define p as a Chebyshev polynomial determined from some knowledge about the distribution of A eigenvalues. This restarting strategy is very efficient in speeding the ERAM convergence (for more details, see [20, 19]). Another way of defining the polynomial p is to compute the restarting vector with a linear combination of s desired Ritz vectors:

$$v(k) = \sum_{i=1}^{s} \alpha_i u_i^{(m)}(k), \tag{4.5}$$

where $u_i^{(m)}(k)$ denotes the ith Ritz vector computed at the iteration k. This restarting strategy is polynomial because $u_i^{(m)}(k) \in \mathbb{K}_{m,v(k-1)}$ implies $u_i^{(m)}(k) = \phi_i(A)v$ for some polynomial ϕ_i and thus $v(k) = \sum_{i=1}^{s} \alpha_i \phi_i(A)v$. There are several ways to choose the scalar values α_i in (4.5). One choice can be α_i equal to the ith residual norm. Other choices can be $\alpha_i = 1$, $\alpha_i = i$, or $\alpha_i = s - i + 1$ for $1 \le i \le s$ (see [21] for more details).

Note that if $v = \sum_{j=1}^{n} \gamma_j u_j$, IRAM with exact shifts [23, 11] provides a specific selection of expansion coefficients γ_j for a new starting vector as a linear combination of the current Ritz vectors for desired eigenvectors. Implicit restarting provides a means to extract eigeninformation of interest from large Krylov subspaces while avoiding the storage and numerical difficulties. This is done by continually compressing eigeninformation of interest into an s-dimensional subspace of fixed size. This means that IRAM continues a BAA step, having kept all Ritz vectors of interest.

To define a restarting strategy for MERAM we have to define the functions f^i introduced in (4.2). These functions with several variables have to be used as input for MERAM. Their variables are calculated dynamically in the case of the asynchronous MERAM. These functions constitute a very important part of this algorithm. Here again we have several possibilities of choosing f^i. The first possibility is to choose them as simple as possible by using the same restarting techniques proposed for ERAM. The second possibility can be defined by using sophisticated restarting techniques such as augmented Krylov or implicit restarting. In the rest of this paper, we use a simple restarting technique based on the one defined in (4.5) and we call it the [illegible]. According to hypothesis (4.1) this hybrid restarting strategy is defined by

$$v^i = f^i(U^{best}) = \sum_{j=1}^{s} \alpha_j^i u_j^{best}, \tag{4.6}$$

where α_j^i (for $j = 1, s$ and $i = 1, \ell$) are some scalar values. The hybrid restarting strategy corresponding to (4.3) is then

$$v^i = f(U^{best}) = \sum_{j=1}^{s} \alpha_j u_j^{best}, \tag{4.7}$$

where $\alpha_j = \alpha_j^1 = \cdots = \alpha_j^\ell$.

We can view $(\ell - 1)$ ERAM processes of MERAM as the convergence accelerators of the last process. We implemented the parallel MERAM with the restarting strategy (4.7). Our experiments in section 6 illustrate that this ERAM process converges quicker than an ERAM with the same subspace size and starting vector. In other words, our experiments show that the restarting vector computed with the hybrid restarting technique is more in the desired invariant subspace than the one computed with (4.5).

4.3. Stopping criterion. The residual norms ρ_i defined in the previous section verify the well-known equation [19]

$$\rho_i = \|(A - \lambda_i^{(m)} I) u_i^{(m)}\|_2 = h_{m+1,m} |e_m^H y_i^{(m)}|, \tag{4.8}$$

where e_m is the mth vector of the canonical basis of $\mathbb{C}^m$. This enables the computation of the residual norms at low cost because the ith residual norm is equal to the product of the last component of the eigenvector $y_i^{(m)}$ by $h_{m+1,m}$. However, in practice the residual norms ρ_i can be more indicative of current errors.

To stop the restarts in ERAM algorithm one has to define the function g given in step 3. Some typical examples are

$$g(r_s) = \|r_s\|_\infty \tag{4.9}$$

or a linear combination of the residual norms

$$g(r_s) = \sum_{j=1}^{s} \alpha_j \rho_j, \tag{4.10}$$

where α_j are scalar values.[3] The same technique applied to $r_s^i = (\rho_1^i, \ldots, \rho_s^i)$ can be used to define the stopping criterion of restarts—the function g in step 2(b)—of MERAM.

4.4. Relationship with Explicitly Restarted Block Arnoldi method. As shown in section 2, the [illegible] allow the approximation of an eigenpair (λ, u) of A by a $(\lambda^{(m)}, u^{(m)})$ solution of (2.2) with $\mathbb{K}_{m,X}$ an $m \times \mu$-dimensional subspace defined by (2.1), where $X = \mathrm{Span}(x_1, \ldots, x_\mu)$ and $\mu > 1$. We assume that the vectors $x_1, \ldots, x_\mu$ are linearly independent.

The Explicitly Restarted Block Arnoldi Method (ERBAM) is used to extract the eigenvalues whose algebraic multiplicity is less than or equal to the block size [9]. Let X_1 be the orthonormal matrix represented by the columns $x_1, \ldots, x_\mu$. The block Arnoldi process generates a set of $X_1, \ldots, X_m$ matrices. Let $\boldsymbol{W}_m$ be the $(n, m \times \mu)$-sized matrix $[X_1, \ldots, X_m]$. The $m \times \mu$ columns of this matrix constitute an orthogonal basis of Krylov subspace $\mathbb{K}_{m,X_1}$:

`Block Arnoldi Reduction :` $\mathtt{BAR}(input : A, m, X_1; output : \boldsymbol{H}_m, \boldsymbol{W}_m)$.

For $j = 1, 2, \ldots, m$ do:

- $H_{i,j} = X_i^H A X_j$, for $i = 1, 2, \ldots, j$.
- $\bar{X}_j = AX_j - \sum_{i=1}^{j} X_i H_{i,j}$.
- Compute QR decomposition of $\bar{X}_j$:
  $\bar{X}_j = Q_j R_j = X_{j+1} H_{j+1,j}$.

The $H_{i,j}$ matrices generated by the BAR process constitute the blocks of an upper block Hessenberg matrix $\boldsymbol{H}_m$. The eigenvalues of this matrix approach the corresponding eigenvalues of A. Then, an equation equivalent to (3.1) will be

$$A\boldsymbol{W}_m = \boldsymbol{W}_m \boldsymbol{H}_m + F_m \boldsymbol{E}_m^H, \tag{4.11}$$

where $F_m = X_{m+1} H_{m+1,m}$ and $\boldsymbol{E}_m$ is the matrix of the last μ columns of $I_{n\mu}$. The block version of BAA is unchanged, except for the first step where we calculate $\mathrm{BAR}(input : A, m, X_1; output : \boldsymbol{H}_m, \boldsymbol{W}_m)$ instead of $\mathrm{AR}(input : A, m, v; output : H_m, W_m)$. We call this algorithm $\mathrm{BBAA}(input : A, s, m, X_1; output : r_s, \Lambda_{m\times\mu}, U_{m\times\mu})$.

Even though both ERBAM and MERAM begin with a set of starting vectors, they are not equivalent. Suppose that $\ell = \mu$ is the number of starting vectors for both methods. $m \times \ell$ is the projection subspace size in ERBAM, and m is the size of each projection subspace in MERAM, which means $m_1 = \cdots = m_\ell = m$. The difference between these two methods is that the projection subspace in ERBAM is $\mathbb{K}_{m,X}$ with $X = \mathrm{Span}(x_1, \ldots, x_\ell)$, while in MERAM the problem is projected onto ℓ subspaces $\mathbb{K}_{m,x_1}, \ldots, \mathbb{K}_{m,x_\ell}$. Now, assuming that the degree of the minimal polynomial of X_1 is greater than m, these Krylov subspaces satisfy the following relation:

$$\mathbb{K}_{m,X} = \mathbb{K}_{m,x_1} \oplus \cdots \oplus \mathbb{K}_{m,x_\ell}. \tag{4.12}$$

Therefore, if $\dim(\mathbb{K}_{m,X}) = m \times \ell$ we have $\dim(\mathbb{K}_{m,x_i}) = m$ for all $i \in [1, \ell]$ and $\dim(X) = \ell$.

Suppose that $\dim(\mathbb{K}_{m,X}) = m \times \ell$. The construction of an orthogonal basis $((x_1^1, \ldots, x_\ell^1), \ldots, (x_1^m, \ldots, x_\ell^m))$ of $\mathbb{K}_{m,X_1}$ allows the construction of an orthogonal

[3] If $\alpha_j = 1$ (for $j = 1, \ldots, s$), then (4.10) defines the sum of the residual norms associated with s desired eigenelements.

basis $\tilde{X}_i = (x_i^1, \ldots, x_i^m)$ for each subspace $\mathbb{K}_{m,x_i}$, $1 \leq i \leq \ell$. The inverse is true only if $\tilde{X}_i$ is orthogonal to $\tilde{X}_j$ for all $i, j \in [1, \ell]$ and $i \neq j$. This means that the block Arnoldi reduction, which is also a Gram–Schmidt orthogonalization of Krylov vectors of $\mathbb{K}_{m,X_1}$, defines ℓ Arnoldi reductions, which again are ℓ Gram–Schmidt orthogonalizations of Krylov vectors of subspaces $\mathbb{K}_{m,x_i}$ $(1 \leq i \leq \ell)$, but the inverse is not always true.

In the first method, s desired eigenvalues are approximated by the eigenvalues of an $(\ell \times m)$-sized block Hessenberg matrix, while in the second, they are approximated by s "best" eigenvalues of ℓ Hessenberg matrices of order m. We also notice that in ERBAM the relation $m \geq s/\ell$ has to be true, while in MERAM the relation $m \geq s$ must be true.

Some parallel and numerical properties of MERAM and ERBAM are discussed in the next two sections.

5. Parallelism analysis.

5.1. Parallel ERAM. Let us consider the Basic Arnoldi algorithm—BAA($input$: $A, s, m, v; output : r_s, \Lambda_m, U_m$)—which is the core of ERAM. This algorithm can be considered as four main tasks corresponding to its four steps. The projection phase of the algorithm manipulates the n-sized data sets. The second phase acts on m-sized data sets, and the third and fourth phases deal with the n- and m-sized data sets.[4] Steps 1, 3, and 4 constitute the expensive parts of this algorithm. They are composed mainly of sparse matrix-vector multiplications and triadic and reduction operations.

The global programming model used is message passing, which is crucial for distributed architectures. This model assumes that the system has a number of processors that have local memories and communicate with each other by means of memory transfer from one processor to another [5]. A classical way to parallelize BAA using the message passing model is to distribute the large arrays (vectors and/or matrices) among the processors while the small arrays are kept on just one processor or redundantly on all processors. This is equivalent to distributing each of the tasks 1, 3, and 4 among the processors and to run step 2 on just one processor or simultaneously on all processors.

In the context of the message passing programming model, the projection phase of BAA is highly parallel. The second task can represent a good vector structure[5], and the last two tasks are also parallel. These different types of parallelism can be exploited efficiently by a distributed memory platform formed, for example, by a distributed memory parallel machine and a vector computer or by a distributed memory parallel machine and a sequential machine. We must mention that we do not have to use the message passing model for step 2, as this task would become a bottleneck in the algorithm [17]. This means, in terms of implementation, that the architecture used for step 2 ought to be different from the one used for steps 1, 3, and 4 (see Figure 1).

In the ERAM algorithm the size of the subspace determines the amount of time necessary to execute each restart (mainly a BAA step). The cost of each restart increases with the subspace size, while in general the number of restarts decreases according to the subspace size. We need to find a balance between the execution time of each restart and the number of restarts to minimize the total execution time [17, 6].

[4] This is not true for the fourth step if the right-hand side of (4.8) is used to compute the vector r_s.

[5] If we use a classical method like QR and/or inverse iteration to solve this step.

Generally, the larger the subspace size, the better the convergence properties, so the subspace size m ought to be chosen as large as possible given the limitations of the target system resources.

5.2. Parallel MERAM. MERAM consists of several ERAMs interacting with some protocol. As we mentioned in section 4, a great advantage of MERAM is its coarse grain parallelism. We consider here the asynchronous parallel MERAM algorithm described in section 4.1. In this algorithm, each ERAM core—BAA($input$: $A, s, m, v; output$: r_s, Λ_m, U_m)—can be executed [illegible] from the others. Consequently, we have ℓ independent coarse grain tasks (ERAM processes). For each of them, one can make use of the above described parallelism.

In addition to the coarse grain parallelism between ℓ ERAM processes, one can also overlap communication steps with computations. The processors would always be active and execute a nonredundant task. Therefore, a significant reduction of the response time of this algorithm is expected. In the next section, we will confirm this with our experiments.

5.3. Parallel ERBAM. Clearly, the parallelization method described in section 5.1 is also valid for ERBAM (i.e., BBAA in section 4.4). However, if the subspace sizes of ERAM and ERBAM are equal to $\mathcal{M} = m \times \mu$ with $\mu > 1$, the coarse grain parallelism in each BBAA step is larger than in the corresponding BAA step. This is due to the BAA core (i.e., AR) computing $\mathcal{M} = m \times \mu$ matrix-vector products with A, while in the BBAA core (i.e., BAR) the computations of A on μ vectors can be done at once, in roughly the same cost as computing a single matrix-vector product. The AR has to compute $\mathcal{M}$ second-level BLAS operations, while BAR has to compute m third-level BLAS operations. Consequently, the computational cost of ERBAM should be less than that of ERAM.

The global programming model used to implement ERBAM and MERAM is message passing. Bearing in mind that large arrays are distributed among the processors and small arrays are kept on one processor or replicated on all processors, this is equivalent to distributing each of the tasks 1, 3, and 4 of BBAA among the processors and running step 2 on just one processor or redundantly on all processors.

Consider a MERAM with (μ) ERAM processes having the subspace sizes m_i with $\sum_{i=1}^{\mu} m_i = \mathcal{M}$ and $m_1 \leq \cdots \leq m_\mu$. The goal is to determine if this MERAM can be competitive with ERBAM with subspace size $\mathcal{M} = m \times \mu$. In other words, we would like to know if ERAM(m_μ) helped by the other ERAM processes of MERAM (i.e., ERAM(m_i) for $i = 1, \ldots, \mu - 1$) can be competitive with ERBAM.

6. Numerical experiments. Data-parallel and vector programming models are very well suited for ERAM [17]. To efficiently exploit the natural parallelism of MERAM, one has to use the coarse grain parallel model, where a number of ERAM processes are running concurrently. In addition, one can exploit the fine grain parallelism within each task (ERAM process) in order to maximize the resource usage of a metacomputing platform. A metacomputer consists of heterogeneous and autonomous computers linked by an interconnection network. This architecture has a tremendous amount of memory and computational power [10] to cope with large industrial applications. However, in practice, the number of parallel machines involved in a metacomputer is limited. We present, in this section, some results of our experiments on homogenous and heterogeneous programming environments.

We denote by ERAM(m, v) (resp., IRAM(m, v)) an ERAM (resp., IRAM) process with the subspace size m and the initial vector v, by ERBAM(m, V^ℓ) an ERBAM

process with the subspace size m and the initial matrix V^ℓ, and by MERAM(M, V^ℓ) a MERAM with ℓ ERAM$(m_i, v^i)_{i=1,\ell}$ processes, where $M = (m_1, \ldots, m_\ell)$ and $V^\ell = [v^1, \ldots, v^\ell]$. The matrix I_k denotes the one whose columns are $e_1, \ldots, e_k$.

6.1. MERAM and ERAM.

We implemented ERAM and MERAM according to the above programming models on different heterogeneous environments [8]. These environments were a 32-node Connection Machine (CM5), a CM5 and a Sparc 20, a CM5 and two Sparc 20's, and a CM5 and a 4K Connection Machine 200 (CM200). The CM5 is a distributed memory MIMD[6] architecture [27]. Each node of CM5 consists of a Sparc processor, a 32 megabyte memory, and four floating point computation units operating at 32 megaflops each. The CM5 supports the message passing model, the data-parallel model, or a combination of the two. The CM5 nodes are connected by three networks. is a point-to-point communication and has 0.5 gigabyte per second global bandwidth. is a binary tree for global operations (as diffusion, reduction, etc.) by pipelining the messages, and is dedicated to error messages. The CM200 is a massively parallel SIMD[7] machine consisting of thousands of bit serial data processors and a front-end for management and control [26]. The processors and memory are packaged as 16 in a chip. Each chip also contains the routing circuitry which allows any processor to send and receive messages from any other processor in the system. The system version used has 0.5 gigabyte memory and operates at 10 megahertz. The CM200 supports the data-parallel programming model. Both CM5 and CM200 have a workstation as front-end. The programming language used on CM5 and CM200 is CM Fortran [28]. This is an extension of Fortran 77 allowing us to handle parallel variables distributed on the processors.

In [7], we showed that a significant acceleration can be achieved when the heterogeneous architecture consists of a CM5 and two Sparc workstations. In this paper, we present the results of our experiments on a CM5 for ERAM and on a heterogeneous environment consisting of a CM5 and a CM200 interconnected by a 10 Mbit/s Ethernet link for MERAM.

As we have seen in the previous section, the BA algorithm can be divided into four main tasks: the projection, the computation in the subspace, the Ritz vectors computation, and the computation of the vector of the residual norms. These tasks can run on different machines. The projection task, the Ritz vectors computation and the computation of the residual norms are highly data-parallel. They will be mapped on a parallel machine. The computation in the subspace, on the other hand, is suited for vector programming. It will be mapped onto the front-end (i.e., control processor) of the parallel machine. Figure 1 presents this implementation of ERAM on a distributed architecture consisting of a parallel architecture and a vector/sequential machine. In our experiments the parallel machine is CM5 or CM200, while the other machine is the corresponding front-end.

Figure 2 describes the parallel MERAM algorithm for $\ell = 2$ on a heterogeneous architecture. TCMa and TCMb (resp., Tcomma and Tcommb) are computation (resp., communication) tasks of the algorithm. TCMa (resp., TCMb) communicates with Tcomma (resp., Tcommb) using the shared memory model. Tcomma and Tcommb are connected by a network. These communication tasks are designed to overlap

[6]Multiple instruction multiple data.
[7]Single instruction multiple data.

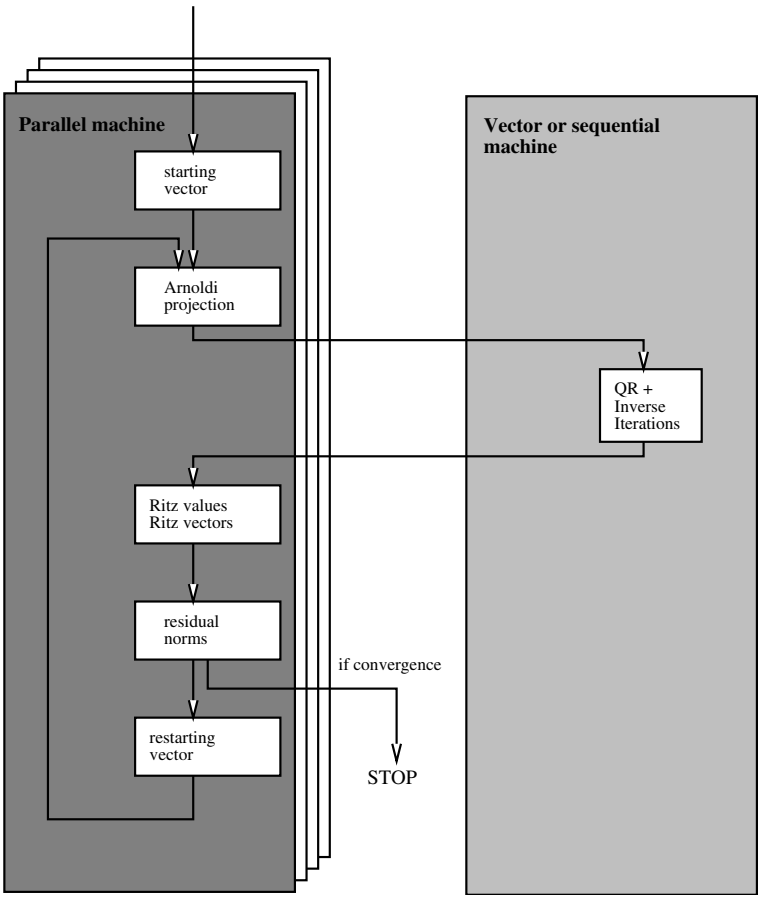


FIG. 1. *ERAM on a distributed architecture.*

communication with computation phases between parallel machines. TCMa (resp., TCMb) implements an ERAM process. After each iteration, if the convergence is not reached, the eigeninformation is stored in the shared memory (SM) and sent by one communication task to the other process. TCMa and TCMb are implemented on a CM5 and a CM200 interconnected by a 10 Mbit/s Ethernet link. The subspace computations of TCMa and TCMb are mapped onto the CM5 front-end and the CM200 front-end, respectively. PVM library is used to implement the communications between Tcomma and Tcommb.

The sparse matrices for our experiments have a C-diagonal pattern. The only nonzero elements of a C-diagonal matrix are on the main diagonal and the diagonals immediately around it. C is the maximum number of nonzero elements per row or per column. These sparse matrices have some interesting properties. They are easy to build and well suited for the sparse general pattern (SGP) data-parallel storage format [18]. It has also been shown in [8] that matrices with a C-diagonal pattern have good performance in sparse matrix-vector multiplication algorithms executed on data-parallel machines. The nonzero upper-diagonal and lower-diagonal elements of our nonsymmetric matrices are chosen in $[-1, +1]$ range, while the nonzero diagonal elements are in $[0, C]$. To study the convergence properties of MERAM and

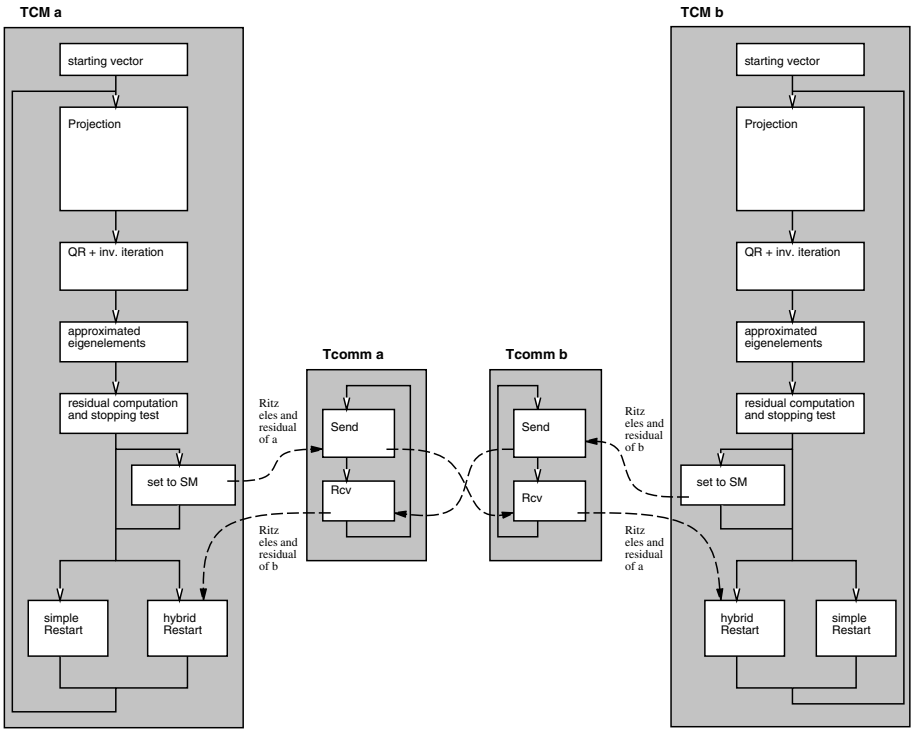


FIG. 2. *The asynchronous MERAM in a heterogeneous environment.*

to compare it with ERAM, we intentionally limited the sizes of our test matrices. The size of the test matrices is fixed to 1024.

The execution of ERAM on CM5 is done with the starting vector equal to v, the subspace size m, and the restarting strategy (4.7) with $\alpha_i = 1$ (for $i = 1, s$). On the other hand, two starting vectors (v^1, v^2), two subspace sizes (m_1, m_2), and two restarting strategies are needed to execute MERAM on both CM5 and CM200. We suppose that we have the same parameters on the CM5 for ERAM and for MERAM. This means that $v^1 = v$, $m_1 = m$, and the restarting strategies for ERAM and MERAM will be the same if the restart of MERAM is simple on the CM5. In the case of hybrid restart, we have two sets of eigeninformation $E_{m_1} = (\Lambda_{m_1}, U_{m_1}, r_s^1)$ and $E_{m_2} = (\Lambda_{m_2}, U_{m_2}, r_s^2)$. We choose "the best" to form a new set of eigeninformation: $E_{best} = (\Lambda^{best}, U^{best}, r_s^{best})$. Then, with these s Ritz vectors, we compute the restarting vector by (4.7) with $\alpha_i = 1$ (for $i = 1, s$). The speedup, S, is calculated as the ratio of the time needed to solve the problem with MERAM (on CM5 and CM200) to the time needed to resolve the same problem with ERAM (on CM5 alone). The execution time of MERAM is that of the converged node (the fastest node which is CM5).

We want to find $s = 4$ algebraically largest eigenvalues and their corresponding eigenvectors. The tolerance value is $tol = 5E - 10$.

We expect that MERAM converges faster than ERAM. In fact, for the worst case scenario MERAM behaves like ERAM applied to the fastest processor (i.e., CM5). This is the case when no other processes yield information that is better than the original ERAM process. In other words, since the CM5 is faster than the CM200, the

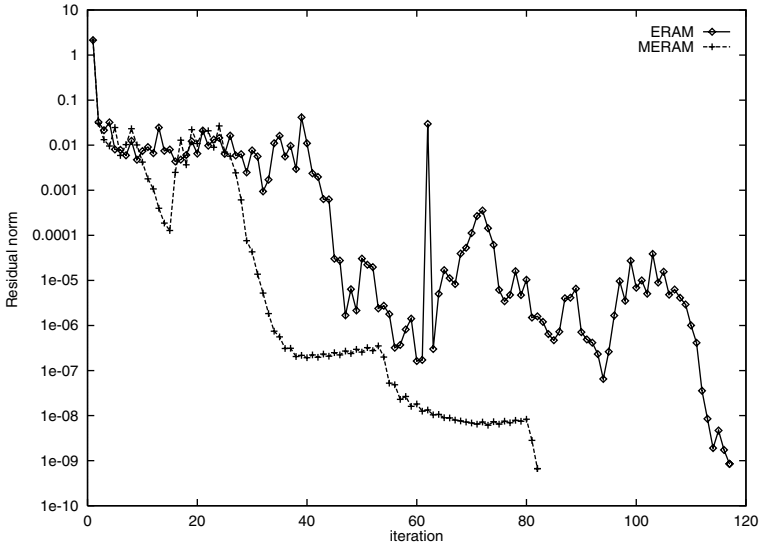


FIG. 3. *MERAM*$((28, 15), [z, \mathrm{r}])$ *versus ERAM*$(28, z)$ *with* 21*-diagonal matrix.*

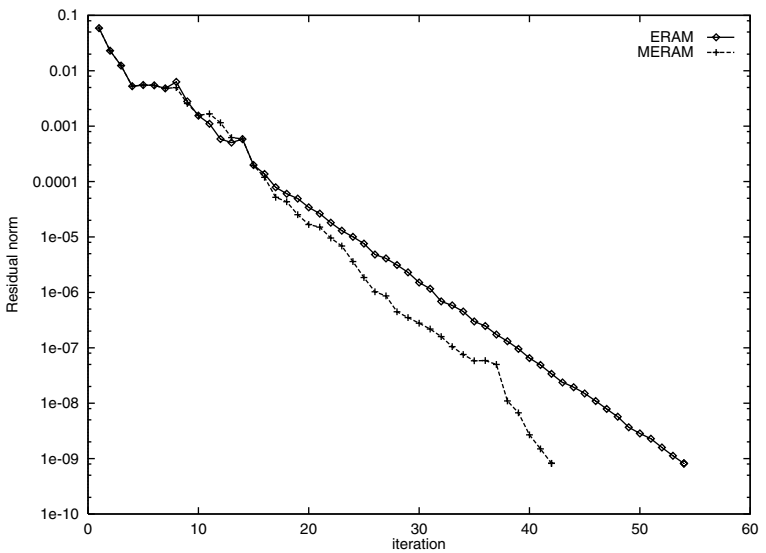


FIG. 4. *MERAM*$((26, 26), [z, \mathrm{r}])$ *versus ERAM*$(26, z)$ *with* 63*-diagonal matrix.*

CM200 ERAM process can be viewed as an accelerator for the CM5 ERAM process. Consequently, the number of iterations (restarts) of MERAM on the CM5 ought to be less than the number of iterations of ERAM on this machine. By the number of iterations of MERAM, we mean the number of iterations of its process on the CM5.

Figures 3–7 illustrate the sum of the residual norms (SRN)[8] (4.10) with $\alpha_i = 1$

[8]SRN is a stronger convergence measure than the residual of each Ritz vector alone. This is because SRN bounds the subspace residual : $\|AU_m - U_m\Lambda_m\|_2 \leq SRN = \sum_{i=1}^{s} \|Au_i^{(m)} - \lambda_i^{(m)} u_i^{(m)}\|_2$.

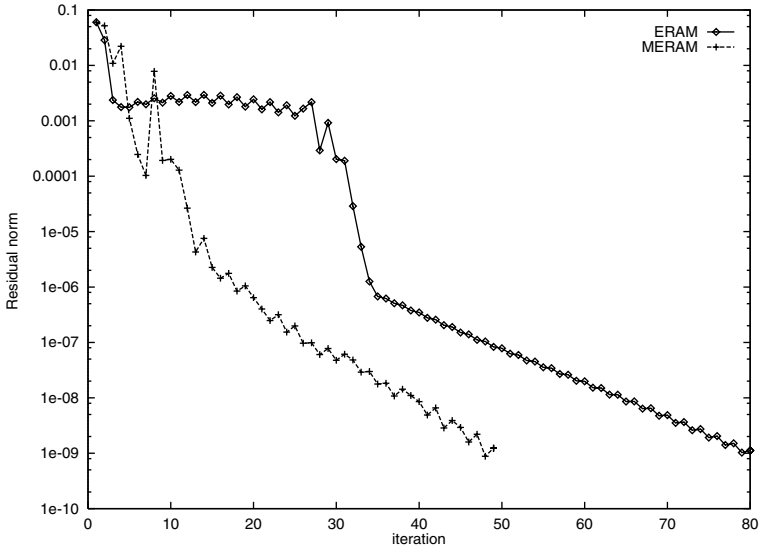


FIG. 5. *MERAM*$((32, 32), [z, \mathrm{r}])$ *versus ERAM*$(32, z)$ *with* 21*-diagonal matrix.*

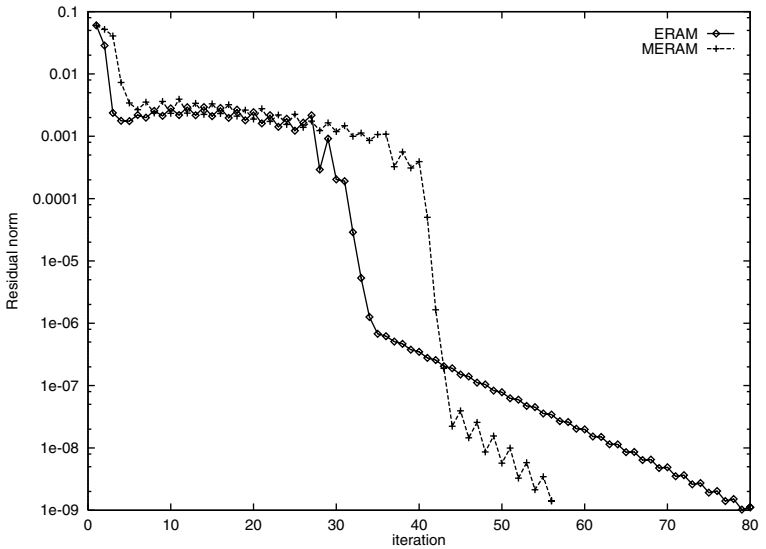


FIG. 6. *MERAM*$((32, 20), [z, z])$ *versus ERAM*$(32, z)$ *with* 21*-diagonal matrix.*

for $i = 1, s$, with respect to the number of restarts for ERAM and MERAM for C-diagonal matrices. They present the number of iterations required by ERAM and MERAM to converge. Table 1 brings together the parameters and the curves of performances presented in Figures 3–7. This table and all of these figures illustrate the influence of the CM200 eigeninformation on the improvement of the restarting vector, and consequently on the convergence of the ERAM process of MERAM on the CM5. We notice that the oscillations of the convergence curves of MERAM are qualitatively and quantitatively less important than the ones of ERAM. According to our experiments, MERAM always converges more rapidly than ERAM to the desired

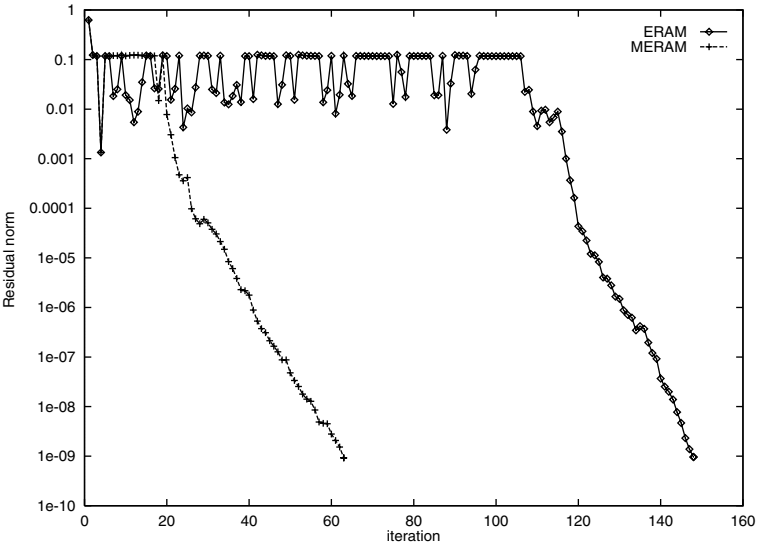


FIG. 7. *MERAM*$((40,40),[z,z])$ *versus ERAM*$(40,z)$ *with* 63*-diagonal matrix.*

TABLE 1
The vector z *is defined by* $z = \frac{1}{\sqrt{n}}(1,\ldots,1)^t$*, and* r *is a normalized random vector.*

		MERAM			ERAM			
Fig.	C	m_1, m_2	v^1, v^2	iteration	$m = m_1$	$v = v^1$	iteration	S
3	21	28,15	z, r	80	28	z	120	0.80
4	63	26,26	z, r	42	26	z	54	0.84
5	21	32,32	z, r	50	32	z	80	0.72
6	21	32,20	z, z	56	32	z	80	0.81
7	63	40,40	z, z	63	40	z	148	0.51

invariant subspace. These results also show that we can obtain a good acceleration of convergence compared to ERAM.

In Figure 6 there is a section of convergence history where ERAM is better than MERAM. This seems to be due to the distribution of the eigenvalues. Indeed, the s Ritz values computed by two ERAM processes with subspace sizes m_1 and m_2 do not always correspond to s wanted eigenvalues. Suppose, for example, that the wanted eigenvalues are $(s = 4)$ algebraically the largest: $\lambda_1 > \lambda_2 > \lambda_3 > \lambda_4 > \lambda_5 > \cdots$ and λ_4, λ_5 are very close to each other. At some iteration, the ERAM process with m_1 can compute Ritz values corresponding to λ_1, λ_2, λ_3, and λ_4, while the ERAM process with m_2 can approach Ritz values corresponding to λ_1, λ_2, λ_3, and λ_5. Now, if the residual norm of the Ritz vector corresponding to λ_5 is less than the one corresponding to λ_4, then the Ritz vector corresponding to λ_5 will be chosen as "the best." Thus, hybrid restart takes into account a "bad" information to update the restarting vector in MERAM. Consequently, the restarting vector moves away from the wanted invariant subspace. An approach to remedy this issue seems to be to use a mixture of all Ritz vectors instead of the best vector. The hybrid restart strategy should make use of all Ritz vectors, and the whole computed information will be taken into account.

6.2. MERAM and other methods. To evaluate MERAM with respect to some other methods, we compared it with ERBAM and IRAM under similar con-

straints. The goal is to know which of the two schemes, MERAM and ERBAM (respectively, MERAM and IRAM), is the fastest given the same computational resources.

Numerical experiments were done on several Sun Ultra Sparc workstations. They were linked by 10 Mbit/s and 100 Mbit/s Ethernet networks for the experimentation of sections 6.2.1 and 6.2.2, respectively. We made use of two or three Sparc Ultra-1 workstations for the experiments of section 6.2.1. A maximum of six workstations—three Sparc Ultra-1 and three Sparc Ultra-5-10—were used for the experiments of section 6.2.2. All of these workstations function with the Sun operating system.

We made use of the LAKe and MPI message passing libraries to implement ERBAM and MERAM. LAKe (`Linear Algebra Kernel`) is a linear algebra class library implemented in C++ that uses MPI through OOMPI [16] for the parallel classes [14, 15]. We used the PARPACK package [24] to run a parallel version of IRAM. PARPACK is a parallel version of the ARPACK software (a package of Fortran 77 subroutines) which implements IRAM. This package is targeted for distributed memory message passing systems. The message passing layers supported are BLACS and MPI. The reverse communication interface of PARPACK allows a simplified SPMD[9] parallelization strategy.

The matrices are taken from MatrixMarket [2], the University of Florida sparse matrix collection[10], and the SPARSKIT package [22]. The matrices $PDE2961$ and $PDE490000$ are from partial differential equations (PDE) and are created by the `MATPDE` matrix generator of MatrixMarket. The finite difference matrix $PDE248004$ is from a second-order elliptic operator and is produced by the generator `GENMAT` of SPARSKIT. Table 2 presents some characteristics of these matrices. The number of nonzero elements of a matrix A is denoted by NNZ and its Frobenius norm by $\|A\|_F$.

TABLE 2
List of matrices.

Collection	Matrix	Matrix size	NNZ	$\|A\|_F$
MatrixMarket	$AF23560$	23560	484256	$1.0E+4$
	$BFW782A$	782	5982	$1.3E+2$
	$PDE2961$	2961	14585	$2.2E+2$
	$PDE490000$	490000	2447200	$3.4E+3$
UF collection	$TWOTONE$	120750	1224224	$8.2E+5$
	$VENKAT01$	62424	1717792	$1.2E+2$
SPARSKIT	$PDE248004$	248004	1238028	$2.3E+3$

6.2.1. MERAM versus ERBAM. For our experiments we have exploited only very high level parallelism of asynchronous MERAM. We did not take advantage of other types of parallelism that can be found in each ERAM process of MERAM. This means that each ERAM process of MERAM runs in serial mode. The parallelization of ERBAM is done according to the scheme presented in section 5.3. Each of the tasks 1, 3, and 4 of BBAA is distributed between the workstations, and step 2 runs redundantly on all processors.

We used these methods in order to find the $s = 4$ eigenvalues of largest modulus. The initial vector v^i is taken as the vector e_i (for $i = 1, \ell$). Let $\mathcal{M} = \sum_{i=1}^{\ell} m_i$ be the

[9]Single program multiple data.
[10]http://www.cise.ufl.edu/research/sparse/matrices/.

TABLE 3
MERAM versus ERBAM on ℓ interconnected workstations.

			MERAM			ERBAM		
Matrix	Fig.	ℓ	$m_1, \ldots, m_\ell$	iteration	time	$\ell \times m$	iteration	time
AF23560	-	2	10, 30	6	554	2×20	9	487
	8	3	20, 25, 35	5	602	3×27	5	1053
	-	3	27, 27, 27	5	574	3×27	5	1053
BFW782	-	2	13, 37	10	35	2×25	8	36
	9	2	5, 45	4	20	2×25	8	36

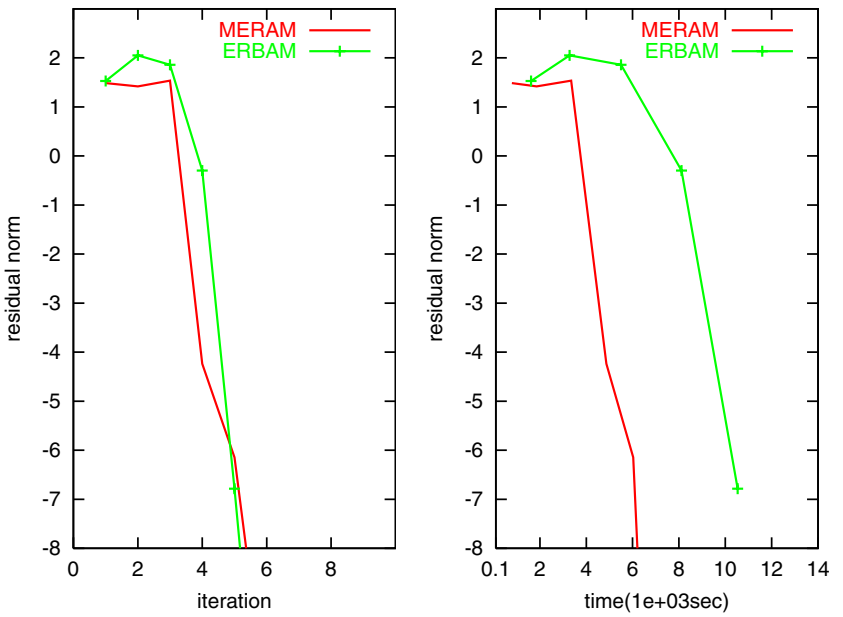


FIG. 8. *MERAM*$((20, 25, 35), I_3)$ *versus ERBAM*$(81, I_3)$ *with AF*23560 *matrix.*

sum of the subspace sizes of the ℓ ERAM processes of MERAM. The subspace size of ERBAM is chosen as $\ell \times m$, where m is defined by $m = \frac{\mathcal{M}}{\ell} + \mathrm{mod}(\ell, \mathrm{mod}(\mathcal{M}, \ell))$. The tolerance value is $tol = 1E-6$. The number of iterations of MERAM is the number of iterations of the ERAM process that reaches convergence (the other ERAM processes are stopped when the first one converges). It is, generally, the ERAM process with the largest subspace size.

We run experimentations with two or three ERAMs in parallel (i.e., $\ell = 2$ or 3). Table 3 brings together the input and output parameters of some comparisons between MERAM and ERBAM. Figures 8 and 9 illustrate Log_{10} of the sum of the residual norms versus the number of restarts and the execution time for ERBAM and MERAM on ℓ workstations. We notice in these figures and Table 3 an important reduction of MERAM response time with respect to ERBAM. The convergence of MERAM can be seen as the convergence of its ERAM process with the largest subspace size (i.e., ERAM(m_ℓ)). According to our experiments, this ERAM process, accelerated by the other ERAM processes of MERAM, is competitive with ERBAM. This is due to different factors: the small subspace size of ERAM (m_ℓ) in comparison to the subspace size of ERBAM (i.e., $\ell \times m$), the mixing of restarting strategies in MERAM, and the use of some additional resources in order to reduce ERAM (m_ℓ) process response time.

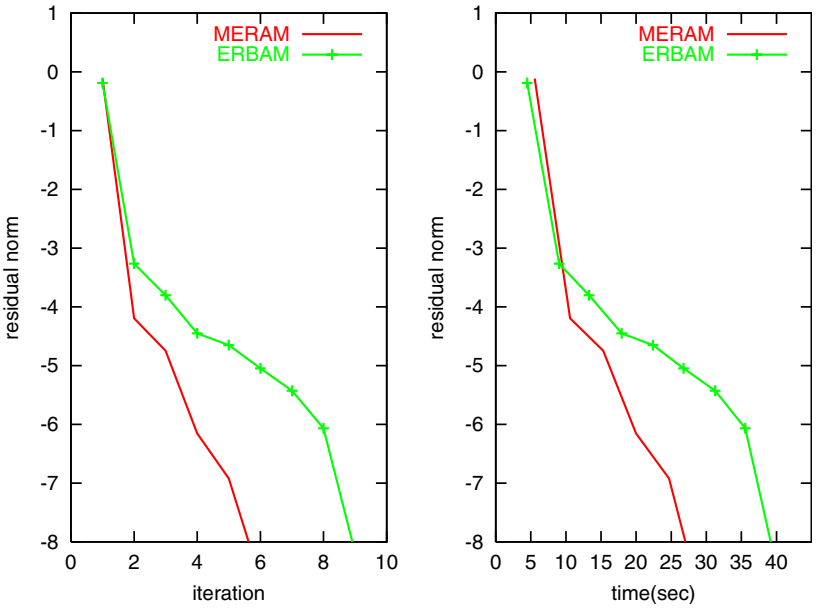


FIG. 9. *MERAM*$((5,45),I_2)$ *versus* *ERBAM*$(50,I_2)$ *with* *BFW*782*A matrix.*

6.2.2. MERAM versus IRAM. The Implicitly Restarted Arnoldi is a more robust method than ERAM. This is because of the powerful restarting strategy used in IRAM [23]. As we have seen before, MERAM improves the convergence of ERAM. This is because the restarting vector of an ERAM process of MERAM is computed by taking into account the eigeninformation obtained by the other ERAM processes. To evaluate the restarting strategy of MERAM with respect to IRAM, we have compared these methods under similar constraints.

The PARPACK package [24] implements a parallel version of IRAM according to the message passing model. We run experimentations with this package on a set of p interconnected workstations.

For our experiments, we have exploited the high level parallelism of asynchronous MERAM (i.e., that which is between its ERAM processes) as well as the parallelism inside each ERAM process. MERAM is implemented according to the model described in section 6.1. That is, we define ℓ computation tasks TCM_i and ℓ communication tasks Tcomm_i (for $i = 1, \ell$). Here TCM_i communicates with Tcomm_i using the message passing model. TCM_i implements the ith ERAM process of MERAM. After each iteration, if the convergence is not reached, the eigeninformation is sent to Tcomm_i which sends it in turn to the Tcomm_j tasks (for all $j \neq i$). Moreover, Tcomm_i receives the eigeninformation coming from Tcomm_j $(j \neq i)$ and communicates it to TCM_i.

MERAM is implemented on a set of p interconnected workstations. The TCM_i task is implemented on p_i workstations; $p_i = 1$ corresponds to a serial run. The subspace computation of TCM_i is replicated to all of the p_i workstations. The Tcomm_i task is implemented on a workstation. We then have $p = (\sum_{i=1}^{\ell} p_i) + \ell$. The MPI library is used to implement the communications between Tcomm_i (for $i = 1, \ell$).

We used these methods in order to find the s eigenvalues of largest modulus. We run experimentations with $p = 5$ or 6 workstations. The stopping criterion SRN is normalized by $\|A\|_F$ in both methods, and the tolerance value is $tol = 1.E-6$.

TABLE 4
MERAM versus IRAM on p workstations.

				MERAM			IRAM		
Matrix	Fig.	p, p_1, p_2	s	$m_1, m_{\ell=2}$	iteration	time	m	iteration	time
AF23560	-	6, 3, 1	2	13, 23	3	119	23	8	118
BFW782	10	5, 2, 1	2	10, 20	6	17	20	23	19
VENKAT01	-	6, 3, 1	2	10, 30	3	399	30	3	161
TWOTONE	-	6, 3, 1	4	15, 25	3	550	25	4	354
PDE2961	12	5, 2, 1	2	15, 20	5	17	20	10	21
PDE248004	11	6, 3, 1	4	20, 35	20	8.87E+03	35	27	4.62E+03
PDE490000	13	5, 2, 1	4	55, 30	2	1.25E+04	55	30	4.07E+04

Table 4 brings together the input and output parameters of some comparisons between MERAM and PARPACK. For some of them, Figures 10–13 illustrate graphically Log_{10} of the normalized sum of the residual norms versus the number of restarts and the execution time for IRAM and MERAM. We notice in these figures and in Table 4 that the number of restarts of MERAM is always smaller than the number of restarts of PARPACK. On the other hand, the PARPACK response time is often faster than that of MERAM. This is due especially to the too heavy load of the workstations supporting the ERAM processes of MERAM. Let us recall that we dedicated $\ell = 2$ workstations to the communication processes Tcomm_1 and Tcomm_2. Consequently, the m_1 and m_2 matrix-vector multiplications of the first and the second ERAM processes of MERAM are distributed, respectively, on p_1 and p_2 workstations, while $(m_1 - s)$ matrix-vector multiplications of PARPACK are distributed on $p = p_1 + p_2 + \ell$ workstations. That means that each ERAM process runs in parallel on 2 or 3 processors or even in serial mode while IRAM runs on 5 or 6 processors. The load of the processors supporting IRAM is lighter than that of the one supporting ERAM processes of MERAM. Despite this unfavorable context for MERAM, the performances presented by Figures 10, 12, and 13 show that, for the corresponding matrices and parameters, MERAM has better convergence properties than IRAM.

7. Conclusion. We have presented the MERAM and some of its characteristics: the restarting strategy and its relationship to the block version of ERAM. We also presented an asynchronous parallel version of this method and performance results of its implementation on heterogeneous and homogeneous environments. MERAM allows us to restart each of its ERAM processes with a vector preconditioning so that it can be forced in the desired direction. This preconditioning can be seen as a particular application of an ERAM process to the starting vector of the other ones.

The restarting strategy is a critical part of MERAM. For an ERAM process of this method, a hybrid restart is defined to be a restarting strategy that takes into account all Ritz vectors including those received from the other ERAM processes. Thus, the restarting vector of this ERAM process is a linear combination of "the best" Ritz vectors of interest. We defined "the best" Ritz vector to be the vector that has the smallest residual norm.

Asynchronous MERAM is characterized by the fact that the communications between its subtasks (i.e., ERAM processes) are totally asynchronous. As a consequence, this parallel algorithm is not the parallel version of a unique serial one. [illegible] of this dynamic algorithm corresponds to one parallel static version. MERAM is a fault tolerant method. A loss of an ERAM process during MERAM execution does not interfere with its termination. Furthermore, this algorithm has a great po-

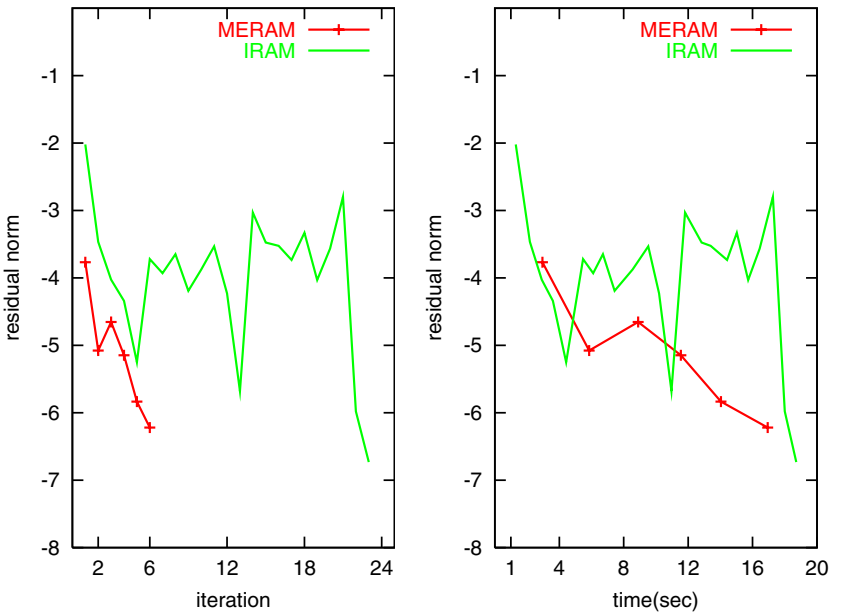


FIG. 10. *MERAM*$((10,20),I_2)$ *versus* *IRAM*$(20,e_1)$ *with* $BFW782A$ *matrix.*

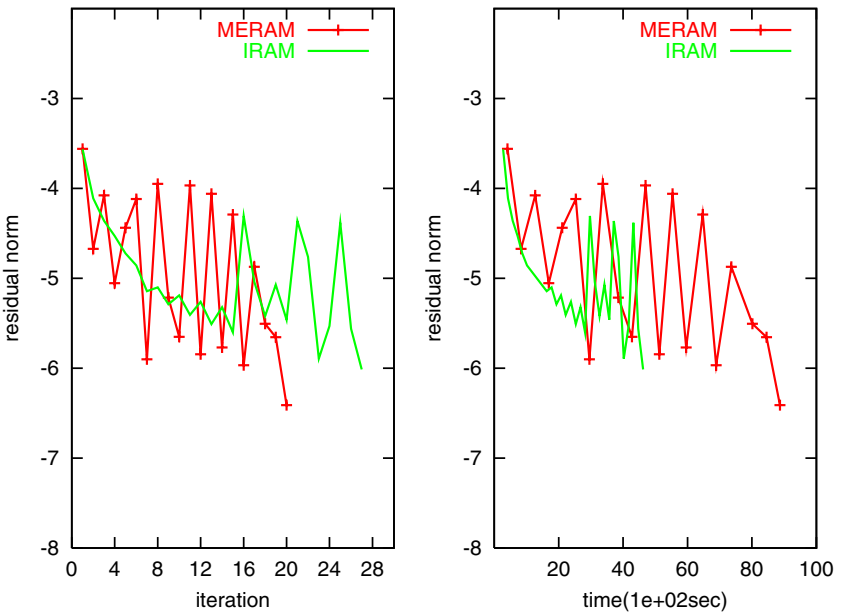


FIG. 11. *MERAM*$((20,35),I_2)$ *versus* *IRAM*$(35,e_1)$ *with* $PDE248004$ *matrix.*

tential for dynamic load balancing. Indeed, the attribution of ERAM processes of MERAM to the available resources can be done as a function of their subspace size at run time. The heterogeneity of computing supports then can be an optimization factor for this method. All of these properties show that MERAM is well suited to the GRID computational environments.

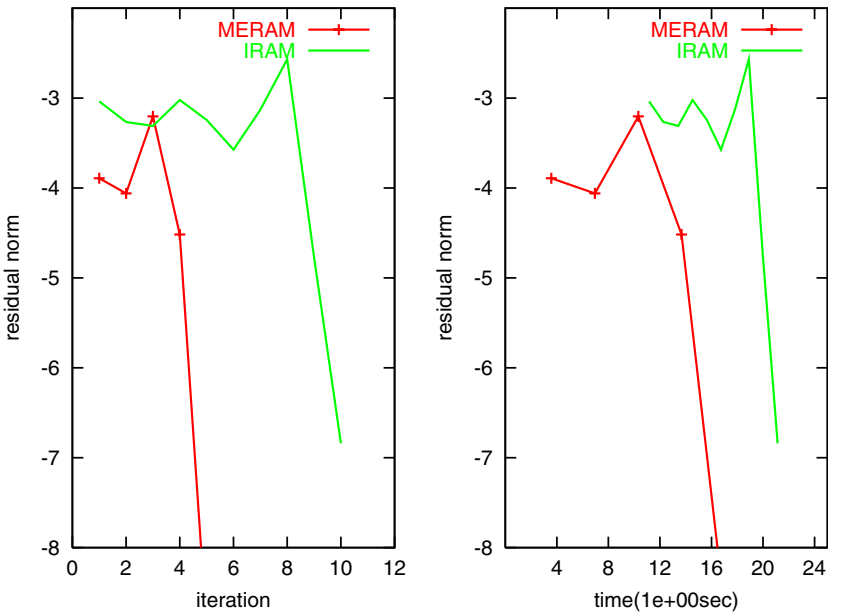


FIG. 12. *MERAM*$((15,20),I_2)$ *versus* *IRAM*$(20,e_1)$ *with* $PDE2961$ *matrix.*

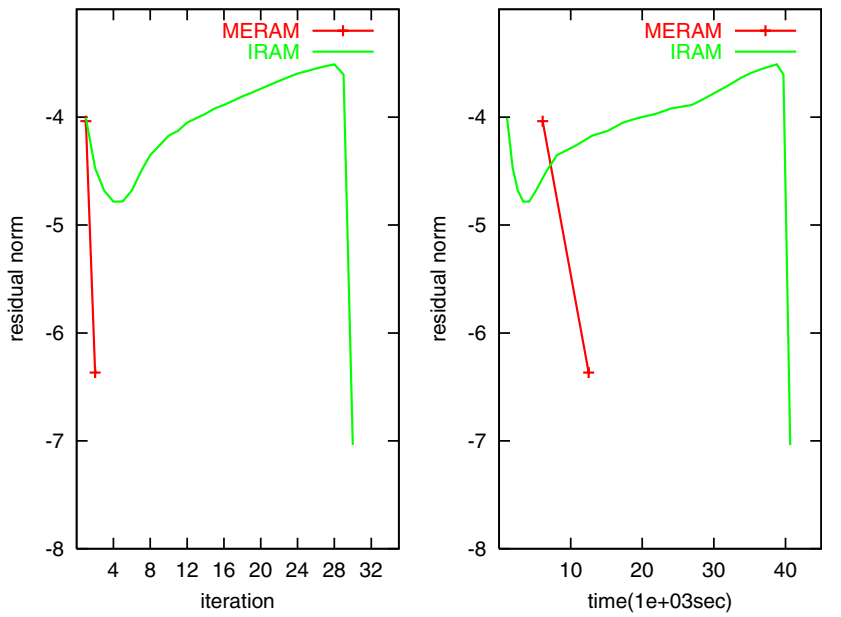


FIG. 13. *MERAM*$((55,30),I_2)$ *versus* *IRAM*$(55,e_1)$ *with* $PDE490000$ *matrix.*

As we have seen, MERAM accelerates the convergence of ERAM. Furthermore, it can cooperate with convergence acceleration methods similar to the iterative Chebyshev polynomials. The presented results of our experiments on a cluster of workstations point out that MERAM has a good convergence behavior compared with ERBAM. Moreover, according to our experiments, for some matrices, MERAM converges faster than IRAM.

We showed that to solve a large eigenproblem, the projection of ERAM on multiple subspaces and the communication of the intermediary results are more interesting than this projection on just one subspace. This concept is simple and can be extended to some other restarted projection methods like IRAM, GMRES, or Hermitian/non-Hermitian Lanczos methods.

There are still many open problems in these methods which involve interesting work from a theoretical point of view as well as practical computation. Some potential improvements to restarting techniques should be considered in future works. Approaches based on sophisticated restart such as augmented Krylov or implicit restarting are also under investigation.

Acknowledgments. The authors wish to thank the SIAM referees for their numerous suggestions and Ani Sedrakian for the realization of some of the experiments.

REFERENCES

[1] W. E. Arnoldi, *The principle of minimized iteration in the solution of matrix eigenvalue problems*, Quart. Appl. Math., 9 (1951), pp. 17–29.
[2] Z. Bai, D. Day, J. Demmel, and J. J. Dongarra, *A Test Matrix Collection for Non-Hermitian Eigenvalue Problems*, http://math.nist.gov/MatrixMarket (October 1996).
[3] F. Chatelin, *Valeurs Propres de Matrices*, Masson, Paris, 1988.
[4] C. Brézinski and M. Redivo-Zaglia, *A hybrid procedure for solving linear systems*, Numer. Math., 67 (1994), pp. 1–19.
[5] J. J. Dongarra, I. S. Duff, D. C. Sorensen, and H. A. Van der Vorst, *Numerical Linear Algebra on High-Performance Computers*, Software Environ. Tools 7, SIAM, Philadelphia, 1998.
[6] G. Edjlali, N. Emad, and S. Petiton, *Hybrid methods on network of heterogeneous parallel computers*, in Proceedings of the 14th IMACS International Symposium on Iterative Methods in Linear Algebra, IMACS World Congress, Atlanta, 1994, pp. 1224-1228.
[7] G. Edjlali, S. Petiton, and N. Emad, *Interleaved Parallel Hybrid Arnoldi Method for a Parallel Machine and a Network of Workstations*, in Proceedings of the International Conference on Information, Systems, Analysis and Synthesis (ISAS'96), Orlando, FL, 1996.
[8] G. Edjlali, *Contribution à la parallélisation des méthodes itératives hybrides pour matrice creuse sur architectures hétérogènes*, Ph.D. thesis, University of Pierre et Marie Curie, Paris, 1994.
[9] G. Golub and C. Van Loan, *Matrix Computation*, 2nd ed., The John Hopkins University Press, Baltimore, MD, 1989.
[10] I. Foster and C. Kesselman, eds., *The GRID: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1999.
[11] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK User's Guide. Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools, SIAM, Philadelphia, 1998.
[12] R. B. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65 (1996), pp. 1213–1230.
[13] R. B. Morgan and M. Zeng, *A Harmonic Restarted Arnoldi Algorithm for Calculating Eigenvalues and Determining Multiplicity*, Tech. report, 2003, available online at http://www3.baylor.edu/∼Ronald_Morgan/papers/hram2.pdf.
[14] E. Noulard, *Object-Oriented Parallel Programming and Reutilisability Applied to Linear Algebra*, Ph.D. thesis, Versailles University, France, 2000.
[15] E. Noulard and N. Emad, *A Key for Reusable Parallel Linear Algebra Software*, Parallel Comput., 27 (2001), pp. 1299–1319.
[16] P. W. Rijks, J. M. Squyres, and A. Lumsdaine, *Object-Oriented MPI (OOMPI) version* 1.0.2g, Tech. report TR-99-14, Department of Computer Science, University of Notre Dame, Notre Dame, IN, 1999.
[17] S. Petiton, *Parallel subspace method for non-Hermitian eigenproblems on the connection machine*, Appl. Numer. Math., 10 (1992), pp. 19–35.
[18] S. Petiton and N. Emad, *A data parallel scientific computing introduction*, in The Data Parallel Programming Model, Lecture Notes in Comput. Sci. 1132, G.-R. Perrin and A. Darte, eds., Springer-Verlag, Berlin, 1996, pp. 45–64.

[19] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, UK, 1992.
[20] Y. Saad, *Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems*, Math. Comp., 42 (1994), pp. 567–588.
[21] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.
[22] Y. Saad, *SPARSKIT: A Basic Tool-Kit for Sparse Matrix Computations*, Version 2, http://www-users.cs.umn.edu/∼saad/software/SPARSKIT/sparskit.html (1994).
[23] D. C. Sorensen, *Implicitly restarted Arnoldi/Lanczos methods for large scale eigenvalue calculations*, in Parallel Numerical Algorithms, D. E. Keyes, A. Sameh, and V. Venkatakrishnan, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 119–165.
[24] K. J. Maschhoff and D. C. Sorensen, *P_ARPACK: An Efficient Portable Large Scale Eigenvalue Package for Distributed Memory Parallel Architectures*, in Applied Parallel Computing in Industrial Problems and Optimization (PARA'96), Lecture Notes in Comput. Sci. 1184, Springer-Verlag, Berlin, 1996, pp. 478–486.
[25] G. W. Stewart, *A Krylov–Schur algorithm for large eigenproblems*, SIAM J. Matrix Anal. Appl., 23 (2001), pp. 601–614.
[26] H. D. Simon and L. Dagum, *Experince in using SIMD and MIMD parallelism for computational fluid dynamics*, Appl. Numer. Math., 12 (1993), pp. 431–442.
[27] Thinking Machines Corporation, *Connection Machine* 5, http://www.cs.berkeley.edu/∼demmel/cs267/cm5docs.html (1992).
[28] Thinking Machines Corporation, *CM Fortran Programming Guide*, http://www.cs.berkeley.edu/∼demmel/cs267/cm5docs.html (1992).
[29] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM Rev., 42 (2000), pp. 267–293.
[30] G. L. G. Sleijpen and H. A. Van der Vorst, *Jacobi–Davidson methods*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., Software Eviron. Tools 11, SIAM, Philadelphia, 2000, pp. 88–105.
[31] K. Wu and H. Simon, *Thick-restart Lanczos method for symmetric eigenvalue problems*, SIAM J. Matrix Anal. Appl., 22 (2000), pp. 602–616.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 278–293
© 2005 Society for Industrial and Applied Mathematics

COMPUTATION OF THE EXPONENTIAL OF LARGE SPARSE SKEW-SYMMETRIC MATRICES*

N. DEL BUONO†, L. LOPEZ†, AND R. PELUSO‡

Abstract. In this paper we consider methods for evaluating both $\exp(A)$ and $\exp(\tau A)q_1$ where $\exp(\cdot)$ is the exponential function, A is a sparse skew-symmetric matrix of large dimension, q_1 is a given vector, and τ is a scaling factor. The proposed method is based on two main steps: A is factorized into its tridiagonal form H by the well-known Lanczos iterative process, and then $\exp(A)$ is derived making use of an effective Schur decomposition of H. The procedure takes full advantage of the sparsity of A and of the decay behavior of $\exp(H)$. Several applications and numerical tests are also reported.

Key words. skew-symmetric matrices, singular value decomposition, geometric integration of ordinary differential equations, splitting techniques

AMS subject classifications. 65F, 65L

DOI. 10.1137/030600758

1. Introduction. Exponential methods for the geometric integration of ordinary differential equations (ODEs) with quadratic invariants [13, 17] or for the solution of partial differential equations (PDEs) of advection type [4, 7, 16, 26] require computing the exponentials of skew-symmetric matrices as well as preserving the orthogonal feature of the obtained matrices. Hence, methods for evaluating both $\exp(A)$ and $\exp(\tau A)q_1$, where A is a real sparse skew-symmetric matrix of large dimension n, q_1 is a given vector of Euclidean norm 1, and τ is a scaling factor associable with the step size in a time integration method for ODEs, are of particular interest. Exact formulas for the computation of the exponential of a low dimension skew-symmetric matrix A are known [1, 21, 24], but unfortunately these are not practical when the dimension n of the matrix A increases. Standard techniques based on Padé and Chebyshev rational approximants can be used, but generally they do not preserve $\exp(\tau A)$ on the orthogonal group except for the diagonal Padé approximants [5]. Polynomial approximants, instead, are based on the approximation of $\exp(\tau A)q_1$ on some Krylov subspace of dimension $m < n$, but these techniques are computationally competitive only when m is small compared with n [11, 14, 25]. Furthermore, when A is skew-symmetric with uniformly distributed eigenvalues, no substantial error reduction occurs for $m \leq \|\tau A\|$ as observed in [14].

In [5, 6, 22] methods based on splitting techniques exploiting the Lie algebra structure of the orthogonal group have been considered to approximate $\exp(\tau A)$ to a given order of accuracy with respect to τ. In [19, 28] schemes based on the [illegible] of A and involving a computational cost of κn^3 flops, where the constant κ increases with the order of the approximation, have also been presented.

In this paper we suggest a two-step procedure to compute $\exp(A)$: A is first reduced into a tridiagonal form H using the Lanczos tridiagonalization process, then

*Received by the editors August 27, 2003; accepted for publication (in revised form) September 2, 2004; published electronically September 12, 2005.
http://www.siam.org/journals/sisc/27-1/60075.html

†Dipartimento di Matematica, Università degli Studi di Bari, Via E. Orabona 4, I-70125 Bari, Italy (delbuono@dm.uniba.it, lopezl@dm.uniba.it).
‡Dipartimento di Matematica, Politecnico di Bari, Via Amendola 126/B, I-70126 Bari, Italy (peluso@poliba.it).

an effective Schur decomposition of H is obtained via the singular value decomposition (SVD) of a bidiagonal matrix B of half dimension $n/2$. The SVD of such a matrix may be computed by efficient methods, for instance the Golub and Kahan algorithm [12] or the accurate [illegible] [9, 10]. In exact arithmetic our procedure preserves $\exp(A)$ in the orthogonal group and its main computational cost is $35/8n^3$ flops to compute $\exp(A)$ and $\mathcal{O}(n^2)$ flops when evaluating $\exp(A)q_1$.

The paper is organized as follows. In section 2 we describe the procedure to derive the Schur decomposition of a tridiagonal, skew-symmetric matrix H, then we show the decay behavior of the entries of $\exp(H)$ proving that such a matrix can be considered as banded. Generalization to smooth functions $f(z)$ is also considered. In section 3 our procedure is numerically compared with MATLAB routines for evaluating exponentials. Furthermore, we show some applications in the approximation of $\exp(\tau A)q_1$ on Krylov subspaces, in the numerical solution of ODEs and finally in the computation of exponentials of general matrices by using splitting techniques.

2. Functions of skew-symmetric matrices.

2.1. Schur decompositions of A. Let us consider a real skew-symmetric matrix $A \in \mathbb{R}^{n\times n}$, a vector $q_1 \in \mathbb{R}^n$ of Euclidean norm 1, the Krylov matrix $K(A, q_1, m) = [q_1\ Aq_1\ A^2q_1 \cdots\ A^{m-1}q_1] \in \mathbb{R}^{n\times m}$, and the Krylov subspace $\mathcal{K}_m = \text{span}\{q_1, Aq_1, \ldots, A^{m-1}q_1\}$. The following result states the conditions for the existence of a Hessenberg form of A.

THEOREM 2.1 (see [12]). [illegible] $Q = [q_1\ q_2 \ldots\ q_n] \in \mathbb{R}^{n\times n}$ [illegible] $Q^\top AQ = H$ [illegible] $R = Q^\top K(A, q_1, n)$ [illegible]

Thus, when $K(A, q_1, n)$ is of full rank n, from the QR factorization of $K(A, q_1, n)$, it follows that an unreduced Hessenberg form H of A exists. The Hessenberg decomposition, $A = QHQ^\top$, is essentially unique when the first column q_1 of Q is fixed and its unreduced Hessenberg form H is skew-symmetric and possesses the following tridiagonal structure:

$$H = \begin{pmatrix} 0 & -h_2 & 0 & \ldots & 0 \\ h_2 & 0 & -h_3 & \ldots & 0 \\ \ldots & & & & \\ \ldots & & & & \\ \ldots & & & & \\ 0 & \ldots & \ldots & \ldots & -h_n \\ 0 & \ldots & \ldots & h_n & 0 \end{pmatrix}. \tag{2.1}$$

For skew-symmetric matrices the reduction to the above tridiagonal form can be performed using the following Lanczos tridiagonalization process:

Let q_1 be a vector of $\mathbb{R}^n$ with $\|q_1\| = 1$ and set $h_1 = 0$ and $q_0 = 0$.
For $j = 1 : n$
  $w_j = Aq_j + h_j q_{j-1}$;
  $h_{j+1} = \|w_j\|$;
  $q_{j+1} = w_j/h_{j+1}$;
end

We have to notice that, in exact arithmetic, the above algorithm is equivalent to the Arnoldi process applied to skew-symmetric matrices. It provides an orthogonal matrix $Q = [q_1\ q_2\ \ldots\ q_n] \in \mathbb{R}^{n\times n}$ such that $Q^\top AQ = H$ where H is the tridiagonal matrix in (2.1). Moreover, it allows one to take full advantage of the possible sparsity of A due to

the matrix-vector products involved. However, in floating-point arithmetic the vectors q_j could progressively lose their orthogonality, in this case, the application of a re-orthogonalization procedure is required [12, 25]. On the contrary, the Arnoldi process preserves better the orthogonality of the matrix Q, but the tridiagonal structure of H could be lost into a Hessenberg one. In the numerical tests we implemented the above tridiagonalization process with a full re-orthogonalization based on the modified Gram–Schmidt algorithm [8, 23].

For very large size problems, the storage requirement for the vectors q_i's, providing the similarity transformation of $\exp(A)$, becomes the main drawback of the process. In this case the Lanczos procedure has to be modified applying a storage technique: a simple way to overcome this problem consists in running twice the process; after the first run the vector $c = \exp(H)e_1$ is computed storing only the sequence $q_1, \ldots, q_r$ with $r < n$ depending on the actual available computer memory [3]. Then the second step consists in restarting the process from q_r to q_n without storing them, and evaluating $\exp(A)q_1 = \sum_{i=1}^{n} c_i q_i$, where c_i is the ith entry of the vector c. Of course, in this version of the algorithm, the cost may increase up to two times the one of a single run, although in the second run the computation of the matrix H can be avoided.

Then, using $A = QHQ^\top$, it follows that $\exp(A) = Q\exp(H)Q^\top$ and $\exp(A)q_1 = Q\exp(H)e_1$, where e_1 is the first vector of the canonical basis of $\mathbb{R}^n$. In this context, the computational problem to deal with is the evaluation of $\exp(H)$, which we will tackle adopting a known Schur decomposition of a tridiagonal skew-symmetric matrix H (see Problem 8.6.6 in [12]).

Suppose n is even. The case of n odd may be approached in a similar way. Let us consider the permutation matrix

$$P = (e_1, e_3, \ldots, e_{n-1}, e_2, e_4, \ldots, e_n), \tag{2.2}$$

where $(e_1, e_2, \ldots, e_n)$ is the canonical basis of $\mathbb{R}^n$. Then, if H is as in (2.1) we have

$$P^\top H P = \begin{pmatrix} 0 & -B \\ B^\top & 0 \end{pmatrix}, \tag{2.3}$$

where B is the lower bidiagonal square matrix of size $w = n/2$ given by

$$B = \begin{pmatrix} h_2 & 0 & \ldots & & 0 \\ -h_3 & h_4 & \ldots & & 0 \\ \ldots & & & & \\ \ldots & & & & \\ 0 & \ldots & \ldots & h_{n-2} & 0 \\ 0 & \ldots & & -h_{n-1} & h_n \end{pmatrix}. \tag{2.4}$$

Since all diagonal and subdiagonal entries of B are nonzero, the w singular values of B are distinct and different from zero.

Let us consider the SVD of B, that is,

$$B = U\Sigma V^\top \tag{2.5}$$

with $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_w)$ and $\sigma_1 > \sigma_2 > \cdots > \sigma_w > 0$, and consider the orthogonal $n \times n$ matrix

$$W = \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix}. \tag{2.6}$$

Then

$$W^\top P^\top H P W = \begin{pmatrix} U^\top & 0 \\ 0 & V^\top \end{pmatrix} \begin{pmatrix} 0 & -B \\ B^\top & 0 \end{pmatrix} \begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix} = \begin{pmatrix} 0 & -\Sigma \\ \Sigma & 0 \end{pmatrix}.$$

By considering the orthogonal matrix $R = PWP^\top$ it follows that

$$P \begin{pmatrix} 0 & -\Sigma \\ \Sigma & 0 \end{pmatrix} P^\top = R^\top H R = \operatorname{diag}(S_1, S_2, \ldots, S_w), \tag{2.7}$$

where

$$S_j = \begin{pmatrix} 0 & -\sigma_j \\ \sigma_j & 0 \end{pmatrix} \qquad \text{for } j = 1, 2, \ldots, w \tag{2.8}$$

is a 2×2 skew-symmetric matrix with eigenvalues given by the pair of pure imaginary values $\lambda_{j,1} = \imath\sigma_j$ and $\lambda_{j,2} = -\imath\sigma_j$ for $j = 1, 2, \ldots, w$, where $\imath$ denotes the imaginary unit. Thus R provides the real Schur decomposition of H, that is,

$$H = R \operatorname{diag}(S_1, S_2, \ldots, S_w) R^\top. \tag{2.9}$$

Finally, the exponential $\exp(H)$ may be computed as

$$\exp(H) = R \operatorname{diag}(\exp(S_1), \exp(S_2), \ldots, \exp(S_w))\ R^\top,$$

where

$$\exp(S_j) = \begin{pmatrix} \cos\sigma_j & -\sin\sigma_j \\ \sin\sigma_j & \cos\sigma_j \end{pmatrix} \qquad \text{for } j = 1, 2, \ldots, w.$$

We can avoid computing R to form $\exp(H)$. In fact, from (2.7) and the definition of $R = PWP^\top$ it follows that

$$\exp(H) = PW \exp\left(\begin{pmatrix} 0 & -\Sigma \\ \Sigma & 0 \end{pmatrix} \right) W^\top P^\top$$

$$= PWP^\top \operatorname{diag}(\exp(S_1), \exp(S_2), \ldots, \exp(S_w)) PW^\top P^\top$$

$$= PW \begin{pmatrix} \cos(\Sigma) & -\sin(\Sigma) \\ \sin(\Sigma) & \cos(\Sigma) \end{pmatrix} W^\top P^\top,$$

where

$$\cos(\Sigma) = \operatorname{diag}(\cos\sigma_1, \cos\sigma_2, \ldots, \cos\sigma_w), \tag{2.10}$$

$$\sin(\Sigma) = \operatorname{diag}(\sin\sigma_1, \sin\sigma_2, \ldots, \sin\sigma_w). \tag{2.11}$$

Finally,

$$\exp(A) = QPT(U, V, \Sigma) P^\top Q^\top, \tag{2.12}$$

where $T(U, V, \Sigma)$ is the following block matrix:

$$T(U, V, \Sigma) = \begin{pmatrix} U\cos(\Sigma)U^\top & -U\sin(\Sigma)V^\top \\ V\sin(\Sigma)U^\top & V\cos(\Sigma)V^\top \end{pmatrix}. \tag{2.13}$$

The vector $\exp(A)q_1$ can be computed as

$$\exp(A)q_1 = QPT(U,V,\Sigma)P^\top Q^\top q_1 = QPT(U,V,\Sigma)e_1. \tag{2.14}$$

It should be noted that, if A is replaced by a scaled matrix τA where τ is a scaling factor, then the n dimensional Krylov space $\mathcal{K}_n$ and its basis Q remain the same, while the bidiagonal matrix B has to be scaled to τB. In the SVD of τB, the matrices U and V remain unchanged while the singular value σ_j has to be scaled to $\tau\sigma_j$. Therefore, we have $\exp(\tau A) = Q\exp(\tau H)Q^\top$ where $\exp(\tau H) = PT(U,V,\tau\Sigma)P^\top$.

The Lanczos tridiagonalization procedure, for computing the tridiagonal form of A, requires at most $(2\upsilon+5)n^2 - (2\upsilon+7)n$ flops plus $n-1$ square roots, where $\upsilon = \max_{1\le j\le n}\upsilon_j$ and υ_j is the number of the nonzero entries of the jth row of A.

To obtain the SVD of the bidiagonal matrix B of dimension $w = \frac{n}{2}$, we employ the Golub and Kahan algorithm which involves $12w^2+30w$ flops plus $2w$ square roots for computing both the singular values and the singular vectors (see [12, page 454]). The cost of the [9, 10] is similar.

Concerning the computation of the block matrix $T(U,V,\Sigma)$, it should be observed that its $(1,2)$ block is the transpose with opposite sign of the block $(2,1)$. Since $\cos(\Sigma)$ and $\sin(\Sigma)$ are diagonal matrices, then the flops required by the three matrix-matrix products $U\cos(\Sigma)$, $V\cos(\Sigma)$, and $U\sin(\Sigma)$ are $3/4n^2$. The three matrix-matrix products $U\cos(\Sigma)U^\top$, $V\cos(\Sigma)V^\top$, and $U\sin(\Sigma)V^\top$ can be computed with a cost of $3/8n^3$ flops, while the product of QP by $T(U,V,\Sigma)$ and of the resulting matrix by $P^\top Q^\top$ needs $4n^3$ flops. Thus the total cost required by our procedure to evaluate $Q\exp(H)Q^\top$ is $35/8n^3+(35/4+2\upsilon)n^2-(2\upsilon-8)n$ flops plus the total cost of function evaluations which is given by $2n-1$ square roots plus the n evaluations of $\cos(\sigma)$ and $\sin(\sigma)$ at the singular values. On the other hand, if the Lanczos process is replaced by n steps of the Arnoldi algorithm, the orthogonalization process needs almost $2n^2\upsilon+2n^3$ flops increasing up to $2n^3$. Then, the main cost of our procedure is $35/8n^3$ flops which should be compared with that of MATLAB routines for matrix exponential which generally varies between $20n^3$ and $30n^3$ flops. We observe that, in the case of full matrices A, the main cost becomes $51/8n^3$ flops.

When $\exp(A)q_1$ has to be computed, from (2.14) it follows that only the first column of $T(U,V,\Sigma)$ is required. Then $U\cos(\Sigma)$ and $V\sin(\Sigma)$ require $1/2n^2$ flops, the first columns of $U\cos(\Sigma)U^\top$ and $V\sin(\Sigma)U^\top$ need n^2 flops, and the matrix-vector product of QP by the first column of T costs $2n^2$ flops. In this case, the total cost of our procedure is $(23/2+2\upsilon)n^2-(2\upsilon-8)n$ flops plus that of function evaluations.

2.2. Generalization. Our procedure may be generalized to evaluating a matrix function $f(A)$, where $f(z)$ is an analytic function inside a closed contour ν encircling the spectrum $[-\imath\sigma_1,\imath\sigma_1]$ of A (i.e., $f(A)$ is). In this case using the Schur decomposition (2.9) we obtain

$$f(H) = R\ \mathrm{diag}(f(S_1), f(S_2), \ldots, f(S_w))\ R^\top,$$

where $f(S_j)$ can be evaluated as described in the following lemma.

LEMMA 2.2. $f(A)$ S_j (2.8)

$$f(S_j) = \gamma(\sigma_j)I + \delta(\sigma_j)J \qquad \text{for } j = 1,2,\ldots,w, \tag{2.15}$$

$J = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ $\gamma(\sigma)$ $\delta(\sigma)$ $\mathbb{R}$

$$\gamma(\sigma) = \frac{1}{2}(f(\imath\sigma) + f(-\imath\sigma)), \qquad \delta(\sigma) = \frac{1}{2\imath}(f(\imath\sigma) - f(-\imath\sigma)), \qquad \sigma \in \mathbb{R}. \tag{2.16}$$

$f(z)$ $\overline{f(z)} = f(\overline{z})$ $z \in \mathbf{C}$ $\gamma(\sigma)$ $\delta(\sigma)$ (2.16)

Since S_j has eigenvalues given by $\lambda_{j,1} = \imath\sigma_j$ and $\lambda_{j,2} = -\imath\sigma_j$, from Sylvester's formula [15] we have

$$f(S_j) = \frac{1}{\lambda_{j,1} - \lambda_{j,2}}[(f(\lambda_{j,1}) - f(\lambda_{j,2}))S_j + (\lambda_{j,1}f(\lambda_{j,2}) - \lambda_{j,2}f(\lambda_{j,1}))I]$$

$$= \frac{1}{2\imath}[\imath(f(\imath\sigma_j) + f(-\imath\sigma_j))I + (f(\imath\sigma_j) - f(\imath\sigma_j))J],$$

where we have used $S_j = \sigma_j J$. Thus (2.15) follows. Now, since $\overline{\imath\sigma} = -\imath\sigma$, when $f(z)$ is an analytic function such that $\overline{f(z)} = f(\overline{z})$ for $z \in C$, then it follows that $\gamma(\sigma)$ and $\delta(\sigma)$ are real infinitely differentiable functions. □

Complex functions $f(z)$ satisfying the condition $\overline{f(z)} = f(\overline{z})$ are $\cos z$; $\sin z$; $\varphi(z) = (\exp(z) - 1)/z$; $\mathrm{cay}(z) = (1 + z/2)/(1 - z/2)$ for $z \neq 2$; $f(z) = (\alpha + z)^{-1}$, with $\alpha \geq 0$ and $z \neq -\alpha$. When $f(z) = \varphi(z)$ for all $z \in C$, we have

$$\gamma(\sigma) = \frac{\sin\sigma}{\sigma}, \qquad \delta(\sigma) = \frac{1 - \cos\sigma}{\sigma}, \qquad \sigma \in \mathbb{R}.$$

When $f(z) = \mathrm{cay}(z)$ for all complex values $z \neq 2$, we have

$$\gamma(\sigma) = \frac{1 - (\frac{\sigma}{2})^2}{1 + (\frac{\sigma}{2})^2}, \qquad \delta(\sigma) = \frac{\sigma}{1 + (\frac{\sigma}{2})^2}, \qquad \sigma \in \mathbb{R}.$$

When $f(z) = (\alpha + z)^{-1}$, with $\alpha \geq 0$ and for all complex values $z \neq -\alpha$, we have

$$\gamma(\sigma) = \frac{\alpha}{\alpha^2 + \sigma^2}, \qquad \delta(\sigma) = -\frac{\sigma}{\alpha^2 + \sigma^2}, \qquad \sigma \in \mathbb{R}.$$

The previous results may be summarized in the following theorem.

THEOREM 2.3. n A $n \times n$ q_1 $\mathrm{rank}[K(A, q_1, n)] = n$ $A = QHQ^\top$ A $f(A)$ $U\Sigma V^\top$ B (2.4) $f(A) = Qf(H)Q^\top$

$$f(H) = P \begin{pmatrix} U\ \Gamma(\Sigma)\ U^\top & -U\ \Delta(\Sigma)\ V^\top \\ V\ \Delta(\Sigma)\ U^\top & V\ \Gamma(\Sigma)\ V^\top \end{pmatrix} P^\top,$$

$\Gamma(\Sigma)$ $\Delta(\Sigma)$

$$\Gamma(\Sigma) = \mathrm{diag}(\gamma(\sigma_1), \gamma(\sigma_2), \ldots, \gamma(\sigma_w)), \quad \Delta(\Sigma) = \mathrm{diag}(\delta(\sigma_1), \delta(\sigma_2), \ldots, \delta(\sigma_w)), \tag{2.17}$$

$\gamma(\sigma)$ $\delta(\sigma)$ (2.16)

2.3. Decay behavior. Although $\exp(A)$ is formally a dense matrix, one can take computational advantage of the possible decay of its entries away from the main diagonal. This kind of behavior has been observed in [2] for symmetric positive definite matrices and in [18] for banded matrices. Here, using the SVD of the matrix B, we will prove that for tridiagonal and skew-symmetric matrices H, the entries of each block in $T(U,V,\Sigma)$ show a decay away from the diagonals. This behavior may be exploited in constructing a banded approximation of $T(U,V,\Sigma)$.

In order to estimate the size of the entries of $T(U,V,\Sigma)$, we derive bounds for the interpolation errors of the functions $\cos\sigma$ and $\sin\sigma$. Let s be an integer greater than 1 and let $\xi_1,\dots,\xi_{s+1}$ be the roots of the Chebyshev polynomial $\tau_{s+1}(\sigma)$ of degree $s+1$ in $[\sigma_w^2,\sigma_1^2]$. Set $\gamma_i=\sqrt{\xi_i}$ for $i=1,\dots,s+1$, and consider the interpolation polynomial $p(\sigma)$ of $\cos\sigma$ at $\pm\gamma_i$ for $i=1,\dots,s+1$. Since $\cos\sigma$ is an even function, it is easy to see that $p(\sigma)$ is an even polynomial of degree $2s$, i.e., $p_{2s}(\sigma)=\sum_{r=0}^{s}c_r\sigma^{2r}$. Then, $\cos\sigma=p_{2s}(\sigma)+R(\sigma)$, with the error function bounded by

$$|R(\sigma)|\le\frac{\left|\prod_{i=1}^{s+1}(\sigma^2-\gamma_i^2)\right|}{(2s+2)!}\qquad\text{for }\sigma\in\mathbb{R}.$$

Since for $i=1,\dots,s+1$, $\gamma_i^2=\xi_i$ are the roots of $\tau_{s+1}(\sigma)$, when $\sigma w\le\sigma\le\sigma_1$ it follows that

$$\left|\prod_{i=1}^{s+1}(\sigma^2-\gamma_i^2)\right|\le\left(\frac{\sigma_1^2-\sigma_w^2}{2}\right)^{s+1}\frac{1}{2^s}.$$

Hence we have

$$|R(\sigma)|\le\frac{(\sigma_1^2-\sigma_w^2)^{s+1}}{2^{2s+1}(2s+2)!}\qquad\text{for }\sigma_w\le\sigma\le\sigma_1. \tag{2.18}$$

Consider the odd function $\sin\sigma$ and the interpolation polynomial at nodes $\pm\gamma_i$ for $i=1,\dots,s+1$. This polynomial is an odd polynomial of degree $2s+1$, i.e., $\sin\sigma=p_{2s+1}(\sigma)+R(\sigma)$, where $p_{2s+1}(\sigma)=\sum_{r=0}^{s}d_r\sigma^{2r+1}$ and

$$|R(\sigma)|\le\frac{\left|\prod_{i=1}^{s+1}(\sigma^2-\gamma_i^2)\right|}{(2s+2)!}\qquad\text{for }\sigma\in\mathbb{R},$$

therefore it follows that

$$|R(\sigma)|\le\frac{(\sigma_1^2-\sigma_w^2)^{s+1}}{2^{2s+1}(2s+2)!}\qquad\text{for }\sigma_w\le\sigma\le\sigma_1. \tag{2.19}$$

THEOREM 2.4. $B=U\Sigma V^\top$ B (2.4) $\cos(\Sigma)$ $\sin(\Sigma)$ (2.10), (2.11) $s\ge1$ $T(U,V,\Sigma)$ (2.12)

$$|[U\cos(\Sigma)U^\top]_{k,l}|\le u(s,\sigma_1,\sigma_w)\qquad\text{for }|k-l|\ge s+1, \tag{2.20}$$

$$|[V\cos(\Sigma)V^\top]_{k,l}|\le u(s,\sigma_1,\sigma_w)\qquad\text{for }|k-l|\ge s+1, \tag{2.21}$$

$$|[U\sin(\Sigma)V^\top]_{k,l}|\le u(s,\sigma_1,\sigma_w)\quad\text{for }|k-l|\ge s+1,\ k-l\ne s+1, \tag{2.22}$$

$$u(s,\sigma_1,\sigma_w)=\frac{(\sigma_1^2-\sigma_w^2)^{s+1}}{2^{2s+1}(2s+2)!}. \tag{2.23}$$

. Consider $E=BB^\top$ and $F=B^\top B$. From the SVD of B, it follows that $E^s=U\Sigma^{2s}U^\top$ and $F^s=V\Sigma^{2s}V^\top$ for $s\geq 1$. Since E and F are tridiagonal matrices, then $E^s=(e_{kl}^{(s)})$ and $F^s=(f_{kl}^{(s)})$ are banded matrices, i.e., $e_{kl}^{(s)}=f_{kl}^{(s)}=0$ for $|k-l|\geq s+1$, and $s\geq 1$. The matrix $G^s=E^sB$, given by $G^s=U\Sigma^{2s+1}V^\top$, is a banded matrix with one more diagonal than E^s and F^s, that is, $g_{kl}^{(s)}=0$ for $|k-l|\geq s+1$ and $k-l\neq s+1$.

Using the interpolation formula $\cos\sigma=p_{2s}(\sigma)+R(\sigma)$ with $p_{2s}(\sigma)=\sum_{r=0}^{s}c_r\sigma^{2r}$, it follows that

$$U\cos(\Sigma)U^\top=\sum_{r=0}^{s}c_rU\Sigma^{2r}U^\top+UR(\Sigma)U^\top=\sum_{r=0}^{s}c_rE^r+UR(\Sigma)U^\top,$$
$$V\cos(\Sigma)V^\top=\sum_{r=0}^{s}c_rV\Sigma^{2r}V^\top+VR(\Sigma)V^\top=\sum_{r=0}^{s}c_rF^r+VR(\Sigma)V^\top,$$

where $R(\Sigma)=\mathrm{diag}(R(\sigma_1),R(\sigma_2),\ldots,R(\sigma_w))$.

Since E^s and F^s are banded matrices, then $[U^\top\cos(\Sigma)U]_{kl}=[U^\top R(\Sigma)U]_{kl}$ and $[V^\top\cos(\Sigma)V]_{kl}=[V^\top R(\Sigma)V]_{kl}$ for $|k-l|\geq s+1$. Hence, by using (2.18), the inequalities (2.20) and (2.21) follow.

Using the interpolation formula $\sin\sigma=\sum_{r=0}^{s}d_r\sigma^{2r+1}+R(\sigma)$, it follows that

$$U\sin(\Sigma)V^\top=\sum_{r=0}^{s}d_rU\Sigma^{2r+1}V^\top+UR(\Sigma)V^\top=\sum_{r=0}^{s}d_rG^r+UR(\Sigma)V^\top,$$

and thus from (2.19), inequality (2.22) follows. ∎

If $\sigma_1^2-\sigma_w^2<2$, the upper bound function $u(s,\sigma_1,\sigma_w)$ in (2.23) rapidly tends to zero as s increases. If $\sigma_1^2-\sigma_w^2\geq 2$ but $(\sigma_1^2-\sigma_w^2)^{s+1}$ increases, less than $2^{2s+1}(2s+2)!$, then $u(s,\sigma_1,\sigma_w)$ tends to zero as s increases. In both cases the estimates of Theorem 2.4 show that the entries of each block of the matrix $T(U,V,\Sigma)$ decay away from the diagonal. Given a threshold parameter ϵ, the bandwidth outside which the entries of $T(U,V,\Sigma)$ are smaller in modulus than ϵ can be defined as $r=\min\{s|u(s,\sigma_1,\sigma_w)\leq\epsilon\}$. The integer r is usually much smaller than w for large size problems, when the distance $\sigma_1^2-\sigma_w^2$ is of moderate size.

The result shown in Theorem 2.4 may be used to define a banded version of our procedure. Let $T_\epsilon(U,V,\Sigma)$ be the matrix derived from $T(U,V,\Sigma)$ by setting to zero the entries with modulus less than ϵ and let $[\exp(A)]_\epsilon$ be the approximation of $\exp(A)$ obtained by

$$[\exp(A)]_\epsilon=QPT_\epsilon(U,V,\Sigma)Q^\top P^\top.$$

It then follows that $\|T_\epsilon(U,V,\Sigma)-T(U,V,\Sigma)\|\leq n\epsilon$, where $\|\cdot\|$ denotes the 2-norm on matrices. Since Q and P are orthogonal matrices, we have $\|\exp(A)-[\exp(A)]_\epsilon\|=\|QP[T(U,V,\Sigma)-T_\epsilon(U,V,\Sigma)]Q^\top P^\top\|=\|T_\epsilon(U,V,\Sigma)-T(U,V,\Sigma)\|\leq n\epsilon$. Thus a suitable choice for ϵ is given by $\epsilon=\varepsilon/n$ where ε is the machine accuracy. Each block of $T_\varepsilon(U,V,\Sigma)$ is a band matrix with bandwidth r. It is not difficult to see that $[\exp(H)]_\epsilon=PT_\varepsilon(U,V,\Sigma)P^\top$ is a band matrix with bandwidth given by $2r$.

Since each block of $T_\epsilon(U,V,\Sigma)$ is a banded matrix of bandwidth r, the two matrix-matrix products in $U\cos(\Sigma)U^\top$ and $V\cos(\Sigma)V^\top$ require $2w(2w-r)(r+1)$ flops, while the matrix-matrix product in $U\sin(\Sigma)V^\top$ costs $2w^2+2w[2(w-1)-r](r+1)$ flops, with $w=\frac{n}{2}$. The matrix-matrix product of QP by $T_\epsilon(U,V,\Sigma)$ can be performed with a cost of $4n^2(2r+1)$ flops. Recalling that $w=n/2$ and that υ is the (maximum by rows) number of nonzero entries, the total cost of our procedure is $2n^3+(2\upsilon+61/4+10r)n^2-(2\upsilon+2r^2+4r+8)n$ plus the cost of function evaluations.

As before, if the vector $\exp(A)q_1$ only has to be computed, we need $4w(2r+1)$ flops to compute the first column vectors of $U\cos(\Sigma)U^\top$ and $V\sin(\Sigma)V^\top$, while $4n(2r+1)$ are the flops required for the matrix-vector product of QP by the first column of $T(U,V,\Sigma)$. Thus, our procedure now requires $(17/2+2\upsilon)n^2-(2\upsilon-12r)n$ plus the cost of function evaluations. The partial costs are not reported since they equal the ones of the nonbanded case (refer to the end of section 2.1).

3. Applications and numerical tests. All numerical tests have been obtained running MATLAB 5.3 on a Pentium IV 2.4 GHz with 1024 MbRAM. In the numerical codes we have implemented the Lanczos tridiagonalization process with a full reorthogonalization based on the iterated modified Gram–Schmidt algorithm [8, 23]; the SVD of the bidiagonal matrix B has been performed exploiting the Golub and Kahan algorithm. Moreover, no storage techniques has been adopted.

The procedure for computing $\exp(A)$ by (2.12) has been denoted by `AExp`, while `AbExp(r)` denotes the procedure in which $T(U,V,\Sigma)$ is replaced by $T_\epsilon(U,V,\Sigma)$, where the bandwidth `r` of each block of $T_\epsilon(U,V,\Sigma)$ is given by $\min\{s|u(s,\sigma_1,\sigma_w)\le\epsilon\}$, with $\epsilon=\varepsilon/n$ and $\varepsilon=2.2206\text{e}-16$ (the MATLAB floating point relative accuracy).

3.1. Numerical comparisons. We have compared our approaches with two MATLAB functions `Expm` and `Expm3`. Particularly, `Expm` computes the exponential of A using a scaling and squaring algorithm with Padé approximations, while `Expm3` evaluates the exponential of A via eigenvalue and eigenvector decomposition. Comparisons are done in terms of: Flops (counted by the built in MATLAB routine `flops`); global error, defined as the 2-norm of the difference of `AExp` (respectively, `AbExp`) and `Expm`; orthogonal error, defined as the distance of the computed exponential from the orthogonal manifold (i.e., $\|\texttt{AExp}^\top\texttt{AExp}-I_n\|_F$, where $\|\cdot\|_F$ is the Frobenius norm on matrices). Tables 3.1 and 3.2 report the comparisons on sparse and dense skew-symmetric matrices A of different dimensions n and entries randomly generated in $[-10,10]$. Observe that the results obtained for `AExp` and `AbExp(r)` are similar, nevertheless `AbExp(r)` saves flops.

3.2. Krylov approximations of $\exp(\tau A)q_1$. When we need to compute $y=\exp(\tau A)q_1$ and n is large, the tridiagonalization process to obtain H may be too expensive. In this case a common way to save computation is to consider the approximation, $y_m=Q_m\exp(\tau H_m)e_1$, on the Krylov subspace of dimension $m<n$, where $e_1=(1,0,\dots,0)^\top\in\mathbb{R}^m$, and Q_m, H_m are the matrices derived by the Krylov algorithm that is the Lanczos tridiagonalization process in section 2.1 run until $m<n$. However, when A is a skew-symmetric matrix with uniformly distributed eigenvalues, no substantial error reduction can be observed for $m\le\|\tau A\|$ ([14]), and thus the Krylov algorithm should be implemented in a different way than for symmetric matrices. In particular, we fix m sufficiently large and consider the approximation $y_m=Q_m\exp(\tau H_m)e_1$ then an a posteriori error estimate has to be used to evaluate the Krylov approximation obtained.

TABLE 3.1
Comparisons on sparse matrices.

n	Method	Flops	Global error	Orthogonal error
50	`AExp`	1145932	9.0129e-15	1.5095e-14
	`AbExp(9)`	1106808	2.9155e-14	7.9203e-14
	`Expm`	2664508	-	1.3323e-15
	`Expm3`	5231760	-	2.2204e-16
100	`AExp`	8791319	2.9465e-14	3.7254e-14
	`AbExp(11)`	8328146	2.9737e-14	3.5312e-14
	`Expm`	22990654	-	-3.7748e-15
	`Expm3`	40867683	-	1.3323e-15
200	`AExp`	68324797	5.4209e-14	8.2757e-14
	`AbExp(13)`	63746408	5.4535e-14	7.4573e-14
	`Expm`	182624884	-	8.6597e-15
	`Expm3`	318490737	-	8.8818e-16
1000	`AExp`	9595372489	1.8063e-11	5.5681e-13
	`AbExp(57)`	9095990316	1.8867e-11	5.5681e-13
	`Expm`	38696969972	-	8.9447e-11
	`Expm3`	38702914947	-	3.3599e-12

TABLE 3.2
Comparisons on dense matrices.

n	Method	Flops	Global error	Orthogonal error
50	`AExp`	1372571	7.6373e-14	1.8952e-14
	`AbExp(18)`	1367152	7.6353e-14	1.9060e-14
	`Expm`	2664508	-	1.3323e-15
	`Expm3`	5231760	-	2.2204e-16
100	`AExp`	10576419	7.1397e-14	3.3902e-14
	`AbExp(21)`	10322746	7.1459e-14	3.5483e-14
	`Expm`	26989568	-	5.5511e-15
	`Expm3`	40807089	-	8.8818e-16
200	`AExp`	82521976	2.7667e-13	1.0002e-13
	`AbExp(27)`	79302399	2.7657e-13	1.1354e-13
	`Expm`	230628400	-	1.7319e-14
	`Expm3`	318630721	-	8.8818e-16
1000	`AExp`	10084256626	2.2523e-12	6.6374e-13
	`AbExp(36)`	9430845640	2.0543e-12	8.9980e-13
	`Expm`	32696962184	-	3.4766e-11
	`Expm3`	38733904231	-	3.0348e-12

`AExpy` denotes the method based on formula (2.14) for evaluating $y = \exp(\tau A)q_1$ while `KExpy(m)` indicates the Krylov approximation, $y_m = Q_m \exp(\tau H_m)e_1$, where Q_m and $\exp(\tau H_m)$ are computed by our procedure, while `KExpmy(m)` marks the Krylov approximation where the exponential $\exp(\tau H_m)$ is computed using the MATLAB function `Expm`. In Table 3.3 we compare the results obtained by `AExpy`, `KExpy(m)`, and `KExpmy(m)`, where A is a dense random matrix with entries in $[-1, 1]$ and q_1 a vector of Euclidean unit norm. The results show that `KExpy(m)` always saves computation with respect to `AExpy` and `KExpmy(m)`. We have also applied our procedure to compute the vector $y = \exp(A)q_1$, where A is now a 300×300 sparse skew-symmetric matrix, with about 10% nonzero elements randomly generated in $[-10, 10]$, and q_1 is a random vector with $\|q_1\| = 1$. Figure 3.1 shows the semilog plot of the global error (dotted line with circle) evaluated as $\|y - y_m\|$, and of the orthogonal error, defined as $|y_m^\top y_m - 1|$, (solid line with stars), against the dimension m of the Krylov subspace. Figure 3.1 shows that accurate Krylov approximations may be obtained for quite large values of m.

TABLE 3.3
Comparisons for computing $\exp(A)q$.

n	Method	Flops	Global error	Orthogonal error
100	`AExpy`	5843843	2.8015e-14	4.4409e-16
	`KExpy(25)`	663918	2.8398e-14	4.4409e-16
	`KExpmy(25)`	955870	2.7832e-14	2.8866e-15
200	`AExpy`	44591067	8.4962e-14	1.1102e-16
	`KExpy(50)`	5319882	8.1275e-14	4.4409e-16
	`KExpmy(50)`	7784495	8.1730e-14	1.1102e-15
300	`AExpy`	148907702	6.9981e-14	8.8818e-16
	`KExpy(70)`	16314427	6.2032e-14	3.1086e-15
	`KExpmy(70)`	22392695	6.1764e-14	2.2204e-16

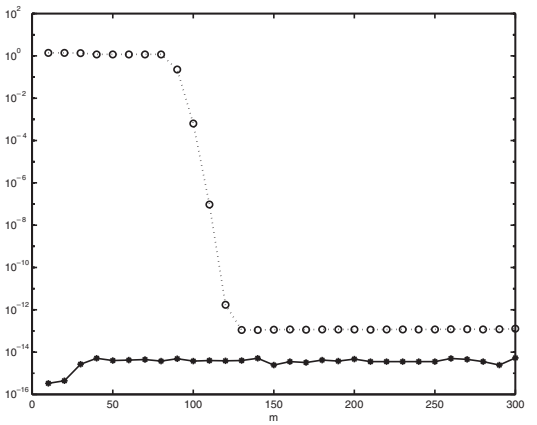


FIG. 3.1. *Errors behavior.*

3.3. Application to ODEs and PDEs. We consider the method of Magnus series, which approximates the solution of linear differential equations $y' = A(t)y$, $y(0) = y_0$ in the form $y(t) = \exp(\Omega(t))y_0$, where $\Omega(t)$ is a matrix function satisfying a suitable ODE [13, 17]. When $A(t)$ is a skew-symmetric matrix function, $\Omega(t)$ is also skew-symmetric and $y(t)$ is a vector with $\|y(t)\| = 1$ for all t. We adopt our approach to compute the exponentials of the second and fourth order Magnus methods, whose definitions are reported below (h is the time step):

MG2	MG4
$A_1 = A(t_n + h/2);$	$A_1 = A(t_n + (\frac{1}{2} - \frac{\sqrt{3}}{6})h);$
$\omega_n = A_1;$	$A_2 = A(t_n + (\frac{1}{2} + \frac{\sqrt{3}}{6})h);$
$y_{n+1} = \exp(h\omega_n)y_n;$	$\omega_n = \frac{1}{2}(A_1 + A_2) + \frac{\sqrt{3}}{12}h[A_2, A_1];$
	$y_{n+1} = \exp(h\omega_n)y_n.$

As an example we consider the skew-symmetric matrix $A(t) = (a_{ij}(t))$, whose upper triangular entries are $a_{ij}(t) = (-1)^{i+j}\frac{i}{j+1}t^{j-i}$, $1 \le i < j \le n$ [20]. In the computation we have taken $n = 100$ and we integrate the differential system $y' = A(t)y$ on the interval $[0, 1]$; as initial vector a vector $y(0)$ randomly generated and with unit Euclidean norm has been employed . Denote by ZpExp the method of order p, with respect to the time step h, proposed in [28]. It preserves exponential matrices on the orthogonal group and its cost is of κn^3 flops, where the constant κ increases with the order p of the approximation. MGp(`AExpy`) and MGp(Zp`Exp`) denote the

Magnus methods in which the exponential matrices, at each step, are computed by `AExpy` and `Z`p`Exp`, respectively. Table 3.4 reports the performance of the methods. It can be seen that MGp(`Z`p`Exp`) are less accurate than MGp(`AExpy`). However the second order Magnus method based on `Z2Exp` saves flops with respect to MG2(`AExpy`) although this favorable feature is less clear when we compare fourth order methods MG4(`Z4Exp`) and MG4(`AExpy`), since `Z`p`Exp` have a cost increasing with the order p of the approximation. The schemes MGp(`KExpy(m)`), based on the approximation of Krylov subspace, show the same performance in terms of orthogonal and global errors as MGp(`AExpy`), while reducing the number of flops required.

TABLE 3.4
Performance of second and fourth order methods.

h	Method	Flops	Orthogonal error	Global error
$h = 0.1$	MG2(`AExpy`)	58898468	1.1990e-14	0.0025
	MG2(`KExpy`(10))	2625917	3.5527e-15	0.0025
	MG2(`Z2Exp`)	8281742	4.4409e-16	0.0216
	MG4(`AExpy`)	99830866	7.9936e-15	3.3569e-4
	MG4(`KExpy`(10))	43620601	5.3291e-15	3.3569e-4
	MG4(`Z4Exp`)	67318182	5.5511e-16	0.0021
$h = 0.05$	MG2(`AExpy`)	117785480	7.7716e-15	7.3285e-4
	MG2(`KExpy`(10))	10500852	7.1054e-15	7.3285e-4
	MG2(`Z2Expy`)	16563482	2.2204e-16	0.0079
	MG4(`AExpy`)	199697270	1.4433e-14	5.2503e-5
	MG4(`KExpy`(10))	87240935	6.2172e-15	5.2503e-5
	MG4(`Z4Exp`)	134636362	8.8818e-16	2.4964e-4
$h = 0.025$	MG2(`AExpy`)	235500110	5.5511e-15	1.9819e-4
	MG2(`KExpy`(10))	10500852	7.1054e-15	1.9819e-4
	MG2(`Z2Expy`)	33126962	6.6613e-16	0.0023
	MG4(`AExpy`)	399342752	1.7986e-14	4.5198e-6
	MG4(`KExpy`(10))	174480892	9.3259e-15	4.5198e-6
	MG4(`Z4Exp`)	269272722	2.5535e-15	1.8543e-5

We now consider a different example, in particular the KdV PDE $u_t = -uu_x - \delta^2 u_{xxx}$, with periodic boundary conditions $u(0,t) = u(L,t)$, where L is the period and δ is a small parameter. As shown in [7], appropriate methods of space discretization lead to solve a set of ODEs of the form $y' = A(y)y$, $y(0) = y_0$, evolving on the sphere of radius $\|y_0\|$, where $y(t) = (u_0(t), u_1(t), \ldots, u_{N-1}(t))^\top$, $u_i(t) \approx u(i\Delta x, t)$ for $i = 0, 1, \ldots, N-1$, and where $\Delta x = \frac{2}{N}$ is the spatial step of $[0, 2]$. For instance, if we consider the space discretization method in [26] we have

$$A(y) = -\frac{1}{6\Delta x} g(y) - \frac{\delta^2}{2\Delta x^3} P,$$

where both $g(y)$ and P are two $N \times N$ skew-symmetric matrices given by

$$g(y) = \begin{pmatrix} 0 & u_0+u_1 & 0 & 0 & 0 & -(u_0+u_{N-1}) \\ -(u_0+u_1) & 0 & u_1+u_2 & \ldots & 0 & 0 \\ 0 & -(u_1+u_2) & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ u_{N-1}+u_0 & 0 & 0 & \ldots & -(u_{N-1}+u_{N-2}) & 0 \end{pmatrix}$$

and

$$P = \begin{pmatrix} 0 & -2 & 1 & 0 & 0 & \dots & 0 & 0 & -1 & 2 \\ 2 & 0 & -2 & 1 & 0 & \dots & 0 & 0 & 0 & -1 \\ -1 & 2 & 0 & -2 & 1 & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \\ 1 & 0 & 0 & 0 & 0 & \dots & -1 & 2 & 0 & -2 \\ -2 & 1 & 0 & 0 & 0 & \dots & 0 & -1 & 2 & 0 \end{pmatrix}.$$

Particularly we adopt the method of iterated commutators of second order [27], denoted by M2, to solve the ODE derived from the KdV equation with $\delta = 0.022$, initial condition $y_0 = y(0)/\|y(0)\|$, where $y(0) = (u_0(0), u_1(0), \dots, u_{N-1}(0))^\top$, $u_i(0) = u(i\Delta x, 0)$ for $i = 0, \dots, N-1$, and $u(x,0) = \cos(\pi x)$. As before, to compute the exponential appearing in the numerical scheme we use `AExpy`, `KExpy(m)`, and `Z2Exp` procedures. Table 3.5 provides the performance of the methods applied to compute the waveform at $\pi t = 1$ with steplength $\Delta t = 0.04/\pi$ and different spacings Δx.

TABLE 3.5
Performance of second order method on different spacings.

Δx	Method	Flops	Distance from the sphere
2/150	M2(`AExpy`)	654872986	9.7700e-15
	M2(`KExpy`(56))	58898468	5.5511e-16
	M2(`Z2Exp`)	133462402	6.6613e-16
2/200	M2(`AExpy`)	1373223649	2.6090e-14
	M2(`KExpy`(65))	130256897	2.2093e-14
	M2(`Z2Exp`)	303614902	1.6653e-15
2/300	M2(`AExpy`)	4517639681	2.7756e-15
	M2(`KExpy`(75))	244039347	1.2212e-15
	M2(`Z2Exp`)	982419902	1.1102e-16

3.4. Exponentials of general matrices and splitting techniques. Slight modifications in the algorithm shown in section 2.1 can allow the computation of the exponential of a tridiagonal symmetric matrix H with zero diagonal entries. Let H be the following matrix:

$$H = \begin{pmatrix} 0 & h_2 & 0 & \dots & 0 \\ h_2 & 0 & h_3 & \dots & 0 \\ 0 & h_3 & 0 & \dots & 0 \\ \dots & \dots & 0 & \dots & h_n \\ 0 & \dots & \dots & h_n & 0 \end{pmatrix}. \tag{3.1}$$

Considering the permutation matrix (2.2) we have that

$$P^\top H P = \begin{pmatrix} 0 & B \\ B^\top & 0 \end{pmatrix},$$

where now B is the square matrix of dimension $w = n/2$ given by

$$B = \begin{pmatrix} h_2 & 0 & \dots & & 0 \\ h_3 & h_4 & \dots & & 0 \\ \dots & & & & \\ 0 & \dots & \dots & h_{n-2} & 0 \\ 0 & \dots & \dots & h_{n-1} & h_n \end{pmatrix} \tag{3.2}$$

with $h_i > 0$ for $i = 2, \ldots, n$.

Using the SVD of $B = U\Sigma V^\top$ with $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_w)$ and $\sigma_1 > \sigma_2 > \cdots > \sigma_w > 0$, in the same fashion as in the skew-symmetric case, the exponential of H of the form (3.1) can be computed as

$$\exp(H) = PT(U, V, \Sigma)P^\top, \tag{3.3}$$

where the matrix $T(U, V, \Sigma)$ is now given by

$$T(U, V, \Sigma) = \begin{pmatrix} U\cosh(\Sigma)U^\top & U\sin h(\Sigma)V^\top \\ V\sinh(\Sigma)U^\top & V\cosh(\Sigma)V^\top \end{pmatrix}$$

and where $\cosh(\Sigma)$ and $\sinh(\Sigma)$ denote the following two diagonal matrices:

$$\cosh(\Sigma) = \mathrm{diag}(\cosh\sigma_1, \cosh\sigma_2, \ldots, \cosh\sigma_w), \tag{3.4}$$

$$\sinh(\Sigma) = \mathrm{diag}(\sinh\sigma_1, \sinh\sigma_2, \ldots, \sinh\sigma_w). \tag{3.5}$$

This observation can be useful in the computation of the exponential $\exp(hA)$ of a general matrix A to a low order accuracy with respect to h. In fact, by using the splitting $A = A_1 + A_2$, where $A_1 = 1/2(A + A^\top)$ and $A_2 = 1/2(A - A^\top)$ are, respectively, the symmetric and skew-symmetric parts of A, then a scheme of first order with respect to h to approximate $\exp(hA)$ is given by

$$\exp(hA_1)\exp(hA_2).$$

Since A_2 is skew-symmetric, then $\exp(hA_2)$ can be computed by our procedure (2.12). To evaluate $\exp(hA_1)$ we observe that, using the Lanczos algorithm, A_1 can be transformed into a tridiagonal symmetric matrix $S = Q_1 A_1 Q_1^\top$ with Q_1 orthogonal.

By considering the splitting $S = S_1 + D_1$, where $D_1 = \mathrm{diag}(S)$ and S_1 is a tridiagonal symmetric matrix with zero diagonal entries, the exponential matrix $\exp(hA)$ can be approximated to first order of accuracy by

$$Q_1 \exp(hS_1)\exp(hD_1)Q_1^\top \exp(hA_2); \tag{3.6}$$

finally, in the computation of $\exp(hS_1)$ we can apply (3.3).

A second order scheme is given by the Strang formula

$$\exp\left(\frac{h}{2}A_2\right)\exp(hA_1)\exp\left(\frac{h}{2}A_2\right)$$

from which, using $S = Q_1 A_1 Q_1^\top$ and then $S = S_1 + D_1$, we obtain

$$\exp\left(\frac{h}{2}A_2\right)Q_1\exp\left(\frac{h}{2}D_1\right)\exp(hS_1)\exp\left(\frac{h}{2}D_1\right)Q_1^\top\exp\left(\frac{h}{2}A_2\right). \tag{3.7}$$

We have applied the above splitting (3.6) and (3.7) to compute the exponential of a sparse randomly generated 200×200 matrix. Table 3.6 reports the number of floating-point operations required by the first and second order splitting methods implemented using our procedures to compute the exponential and the MATLAB function `Expm` and `Expm3`.

Table 3.7 shows the performance of the splitting procedures when the step-size value is halved. It can be observed that the global error reduces with the order of the method.

TABLE 3.6
Costs in the evaluation of $\exp(A)$.

Method	Flops
`Expm`	198544582
`Expm3`	353673645
Splitting1	112392463
Splitting2	112599180

TABLE 3.7
Evaluation of $\exp(tA)$.

h	Method	Flops	Global error at $t = 0.01$
0.01	Splitting1	112095693	0.0023
	Splitting2	112589080	4.9001e-5
0.005	Splitting1	128387212	0.0011
	Splitting2	128582213	1.2250e-5
0.0025	Splitting1	160381558	5.6943e-4
	Splitting2	160589487	3.0625e-6

4. Conclusions. In this paper we have provided a method to compute both $\exp(\tau A)$ and $\exp(\tau A)q_1$, when A is a sparse skew-symmetric matrix. The proposed algorithm preserves $\exp(\tau A)$ in the orthogonal group and requires $35/8n^3$ floating-point operations for this computation, while it compels $(23/2 + 2v)n^2$ floating-point operations to evaluate $\exp(\tau A)q_1$ conserving the length of the obtained vector. We have also generalized our procedure to calculate smooth functions of skew-symmetric matrices and highlighted the use of our algorithm in the approximation of $\exp(\tau A)q_1$ on Krylov subspace, in the context of exponential integrators of ODEs and PDEs and in the evaluation of the exponential of a general matrix.

Acknowledgment. The authors wish to thank the anonymous referees for their many helpful suggestions and constructive remarks.

REFERENCES

[1] A. O. BARUT, J. R. ZENI, AND A. LAUFER, *The exponential map for the conformal group* $O(2,4)$, J. Phys. A, 27 (1994), pp. 5239–5250.
[2] M. BENZI AND G. H. GOLUB, *Bounds for the entries of matrix functions with applications to preconditioning*, BIT, 39 (1999), pp. 417–438.
[3] L. BERGAMASCHI AND M. VIANELLO, *Efficient computation of the exponential operator for large, sparse, symmetric matrices*, Numer. Linear Algebra Appl., 7 (2000), pp. 27–45.
[4] E. CELLEDONI, *A note on the numerical integration of the KdV equation via isospectral deformations*, J. Phys. A, 34 (2001), pp. 2205–2214.
[5] E. CELLEDONI AND A. ISERLES, *Approximating the exponential from a Lie algebra to a Lie group*, Math. Comp., 69 (2000), pp. 1457–1480.
[6] E. CELLEDONI AND A. ISERLES, *Methods for the approximation of the matrix exponential in a Lie-algebraic setting*, IMA J. Numer. Anal., 21 (2001), pp. 463–488.
[7] J.-B. CHEN, H. MUNTHE-KAAS, AND M.-Z. QIN, *Square-conservative schemes for a class of evolution equations using Lie-group methods*, SIAM J. Numer. Anal., 39 (2002), pp. 2164–2178.
[8] J. W. DANIEL, W. B. GRAGG, L. KAUFMAN, AND G. W. STEWART, *Reorthogonalization and stable algorithms updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.
[9] J. DEMMEL AND W. KAHAN, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.
[10] K. V. FERNANDO AND B. N. PARLETT, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[11] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1236–1264.
[12] G. H. Golub and C. F. van Loan, *Matrix Computation*, 3rd ed., The John Hopkins University Press, Baltimore, 1996.
[13] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag, Berlin, 2002.
[14] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997) pp. 1911–1925.
[15] R. Horn and C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1995.
[16] W. H. Hundsdorfer, *Numerical Solution of Advection-Diffusion-Reaction Equations*, CWI Report NM-N9603, Center for Mathematics and Computer Science CWI, Amsterdam, 1996.
[17] A. Iserles, H. Munthe-Kaas, S. Nørsett, and A. Zanna, *Lie-group methods*, in Acta Numerica, Acta Numer. 9, Cambridge University Press, Cambridge, UK, 2000, pp. 215–365.
[18] A. Iserles, *How large is the exponential of a banded matrix?*, New Zealand J. Math., 29 (2000), pp. 177–192.
[19] A. Iserles and A. Zanna, *Efficient computation of the matrix exponential by generalized polar decompositions*, SIAM J. Numer. Anal., 42 (2005), pp. 2218–2256.
[20] A. Iserles, E. Marthinsen, and S. Nørsett, *On the implementation of the method of Magnus series for linear differential equations*, BIT, 39 (1999), pp. 281–304.
[21] F. S. Leite and P. Crouch, *Closed forms for the exponential mapping on matrix Lie groups based on Putzer's method*, J. Math. Phys., 40 (1999), pp. 3561–3568.
[22] B. Leimkuhler, *Adaptive geometric integrators based on splitting, scaling, and switching*, in International Conference on Differential Equations, Vol. 1, 2, World Scientific, River Edge, NJ, 2000, pp. 988–993.
[23] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.
[24] T. Politi, *A formula for the exponential of a real skew symmetric matrix of order* 4, BIT, 41 (2001), pp. 842–845.
[25] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[26] N. J. Zabusky and M. D. Kruskal, *Interaction of solitons in a collisionless plasma and the recurrence of initial states*, Phys. Rev. Lett., 15 (1965), pp. 240–243.
[27] A. Zanna, *The Method of Iterated Commutators for Ordinary Differential Equations on Lie Groups*, DAMTP Technical report NA1996/12, University of Cambridge, Cambridge, UK, 1996.
[28] A. Zanna and H. Z. Munthe-Kaas, *Generalized polar decompositions for the approximation of the matrix exponential*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 840–862.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 294–314
© 2005 Society for Industrial and Applied Mathematics

CONSERVATIVE LOGARITHMIC RECONSTRUCTIONS AND FINITE VOLUME METHODS*

ROBERT ARTEBRANT† AND H. JOACHIM SCHROLL†

Abstract. A class of high-order reconstruction methods based on logarithmic functions is presented. Inspired by Marquina's hyperbolic method, we introduce a double logarithmic ansatz of fifth order of accuracy. Low variation is guaranteed by the ansatz and (slope-) limiting is avoided. The method can reconstruct smooth extrema without order reduction. Fifth order of convergence is verified in a numerical experiment governed by the nonlinear Euler system. Numerical experiments, including the Osher–Shu shock/acoustic interaction, are presented.

Key words. conservation law, finite volume method, high-order reconstruction

AMS subject classifications. 35L65, 35L45, 65M06

DOI. 10.1137/03060240X

1. Introduction and background.

The present paper presents a new reconstruction method to compute accurate numerical fluxes within the context of finite volume methods for conservation laws. These schemes are derived from the integro-differential formulation of the conservation law

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_\Omega u(t,x)\,\mathrm{d}x + \int_{\partial\Omega} f(u(t,x))\cdot n\,\mathrm{d}S = 0\ .$$

Within a spatial domain Ω the total amount of a certain quantity u is conserved; i.e., it does not change with time except by flow across the boundary $\partial\Omega$. It seems worthwhile to point out the duality in the formulation of this dynamic: The conserved quantity is the total amount or the [illegible] of u, while the dynamics of this average is determined by [illegible] of the flux $f(u(t,x))$ along the boundary. Clearly, to simulate the dynamics accurately, it is absolutely essential to determine accurate point values for the flux along the boundary. That is the [illegible] task: From given cell averages, determine point values along the boundary.

Since the 1970's several reconstruction approaches have been developed. The starting point was a piecewise linear reconstruction in the monotonic upstream-centered scheme for conservation laws (MUSCL) scheme which resulted in a second-order sequel to Godunov's method [13]. At that time it seemed natural to use higher-degree polynomials for reconstruction aiming for higher-order methods. However, polynomials of higher degree tend to oscillate and solutions of conservation laws are very sensitive to artificial oscillations. To minimize these artificial oscillations, essentially nonoscillatory (ENO) schemes select one (or, in weighted ENO (WENO) schemes, a combination) of several interpolation polynomials defined by neighboring cell averages such that the variation of the reconstructing polynomial remains as small as possible. Several modifications to improve the performance of ENO-type schemes were proposed recently; see [10] and the references therein.

*Received by the editors December 12, 2003; accepted for publication (in revised form) September 13, 2004; published electronically September 12, 2005. This research was supported by the European network HYKE, funded by the EC under contract HPRN-CT-2002-00282.

http://www.siam.org/journals/sisc/27-1/60240.html

†Centre for Mathematical Sciences, Lund University, Box 118, SE–221 00 Lund, Sweden (robert.artebrant@na.lu.se, achim.schroll@na.lu.se).

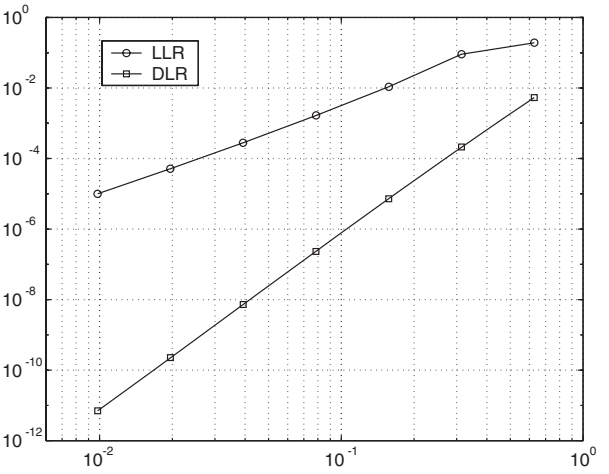


FIG. 1. *L_1-errors for the Euler system.*

To the knowledge of the authors there is only one higher-order method which avoids polynomials for reconstruction; that is the piecewise hyperbolic method (PHM) by Marquina [5]. This reconstruction is monotone and hence oscillation-free and third-order accurate for monotone data. In the case of local extrema it degenerates to second order. Motivated by promising results with this reconstruction method when applied in relaxed finite volume schemes [8], it is tempting to look for similar (monotone) reconstruction methods which may be extended to an order of accuracy higher than three. To achieve this, additional conditions will be imposed on the reconstructing function, in particular, on its derivatives; see section 3. In order to handle these conditions it will be advantageous to work with a logarithmic ansatz instead of hyperbolic functions, thereby "gaining" one derivative via integration. In fact, it turns out that, by a [illegible] ansatz, [illegible] order of accuracy is possible even at smooth local extrema. In one space dimension the ansatz reads

$$r(x) = A + B\log(x + C) + D\log(x + E) \ .$$

The five coefficients are determined from conditions on the average as well as first and second derivatives. By combining two monotone logarithms we allow for one isolated extreme value in the reconstruction such that smooth extrema can be reconstructed with full fifth order of accuracy. The following experiment [7] verifies this capability. Consider the Euler system

$$\begin{pmatrix} \rho \\ \rho u \\ e \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ u(e+p) \end{pmatrix}_x = 0 \ ,$$

with pressure $p = (\gamma - 1)\left(e - \rho u^2/2\right)$, $\gamma = 1.4$, and initial data $\rho(0, x) = 1 + 0.2\sin(x)$, $u \equiv 1$, and $p \equiv 1$. The solution to this initial value problem is $\rho(t, x) = 1 + 0.2\sin(x - t)$, $u \equiv 1$, and $p \equiv 1$. We compute an approximation to the solution by a standard finite volume method using the double logarithmic reconstruction (DLR) for the evaluation of the boundary fluxes; for details see section 7. The results are displayed in Figure 1 and Table 1. Full [illegible] of convergence is clearly verified. Note that the governing equation is a [illegible] of conservation laws.

TABLE 1
Estimated rates for the Euler system.

	$N = 640$	320	160	80	40	20
LLR	2.36	2.44	2.57	2.71	3.06	1.08
DLR	5.00	5.00	5.00	4.97	4.88	4.64

In the following a local reconstruction using a single logarithmic ansatz is described first. Then section 3 presents several realizations for higher-order reconstructions. The double logarithmic ansatz is worked out in detail in sections 4 and 5. The complete, hybrid algorithm is described in section 6. Numerical experiments, including convergence and accuracy tests, are found in section 7.

2. Local logarithmic reconstruction. In this section the piecewise reconstruction of an unknown smooth function g using a single logarithmic function is described. The resulting reconstruction method uses data only from the central cell and its nearest neighbors and is therefore called [illegible] (LLR). The grid data are the cell averages and approximations to the lateral derivatives of g.

We consider the computational grid $x_i = ih$, i integer and h a positive number, with cells $C_i = [x_{i-1/2}, x_{i+1/2})$.

The reconstruction is piecewise defined in each cell; therefore we shall consider without restriction a reference cell C_0. The reconstructing function in this cell has the form

$$r_0(x) \sim A + B\log(x + C).$$

2.1. The monotone case. Since the reconstructing function is monotone, we assume for the moment that g is also monotone in C_0. Now, following Marquina [5], we impose three conditions on r_0 to determine the three coefficients A, B, and C. Given v_0 the cell average of g in C_0 and d_1 and d_2 approximations to the left and right derivatives, we require

$$\frac{1}{h}\int_{C_0} r_0(x)dx = v_0, \tag{1}$$

$$r_0'\left(x_0 - \frac{h}{2}\right) = d_1, \tag{2}$$

$$r_0'\left(x_0 + \frac{h}{2}\right) = d_2. \tag{3}$$

To this end define for $x \in C_0$ and a a parameter to be determined

$$\phi_0(x) := -\frac{d_1 h}{a}\log\left[x - x_0 - \frac{h}{2}\left(\frac{2}{a} - 1\right)\right]. \tag{4}$$

By inspection it is observed that the argument of the log function is positive for $a \in (-\infty, 0)$ when $x \in C_0$. Now put

$$r_0(x) := v_0 + \phi_0(x) - \frac{1}{h}\int_{C_0}\phi_0(\xi)d\xi. \tag{5}$$

We immediately see that (5) satisfies the condition (1) of preserved cell average. Further,

$$r_0'(x) = \phi_0'(x) = -\frac{d_1 h}{a\left[x - x_0 - \frac{h}{2}\left(\frac{2}{a} - 1\right)\right]},$$

so $r_0'(x_0 - h/2) = d_1$ and also condition (2) is fulfilled. The parameter a is now used to realize condition (3):

$$r_0'(x_0 + h/2) = -\frac{d_1}{a-1} = d_2 \quad \Rightarrow \quad a = 1 - d_1/d_2.$$

Since g is monotone, $d_1 d_2 > 0$ and therefore $a \in (-\infty, 1)$. Above we observed that the argument of the log function is positive when $a \in (-\infty, 0)$; however for $a \in [0, 1)$ the reconstructing function r_0 is also real and smooth (see Corollary 5.8).

In the finite volume context the point values at the cell boundaries are of particular interest. The LLR approximations to $g\left(x_0 \pm \frac{h}{2}\right)$ are conveniently computed as

$$r_0\left(x_0 \pm \frac{h}{2}\right) = v_0 + d_1 h \eta^{\pm}(a), \tag{6}$$

where

$$\eta^{+}(a) = -\frac{\log(1-a) + a}{a^2}, \tag{7}$$

and

$$\eta^{-}(a) = \frac{(a-1)\log(1-a) - a}{a^2}. \tag{8}$$

The functions $\eta^{\pm}$ have a removable singularity at zero by putting $\eta^{\pm}(0) = \pm\frac{1}{2}$. The function η^{-} has also a removable singularity at $+1$ by setting $\eta^{-}(1) = -1$.

2.2. Preprocessing of nonmonotone data. If g is not monotone in C_0, we replace the approximative derivate with the largest absolute value by the other one multiplied with h^2; see [5]. To avoid division by zero we also make this modification if one of both derivatives is smaller than h^2. In the case that both derivatives are below h^2, the data is essentially constant and the reconstruction should be skipped. Note that this preprocessing of general data results in monotone data $d_1 d_2 > 0$ for the reconstruction in any case. Hereby the parameter $a = 1 - d_1/d_2$ in (5) also transforms into the allowed range $(-\infty, 1)$, resulting in a [illegible]; cf. Corollary 5.8.

A numerical convergence test with the Euler system as governing equations has been computed using the LLR in connection with Van Leer flux splitting; see Figure 1. The initial data and the smooth solution of the problem are given in the introduction. For further details see section 7.

[illegible] 2.1 (local variation). The [illegible] of a monotone function f in a cell C_i is given by $LV_i(f) = |f(x_i + h/2) - f(x_i - h/2)|$. In the case of a discontinuity within the cell, i.e., $d_1 = \mathcal{O}(1/h)$ and $d_2 = \mathcal{O}(1)$ (or vice versa), the local variation of LLR is $LV_0(r_0) = \mathcal{O}(h|\log h|)$. This variation is smaller compared to the hyperbolic reconstruction (LHR) in [5] for which $LV_0 = \mathcal{O}(\sqrt{h})$. This is the reason why hyperbolic methods need additional smoothing before reconstruction. Marquina's local, harmonic, hyperbolic reconstruction (LHHR) indeed has sufficiently small variation of $\mathcal{O}(h)$; see [5].

[illegible] 2.2 (limiters). As in Marquina's hyperbolic method (LHR), no limiter is used in the reconstruction.

3. Higher order: The reconstruction principle. We are interested in higher-order [illegible] in space. The reconstruction is carried out at a certain time, and therefore we omit in this section the time dependence of the function u to be reconstructed. Since we are dealing with finite volume schemes the available information about the unknown function u is its cell averages:

$$u_i = \frac{1}{h}\int_{C_i} u(x)dx. \tag{9}$$

The goal is to find a reconstruction $r(x)$ from given cell averages u_i such that in each cell

$$(r-u)(x) = \mathcal{O}(h^p), \quad x \in C_i.$$

To this end the following observation, which is easily verified by Taylor expansion, will be useful.

OBSERVATION 3.1. [illegible] $f, g \in C^p[a,b]$ [illegible] $h := b-a$ [illegible] $(f-g)(x) = \mathcal{O}(h^p)$ [illegible] $x \in [a,b]$ [illegible] p [illegible] $x_k \in [a,b]$ $k = 0, \dots, p-1$ [illegible] $(f-g)^{(k)}(x_k) = \mathcal{O}(h^{p-k})$

Now we can verify that LLR is a third-order reconstruction for monotone functions.

LEMMA 3.2 (LLR). [illegible] $f, g \in C^3[a,b]$ [illegible] $h := b-a$ [illegible]

$$\int_a^b f(x)dx = \int_a^b g(x)dx, \tag{10}$$

$$(f-g)'(a) = \mathcal{O}(h^2) = (f-g)'(b); \tag{11}$$

[illegible] $(f-g)(x) = \mathcal{O}(h^3)$ [illegible] $x \in [a,b]$ [illegible]. It is enough to find points satisfying the requirements in Observation 3.1. First, (10) and continuity give that there exists an $x_0 \in (a,b)$ such that $f(x_0) = g(x_0)$. By (11) we have with $x_1 = a$ (or b), $(f-g)'(x_1) = \mathcal{O}(h^2)$. By the mean value theorem (MVT), $(f-g)'(b) - (f-g)'(a) = (f-g)''(x_2)h$ and now (11) implies $(f-g)''(x_2) = \mathcal{O}(h)$. Thus the points are found. □

We can easily add one condition to obtain a fourth-order reconstruction. The extra condition is set on the derivative in the cell center.

LEMMA 3.3. [illegible] $f, g \in C^4[a,b]$ [illegible] $h := b-a$ $c := (b+a)/2$ [illegible]

$$\int_a^b f(x)dx = \int_a^b g(x)dx, \tag{12}$$

$$(f-g)'(a) = \mathcal{O}(h^3) = (f-g)'(b), \tag{13}$$

$$(f-g)'(c) = \mathcal{O}(h^3); \tag{14}$$

[illegible] $(f-g)(x) = \mathcal{O}(h^4)$ [illegible] $x \in [a,b]$

We omit the proof.

The next goal is to extend this reconstruction principle to fifth order of accuracy. It turns out that there are several possibilities. We present three of them.

LEMMA 3.4 (realization 1). Let $f, g \in C^5[a,b]$ and $h := b - a$.

$$\int_a^b f(x)dx = \int_a^b g(x)dx, \tag{15}$$

$$(f-g)'(a) = \mathcal{O}(h^4) = (f-g)'(b), \tag{16}$$

$$(f-g)''(a) = \mathcal{O}(h^3) = (f-g)''(b); \tag{17}$$

then $(f-g)(x) = \mathcal{O}(h^5)$ for all $x \in [a,b]$.

Again, we verify the requirements in Observation 3.1. First, (15) and continuity give that there exists an $x_0 \in (a,b)$ such that $f(x_0) = g(x_0)$. By (16) we have with $x_1 = a$ (or b) $(f-g)'(x_1) = \mathcal{O}(h^4)$. By the MVT, $(f-g)'(b) - (f-g)'(a) = (f-g)''(x_2)h$ and now (16) implies

$$(f-g)''(x_2) = \mathcal{O}(h^3). \tag{18}$$

Again the MVT gives $(f-g)''(x_2) - (f-g)''(a) = (f-g)^{(3)}(x_3)(x_2-a)h$ and by (17) and (18)

$$(f-g)^{(3)}(x_3) = \mathcal{O}(h^2). \tag{19}$$

Another use of the MVT yields $(f-g)''(b) - (f-g)''(x_2) = (f-g)^{(3)}(x_3')(b-x_2)h$ and by (17) and (18)

$$(f-g)^{(3)}(x_3') = \mathcal{O}(h^2). \tag{20}$$

Finally, the MVT implies $(f-g)^{(3)}(x_3') - (f-g)^{(3)}(x_3) = (f-g)^{(4)}(x_4)(x_3'-x_3)h$, and by (19) and (20) $(f-g)^{(4)}(x_4) = \mathcal{O}(h)$. Thus the points are found. □

The second realization sets conditions in the cell center on the benefit of requiring one second derivative less.

LEMMA 3.5 (realization 2). Let $f, g \in C^5[a,b]$ and $h := b - a$, $c := (b+a)/2$.

$$\int_a^b f(x)dx = \int_a^b g(x)dx, \tag{21}$$

$$(f-g)'(a) = \mathcal{O}(h^4) = (f-g)'(b), \tag{22}$$

$$(f-g)'(c) = \mathcal{O}(h^4), \tag{23}$$

$$(f-g)''(c) = \mathcal{O}(h^3); \tag{24}$$

then $(f-g)(x) = \mathcal{O}(h^5)$ for all $x \in [a,b]$.

The proofs of this and also the next lemma follow the lines of the previous one and are therefore omitted.

The third and last realization does not involve requirements on the second derivative at all but sets conditions on the first derivative at four points in the interval of consideration.

LEMMA 3.6 (realization 3). Let $f, g \in C^5[a,b]$ and $h := b - a$, $c := (2a+b)/3$, $d := (2b+a)/3$.

$$\int_a^b f(x)dx = \int_a^b g(x)dx, \tag{25}$$

$$(f-g)'(a) = \mathcal{O}(h^4) = (f-g)'(b), \tag{26}$$

$$(f-g)'(c) = \mathcal{O}(h^4) = (f-g)'(d); \tag{27}$$

$(f-g)(x) = \mathcal{O}(h^5)$ $x \in [a,b]$

Some remarks are appropriate.

3.7 (conservative finite differences). The realizations above all require higher-order approximations to derivatives of the unknown function at some points. However, standard finite difference (fd) formulas cannot be applied since point values of the function are unknown. formulas are computed from cell averages (9); cf. sections 6 and 9.

In the following we shall describe double logarithmic reconstructions based on lateral derivatives, i.e., realization 1.

4. Double logarithmic reconstruction. In this section we describe a reconstruction method where the reconstructing function is up to a constant the sum of two logarithmic functions, hence the name (DLR).

The goal is to develop a low variation reconstruction of order higher than three. Having in mind Remark 2.1, a logarithmic ansatz is a natural candidate. For a fourth-order method, a four-parameter, double logarithmic ansatz based on Lemma 3.3 would be tempting

$$r_0(x) \sim A + B[\log(x+C) + \log(x+D)].$$

However, this ansatz is still monotone. Order reduction at local extrema is therefore inevitable and a significant improvement over LLR cannot be expected. Incorporating one more parameter, thus allowing for a combination of two monotone functions that are possibly of different signs, higher-order reconstruction of smooth extrema is feasible:

$$r_0(x) \sim A + B\log(x+C) + D\log(x+E).$$

Summing up two monotone functions also ensures that not more than one new extremum is introduced. Fifth order of accuracy is realized by adjusting the parameters according to the five order conditions in one of Lemmas 3.4–3.6.

4.1. Construction of the fifth-order DLR. Following the procedure for LLR, we define for $x \in C_0$

$$\phi_0(x) := -\frac{ch}{a}\log\left[x - x_0 - \frac{h}{2}\left(\frac{2}{a} - 1\right)\right] - \frac{dh}{b}\log\left[x - x_0 - \frac{h}{2}\left(\frac{2}{b} - 1\right)\right], \tag{28}$$

where a, b, c, and d are parameters to be determined, and we let

$$r_0(x) := v_0 + \phi_0(x) - \frac{1}{h}\int_{C_0}\phi_0(\xi)d\xi. \tag{29}$$

As for the LLR the question arises if r_0 is well defined in the sense that we obtain a real-valued smooth function. However, for DLR this is not obvious and requires a more careful analysis, which is presented in the next section. Lemma 5.9 states that the reconstruction (29) is a real and smooth function if (i) $a \in \mathbb{C} \setminus \mathbb{R}$, (ii) $a \in (-\infty, 1)$ and $b \in (-\infty, 1)$.

In the following we consider the conditions set in Lemma 3.4 (realization 1 of DLR), that is,

$$\frac{1}{h}\int_{x_0-h/2}^{x_0+h/2} r_0(x)dx = v_0, \tag{30}$$

$$r_0'\left(x_0 - \frac{h}{2}\right) = d_1, \tag{31}$$

$$r_0'\left(x_0 + \frac{h}{2}\right) = d_2, \tag{32}$$

$$r_0''\left(x_0 - \frac{h}{2}\right) = D_1, \tag{33}$$

$$r_0''\left(x_0 + \frac{h}{2}\right) = D_2, \tag{34}$$

where v_0 is the cell average of the unknown function g in C_0 and d_1 and d_2 are fourth-order approximations to the left and right derivatives and D_1 and D_2 are third-order approximations to the left and right second derivatives. We see that (29) preserves the cell average v_0. Now conditions (31)–(34) lead to the following equations for the parameters a, b, c, and d:

$$c + d = d_1, \tag{35}$$

$$-\frac{d(a-1) + c(b-1)}{(a-1)(b-1)} = d_2, \tag{36}$$

$$\frac{ca + db}{h} = D_1, \tag{37}$$

$$\frac{db(a-1)^2 + ca(b-1)^2}{h(a-1)^2(b-1)^2} = D_2. \tag{38}$$

Solving equations (35)–(37) for b, c, and d, putting the solution in (38), and then solving for a yield after some manipulation

$$a = a_1 + a_2, \tag{39}$$

$$b = a_1 - a_2, \tag{40}$$

$$c = d_1/2 + q(a), \tag{41}$$

$$d = d_1/2 - q(a), \tag{42}$$

where

$$\begin{aligned} a_1 &= [D_1 D_2 h^2 + (d_1 D_2 + d_2 D_1)h + 2d_2(d_1 - d_2)]/a_0, \\ a_2 &= \operatorname{sqrt}[D_1^2 D_2^2 h^4 + 2(d_2 D_1^2 D_2 - d_1 D_1 D_2^2)h^3 \\ &\quad + (d_1^2 D_2^2 + 2d_1 d_2 D_1 D_2 + d_2^2 D_1^2)h^2 \\ &\quad + 4(d_1^3 D_2 - d_1^2 d_2 D_2 + d_1 d_2^2 D_1 - d_2^3 D_1)h \\ &\quad + 4d_1 d_2 (d_1 - d_2)^2]/a_0 \end{aligned} \tag{43, 44}$$

with

$$a_0 = 2(d_2(d_1 - d_2) + h d_1 D_2) \tag{45}$$

and q is the function

$$q(z) = \frac{1}{2}\frac{d_1 d_2(1 - z^2) - 2h d_2 D_1(1 - z) + d_1(hD_1 - d_1)}{d_2(z-1)^2 + d_1(2z - 1) - hD_1}. \tag{46}$$

Now when the coefficients are found, assuming $a,b \notin [1, \infty)$, we can in principle put them in (28) and use the reconstructing function (29) to obtain pointwise approximations to g for all $x \in C_0$. It is, however, convenient to compute the DLR

approximations to the lateral function values as

$$r_0\left(x_0+\frac{h}{2}\right)=v_0+ch\eta^+(a)+dh\eta^+(b), \tag{47}$$

$$r_0\left(x_0-\frac{h}{2}\right)=v_0+ch\eta^-(a)+dh\eta^-(b), \tag{48}$$

where the functions $\eta^{\pm}$ are the LLR lateral functions (7) and (8).

4.2. DLR preprocessing. The DLR coefficients (39)–(42) which are obtained from the order conditions (30)–(34) are not always well defined. For example, the denominator of (41) and (42), $q_0 = 2(d_2(a-1)^2 + d_1(2a-1) - hD_1)$, as well as a_0 in (45), the denominator in (43) and (44), might be zero. To avoid division by zero, respecting the order conditions in Lemma 3.4, first- and second-order derivatives are modified as follows: For tol $= \epsilon_{mach}$ we set

if $|a_0|$ <tol **and** $|d_1| > h$
  $D_2 = D_2 + h^3$
else if $|a_0|$ <tol **and** $|d_1| \le h$
  $d_2 = d_2 + h^4$
if $|q_0|$ <tol
  $D_1 = D_1 - h^3$.

Be aware that in contrast to LLR preprocessing (section 2.2), [illegible]. It is just the DLR coefficients (39)–(42) which are well defined, finite numbers after DLR preprocessing. In this case, whether the reconstruction is well defined is analyzed in the next section; see Lemma 5.9. Based on this analysis a hybrid algorithm is built in section 6, using DLR wherever possible and LLR otherwise. Nonexistence of a well-defined DLR is exemplified in section 7.1, Example 3.

For the other realizations (Lemmas 3.5 and 3.6), (37) and (38) are replaced by the relevant ones but the solution, coefficients a, b, c, and d, can be written in the form (39)–(42) with new expressions for a_1, a_2, and q. We list these expressions in the appendix (section 9). Since the reconstructing function (29) is the same, the approximations to the lateral function values are still given by (47) and (48). The aim of the preprocessing is again solely to avoid dividing by zero, so it is straightforward to modify the preprocessing previously stated for these cases.

5. Analysis of the reconstructing functions. In this section we prove that the reconstructing functions in the LLR and DLR are real and smooth for the previously stated values of the parameters.

Let ϕ_0 be the function (28) defined for the DLR,

$$\phi_0(x) = -\frac{ch}{a}\log\left[x-x_0-\frac{h}{2}\left(\frac{2}{a}-1\right)\right] - \frac{dh}{b}\log\left[x-x_0-\frac{h}{2}\left(\frac{2}{b}-1\right)\right],$$

and consider the DLR function (29)

$$r_0(x) = v_0 + \phi_0(x) - \frac{1}{h}\int_{C_0}\phi_0(\xi)d\xi.$$

The coefficients for DLR are given in (39)–(42),

$$a = a_1 + a_2,$$

$$b = a_1 - a_2,$$
$$c = d_1/2 + q(a),$$
$$d = d_1/2 - q(a),$$

with a_1 real, a_2 real or purely imaginary, d_1 real, and q a rational function of a complex variable. In this section we assume preprocessed data; that is, all coefficients are well-defined finite numbers.

Complex coefficients. We consider first the case that a is not real. It is trivial to see the following.

OBSERVATION 5.1. $a \in \mathbb{C} \setminus \mathbb{R} \Rightarrow b = \bar{a}$

Evaluating q at a and b it is equally simple to see the following lemma.

LEMMA 5.2. [illegible] a [illegible] b [illegible] $q(b) = -q(a)$

In the proof of the next lemma we will need the following theorem from complex analysis.

THEOREM 5.3 (reflection principle). [illegible] f [illegible] Ω [illegible] $f(z) = \overline{f(\bar{z})}$ [illegible] $z \in \Omega$.

LEMMA 5.4. [illegible] $a \in \mathbb{C} \setminus \mathbb{R}$ [illegible] $c \in \mathbb{C}$ [illegible] $d = \bar{c}$

[illegible]. The function q is rational and therefore analytic and real on a real interval. From Observation 5.1 and Lemma 5.2 it follows that $q(a) = -q(\bar{a})$. Moreover, by Theorem 5.3 $q(a) = \overline{q(\bar{a})}$, so $q(a) = -\overline{q(a)}$ and therefore $q(a)$ is purely imaginary and the proof is complete. □

LEMMA 5.5. [illegible] $a \in \mathbb{C} \setminus \mathbb{R}$ [illegible] (29) [illegible]

[illegible]. Define $\alpha(x) := x - x_0 - h(2/a - 1)/2$, $\beta(x) := x - x_0 - h(2/b - 1)/2$, $\gamma := -ch/a$, and $\delta := -dh/b$. Now write (28) as

$$\phi_0(x) = \gamma \log(\alpha(x)) + \delta \log(\beta(x)).$$

From Observation 5.1 and Lemma 5.4 it follows that $b = \bar{a}$ and $d = \bar{c}$; therefore $\beta(x) = \bar{\alpha}(x)$ and $\delta = \bar{\gamma}$ for all $x \in C_0$. Let $\gamma := \xi + i\eta$ with ξ and η real. Compute

$$\begin{aligned}\phi_0(x) = \gamma \log(\alpha) + \bar{\gamma} \log(\bar{\alpha}) &= (\xi \log|\alpha| - \eta \arg\alpha) \\ &+ i(\eta \log|\alpha| + \xi \arg\alpha) + (\xi \log|\alpha| - \eta \arg\alpha) \\ &+ i(-\eta \log|\alpha| - \xi \arg\alpha) = 2(\xi \log|\alpha| - \eta \arg\alpha) \in \mathbb{R}.\end{aligned}$$

Since ϕ_0 is real the reconstructing function (29) is real. □

Real coefficients. Consider now the case that a is real.

LEMMA 5.6. [illegible] $\phi(x) = \frac{h}{\lambda} \log\left[x - x_0 - \frac{h}{2}\left(\frac{2}{\lambda} - 1\right)\right]$ [illegible] $\psi(x) := \phi(x) - \frac{1}{h}\int_{C_0} \phi(\xi)d\xi$ [illegible] $x \in C_0$ [illegible] ψ [illegible] $\lambda \in (-\infty, 1)$

[illegible]. Let

$$t(x) := x - x_0 + h/2 - h/\lambda$$

be the argument of the log function. The function ϕ has a singularity for $x \in C_0$ when $t(x) = 0$. Since t is linear in x and

$$\min_{x \in C_0} t(x) = -h/\lambda \quad \text{and} \quad \max_{x \in C_0} t(x) = h - h/\lambda, \tag{49}$$

t has a zero in C_0 for $\lambda \in [1, \infty)$. For $\lambda < 0$ we see that $t > 0$ and therefore ϕ, and hence ψ, is real and smooth for λ negative. Now consider $\lambda \in (0, 1)$. From (49) it

follows that $t(x) < 0$ for $\lambda \in (0,1)$; hence $\phi(x) = \frac{h}{\lambda}(\log|t(x)| + i\pi)$ and

$$\begin{aligned}\frac{\lambda}{h}\psi(x) &= \log|t(x)| + i\pi - \frac{1}{h}\int_{C_0} \log|t(x)| + i\pi dx \\ &= \log|t(x)| - \frac{1}{h}\int_{C_0} \log|t(x)| dx,\end{aligned}$$

so ψ is real and smooth for $\lambda \in (0,1)$. Finally ψ has a removable singularity at $\lambda = 0$ by putting $\psi(x)|_{\lambda=0} = x_0 - x$. □

Now we turn back to the DLR. Observe that if a is real, then b, c, and d are also real. Then the following lemma follows from Lemma 5.6.

LEMMA 5.7. [illegible] $a \in (-\infty, 1)$ [illegible] $b \in (-\infty, 1)$ [illegible] (29) [illegible]

As a simple consequence of the above lemma we get the following corollary.

COROLLARY 5.8 (LLR). [illegible] (5) [illegible] $a \in (-\infty, 1)$ [illegible] 2.2) [illegible]

From Lemmas 5.5 and 5.7 and the observation that the DLR function (29) can have a singularity in C_0 only if a and b are real, the following final result follows.

LEMMA 5.9 (DLR). [illegible] $a \in \mathbb{C} \setminus \mathbb{R}$ [illegible] $a \in (-\infty, 1)$ [illegible] $b \in (-\infty, 1)$ [illegible] (29) [illegible]

6. Logarithmic reconstruction—the algorithm. In this section the previously developed reconstruction techniques are composed into a complete algorithm. This requires two aspects to be handled: First, the reconstruction method has to be well defined, and second, the desired result should be a robust, low variation but high-order approximation of the unknown function.

The above analysis (in section 5) has shown that the fifth-order double logarithmic reconstruction is not always possible with the most general in-data given in terms of cell averages in nearby cells. If the DLR parameters a, b which are determined from the system (35)–(38) happen to be real but greater than one, then the double logarithmic ansatz (29) develops a singularity within the actual cell—this is undesirable (to say the least). In this case the single logarithmic reconstruction LLR is preferred, which—by preprocessing of nonmonotone data (section 2.2)—in any case is a well-defined smooth function. So, if during DLR, either $1 < a \in \mathbb{R}$ or $1 < b \in \mathbb{R}$, then DLR is replaced by preprocessed LLR, achieving the first goal: a well-defined reconstruction. This hybrid algorithm is summarized as follows.

Logarithmic reconstruction. Given cell averages:

1. Determine (sufficiently accurate) first- and second-order lateral derivatives $d_{1,2}$ and $D_{1,2}$, respectively (see below).
2. Determine preprocessed DLR coefficients (sections 4.1 and 4.2).
3. [illegible] (Lemma 5.9), evaluate DLR
   [illegible] evaluate preprocessed LLR (sections 2.1 and 2.2) instead.

The next issue addressed is robustness; i.e., how to construct appropriate approximations to the required derivatives based on cell averages. The selection of stencils affects the robustness of the scheme.

The double logarithmic reconstruction (realization 1) relies on cell averages (30) and discrete approximations to the first and second lateral derivatives (31)–(34). Clearly such numerical derivatives are not unique and the selection of appropriate fd-formulas affects the stencil of the resulting reconstruction. Note that in the finite

volume context, the available data are averages instead of point values. For example, consider cells centered around grid points $x_i = ih$. Based on cell averages

$$f_i = \frac{1}{h}\int_{x_{i-1/2}}^{x_{i+1/2}} f(\xi)d\xi$$

we need to approximate the point value of the derivative $f'(x_{i-1/2})$ on the cell boundary. A backward 3-point fd-formula would look like

$$\frac{\alpha f_{i-3} + \beta f_{i-2} + \gamma f_{i-1}}{h} \approx f'(x_{i-1/2}). \tag{50}$$

In terms of the primitive of f, the averages are expressed as divided differences

$$f_i = \frac{1}{h}\left(F_{i+1/2} - F_{i-1/2}\right),$$

where $F_{i+1/2} = F(x_{i+1/2})$ denotes the point value of the primitive $F' = f$ of f. Now (50) reads

$$\frac{-\alpha F_{i-7/2} + (\alpha-\beta)F_{i-5/2} + (\beta-\gamma)F_{i-3/2} + \gamma F_{i-1/2}}{h^2} \approx F''(x_{i-1/2}).$$

Applying the well-known second-order backward 4-point formula for point values, we find $\alpha = 1$, $\alpha - \beta = 4$, $\beta - \gamma = -5$, and $\gamma = 2$. The resulting conservative backward 3-point formula reads

$$d^-_{i-1/2} = \frac{f_{i-3} - 3f_{i-2} + 2f_{i-1}}{h} = f'(x_{i-1/2}) + \mathcal{O}(h^2). \tag{51}$$

Similarly the conservative central and forward formulas of second order for $f'(x_{i-1/2})$ are

$$d^0_{i-1/2} = \frac{f_i - f_{i-1}}{h}, \tag{52}$$

$$d^+_{i-1/2} = \frac{-2f_i + 3f_{i+1} - f_{i+2}}{h}. \tag{53}$$

Below, conservative second-order formulas for the second derivative $f''(x_{i-1/2})$ are also required:

$$D^-_{i-1/2} = \frac{-3f_{i-4} + 11f_{i-3} - 13f_{i-2} + 5f_{i-1}}{2h^2}, \tag{54}$$

$$D^0_{i-1/2} = \frac{f_{i-2} - f_{i-1} - f_i + f_{i+1}}{2h^2}, \tag{55}$$

$$D^+_{i-1/2} = \frac{5f_i - 13f_{i+1} + 11f_{i+2} - 3f_{i+3}}{2h^2}. \tag{56}$$

In fact, to establish a fifth-order DLR, the lateral first derivatives are required in fourth order of accuracy. In principle, such formulas can be derived via the primitive as well. However, to avoid numerical differentiation across discontinuities, we prefer to establish fourth order by convex combination of the three second-order formulas $d^-_{i-1/2}$, $d^0_{i-1/2}$, and $d^+_{i-1/2}$. Such a convex combination allows one to weight the influence of neighboring cells adaptively depending on the local smoothness of the solution, just as in WENO reconstructions. This will be explained at a generic cell

boundary $x_{i-1/2}$; thus the special index $i-1/2$ can be omitted in what follows. A convex combination of the conservative fd formulas reads

$$d = c^- d^- + c^0 d^0 + c^+ d^+ \tag{57}$$

with nonnegative weights, summing up to one. The optimal weights are found to be

$$c^- = c^+ = 1/24, \quad c^0 = 11/12. \tag{58}$$

They eliminate the second- and third-order terms in the truncation error $d_{i-1/2} - f'(x_{i-1/2})$, thus resulting in a fourth-order formula for the lateral derivatives, as required by Lemma 3.4. In the same way

$$D = c^- D^- + c^0 D^0 + c^+ D^+ \tag{59}$$

is a fourth-order conservative approximation to the second derivative $f''(x_{i-1/2})$ with the optimal weights

$$c^- = c^+ = 1/16, \quad c^0 = 7/8. \tag{60}$$

In fact, the reconstruction in Lemma 3.4 requires only third order of accuracy for this lateral second derivative. However, we found it convenient to use a three-term convex combination as for the first derivative. The robustness of the method improves by adapting the weights in both formulas (57) and (59) to the local smoothness of the solution. In this way, numerical differentiation across discontinuities is avoided. This concept calls for a [illegible]. In all our numerical experiments we use

$$s^* = (hd^*)^2 + (h^2 D^*)^2, \tag{61}$$

where * may be one of the superscripts –, 0, or +. The scaling with h is to make $s^* = \mathcal{O}(1)$ when a discontinuity is present. Similarly as in WENO reconstructions, the weights may be adapted by the sensor via

$$w^* = \frac{c^* \alpha^*}{c^- \alpha^- + c^0 \alpha^0 + c^+ \alpha^+}, \quad \alpha^* = \frac{h^2}{\epsilon + s^*}, \tag{62}$$

with $\epsilon \sim 10^{-8}$ in order to avoid division by zero. The resulting conservative fd approximations are

$$\delta = w^- \delta^- + w^0 \delta^0 + w^+ \delta^+, \tag{63}$$

where δ stands for discrete first- or second-order derivatives d or D, respectively. Note that, ideally, in smooth regions the weights satisfy $w^* = c^* + \mathcal{O}(h^2)$, implying formula (63) is fourth-order accurate. For the first derivative this can be seen by noticing that $d^* = f' + e^* + \mathcal{O}(h^3)$, where $e^* = \mathcal{O}(h^2)$. Consider then $\sum (w^* - c^*) d^* = f' \sum (w^* - c^*) + \sum (w^* - c^*) e^*$ and use that $\sum c^* = \sum w^* = 1$.

We demonstrate the behavior of the weights on the function proposed in [10] with two discontinuities in the first derivative:

$$u(x) = \begin{cases} \sin\left(\pi \frac{x+0.6}{1.2}\right), & |x| \le 0.6, \\ 0, & |x| > 0.6. \end{cases} \tag{64}$$

The weights w^0 and w^+ computed on a 40-point grid on $[-1, 1]$ are displayed in Figure 2. We observe that discrepancy from the optimal weights occurs for at most

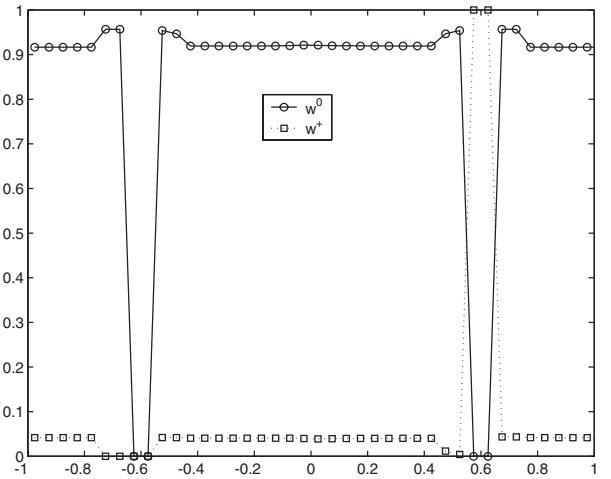


FIG. 2. *Weights for a function with corners.*

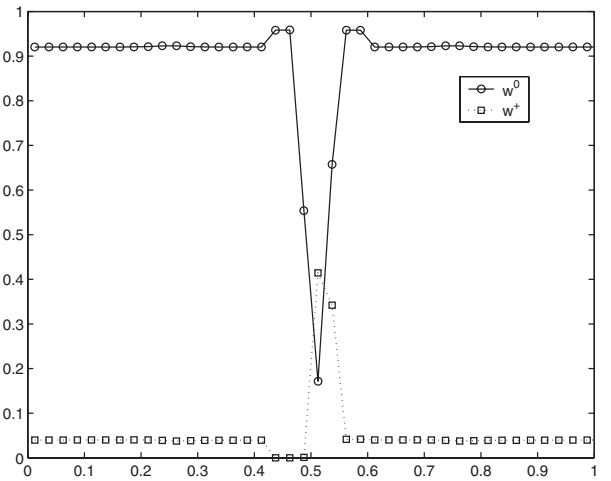


FIG. 3. *Weights for a function with jump discontinuity.*

three points next to the corners. The computations in [10] show that this is one point more compared to the Jiang–Shu WENO5 and two points more than the power ENO method developed there. The weights (62), however, have calmer behavior around the extrema at $x = 0$ than the weights for the above-mentioned methods.

Next we test the performance of the smoothness indicator on the function suggested in [4] with a jump discontinuity at zero:

$$u(x) = \begin{cases} \sin(2\pi x), & 0 \le x \le 0.5, \\ 1 - \sin(2\pi x), & 0.5 < x \le 1. \end{cases} \tag{65}$$

Depicted in Figure 3 are the weights w^0 and w^+ computed using 40 points. We note that the weight w^0 has not assumed the value zero at the discontinuity; increasing the number of points makes the selection more digital, showing that the indicator is not in its asymptotic regime with this coarse grid. As in the previous example we

note the calm behavior at the smooth local extrema.

7. Numerical results. In this section numerical experiments with the local logarithmic and the double logarithmic reconstructions are presented. We use the third-order strong stability-preserving (SSP) Runge–Kutta [1] time integrator in all simulations. For all DLR approximations we set $\epsilon = 10^{-8}$ in (62) and the weights w^* are adapted componentwise.

7.1. Convergence and accuracy tests.

1. A convergence test. We test the rate of convergence for the DLR with a as the governing equation. Consider the Euler system

$$u_t + f(u)_x = 0$$

with

$$u = (\rho, \rho v, E), \quad f(u) = (\rho v, \rho v^2 + p, v(E + p)),$$

and

$$p = (\gamma - 1)(e - \rho u^2/2), \quad \gamma = 1.4.$$

With initial data $\rho(0, x) = 1 + 0.2\sin(x)$, $v \equiv 1$, and $p \equiv 1$ the Euler system has the solution $\rho(t, x) = 1 + 0.2\sin(x - t)$, $v \equiv 1$, and $p \equiv 1$ [7]. We compute approximations to the solution at $t = 1$, on the interval $-\pi \leq x \leq \pi$ with periodic boundary conditions, by double logarithmic reconstruction of the conserved variables in the Van Leer upwind scheme. To reach fifth order in time we take $\Delta t \sim \Delta x^{5/3}$. The error in density is displayed in Table 2 (see also Figure 1 in the introduction). Full of convergence is clearly verified.

TABLE 2
Euler system. Absolute errors and convergence rates for DLR.

N	L_1-error	L_1-order	L_∞-error	L_∞-order
10	5.33e-03	-	1.22e-03	-
20	2.13e-04	4.64	7.33e-05	4.06
40	7.25e-06	4.88	2.88e-06	4.67
80	2.32e-07	4.97	9.49e-08	4.93
160	7.24e-09	5.00	2.77e-09	5.10
320	2.26e-10	5.00	6.45e-11	5.42
640	7.03e-12	5.00	2.07e-12	4.96

2. Linear advection. We solve the periodic advection problem

$$u_t + u_x = 0, \quad -\pi < x < \pi,$$

with initial data $u(x, 0) = \sin^4(x)$ and compute the errors at $t = 1$. The results, presented in Table 3, show that, in contrast to ENO schemes, which have problems reaching their designed order for this initial condition [11], full order of convergence is reached.

3. Linear advection. The equation

$$u_t + u_x = 0, \quad -1 < x < 1,$$

is solved with periodic boundary condition and initial data as proposed in [4]. The signal consists of a smooth and narrow combination of Gaussians, a square wave, a

TABLE 3
Linear advection of $\sin^4$*. Absolute errors and convergence rates for DLR.*

N	L_1-error	L_1-order	L_∞-error	L_∞-order
10	4.56e-01	-	1.78e-01	-
20	1.34e-01	1.77	3.90e-02	2.19
40	8.55e-03	3.97	3.12e-03	3.65
80	8.58e-04	3.32	5.33e-04	2.55
160	6.79e-05	3.66	7.47e-05	2.84
320	1.71e-06	5.32	2.36e-06	4.98
640	4.46e-08	5.26	5.48e-08	5.43
1280	1.26e-09	5.14	5.14e-09	3.41

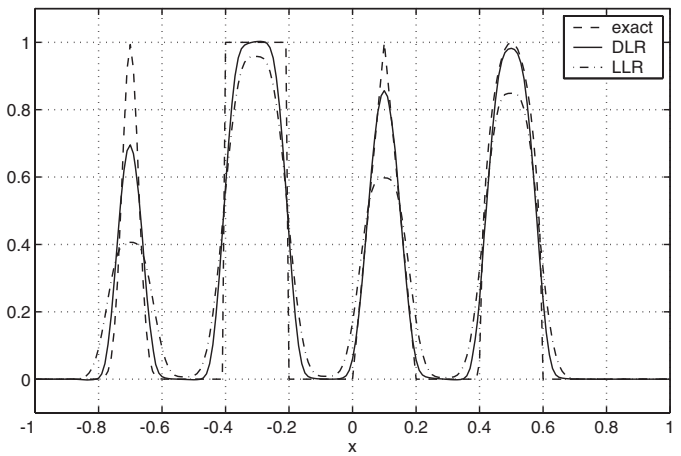


FIG. 4. *Advection.*

triangle wave, and a half ellipse. We refer the reader to [4] for the details. We compute LLR and DLR approximations with 200 cells using CFL $= 0.6$ and examine the results at $t = 8$. In Figure 4 we clearly see the improved quality for DLR compared to LLR. Comparing the DLR approximation to the fifth-order WENO approximation in [4], we note that the first peak is not so well resolved for DLR but that DLR improves over WENO in the resolution of corners.

To get an idea of the existence of the double logarithmic approximating function, which is hard to tell from (39)–(40) and (43)–(45) (especially in the limit $h \to 0$), we compute the DLR coefficients for the signal above and check whether the coefficients a or b are in the allowable range or not. In Figure 5 the result using one million points is displayed; cells where DLR fails to exist are marked. We observe that the DLR exists in this example, even for the extremely small cell size, wherever the second derivatives are nonzero and bounded. For the linear parts DLR may fail to exist, as expected when inspecting a_0 in (45) with data $d_1 = d_2$ and $D_2 = 0$. However, in this case it is overly ambitious to approximate the linear function with DLR since this is accurately done with the then active LLR; cf. Lemma 5.6 or the functions (7)–(8) and the removable singularity at zero.

4 (Buckley–Leverett). The nonlinear and nonconvex scalar Buckley–Leverett problem

$$u_t + \left(\frac{4u^2}{4u^2 + (1-u)^2} \right)_x = 0$$

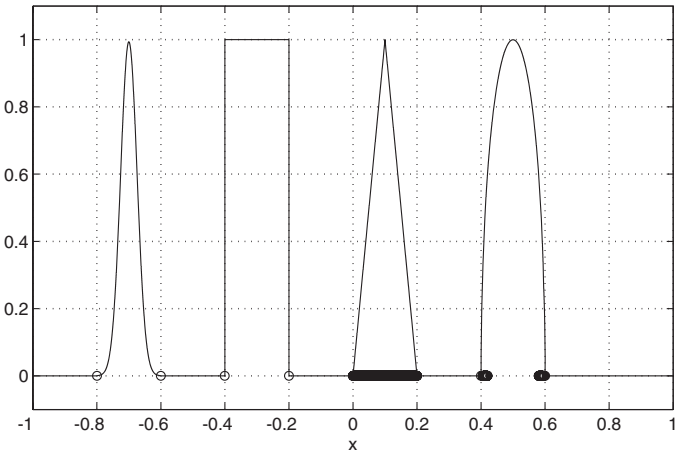


FIG. 5. *Existence of DLR. "o" marks nonexistence.*

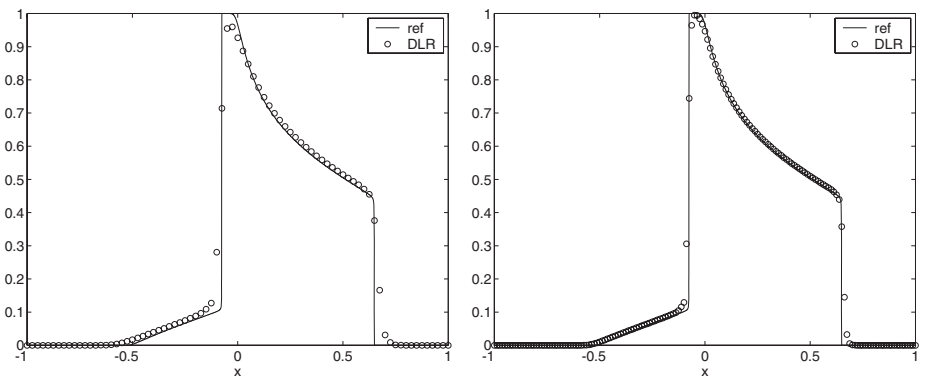


FIG. 6. *Buckley–Leverett. Left:* 80 *cells. Right:* 160 *cells.*

is solved using a relaxation method [8] with double logarithmic reconstruction of the characteristic variables of the relaxed system. The initial data is $u = 1$ for $-\frac{1}{2} \le x \le 0$ and $u = 0$ elsewhere. Approximations using 80 and 160 cells with CFL $= 0.7$ are shown in Figure 6. We conclude that it converges to the correct entropy solution.

7.2. Euler system.

1 (Shu–Osher shock-acoustic problem). In this test problem proposed by Shu and Osher [12] a supersonic shock wave moves into a smooth acoustic entropy wave. The initial data are

$$u(x,0) = (\rho, v, p) = \begin{cases} (3.857143, 2.629369, 10.333333), & -5 \le x < -5, \\ (1 + 0.2\sin(5x), 0, 1), & -4 \le x \le 5. \end{cases}$$

Behind the shock the acoustic wave is amplified. We run the experiment with componentwise reconstruction of the conserved variables in connection with Van Leer flux splitting. In Figure 7 DLR and LLR approximations to the shock-acoustic problem are displayed. The simulations are done with 400 cells using CFL $= 0.8$ in both cases. The benefit of reconstruction of smooth extrema is quite obvious in this test problem and we see that the DLR approximation resolves the waves behind the shock well.

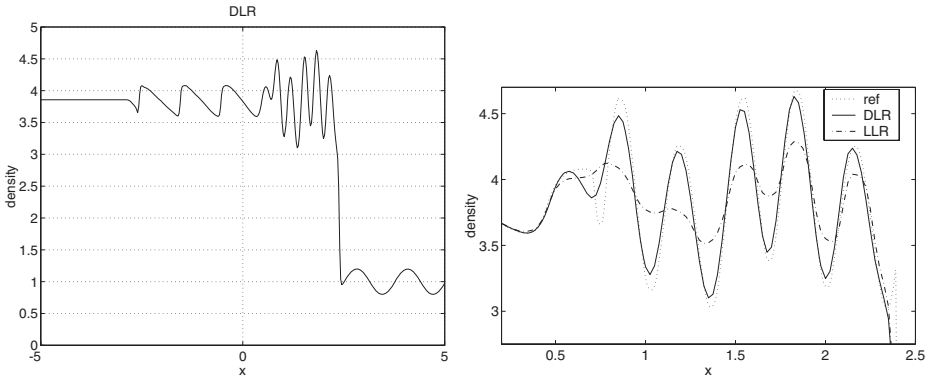


FIG. 7. *Shu–Osher shock-acoustic problem.*

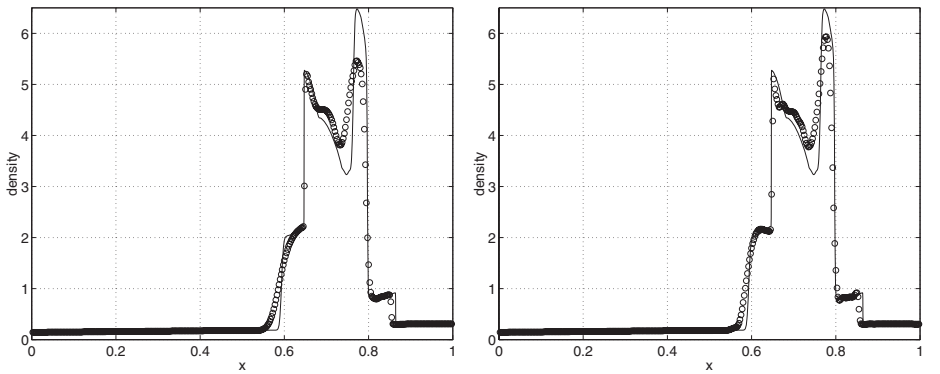


FIG. 8. *Shock-blast interaction. Left: LLR. Right: DLR.*

, , 2 (two interacting blast waves). This experiment was proposed as a benchmark in [14]. The initial data are

$$u(x,0) = (\rho, v, p) = \begin{cases} (1, 0, 10^3), & 0 \le x < 0.1, \\ (1, 0, 10^{-2}), & 0.1 \le x < 0.9, \\ (1, 0, 10^2), & 0.9 \le x \le 1. \end{cases}$$

Reflective boundary conditions are used. Numerical solutions on a 400-cell grid are computed with LLR and DLR applied to the Van Leer fluxes under CFL = 0.6. In Figure 8 the approximations at $t = 0.038$ are displayed. It is observed that the local extrema, especially the peak, are better resolved using the DLR but that there are also small oscillations present, indicating that the DLR introduces rather too much variation in this context. The LHR leads to a negative density in this test example. The reference solution, however, is obtained by the harmonic LHHR using 3200 cells.

We close this section by presenting some preliminary results in two space dimensions. Both experiments presented here are flows in rectangular domains simulated on Cartesian meshes. In this situation the one-dimensional reconstructions can be applied directionwise.

, , 3 (two-dimensional Riemann problem). This is an example from gas dynamics, Configuration F in [9]. In contrast to one-dimensional Riemann problems,

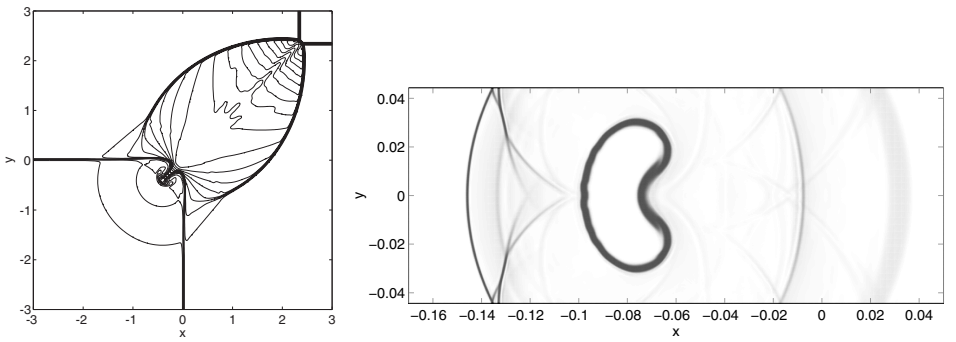


FIG. 9. *Left: Two-dimensional Euler Riemann problem. Right: Shock-bubble interaction.*

the density contours presented in Figure 9 develop some quite complex structures, demanding higher-order methods. For this simulation a locally relaxed method [8] with artificial compression on a 400×400 mesh is used. The LLR is applied on the characteristic variables of the relaxation approximation, as described in [8]. For easy comparison the contour levels are chosen as in [9].

' . ,' 4 (a two component shock-bubble interaction). The initial configuration is an air-helium bubble with a 1.22 Mach shock situated to the right of it [2]. In Figure 9 a locally relaxed LLR approximation at $t = 260\,\mu$s using 1600×400 cells is shown; cf. [3]. By comparing certain time-averaged velocities this simulation shows good agreement with the experimental study in [2] and the numerical results found in [6].

8. Summary and comments. In this paper a monotone third-order LLR and a fifth-order reconstruction DLR were developed, inspired by the LHR by Marquina. The LLR has a smaller local variation than the LHR and is therefore more robust; see the shock-blast experiment (Example 2, section 7.2). Marquina's harmonic version of LHR, the LHHR, is locally TV-bounded and it produces good results; as with LLR, corners are especially well resolved.

The main achievement is, however, that logarithmic reconstructions can be extended to fifth order of accuracy. The DLR based on a WENO-type stencil is a fifth-order reconstruction, even at local extrema. The choice of the smoothness indicator is, of course, essential. Despite the simplicity of the current indicator the numerical results were of high quality, but there is room for improvement on the sensor.

To construct a fifth-order accurate reconstruction using the minimal 5-point stencil one can consider a hybrid version of DLR and LLR. Then (central) conservative fds of high order are needed. Examples are realization 2 with (66)–(68) and realization 3 with (69)–(71). For this hybrid scheme a well-functioning switch criterion is needed; the necessary (problem-dependent) threshold value associated with a hybrid scheme on a fixed stencil will of course be a drawback.

9. Appendix. Technical details.

9.1. Conservative fd-formulas. Realization 2:

$$f'(x_{i-1/2}) = \frac{-f_{i+1} + 15f_i - 15f_{i-1} + f_{i-2}}{12h} + \mathcal{O}(h^4), \tag{66}$$

$$f'(x_i) = \frac{-5f_{i+2} + 34f_{i+1} - 34f_{i-1} + 5f_{i-2}}{48h} + \mathcal{O}(h^4), \tag{67}$$

$$f''(x_i) = \frac{-f_{i+2} + 12f_{i+1} - 22f_i + 12f_{i-1} - f_{i-2}}{8h^2} + \mathcal{O}(h^4). \tag{68}$$

Realization 3:

$$f'(x_{i-1/2}) = \frac{-f_{i+1} + 15f_i - 15f_{i-1} + f_{i-2}}{12h} + \mathcal{O}(h^4), \tag{69}$$

$$f'(x_{i-1/6}) = \frac{-25f_{i+2} + 145f_{i+1} + 147f_i - 305f_{i-1} + 38f_{i-2}}{324h} + \mathcal{O}(h^4), \tag{70}$$

$$f'(x_{i+1/6}) = \frac{-38f_{i+2} + 305f_{i+1} - 147f_i - 145f_{i-1} + 25f_{i-2}}{324h} + \mathcal{O}(h^4). \tag{71}$$

9.2. DLR coefficients.

9.2.1. Realization 2. Denote by d_0 and D_0 the approximation to middle derivative and the middle second derivative, respectively. Then

$$\begin{aligned} a_1 &= [(3d_2 - d_1)D_0h + 8d_0^2 - (6d_1 + 10d_2)d_0 + 8d_1d_2]/a_0, \\ a_2 &= [D_0^2(d_1 + d_2)^2h^2 + 4D_0(d_2 - d_1)(-3d_1d_0 + 4d_1d_2 - 3d_2d_0)h \\ &\quad - 32(d_1 + d_2)d_0^3 + 12(10d_1d_2 + 3d_1^2 + 3d_2^2)d_0^2 \\ &\quad - 96(d_1^2d_2 + d_2^2d_1)d_0 + 64d_1^2d_2^2]^{1/2}/a_0 \end{aligned} \tag{72, 73}$$

with

$$a_0 = 2(2d_0(d_2 - d_0) - D_0hd_2), \tag{74}$$

and q is the function

$$\begin{aligned} q(z) &= \frac{1}{2}\frac{(2d_2d_0 - 2d_1d_2 + d_1d_0)z^2 + 2(4d_1d_2 - d_1d_0 - 3d_2d_0)z}{d_0(z-2)^2 - 2d_2(z-1)^2 - 2d_1} \\ &\quad + \frac{d_1^2 - 3d_1d_2 + 2d_2d_0}{d_0(z-2)^2 - 2d_2(z-1)^2 - 2d_1}. \end{aligned} \tag{75}$$

9.2.2. Realization 3. Denote by d_3 the conservative fd-approximation to $f'(x_{i-1/6})$ and by d_4 the approximation to $f'(x_{i+1/6})$. Then

$$\begin{aligned} a_1 &= [(-9d_2 + 8d_4 - d_3)d_1 + (16d_3 - 5d_4)d_2 - 9d_4d_3]/a_0, \\ a_2 &= [(81d_2^2 + 18(-8d_4 + d_3)d_2 + 16d_4(4d_4 - d_3) + d_3^2)d_1^2 \\ &\quad + (18(d_4 - 8d_3)d_2^2 + 4(4(d_3^2 + d_4^2) + 35d_4d_3)d_2 - 6d_3d_4(d_3 + 8d_4))d_1 \\ &\quad + (-16d_4d_3 + d_4^2 + 64d_3^2)d_2^2 - 6d_3d_4(8d_3 + d_4)d_2 + 9d_4^2d_3^2]^{1/2}/a_0 \end{aligned} \tag{76, 77}$$

with

$$a_0 = 4[(2d_3 - d_4)d_2 - d_4d_3], \tag{78}$$

and q is the function

$$\begin{aligned} q(z) &= \frac{1}{2}\frac{(3d_1d_2 - d_1d_3 - 4d_2d_3)z^2 + 2(-9d_1d_2 + d_1d_3 + 8d_2d_3)z}{(3d_2 - d_3)z^2 + 6(d_3 - d_2)z + 3(d_2 - 3d_3 + 2d_1)} \\ &\quad + \frac{1}{2}\frac{3(-2d_1^2 + (d_3 + 5d_2)d_1 - 4d_2d_3)}{(3d_2 - d_3)z^2 + 6(d_3 - d_2)z + 3(d_2 - 3d_3 + 2d_1)}. \end{aligned} \tag{79}$$

Acknowledgments. We thank the referees for several useful suggestions.

REFERENCES

[1] S. Gottlieb, C.-W. Shu, and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Rev., 43 (2001), pp. 89–112.

[2] J.-F. Haas and B. Sturtevant, *Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities*, J. Fluid Mech., 181 (1987), pp. 41–76.

[3] O. Hall, *The shock-bubble problem computed with three different reconstruction methods*, Diploma thesis, Lund University, Lund, Sweden, 2004.

[4] G.-S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[5] A. Marquina, *Local piecewise hyperbolic reconstruction of numerical fluxes for nonlinear scalar conservation laws*, SIAM J. Sci. Comput., 15 (1994), pp. 892–915.

[6] A. Marquina and P. Mulet, *A flux split algorithm applied to conservative models for multicomponent compressible flows*, J. Comput. Phys., 185 (2003), pp. 120–138.

[7] J. Qiu and C.-W. Shu, *On the construction, comparison, and local characteristic decomposition for high-order central WENO schemes*, J. Comput. Phys., 183 (2002), pp. 187–209.

[8] H. J. Schroll, *High resolution schemes for hyperbolic conservation laws*, J. Sci. Comput., 21 (2004), pp. 251–279.

[9] C. W. Schulz-Rinne, J. P. Collins, and H. M. Glaz, *Numerical solution of the Riemann problem for two-dimensional gas dynamics*, SIAM J. Sci. Comput., 14 (1993), pp. 1394–1414.

[10] S. Serna and A. Marquina, *Power ENO methods: A fifth-order accurate weighted power ENO method*, J. Comput. Phys., 194 (2004), pp. 632–658.

[11] C.-W. Shu, *Numerical experiments on the accuracy of ENO and modified ENO schemes*, J. Sci. Comput., 5 (1990), pp. 127–150.

[12] C.-W. Shu and S. Osher, *Efficient implementation of essentially nonoscillatory shock-capturing schemes,* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[13] B. van Leer, *Towards the ultimate conservative difference scheme* V. *A second order sequel to Godunov's method*, J. Comput. Phys., 32 (1979), pp. 101–136.

[14] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 315–345
© 2005 Society for Industrial and Applied Mathematics

A SEMI-LAGRANGIAN APPROACH FOR AMERICAN ASIAN OPTIONS UNDER JUMP DIFFUSION*

Y. D'HALLUIN†, P. A. FORSYTH†, AND G. LABAHN†

Abstract. A semi-Lagrangian method is presented to price continuously observed fixed strike Asian options. At each timestep a set of one-dimensional partial integrodifferential equations (PIDEs) is solved and the solution of each PIDE is updated using semi-Lagrangian timestepping. Crank–Nicolson and second-order backward differencing timestepping schemes are studied. Monotonicity and stability results are derived. With low volatility values, it is observed that the nonsmoothness at the strike in the payoff affects the convergence rate; a subquadratic convergence rate is observed.

Key words. continuously observed Asian option, semi-Lagrangian, American option, jump diffusion, implicit discretization

AMS subject classifications. 65M12, 65M60, 91B28

DOI. 10.1137/030602630

1. Introduction.

An Asian option gives the holder a payoff that depends on the average price of the underlying asset over a specified period of time [12]. Asian-style derivatives have a wide variety of applications in equity, energy, interest rate, and insurance markets. These options tend to be less expensive than vanilla options (since the volatility of the average price is less than the price itself) and average prices over a period of time are far more difficult to manipulate for illiquid commodities. For example, airline companies are certainly more interested in buying oil based on its average price instead of its spot price. For a historical review of Asian options we refer the reader to [11].

The price of an Asian option at any time is a function of both the underlying asset at that time and the average of the underlying prices up to that time. As such these options are considered path-dependent. In practice, Asian option contracts typically specify that the average is monitored discretely. A typical situation would be to base the average on the daily closing price. If daily averaging is used, then for typical market parameters, for options with expiry times more than three months, we can consider these options as being continuously monitored, for all practical purposes. In addition, if we need to price long-term Asian options (for example, greater than one year), then using timesteps of one day (which would be required in a discrete observation model [51]) would clearly be computationally wasteful. Consequently, in this paper we focus on continuously observed Asian options. For details on numerical methods for discretely observed Asian options, we refer the reader to [17, 16, 51, 24].

In this paper we are interested in the pricing of Asian options with various general payoff conditions (for example, American style, asset-dependent barriers or volatility). When the underlying asset follows a standard Brownian-motion stochastic process then the price of an Asian option can be determined by solving a two-dimensional PDE. In some special cases (e.g., constant volatility, no barrier features, and a float-

*Received by the editors December 19, 2003; accepted for publication (in revised form) September 1, 2004; published electronically September 12, 2005. This work was supported by the Natural Sciences and Engineering Research Council of Canada and by Bell University Laboratories at the University of Waterloo.

http://www.siam.org/journals/sisc/27-1/60263.html

†School of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1 (ydhallui@uwaterloo.ca, paforsyt@uwaterloo.ca, glabahn@uwaterloo.ca).

ing strike contract) this problem can be reduced to a one-dimensional PDE [4]. In addition, for either floating or fixed strike, but not American style or asset-dependent features (e.g., volatility a function of asset price), a one-dimensional PDE can also be derived [45]. However, in the general case which we are interested in, the two-dimensional PDE cannot be reduced to one dimension.

The two-dimensional PDE that appears in our general case of the Asian option pricing problem has no diffusion in one of the coordinate directions and as such is well known to be difficult to solve numerically. In [50, 49], a flux limiter was used to retain accuracy while preventing oscillations. In [31], the first-order hyperbolic term was discretized using a first-order upwind type method, resulting in at most first-order accuracy. A related approach based on a combination of a WENO discretization and grid stretching was used for Asian options in [36]. In [37], a semi-Lagrangian method was used to discretize the hyperbolic term in the average direction. Semi-Lagrangian schemes were first introduced in [20, 39] for atmospheric and weather numerical predictions. These are time-marching schemes that integrate convection-diffusion equations by tracing backward in time the position of the flow. These schemes are used to reduce numerical problems raised by convection dominated equations. In principle, provided an appropriate time discretization is used, and a high enough order of interpolation is used to recover values at the feet of the characteristic curves [22, 1, 10], then this method is capable of greater than first-order convergence as the grid and timestep size is reduced.

In this paper, we will explore the use of a semi-Lagrangian method for pricing Asian options. The semi-Lagrangian method has many advantages in our case. For example we are able to easily solve the pricing problem in the more general context of a jump diffusion stochastic processes, and contracts with American early exercise features. Jump diffusion models were introduced in the option valuation context in [32]. They are important because of the increasing empirical evidence that the usual assumption of geometric Brownian motion should be augmented by discontinuous jump processes [21].

We make no assumptions regarding the contract or the form of the deterministic volatility function. We remark that although we focus exclusively on Asian options in this paper, similar PDEs (no diffusion in one of the space-like directions) occur in certain interest rate models [43] and employee reload options with various constraints [9]. Hence the methods developed here will be applicable to these cases as well.

The main results in this paper are the following.

- We demonstrate that a semi-Lagrangian method can be used to price continuously observed American Asian options under jump diffusion processes. The implementation suggested here reduces this problem to solving a decoupled set of one-dimensional nonlinear discrete partial integrodifferential equations (PIDEs) at each timestep. This makes implementation of this method very straightforward in a software library which is capable of pricing discretely observed path-dependent options [51]. In the general case, a numerical method must be used to integrate the characteristic equation in a semi-Lagrangian method. However, in the case of Asian options, the characteristic equation can be solved analytically, which makes implementation very straightforward.
- We show that in the fully implicit case, the semi-Lagrangian method is algebraically identical to a standard numerical method for pricing discretely observed Asian options, if the observation interval is equal to the discrete timestep. Since lattice methods [46] can be regarded as explicit finite

difference methods, it follows that the usual binomial forest method for Asian options [27] can also be regarded as an explicit semi-Lagrangian method.

- Since the discretized problem at each timestep reduces to a set of decoupled one-dimensional PIDEs, we can make use of the techniques developed in [23, 18, 19] to prove certain properties of the discrete scheme, including convergence of the iterative method used to solve the implicit discrete equations. In the fully implicit case, it is straightforward to prove l_∞ stability and monotonicity, which are important properties of discrete schemes for option pricing [5, 40, 15, 13].

In addition to the above contributions, we also include experimental computations which indicate that, even if second-order timestepping methods are used, observed convergence as the mesh and timestep sizes are refined occurs at a subsecond-order rate. The problem can be traced to the nonsmoothness of the payoff function.

2. Mathematical model. In this section we give the mathematical model for options with jump diffusion processes. We do this for both European and American options. If the underlying asset follows a jump diffusion process, the usual portfolio hedging arguments cannot be used. As such, we will also present a brief discussion of various strategies for hedging jump risk.

Let S represent the underlying stock price. We restrict attention in the following to finite activity processes, that is, we assume that the probability density of a jump occurring with a given jump size, in some interval $[t, t+dt]$ is always finite [14]. The potential stock paths followed by the stock can be modeled by a stochastic differential equation given by

$$\frac{dS}{S} = (\xi - \kappa\lambda)dt + \sigma dZ + (\eta - 1)dq, \tag{2.1}$$

where

ξ is the drift rate,

dq is the independent Poisson process $= \begin{cases} 0 & \text{with probability } 1 - \lambda dt, \\ 1 & \text{with probability } \lambda dt, \end{cases}$

λ is the mean arrival time of the Poisson process,

$\eta - 1$ is an impulse function producing a jump from S to $S\eta$,

σ is the volatility,

dZ is an increment of the standard Gauss–Wiener process,

κ is $E[\eta - 1]$, where $E[\cdot]$ is the expectation operator.

When the average is monitored continuously [7, 46, 24], the arithmetic average A is defined as

$$A = \frac{\int_0^t S(u)du}{t} \quad \text{with} \quad dA = \frac{(S - A)}{t}dt. \tag{2.2}$$

2.1. The PIDE for Asian options. Using standard arguments [7], the value of an option depending on S, (2.1), and A, (2.2), and assuming no jumps (that is, $\lambda = 0$) is given by

$$V_t + \frac{1}{2}\sigma^2 S^2 V_{SS} + \frac{(S - A)}{t}V_A + rSV_S - rV = 0, \tag{2.3}$$

where r is the continuously compounded risk-free interest rate. Since we are solving backward in time from the expiration time $t = T$ to the present time $t = 0$, (2.3) becomes

$$V_\tau = \frac{1}{2}\sigma^2 S^2 V_{SS} + \frac{(S-A)}{T-\tau}V_A + rSV_S - rV, \tag{2.4}$$

where $\tau = T - t$. It is important to note that (2.4) has no diffusion term in the A direction and this is the source of many numerical difficulties [50].

Extending (2.4) to the case of jumps gives

$$V_\tau = \frac{(S-A)}{T-\tau}V_A + \frac{1}{2}\sigma^2 S^2 V_{SS} + (r - \lambda\kappa)SV_S - rV + \left(\lambda \int_0^\infty V(S\eta)g(\eta)d\eta - \lambda V\right), \tag{2.5}$$

where

$$\begin{aligned} &g(\eta) \quad \text{is the probability density function of the jump amplitude } \eta, \\ &\text{thus for all } \eta, g(\eta) \geq 0 \text{ and } \int_0^\infty g(\eta)d\eta = 1. \end{aligned} \tag{2.6}$$

As a specific example, if

$$g(\eta) = \frac{e^{\left(-\frac{(\log(\eta)-\mu)^2}{2\gamma^2}\right)}}{\sqrt{2\pi}\gamma\eta}, \tag{2.7}$$

the probability density function suggested by [32, 44], then its expectation is given by $E[\eta] = \exp(\mu + \gamma^2/2)$. This means that the expected relative change in the stock price is given by $\kappa = E[\eta - 1] = \exp(\mu + \gamma^2/2) - 1$.

For brevity, the details of the derivation of (2.5) have been omitted (see [3, 32, 46]). In general, it is not possible to construct a hedging portfolio which eliminates jump risk. However, by adding options to the hedging portfolio, a hedging strategy can be constructed which minimizes jump risk [3]. If (2.5) is calibrated to market prices, then the parameters so obtained should be regarded as risk-adjusted [30], not historical.

If we define

$$\mathcal{H}V \equiv V_\tau - \left[\frac{(S-A)}{T-\tau}V_A + \frac{\sigma^2 S^2}{2}V_{SS} + (r - \lambda\kappa)SV_S - (r+\lambda)V + \lambda\int_0^\infty V(S\eta)g(\eta)d\eta\right] \tag{2.8}$$

and if $V^*(S, A)$ is the payoff, then the American option pricing problem can be stated as [38]

$$\min(\mathcal{H}V; V - V^*) = 0. \tag{2.9}$$

2.2. Boundary conditions for our PIDE. In order to completely specify our problem we still need to give boundary conditions for our American Asian option pricing PIDE. For the terminal boundary conditions, a number of common payoffs for pricing different types of Asian securities can be used. Typical examples include

- fixed strike call: $V(S, A, \tau = 0) = \max(A - K, 0.0)$,
- fixed strike put: $V(S, A, \tau = 0) = \max(K - A, 0.0)$.

Although the original domain is $0 \leq S < \infty$, $0 \leq A < \infty$, for computational purposes we make the typical assumption of using a finite computational domain $[0, S_{max}] \times [0, A_{max}]$, with $A_{max} = S_{max}$. A discussion of the errors involved in using a finite computational domain (the localization error) for the specific case of option pricing PIDEs is discussed in [15]. However, as pointed out in [15, 13], unlike the usual option pricing PDE case, it is necessary to approximate the solution outside the computational domain $S > S_{max}$ in order that the integral in (2.5) is well defined. The results in [15] suggest that the error in misspecification of the boundary condition at $S = S_{max}$ (for values of S of practical interest) can be made arbitrarily small by making S_{max} (and hence $A_{max} = S_{max}$) sufficiently large. This will be verified in some numerical tests in section 8.

In the case of Asian options without jumps, (2.4), Meyer [33] provides an extensive discussion of the boundary conditions. Using the results in [35], Meyer concludes that no boundary conditions are required at $S = 0$, $A = 0$, $A = A_{max}$. Intuitively, this is because the PDE degenerates on the boundary in such a way that either the PDE becomes an ODE, or the resulting degenerate parabolic PDE has a normal hyperbolic term with outgoing characteristics. Note that rigorous arguments justifying this result can be deduced from [35].

We now turn our attention to the complete PIDE with a nonzero jump term (2.5). We note that the jump integral term contains no A dependence and so consequently, we can simply solve the PIDE along the boundaries $A = A_{max}$, $A = 0$ as in the no-jump case. In this case, the [illegible] for the PIDE is simply the continuous limit of the PIDE as $A \to 0$, $A \to A_{max}$, from points interior to the computational domain.

Taking the limit of the PIDE (2.5) as $S \to 0$, we obtain

$$\lim_{S \to 0} \mathcal{H}V = V_\tau - \left[\frac{-A}{T - \tau} V_A - rV \right]. \tag{2.10}$$

Note that the above limit is obtained by formally interchanging the limit and the integral in (2.5). Equation (2.10) is, of course, the same limit as in the PDE (no-jump) case, since the integral terms disappear as $S \to 0$. Again, in this case, the boundary condition is simply the continuous limit of the PIDE as $S \to 0$. We will develop consistent, stable, and monotone discrete schemes which do not require any data outside the computational domain at $S = 0$, and hence satisfaction of the boundary condition as $S = 0$ is ensured (see the discussion of viscosity solutions in subsection 2.3).

For the boundary condition as $S \to \infty$ we need to deal with two major issues. The first problem is that there is no obvious Dirichlet-type condition which can be imposed for S large. The second issue concerns the fact that on any finite domain $[0, S_{max}]$, the integral term appears to require information from outside the computational domain.

If we make the common assumption that $V_{SS} \to 0$ as $S \to \infty$ [26], then this implies that

$$V \simeq f(A, \tau)S + g(A, \tau) \tag{2.11}$$

as $S \to \infty$ which then means that (2.8) becomes

$$\mathcal{H}V \simeq V_\tau - \left(\frac{S - A}{T - \tau} V_A + rSV_S - rV \right); \quad S \to \infty. \tag{2.12}$$

For $A \gg K$ and $S \to \infty$, we can approximate the solution to $\mathcal{H}V = 0$ by a linear function in A and S

$$V \simeq H_1(\tau)A + H_2(\tau)S + H_3(\tau) \tag{2.13}$$

so that

$$V \simeq \frac{D_1}{T}e^{-r\tau}(T-\tau)A + \left[\frac{D_1}{rT}(1-e^{-r\tau}) + D_2\right]S + D_3 e^{-r\tau}, \tag{2.14}$$

where D_1, D_2, D_3 are independent of (S, A, τ) and are determined by the payoff. For example, for a fixed strike call, $D_1 = 1$, $D_2 = 0$, $D_3 = -K$. Assuming (2.14) holds for $S \to \infty$, for all A (which is clearly an approximation for A small) we obtain

$$V_S \simeq \left[\frac{D_1}{rT}(1-e^{-r\tau}) + D_2\right]; \quad S \to \infty. \tag{2.15}$$

Substituting (2.15) into (2.12) gives

$$\mathcal{H}V \equiv V_\tau - \left(\frac{(S-A)}{T-\tau}V_A + \chi(S,\tau) - rV\right); \quad S \to \infty, \tag{2.16}$$

where

$$\chi(S,\tau) = \left[\frac{D_1}{rT}(1-e^{-r\tau}) + D_2\right]rS. \tag{2.17}$$

For the payoffs mentioned earlier, we have that $\chi(\tau, S) = \frac{(1-e^{-r\tau})}{T}S$ for a fixed strike call and $\chi(\tau, S) = 0$ for a fixed strike put.

The use of approximation (2.15) is discussed in [34], where it is mentioned that estimate (2.15) is in fact an upper bound for V_S. It must be admitted that use of (2.14) for all A as $S \to \infty$ is not rigorously justified. However, we note that other authors [31] simply specify that the boundary condition at $S = S_{max}$ is set to the payoff. In [31], the size of the computational domain is increased as the grid size is reduced, so that the effect of poor specification of the boundary condition becomes negligible. However, as discussed in [34], the boundary condition (2.16) is at least qualitatively correct.

In view of the above, we impose the boundary conditions as $S \to \infty$ in the following manner. We assume that S_{max} is sufficiently large so that the solution can be well approximated by a linear function of S in the region $[S_{max}-\delta, S_{max}]$. Provided that we choose δ carefully (see [19]), the integral term in the PIDE (2.8) in $[0, S_{max}-\delta]$ has sufficient data for accurate computation. In the region $[0, S_{max}-\delta] \times [0, A_{max}]$, the operator $\mathcal{H}$ is defined as in (2.8). In the region $[S_{max}-\delta, S_{max}] \times [0, A_{max}]$, we assume that V is linear in S, so that in this region the operator $\mathcal{H}V$ reduces to the PDE

$$\mathcal{H}V = V_\tau - \left[\frac{\sigma^2 S^2}{2}V_{SS} + \left(\frac{S-A}{T-\tau}\right)V_A + rSV_S - rV\right] \quad \text{for } S \in [S_{max}-\delta, S_{max}]. \tag{2.18}$$

Although we have assumed that $V_{SS} = 0$ in $[S_{max}-\delta, S_{max}]$, we leave the V_{SS} term in the operator (2.18) so that numerical solution is straightforward.

At $S = S_{max}$, based on (2.16), we define $\mathcal{H}V$ as

$$\mathcal{H}V \equiv V_\tau - \left(\frac{S-A}{T-\tau} V_A + \chi(S,\tau) - rV \right). \tag{2.19}$$

The above arguments can be used to justify formally solving the PIDE in the region $[0, S_{max}] \times [0, A_{max}]$, with the requirement that $\lambda = 0$ for $S \in [S_{max} - \delta, S_{max}]$. We then have a well-defined problem in $[0, S_{max}] \times [0, A_{max}]$, which does not require data outside the computational domain. Note that in $[S_{max} - \delta, S_{max}]$, only a few nodes are required, since the solution is assumed to be linear. Hence this method of handling the difficulties as $S \to \infty$ is not computationally expensive (at least as far as the solution of the PIDE is concerned).

2.3. Viscosity solutions. In general, there may be no smooth solutions to (2.9). In what follows it will be understood that we are seeking weak viscosity solutions. A detailed discussion concerning existence and uniqueness of viscosity solutions to (2.9), can be found in [38, 2], for European and American options under jump diffusion. In addition, sufficient conditions to ensure convergence of a discrete numerical scheme to the viscosity solution in the PDE case, are given in [6, 5]. Finally, an extension of the results in [6, 5] to the case of option pricing problems with integrodifferential equations (jump diffusions) is given in [13] for nonlinear, multidimensional, degenerate problems and in [15] for linear problems. By exploiting the equivalence of Binomial Trees and explicit finite difference methods, viscosity solution methods are used to prove convergence of a lattice method for Asian options in [28].

3. Semi-Lagrangian discretization. Typically when pricing continuously observed arithmetic average Asian option, the two-dimensional problem (2.4) must be solved. In [50], the authors use a finite volume approach combined with flux limiters to solve (2.4). This requires solution of a set of nonlinear discretized algebraic equations at each timestep. When the convection terms become very large (note that the convection term in the A direction in (2.5) becomes infinite as $\tau \to T$), flux limiters revert to a first-order upwind scheme which affects the accuracy of the solution [1]. A related approach was used for Asian options in [36]. In this paper we solve (2.4) using a semi-Lagrangian scheme. This idea was also suggested for American Asian options (without jumps but with stochastic volatility) in [37]. In this section we explore different discretization methods for the PDE using the semi-Lagrangian approach.

Before proceeding any further, let us introduce the following definitions. We use an unequally spaced grid in S coordinates for the PDE discretization $[S_1, \ldots, S_M]$, and similarly use an unequally spaced grid in the A direction $[A_1, \ldots A_M]$. Let

$$V_{i,j}^n = V(S_i, A_j, \tau^n) \tag{3.1}$$

denote the solution at asset price node S_i for the average A_j and time level n. Let $\mathcal{C}$ be the differential operator represented by

$$\mathcal{C}V = \frac{1}{2}\sigma^2 S^2 V_{SS} + (r - \lambda\kappa) S V_S - (r - \lambda) V \tag{3.2}$$

and

$$\mathcal{B}V = \lambda \int_0^\infty V(S\eta) g(\eta) d\eta. \tag{3.3}$$

Equation (2.5) can then be rewritten as

$$V_\tau + \frac{(A-S)}{T-\tau} V_A = \mathcal{C}V + \mathcal{B}V. \tag{3.4}$$

In order to impose the American early exercise constraint we will use a penalty method [18, 23, 48]. Briefly, this replaces (2.9) by a nonlinear PIDE

$$V_\tau + \frac{(A-S)}{T-\tau} V_A = \mathcal{C}V + \mathcal{B}V + q(V), \tag{3.5}$$

where the penalty term is

$$q(V) = \rho \max(V^* - V, 0). \tag{3.6}$$

The positive penalty parameter ρ is selected sufficiently large so that either $V \geq V^*$ or

$$\frac{|V - V^*|}{V^*} < \epsilon; \quad \epsilon \ll 1; \quad \text{when } V < V^*, V^* > 0. \tag{3.7}$$

After discretizing the PDE with the nonlinear penalty term, it is straightforward to select a suitably normalized discrete penalty parameter ρ', so that ϵ in (3.7) is less than a user specified tolerance. We refer the reader to [23] for more details about this method.

We use standard finite difference methods to discretize the operator $\mathcal{C}V$ [40] (see Appendix A). If we impose boundary condition (2.16), and use forward and backward differencing as appropriate, it is easy to see that the discrete form of $I - \mathcal{C}V$ is an M-matrix (see section 5.2). As discussed in [40], for typical values of σ, r, upwind differencing of the V_S term in (3.5) is required only rarely, and usually remote from regions of interest, so that in practice this does not impact solution quality. Requiring the discrete form of $\mathcal{C}V$ to be an M-matrix has interesting theoretical properties. In the following, we denote the discrete form of $\mathcal{C}V$ at $S = S_i$, $A = A_j$, $\tau = \tau^n$ by $(\mathcal{C}V)^n_{i,j}$. As described in [19, 18] the integral $(\mathcal{B}V)_{i,j}$ can be efficiently computed by transforming to equally spaced $\log S$ coordinates, approximating the integral using a Trapezoidal rule, using an FFT, and then transforming back to S coordinates. Special care is taken to avoid problems with wrap around [19]. If linear interpolation is used to transform from equally spaced $\log S$ coordinates to unequally spaced S coordinates (and vice versa), this introduces a second-order error consistent with the discretization of the PDE terms [19]. Effectively, we are approximating $\mathcal{B}V$ by

$$\begin{aligned} (\mathcal{B}V)_{i,j} &\simeq \lambda \sum_k b_{ik} V_{kj} = \lambda B \cdot V_j \\ &\text{with } 0 \leq b_{ik} \leq 1 \quad \text{and} \quad \sum_k b_{ik} \leq 1. \end{aligned} \tag{3.8}$$

Note that conditions (2.10) and (2.18) imply that we must also have

$$b_{ik} = 0 \quad \text{for} \quad i = 1 \quad \text{and} \quad i = M \quad \forall k. \tag{3.9}$$

The dense matrix multiply $B \cdot V_j$ can be evaluated efficiently using an FFT. For details, see [19, 18].

The Lagrangian derivative along a trajectory $A = A(S, \tau)$, for S fixed, is

$$\frac{DV}{D\tau} = \frac{\partial V}{\partial \tau} + \frac{\partial V}{\partial A}\frac{dA}{d\tau}. \tag{3.10}$$

Along the trajectory

$$\frac{dA}{d\tau} = \frac{A - S}{T - \tau} \tag{3.11}$$

(3.5) can be written as

$$\frac{DV}{D\tau} = \mathcal{C}V + \mathcal{B}V + q(V). \tag{3.12}$$

Let $A = A(S_i, A_j, \tau^{n+1}, \tau)$ along a trajectory satisfying (3.11), which passes through the discrete grid point (S_i, A_j) at $\tau = \tau^{n+1}$ for S_i being held constant. Let $A^n_{j(i,n+1)}$ be the departure point of this trajectory at $\tau = \tau^n$. Note that $A^n_{j(i,n+1)}$ will not necessarily coincide with a grid point A_j. Rather $A^n_{j(i,n+1)}$ is determined by solving

$$\frac{dA}{d\tau} = \frac{A - S_i}{T - \tau}, \text{ where } A = A_j \text{ for } \tau = \tau^{n+1}, \tag{3.13}$$

from $\tau = \tau^{n+1}$ to $\tau = \tau^n$, that is,

$$A^n_{j(i,n+1)} = A_j + \lim_{\tau^* \to (\tau^{n+1})} \int_{\tau^*}^{\tau^n} \frac{A - S_i}{T - \tau} d\tau. \tag{3.14}$$

The limit in (3.14) is used to avoid problems at $\tau = T$, where ODE (3.13) becomes undefined.

Let $V^n_{i,j(i,n+1)} = V(S_i, A^n_{j(i,n+1)}, \tau^n)$ denote the value of the option price at the departure point of the trajectory. Then discretizing (3.12) along the characteristic trajectory for different timestepping schemes gives, in the case of fully implicit timestepping,

$$\frac{V^{n+1}_{i,j} - V^n_{i,j(i,n+1)}}{\Delta\tau} = (\mathcal{C}V)^{n+1}_{i,j} + (\mathcal{B}V)^{n+1}_{i,j} + q(V^{n+1}_{i,j}), \tag{3.15}$$

and for Crank–Nicolson (CN) timestepping,

$$\begin{aligned}\frac{V^{n+1}_{i,j} - V^n_{i,j(i,n+1)}}{\Delta\tau} &= \frac{1}{2}\left((\mathcal{C}V)^{n+1}_{i,j} + (\mathcal{B}V)^{n+1}_{i,j}\right) \\ &\quad + \frac{1}{2}\left((\mathcal{C}V_{i,j(i,n+1)})^n + (\mathcal{B}V)^n_{i,j(i,n+1)}\right) + q(V^{n+1}_{i,j}),\end{aligned} \tag{3.16}$$

and for second-order backward differencing (BDF) [8],

$$\frac{\frac{1}{2}V^{n+1}_{i,j} - 2V^n_{i,j(i,n+1)} + \frac{1}{2}V^{n-1}_{i,j(i,n+1)}}{\Delta\tau} = (\mathcal{C}V)^{n+1}_{i,j} + (\mathcal{B}V)^{n+1}_{i,j} + q(V^{n+1}_{i,j}). \tag{3.17}$$

For ease of exposition, we have written (3.17) for constant timesteps. This is trivially generalized to nonconstant timesteps [8].

Unlike traditional applications of the semi-Lagrangian approach where the characteristic curve must be estimated numerically, for Asian options the solution along the characteristic curve can be determined exactly. Regarding S as a constant, and solving (3.11) gives

$$A = S_i + \frac{D}{T-\tau}, \tag{3.18}$$

where D is a constant independent of A (but a function of S_i). At $\tau = \tau^{n+1}$, $A = A_j$, so that

$$\begin{aligned} \text{at time } \tau^n : A^n_{j(i,n+1)} &= A_j + \frac{(S_i - A_j)(\tau^{n+1} - \tau^n)}{T - \tau^n}, \\ \text{at time } \tau^{n-1} : A^{n-1}_{j(i,n+1)} &= A_j + \frac{(S_i - A_j)(\tau^{n+1} - \tau^{n-1})}{T - \tau^{n-1}}, \end{aligned} \tag{3.19}$$

where $T \geq \tau^{n+1} > \tau^n > \tau^{n-1}$. Note that for the last step when $\tau^{n+1} = T$, (3.19) simplifies to $A^n_{j(i,n+1)} = A^{n-1}_{j(i,n+1)} = S_i$, which is the correct limiting behaviour of (3.14). The various quantities $(\cdot)^n_{i,j(i,n+1)}$ in (3.15)–(3.17) are determined by interpolation along lines of constant $S = S_i$. Assuming that the S derivatives and the integral term are discretized using second-order accurate methods, then it follows from [10, 22] that at least quadratic interpolation should be used for $(\cdot)^n_{i,j(i,n+1)}$ in order to retain global second-order convergence.

4. Semi-Lagrangian timestepping and discrete observations. Asian options which are continuously observed are typically considered to be the limit of discretely observed Asian options as the observation interval tends to zero [17, 44]. A similar statement can also be made about the discretization of the continuous problem given in the previous section. In this section we compare these two discrete problems. We show that if the discrete sampling period is equal to the discrete PIDE timestep, then a fully implicit, discretely sampled model is algebraically identical to a fully implicit semi-Lagrangian discretization of a continuously observed model. In the following, we ignore the effect of the boundary condition (2.16).

Consider the discrete average computed at the discrete forward observation times $t^\ell = \ell \Delta t$

$$\begin{aligned} \hat{A}(t^\ell) &= \frac{1}{\ell} \sum_{p=1}^{\ell} S(t^p) \\ &= \hat{A}(t^{\ell-1}) + \frac{S(t^\ell) - \hat{A}(t^{\ell-1})}{\ell}. \end{aligned} \tag{4.1}$$

Note that we are careful to distinguish the average computed at forward times as a function of t from the average computed at backwards times as a function of τ, that is, $A(\tau) = \hat{A}(t)$ where $\tau = T - t$. Similarly, we can define the value of an Asian option as a function of $(S, \hat{A}, t)$ as $\hat{V}(S, \hat{A}, t)$, where, in terms of $V(S, A, \tau)$ we have the identity $V(S, A(T-t), T-t) = \hat{V}(S, \hat{A}(t), t)$.

When using a PIDE method to price a discretely observed Asian option, we consider that $\hat{V} = \hat{V}(S, \hat{A}, t)$, and regard $(S, \hat{A})$ as independent variables. Suppose we have N observation dates, at the times $\Delta t, 2\Delta t, \ldots, N\Delta t$ with $N\Delta t = T$. Let $\hat{A}^\ell = \hat{A}(t^\ell)$. Then at the ℓth observation date we must have, by no arbitrage [46],

$$\hat{V}(S, \hat{A}^{\ell+1}, t^{(\ell+1)+}) = \hat{V}(S, \hat{A}^\ell, t^{(\ell+1)-}), \tag{4.2}$$

where $t^{(\ell+1)+}, t^{(\ell+1)-}$ are the instants just after and just before the observation date $t^{\ell+1}$. In (4.2) $\hat{A}^{\ell+1} = \hat{A}^{\ell} + \frac{S-\hat{A}^{\ell}}{\ell+1}$ and is regarded as constant for $t^{\ell+1} < t < t^{\ell+2}$.

Let $A^k = A(\tau^k)$ and set $k = N - \ell$, so that k counts backwards. Since $\tau^k = k\Delta\tau$, we have that $t^{\ell} = T - \tau^k$, $t^{\ell+1} = T - \tau^{k-1}$ and as well, $t^{(\ell+1)+} = T - \tau^{(k-1)-}$ and $t^{(\ell+1)-} = T - \tau^{(k-1)+}$. Writing the no-arbitrage condition (4.2) in terms of the variables $\tau, A(\tau)$ rather than t, $\hat{A}(t)$ then gives

$$V(S, A^{k+1}, \tau^{k+}) = V(S, A^k, \tau^{k-}), \text{ where } A^k = A^{k+1} + \frac{S - A^{k+1}}{N-k}. \tag{4.3}$$

As before we regard A^{k+1} as fixed during $\tau^k < \tau < \tau^{k+1}$.

Consider the case of a discretely observed European Asian option. In this case we solve

$$V_\tau = \mathcal{C}V + \mathcal{B}V + q(V) \tag{4.4}$$

on the domain $[0, S_{max}] \times [0, A_{max}]$, with the no-arbitrage conditions (4.3) imposed at observation times. Away from observation dates, if we discretize (4.4) in the A direction, then (4.4) represents a set of one-dimensional PIDEs, which communicate only through no-arbitrage conditions [17].

From the no-arbitrage condition (4.3) we have that a fully implicit discretization of (4.4) gives

$$\frac{V(S_i, A_j^{k+1}, \tau^{(k+1)-}) - V(S_i, A_j^{k+1}, \tau^{k+})}{\Delta\tau} = (\mathcal{C}V)_{i,j}^{k+1} + (\mathcal{B}V)_{i,j}^{k+1} + q(V_{i,j}^{k+1}). \tag{4.5}$$

Note that this is a set of independent one-dimensional PDEs (there are no A derivatives in (4.5), A_j^{k+1} appears only as a parameter). Using the no-arbitrage condition (4.3) in (4.5) gives

$$\frac{V(S_i, A_j^{k+1}, \tau^{(k+1)-}) - V(S_i, A_{j(i,k+1)}^{k}, \tau^{k-})}{\Delta\tau} = (\mathcal{C}V)_{i,j}^{k+1} + (\mathcal{B}V)_{i,j}^{k+1} + q(V_{i,j}^{k+1}), \tag{4.6}$$

where

$$A_{j(i,k+1)}^{k} = A_j^{k+1} + \frac{(S - A_j^{k+1})\Delta\tau}{T - \tau^k}, \tag{4.7}$$

which (setting $A_j^{k+1} = A_j$ in (4.6)–(4.7)) we recognize from (3.15) and (3.19) as being algebraically identical to a semi-Lagrangian, fully implicit discretization.

Note that in order for this result to hold, we must have discrete observations at $t = \Delta t, 2\Delta t, \ldots, N\Delta t$, that is, no observation at $t = 0$. Of course, in the limit as $\Delta t \to 0$, adding an extra observation at $t = 0$ will be the same to $O(\Delta t)$ as the semi-Lagrangian solution.

REMARK 4.1. [24] [27]

[24]
[24, 7]
[28]

5. Monotonicity and stability of the discrete equations. As shown in [40], in the case of nonlinear option pricing problems, seemingly reasonable discretization schemes can converge to an incorrect (i.e., nonviscosity) solution. Convergence to the viscosity solution is guaranteed if the discretization is consistent, monotone and l_∞ stable [5, 13, 15]. Usually, consistency follows if any reasonable discretization method is used, although in the case of jump-diffusion, the nonlocality of the integral term requires care in showing consistency [13, 15]. l_∞ stability is usually a consequence of monotonicity. Consequently, the most interesting requirement is monotonicity.

In the following, we will investigate the monotonicity and stability properties of the discrete equations. We will use a definition of monotonicity which is somewhat stricter than is usually the case in financial applications [5], but more in line with the definition used in computational fluid dynamics (CFD) [29]. It appears to us that the CFD definition is a more useful aid to the design of suitable discrete schemes. In Appendix B, we will show that CFD definition of monotonicity can be deduced from the usual definition of monotonicity in the viscosity solution literature, as long as the discretization is consistent.

We remind the reader that use of semi-Lagrangian timestepping decouples the discrete equations at each timestep, resulting in a set of one-dimensional discrete PIDEs. Hence we can use our techniques in [23, 40, 18, 19] to prove the desired properties of the discretized equations.

5.1. Preliminaries. Define the matrices B and C such that

$$\lambda \left[B \cdot V_j^n\right]_i = (\mathcal{B}V)_{i,j}^n + \text{ truncation error}, \tag{5.1}$$

$$\left[C \cdot V_j^n\right]_i = (\mathcal{C}V)_{i,j}^n + \text{ truncation error}, \tag{5.2}$$

where V_j^n is the vector of discrete solution values $\left[V_j^n\right]_i = V(A_j, S_i, \tau^n)$ for fixed A_j. A detailed description of B is given in [19, 18]. For our purposes, we note that B has the properties given by (3.8). A detailed description of matrix C is given in Appendix A.

To avoid algebraic complication, we will describe the discrete equations and the method used to solve the algebraic equations, only for the fully implicit and CN timestepping methods. The reader should have no difficulty generalizing the results to the BDF case.

Let Φ^{n+1} be the Lagrange interpolation operator such that

$$(\Phi^{n+1} \cdot V^n)_{i,j} = V(S_i, A^n_{j(i,n+1)}, \tau^n) + \text{ interpolation error}, \tag{5.3}$$

where Φ^{n+1} is a linear operator for any order (linear, quadratic) of interpolation. We also let V^* be the vector of payoffs obtained upon exercise and let P be the diagonal matrix given by

$$P(V_j^{n+1})_{ii} = \begin{cases} \cdot & \text{if } V_{i,j}^{n+1} < V_{i,j}^*, \\ 0 & \text{otherwise.} \end{cases} \tag{5.4}$$

Then the matrix form of the discrete equations for the penalized method is given by

(5.5)
$$[I-(1-\theta)\Delta\tau C+P(V_j^{n+1})]V_j^{n+1}=[\Phi^{n+1}[I+\theta\Delta\tau C]V^n]_j+(1-\theta)\lambda\Delta\tau BV_j^{n+1} +[\Phi^{n+1}\theta\lambda\Delta\tau BV^n]_j+\left[P(V_j^{n+1})\right]V_j^*+\Delta\tau F_j^{n+1}$$

for $j=1,\ldots,M$. Here $\theta=0$ is fully implicit, and $\theta=1/2$ is CN timestepping. The term F_j^{n+1} is used to approximate the boundary condition at $S=S_{max}$, as discussed in subsection 2.2. Note that $B_{i,l}=0$ for $S_i\in[S_{max}-\delta,S_{max}]$, as discussed in subsection 2.2. Condition (2.16) is enforced at $i=M$ by adjusting $C_{M,l}$ as discussed in Appendix A, and letting

$$\left[F_j^{n+1}\right]_i=\begin{cases}0, & i\neq M,\\ (1-\theta)\chi(S_{max},\tau^{n+1})+\theta\chi(S_{max},\tau^n), & i=M,\end{cases} \tag{5.6}$$

where $\chi(S,\tau)$ is discussed in subsection 2.2.

We note here that the penalty formulation of the American option pricing problem reduces problem (2.9) to the nonlinear PIDE (3.5), and hence the results in [13] apply. Briefly, if the numerical scheme is consistent, l_∞ stable, and monotone, then convergence to the viscosity solution is guaranteed. In order to obtain a monotone scheme, we can use at most linear interpolation in (5.3).

The choice of interpolation scheme is discussed in [24, 22]. Specifically, if the interpolation error does not get damped out, the global interpolation error after N timesteps is $O(\frac{((\Delta S)_{\max})^q}{\Delta\tau})$, where $q=2$ for linear interpolation, $q=3$ for quadratic interpolation and $(\Delta S)_{\max}=\max_i(S_{i+1}-S_i)$. Assuming second order in space and time truncation errors, the global discretization error for a semi-Lagrangian method applied to a pure hyperbolic problem is [22, 10]

$$\text{global discretization error}=O\left[\frac{((\Delta S)_{\max})^q}{\Delta\tau}+((\Delta S)_{\max})^2+(\Delta\tau)^2\right]. \tag{5.7}$$

If we assume $(\Delta S)_{max}=\text{const.}\,h$ and $\Delta\tau=\text{const.}\,h$, then (5.7) reduces to

$$\text{global discretization error}=O\left[\min(h^{q-1},h^2)\right]. \tag{5.8}$$

As discussed in [10], estimate (5.8) is valid only for smooth solutions.

5.2. Monotonicity and stability. The highest-order interpolation method Φ which will result in a monotone scheme is linear interpolation. Equation (5.8) suggests that if linear interpolation is used ($q=2$), we can obtain no more than first-order convergence. With this in mind, in the following analysis, we will consider only a fully implicit timestepping, and a linear interpolant Φ (as in (5.3)). We will, however, carry out numerical experiments with CN and BDF timestepping, and higher-order interpolants. The fully implicit version of (5.5) is

$$[I-\Delta\tau C-\lambda\Delta\tau B+P(V_j^{n+1})]V_j^{n+1}=[\Phi^{n+1}V^n]_j+P(V_j^{n+1})V_j^*+\Delta\tau F_j^{n+1} \tag{5.9}$$

for $j=1,\ldots,M$.

As discussed in [5, 13], consistency, stability, and monotonicity are sufficient conditions for a numerical scheme to ensure convergence to the viscosity solution. In view of the importance of discretizations which are stable and monotone, both from a

theoretical and practical point of view, it is useful to gather together a set of results for the implicit discretization schemes.

LEMMA 5.1 (properties of matrix C). [illegible] C [illegible] (5.9) [illegible]

$$\begin{aligned}\sum_k C_{ik} &= -(r+\lambda); \quad i = 2, \ldots, M-1 \\ &= -r; \quad i = 1 \quad\quad i = M, \\ C_{ik} &\geq 0; \quad i \neq k; \quad i = 1, \ldots, M. \end{aligned} \tag{5.10}$$

[illegible]. The proof follows directly from the discussion in Appendix A. ∎

Set

$$Q = I - \Delta\tau C - \lambda\Delta\tau B. \tag{5.11}$$

LEMMA 5.2 (M-matrix property of Q). [illegible] Q [illegible] M [illegible]. From (3.8)–(3.9) we have that $-B$ has nonpositive offdiagonal elements. From Lemma 5.1, we have that $-C - \lambda B$ has nonpositive offdiagonal elements. From Lemma 5.1, and properties (3.8)–(3.9), we have that

$$\sum_k [-C - \lambda B]_{ik} \geq 0; \quad i = 1, \ldots, M, \tag{5.12}$$

and hence Q is an M-matrix. ∎

We can write the discrete equations at each node (S_i, A_j) as

$$\begin{aligned} g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}) &= -[QV_j^{n+1}]_i + [\Phi^{n+1}V^n]_{i,j} \\ &\quad + \left[P(V_j^{n+1})\right]_{ii}(V_{i,j}^* - V_{i,j}^{n+1}) + \Delta\tau F_{i,j}^{n+1} \\ &= 0, \end{aligned} \tag{5.13}$$

where $\{V_{k,j}^{n+1}\}_i$ is to be interpreted as the set of values $V_{k,j}^{n+1}$, $k \neq i$, $k = 1, \ldots, M$, and $\{V^n\}$ is the set $V_{k,\ell}^n$, $k = 1, \ldots, M$; $\ell = 1, \ldots, M$.

DEFINITION 5.3 (monotone discretizations). [illegible] (5.13) [illegible]

(5.14)

$$\begin{aligned} g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1} + \rho_{k,j}^{n+1}\}_i, \{V_{k,\ell}^n + \rho_{k,\ell}^n\}) &\geq g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}) \quad \forall i,j; \\ &\quad \forall k \neq i,\ \forall \rho_{k,\ell}^n \geq 0,\ \forall \rho_{k,j}^{n+1} \geq 0, \end{aligned}$$

$$\begin{aligned} g_{i,j}(V_{i,j}^{n+1} + \rho_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}) &< g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}) \quad \forall i,j; \\ &\quad \forall k \neq i,\ \forall \rho_{i,j}^{n+1} > 0. \end{aligned} \tag{5.15}$$

REMARK 5.1. [illegible] (5.15) [5] (5.14) [illegible] [29, 25] [illegible] (5.15) B [illegible] (5.14) [illegible] (5.15) [illegible] (5.15) [illegible]

THEOREM 5.4 (monotonicity of the discretization). [illegible] (5.13) [illegible]

[illegible]. We rewrite (5.13) as

$$g_{i,j} = -[QV_j^{n+1}]_i + [\Phi^{n+1}V^n]_{i,j} + \left[P(V_j^{n+1})\right]_{ii}(V_{i,j}^* - V_{i,j}^{n+1}) + \Delta\tau F_{i,j}^{n+1} \tag{5.16}$$

and examine each term in (5.16). From Lemma 5.2, matrix Q is an M-matrix, hence $-[QV_j^{n+1}]_i$ is a strictly decreasing function of $V_{i,j}^{n+1}$, and a nondecreasing function of $\{V_{k,j}^{n+1}\}_i$. Since Φ^{n+1} is a linear interpolant operator, $[\Phi^{n+1}V^n]_{i,j}$ is a nondecreasing function of $\{V^n\}$. Finally we see that the term $\left[P(V_j^{n+1})\right]_{ii}(V_{i,j}^* - V_{i,j}^{n+1})$ is a nonincreasing function of $V_{i,j}^{n+1}$. Hence the discretization is monotone from Definition 5.3. □

THEOREM 5.5 (stability of the fully implicit scheme). [illegible]

$$\|V^{n+1}\|_\infty \le \max(\|V^n\|_\infty, \|V^*\|_\infty) + \Delta\tau\chi_{max}, \tag{5.17}$$

[illegible] $\chi(S_{max},\tau)$ [illegible] 2.2 [illegible]

$$\chi_{max} = \max_{0\le\tau\le T} |\chi(S_{max},\tau)|. \tag{5.18}$$

[illegible]

$$\|V^{n+1}\|_\infty \le \|V^*\|_\infty + T\chi_{max}, \tag{5.19}$$

[illegible] $(n+1)\Delta\tau \le T$

[illegible]. Writing out (5.9) in component form gives (see Appendix A)

$$\begin{aligned} V_{i,j}^{n+1}(1 + (\alpha_i + \beta_i + r + \lambda)\Delta\tau) - \alpha_i\Delta\tau V_{i-1,j}^{n+1} - \beta_i\Delta\tau V_{i+1,j}^{n+1} \\ -\lambda\Delta\tau\sum_k b_{ik}V_{k,j}^{n+1} + P_{ii}(V_j^{n+1})V_{i,j}^{n+1} \\ = \sum_{k,l} w_{k,l}^{i,j}V_{k,l}^n + P_{ii}(V_j^{n+1})V_{i,j}^* + \Delta\tau F_{i,j}^{n+1}, \end{aligned} \tag{5.20}$$

where $w_{k,l}^{i,j}$ are linear interpolation weights satisfying

$$0 \le w_{k,l}^{i,j} \le 1 \quad \text{and} \quad \sum_{k,l} w_{k,l}^{i,j} = 1. \tag{5.21}$$

In addition, we recall that B has properties (3.8)

$$0 \le b_{ik} \le 1 \quad \text{and} \quad \sum_k b_{ik} \le 1, \tag{5.22}$$

while from Appendix A we have that

$$\alpha_i \ge 0 \quad \text{and} \quad \beta_i \ge 0. \tag{5.23}$$

Let m be an index such that

$$|V_{m,j}^{n+1}| = \|V_j^{n+1}\|_\infty. \tag{5.24}$$

Then (5.20)–(5.23) imply that

$$\|V_j^{n+1}\|_\infty \left(1 + r\Delta\tau + P(V_j^{n+1})_{mm}\right) \le \|V^n\|_\infty + P(V_j^{n+1})_{mm}\|V^*\|_\infty + \Delta\tau\chi_{max} \tag{5.25}$$

and so

$$\begin{aligned}\|V_j^{n+1}\|_\infty &\le \max(\|V^n\|_\infty, \|V^*\|_\infty)\frac{1 + P(V_j^{n+1})_{mm}}{1 + r\Delta\tau + P(V_j^{n+1})_{mm}} + \frac{\Delta\tau\chi_{max}}{1 + r\Delta\tau + P(V_j^{n+1})_{mm}} \\ &\le \max(\|V^n\|_\infty, \|V^*\|_\infty) + \Delta\tau\chi_{max}.\end{aligned} \tag{5.26}$$

Hence

$$\|V^{n+1}\|_\infty \le \max(\|V^n\|_\infty, \|V^*\|_\infty) + \Delta\tau\chi_{max}. \tag{5.27}$$

Therefore, by induction we have

$$\|V^{n+1}\|_\infty \le \max(\|V^{n-i}\|_\infty, \|V^*\|_\infty) + (i+1)\Delta\tau\chi_{max} \tag{5.28}$$

for all i. Equation (5.19) follows from setting $i = n$ in (5.28). □

REMARK 5.2 (extension to nonlinear models). [illegible] [41] [illegible] [41] [illegible]

5.3. Properties of a semi-implicit discretization. Suppose we alter the discretization (5.9) so that the jump integral term is evaluated explicitly. Then

$$[I - \Delta\tau C + P(V_j^{n+1})]V_j^{n+1} = \lambda\Delta\tau B V_j^n + [\Phi^{n+1}V^n]_j + P(V_j^{n+1})V_j^* + \Delta\tau F_j^{n+1} \tag{5.29}$$

for $j = 1, \ldots, M$. The previous methods can also be applied to determine stability and monotonicity properties of this second discretization.

THEOREM 5.6 (stability and monotonicity of explicit evaluation of the jump term). [illegible] (5.29) [illegible]. Set $R = [I - \Delta\tau C]$. Then we can rewrite (5.29) as

$$\begin{aligned}g_{i,j} = &- [RV_j^{n+1}]_i + [\Phi^{n+1}V^n]_{i,j} + \left[P(V_j^{n+1})\right]_{ii}(V_{i,j}^* - V_{i,j}^{n+1}) \\ &+ \Delta\tau F_{i,j}^{n+1} + [\Delta\tau\lambda B V^n]_{i,j}\end{aligned} \tag{5.30}$$

From Lemma 5.1 and the properties of matrix B, (3.8), and following along the lines used to prove Theorem 5.4, it is straightforward to see that Definition 5.3 holds unconditionally for (5.30). Using a similar maximum analysis as in the proof of Theorem 5.5, we obtain unconditional stability. □

REMARK 5.3. [illegible] (5.29) [illegible]

6. Additional properties of the discrete equations. In this section we investigate how well our discrete approximation (3.5) preserves important properties of our original problem (2.9). We focus on two important properties: how well does the discrete penalty method satisfy the inequality constraints in problem (2.9) and does the discretization preserve arbitrage inequalities [15].

6.1. Error in the penalty formulation. In our original problem (2.9) we need to solve $\min(\mathcal{H}V; V - V^*) = 0$. In particular, we require that

$$(V - V^*) \geq 0. \tag{6.1}$$

In discrete terms this becomes

$$(V_{i,j}^{n+1} - V_{i,j}^*) \geq 0. \tag{6.2}$$

However, the penalty formulation (5.5) will result in $V_{i,j}^{n+1} < V_{i,j}^*$ at nodes in the exercise region. In this subsection we show that at these nodes we have $V_{i,j}^{n+1} = V_{i,j}^* - \epsilon$, where $0 < \epsilon \ll 1$. In particular, we have the following bound on the error in the penalty term.

LEMMA 6.1 (error generated by the penalty formulation). [illegible] V^* [illegible]

$$\frac{\Delta\tau}{\Delta S_{min}} < \text{const.} \quad \Delta\tau, \Delta S_{min} \to 0, \tag{6.3}$$

[illegible] $\Delta S_{min} = \min_i(S_{i+1} - S_i)$ [illegible]

$$V_{i,j}^{n+1} - V_{i,j}^* \geq -\frac{C_1}{Large}, \tag{6.4}$$

[illegible] C_1 [illegible] $\Delta S, \Delta\tau$ [illegible]. Let k be an index such that

$$(V_{k,j}^* - V_{k,j}^{n+1}) = \max_i(V_{i,j}^* - V_{i,j}). \tag{6.5}$$

Since the matrix Q defined by (5.11) is an M-matrix from Lemma 5.2, it follows from (6.5) that

$$\left[Q(V_j^* - V_j^{n+1})\right]_k \geq 0, \tag{6.6}$$

and hence

$$\left[QV_j^*\right]_k \geq \left[QV_j^{n+1}\right]_k. \tag{6.7}$$

From (5.16) we have that for all j

$$QV_j^{n+1} = [\Phi V^n]_j + P(V_j^{n+1})(V_j^* - V_j^{n+1}) + \Delta\tau F_j^{n+1}. \tag{6.8}$$

In particular, row k of (6.8) is

$$\left[QV_j^{n+1}\right]_k = \left[[\Phi V^n]_j\right]_k + \left[P(V_j^{n+1})(V_j^* - V_j^{n+1})\right]_k + \Delta\tau F_{k,j}^{n+1}. \tag{6.9}$$

Since

$$\left[P(V_j^{n+1})(V_j^* - V_j^{n+1})\right]_k = \|P(V_j^{n+1})(V_j^* - V_j^{n+1})\|_\infty, \tag{6.10}$$

then (6.9) gives (using (6.7))

$$\|P(V_j^{n+1})(V_j^* - V_j^{n+1})\|_\infty \le \|V^n\|_\infty + \|QV^*\|_\infty + \Delta\tau\chi_{max}. \tag{6.11}$$

From Theorem 5.5 and (6.11) we have

$$\|P(V_j^{n+1})(V_j^* - V_j^{n+1})\|_\infty \le C_2 + \|QV^*\|_\infty, \tag{6.12}$$

where $C_2 = \|V^*\|_\infty + T\chi_{max}$, with $(n+1)\Delta\tau \le T$. Assuming that V^* satisfies a Lipschitz condition, then

$$\|QV^*\|_\infty \le C_3 \frac{\Delta\tau}{\Delta S_{min}} \tag{6.13}$$

which follows from Lemma 5.2 and Appendix A. Assuming $\Delta\tau/\Delta S_{min}$ is bounded, we have

$$(V_{i,j}^{n+1} - V_{i,j}^*) \ge -\frac{C_1}{Large} \tag{6.14}$$

with $C_1 = C_2 + \frac{C_3\Delta\tau}{\Delta S_{min}} > 0$. □

REMARK 6.1 (significance of Lemma 6.1). 6.1 (2.9) (3.6) *Large* (5.4), $\Delta\tau/\Delta S_{min}$ $(V_{i,j}^{n+1} - V_{i,j}^*)$ [23] (6.2)

THEOREM 6.2 (discrete comparison principle). (5.9) $V^n > W^n$ V^{n+1}, W^{n+1} (5.9) $V^{n+1} > W^{n+1}$

. Suppose $V^n > W^n$. Write (5.9) for V, W

$$\begin{aligned} QV_j^{n+1} &= [\Phi^{n+1}V^n]_j + \left[P(V_j^{n+1})\right](V_j^* - V_j^{n+1}) + \Delta\tau F_j^{n+1}, \\ QW_j^{n+1} &= [\Phi^{n+1}W^n]_j + \left[P(W_j^{n+1})\right](V_j^* - W_j^{n+1}) + \Delta\tau F_j^{n+1}. \end{aligned} \tag{6.15}$$

Some manipulation of (6.15) results in

$$\begin{aligned} Q(V_j - W_j)^{n+1} = &-P(W_j^{n+1})(V_j - W_j)^{n+1} + (P(V_j^{n+1}) \\ &-P(W_j^{n+1})(V_j^* - V_j^{n+1}) + [\Phi^{n+1}(V^n - W^n)]_j \end{aligned} \tag{6.16}$$

or

$$\begin{aligned} \left[Q + P(W_j^{n+1})\right](V_j - W_j)^{n+1} = &(P(V_j^{n+1}) - P(W_j^{n+1})(V_j^* - V_j^{n+1}) \\ &+ [\Phi^{n+1}(V^n - W^n)]_j. \end{aligned} \tag{6.17}$$

Since Q is an M-matrix we have that $[Q + P(W_j^{n+1})]$ is also an M-matrix. From (5.4), we have that

$$(P(V_j^{n+1}) - P(W_j^{n+1})(V_j^* - V_j^{n+1}) \ge 0. \tag{6.18}$$

If linear interpolation is used, then $(V^n - W^n) > 0$ implies that $[\Phi^{n+1}(V^n - W^n)]_j > 0$. Finally, since $[Q + P(W_j^{n+1})]$ is an M-matrix, its inverse satisfies $[Q + P(W_j^{n+1})]^{-1} \geq 0$, and diag.$([Q + P(W_j^{n+1})]^{-1}) > 0$, and hence $(V_j - W_j)^{n+1} > 0$. □

REMARK 6.2. [illegible] [15] [illegible] 6.2 [illegible]

7. Iterative solution of the discretized equations. In order to solve (5.5), we use the following iteration scheme.

Iteration

For $j = 1, 2, \ldots$

Let $(V_j^{n+1})^0 = (V_j^n)$

Let $\hat{V}_j^{\,k} = (V_j^{n+1})^k$

Let $\hat{P}^k = P((V_j^{n+1})^k)$

For $k = 0, 1, 2, \ldots$ until convergence

Solve

$$\left[I - (1-\theta)C + \hat{P}^k\right] \hat{V}_j^{\,k+1} = \left[\Phi^{n+1}\left[I + \theta C\right] V^n\right]_j + \hat{P}^k V_j^* + \Delta\tau F_j^{n+1} + (1-\theta)\lambda\Delta\tau B\hat{V}_j^{\,k} + \theta\lambda\Delta\tau\left[\Phi^{n+1} B V^n\right]_j \tag{7.1}$$

If $\max_i \dfrac{|\hat{V}_{i,j}^{k+1} - \hat{V}_{i,j}^{k}|}{\max(1, |\hat{V}_{i,j}^{k+1}|)} <$ [illegible] then break

EndFor

EndFor

For clarity, we have given algorithm (7.1) only for CN and fully implicit timestepping. However, it is trivial to generalize this method to BDF timestepping. Note that each iteration of algorithm (7.1) requires a tridiagonal factor and solve, and a forward and back FFT (to evaluate $B \cdot \hat{V}_j^{\,k}$).

The following theorem indicates that iteration scheme (7.1) is globally convergent.

THEOREM 7.1 (convergence of iteration). [illegible] C B $\hat{P}$ [illegible] (5.2) (5.1) [illegible] (5.4) [illegible] $I - (1-\theta)C$ [illegible] 5.1 [illegible] B [illegible] (3.8) [illegible] (7.1) [illegible] (5.5) [illegible] $\hat{V}^0$ [illegible]. Note that the algebraic equations (5.5) are decoupled for each line of constant A_j. Hence the issue of convergence of scheme (7.1) reduces to the convergence of each set of equations for constant j. But for constant j, this iteration is equivalent to solution of the discrete penalized equations for one dimensional American options with jump diffusion as described in [18]. Hence the result follows directly from Theorem 4.2 in [18]. □

8. Computational details and numerical results. This section presents numerical results for various options and payoffs, including vanilla European call/put and American options. We will use an unequally spaced grid in the A, S directions, on the domain $[0, S_{max}] \times [0, A_{max}]$, with $A_{max} = S_{max}$. We remind the reader that δ is selected so that in $[S_{max} - \delta, S_{max}]$ we have that $\lambda = 0$, and we thus have sufficient data for accurate computation of the jump integral term in $[0, S_{max}]$. The method used to determine δ is discussed in [19]. Probabilistic arguments can be used to determine an appropriate value for S_{max} [44], so that a linear approximation to V is justified in $[S_{max} - \delta, S_{max}]$. We use $S_{max} = 50K$, where K is the strike. We describe below some tests which were carried out to verify that the effect of imposing boundary conditions at $S = 50K$ results in insignificant error. The convergence tolerance in iteration 7.1 was $tol = 10^{-6}$. As suggested in [23], we choose $Large = 1/tol$. If CN or BDF timestepping is used, then quadratic interpolation is used in (5.3). If fully implicit timestepping is employed, then linear interpolation is used in (5.3).

Given an A grid discretization, the discrete PIDEs (3.15)–(3.16) become decoupled. At each timestep, we have a set of independent one-dimensional discrete PIDEs to solve. This property makes solution of the continuously observed Asian option straightforward to implement, given an existing library which supports pricing of path dependent options.

As pointed out in (5.8), it is necessary to use at least a quadratic Lagrange interpolation scheme to find the solution at the foot of the characteristic curve, if we hope to obtain quadratic convergence. This will, however, result in a scheme which is not monotone.

The convergence ratio R is defined in the following way. For each test, as we double the number of grid points in both S and A directions, we cut the timesteps $(\Delta\tau)$ in half. Let $\Delta\tau = \max_n(\tau^{n+1} - \tau^n)$, $(\Delta A)_{max} = \max_j(A_{j+1} - A_j)$. Note that we are allowing here for the possibility of using variable timestep sizes (to be explained later), although most of our tests will simply use a constant timestep size. If we then carry out a convergence study, letting $h \to 0$, where $(\Delta S)_{max} = \text{const. } h$, $(\Delta A)_{\max} = \text{const. } h$, and $\Delta\tau = \text{const. } h$, then we can assume that the error in the solution (at a given node) is

$$V_{\text{approx}}(h) = V_{\text{exact}} + \text{const. } h^{\xi}.$$

The convergence ratio is then defined as

$$R = \frac{V_{\text{approx}}(h/2) - V_{\text{approx}}(h)}{V_{\text{approx}}(h/4) - V_{\text{approx}}(h/2)}. \tag{8.1}$$

In the case of quadratic convergence ($\xi = 2$), then $R = 4$, while for linear convergence ($\xi = 1$), $R = 2$.

Table 8.1 shows results for a low volatility case, European Asian option (no jumps), using the semi-Lagrangian approach. In this special case, the two-dimensional PDE can be reduced to one dimension [45], which we will refer to as the Večeř PDE [45] in the following. Results obtained by solving the Večeř PDE numerically are also given in Table 8.1.

In Table 8.1, we can see that the convergence ratio R for the semi-Lagrangian method is not quadratic ($R \neq 4$), while for the Večeř PDE [45] quadratic convergence is found. As discussed in [45], the Večeř PDE is not convection dominated, hence it is straightforward to obtain accurate numerical solutions. We remind the reader that this clever reduction to one dimension cannot be used for American options.

TABLE 8.1

Value of a continuously observed fixed strike European Asian call option (no jumps) with constant timesteps. The input parameters are $\sigma = .1$, $r = .1$, $T = .25$, $\lambda = 0$, and $K = 100$. We compare the results given using the Večeř [45] one-dimensional model, and the semi-Lagrangian method presented here. CN timestepping was used.

semi-Lagrangian				Večeř 1-D PDE [45]			
Size of S and A grids	No. of timesteps	$S = 100$ Value	R	Size of S grids	No. of timesteps	$S = 100$ Value	R
51	25	1.857193	n.a.	51	25	1.839863	n.a.
101	50	1.853254	n.a.	101	50	1.848642	n.a.
201	100	1.852120	3.475	201	100	1.850851	3.974
401	200	1.851781	3.338	401	200	1.851407	3.979
801	400	1.851660	2.815	801	400	1.851546	3.987

TABLE 8.2

Value of a continuously observed fixed strike Asian call option (no jumps) at $S = K = 100$, constant CN timestepping. The input parameters are $\sigma = .1$, $\lambda = 0$, $r = .1$, $T = .25$, and $K = 100$. The right boundary of the space discretization $[0, S_{max}]$ domain is truncated at different values.

	$S_{max} = 5 \times K$		$S_{max} = 50 \times K$	
Timesteps	S, A grid nodes	Value	S, A grid nodes	Value
25	51	1.857193	54	1.857193
50	101	1.853254	109	1.853254
100	201	1.852120	217	1.852120
200	401	1.851781	433	1.851781
400	801	1.851660	865	1.851660

The discontinuity present in the payoff greatly affects the convergence of the semi-Lagrangian method, since there is very little diffusion in the A direction, and the nonsmoothness in the payoff is not smoothed out during the solution phase. Since we need to use quadratic interpolation in the A direction in order to determine the values of the solution at the feet of the characteristic curves, the interpolation may be affected by the nonsmooth payoff, and may lower the observed rate of convergence.

In order to test the effect of the boundary condition (2.16) at $S = S_{max}$, we show results using two different values of S_{max} in Table 8.2. This table would seem to indicate that there is a negligible error for options of this maturity incurred setting $S_{max} = 50K$, and all subsequent results will be reported imposing condition (2.16) at $S_{max} = 50K$.

Figures 8.1 and 8.2 graphically present the solution V and the first derivative of the solution with respect to the stock price V_S when CN is used. The plots are all smooth and do not exhibit any oscillations. While not shown here, V_{SS} also did not show any oscillations.

We now explore numerical convergence for pricing Asian options for large values of volatility σ. Table 8.3 presents our results. As expected quadratic convergence is recovered. In this case, a sufficient amount of diffusion in the S direction appears to compensate for zero diffusion in the A direction.

8.1. An in-depth study of the convergence ratio. The results of the previous section indicated that the semi-Lagrangian approach, coupled with CN timestepping, results in quadratic convergence, for large volatilities. However, for small volatility values, quadratic convergence was not recovered. The goal of this subsection is to explore in detail different numerical techniques that could improve the convergence rate.

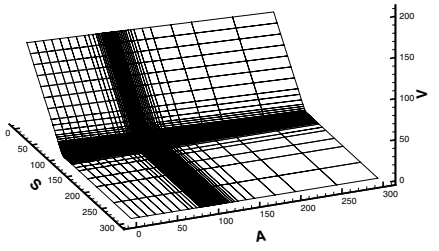


FIG. 8.1. *Value of a European fixed strike Asian put using CN with constant timestepping ($\Delta\tau = .01$). 51 grid points are used both in the A and S direction. The input parameters are $\sigma = .1$, $r = .1$, $T = .25$, $K = 100$, and $\lambda = 0$.*

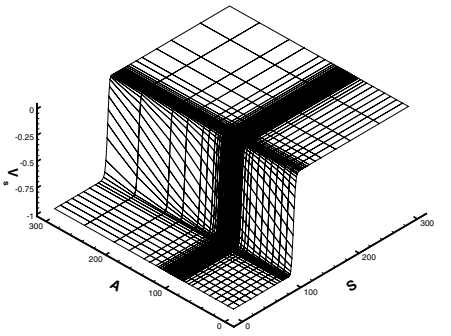


FIG. 8.2. *First derivative (V_S) value of a European fixed strike Asian put using CN with constant timestepping ($\Delta\tau = .01$). 51 grid points are used both in the A and S direction. The input parameters are $\sigma = .1$, $r = .1$, $T = .25$, $K = 100$, and $\lambda = 0$.*

TABLE 8.3

Value of a continuously observed fixed strike Asian call option (no jumps) with constant timesteps at $S = K$. The input parameters are $\sigma = .5$, $r = .05$, $T = .25$, $\lambda = 0$, and $K = 100$. We compare the results given using the Večeř 1-D PDE [45]*, and the semi-Lagrangian method presented here. CN timestepping was used.*

semi-Lagrangian				Večeř 1-D PDE [45]			
Size of	No. of	$S = 100$		Size of	No. of	$S = 100$	
S and A grids	timesteps	Value	R	S grids	timesteps	Value	R
51	25	6.010203	n.a.	51	25	6.009821	n.a.
101	50	6.015092	n.a.	101	50	6.014848	n.a.
201	100	6.016344	3.905	201	100	6.016251	3.582
401	200	6.016651	4.085	401	200	6.016619	3.816
801	400	6.016723	4.219	801	400	6.016713	3.915

TABLE 8.4

Value of a continuously observed fixed strike Asian call option (no jumps) at the strike, constant timesteps. The input parameters are $\sigma = .1$, $r = .1$, $T = .25$, $\lambda = 0$, and $K = 100$. Convergence ratios (8.1) *are presented for different timestepping schemes: implicit, CN, and second-order BDF.*

		Implicit timestepping		CN timestepping		BDF timestepping	
Size of	No. of	$S = 100$		$S = 100$		$S = 100$	
S and A grids	timesteps	Value	R	Value	R	Value	R
51	25	1.911865	n.a.	1.857193	n.a.	1.86096	n.a.
101	50	1.880801	n.a.	1.853254	n.a.	1.854310	n.a.
201	100	1.865907	2.086	1.852120	3.475	1.852416	3.513
401	200	1.858681	2.061	1.851781	3.338	1.851868	3.453
801	400	1.855112	2.025	1.851660	2.815	1.851686	3.014

Table 8.4 contains the convergence rate results for different timestepping schemes for small volatility ($\sigma = .1$ and $r = .1$). For implicit timestepping linear convergence is recovered ($R = 2$), as expected. However for higher-order timestepping schemes such as CN and second-order BDF, quadratic convergence is not found (see Table 8.4). These results are not surprising since the combination of small volatility with

TABLE 8.5

Value of a continuously observed fixed strike call Asian call option (no jumps) at the strike with constant timesteps. The initial payoff is smoothed using the average scheme described by (8.2). *The input parameters are* $\sigma = .1$, $r = .1$, $T = .25$, $\lambda = 0$, *and* $K = 100$. *Convergence ratios* (8.1) *are presented for different timestepping schemes: CN and second-order BDF.*

Size of	No. of	CN timestepping $S = 100$		BDF timestepping $S = 100$	
S and A grids	timesteps	Value	R	Value	R
51	25	1.870322	n.a.	1.874276	n.a.
101	50	1.856377	n.a.	1.857462	n.a.
201	100	1.852873	3.981	1.853179	3.925
401	200	1.851963	3.849	1.852053	3.803
801	400	1.851704	3.513	1.851731	3.497

the nonsmooth payoff, means that quadratic interpolation in the A direction is not $O((\Delta A)^3_{max})$, for small τ.

To try to remedy this problem, the initial payoff function is smoothed out. A classic method for handling discontinuities involves averaging the initial data. Specifically, values at each point are replaced with an average value over nearby space. Mathematically, we set

$$\text{PAYOFF}_{\text{smoothed}}(S_i, A_j) = \int_{K-\frac{\Delta A}{2}}^{K+\frac{\Delta A}{2}} \text{PAYOFF}(S_i, A) dA. \tag{8.2}$$

For a complete description of various smoothing methods the readers are referred to [41].

Table 8.5 contains the convergence rate results. From a convergence point of view, the ratios have improved in comparison with the convergence ratio without smoothing (see Table 8.4). However, quadratic convergence is still not obtained. From a theoretical point of view, all the convergence analysis for semi-Lagrangian scheme is based on the smooth properties of the solution [10, 22]. If the solution is smooth, then quadratic convergence is recovered. However, if the solution is nonsmooth, then we can expect some reduction in the convergence rate.

To confirm our intuition that the nonsmooth payoff is in fact the reason why quadratic convergence is not recovered, we create an artificial payoff that has the property of being quadratically smooth over the entire domain, e.g., $\text{PAYOFF}(A, K) = [\max(0, A - K)]^2$. In this case quadratic convergence is recovered for both CN and second-order BDF. Table 8.6 shows detailed convergence results for CN timestepping.

Several other approaches were considered in an effort to improve convergence. We tried to use Rannacher timestepping [42]; two or more implicit timesteps are taken before reverting to a higher-order timestepping scheme such as CN for example. Numerical experiments indicated that this did not improve the convergence rate. A convergence rate of approximately 3.5 is found in this case. Adaptive timestepping was also considered [23] but this technique did not improve the convergence rate.

8.2. Exotic Asian options. It is not generally possible to achieve second-order convergence for American options using constant timesteps. In [23] it was demonstrated that in order to achieve second-order convergence, it is necessary to use variable timestepping for American options. However, some initial tests showed that due to the large convective term in the A direction, near $\tau = T$, the timestep selector suggested in [23] required very small timesteps near $\tau = T$. Consequently, we will show results in the following using constant timesteps.

TABLE 8.6

Value of a continuously observed Asian call option (no jumps) at the strike with constant timesteps. The input parameters are $\Delta\tau = .01$, $\sigma = .1$, $r = .1$, $T = .25$, $\lambda = 0$, *and* $K = 1$. *Convergence ratios* (8.1) *are presented for the CN timestepping scheme.*

Size of	No. of	Call option (PAYOFF$(A, K) = [\max(0, A - K)]^2$) $S = 1$	
S and A grids	timesteps	Value	R
51	25	0.025749	n.a.
101	50	0.025622	n.a.
201	100	0.025591	4.131
401	200	0.025584	4.093
801	400	0.025582	4.054

TABLE 8.7

Input data used to value American fixed strike Asian options under the lognormal jump diffusion process (2.7). *These parameters are approximately the same as those reported in* [3] *using European call options on the S&P* 500 *stock index in April of* 1999.

Parameter values			
σ	0.15	λ	0.10
r	0.05	T	0.25
γ	0.45	K	100.00
μ	-0.90	σ_{implied}	0.1886

Table 8.7 presents the input parameters. The mean of the jump distribution is denoted by μ and the jump distribution standard deviation is denoted by γ (see (2.7)).

To ensure consistent comparison between American Asian options with jumps and American Asian options without jumps, we proceed as follows.

1. Given the parameters in Table 8.7, we compute the analytical solution V_{jump} at the strike K of a vanilla put option, under jump diffusion.
2. Use a constant volatility Black–Scholes model with no jump to determine the implied volatility σ_{implied} which matches the jump diffusion value V_{jump} at the strike K.
3. Price the American Asian option with jumps using the parameters in Table 8.7.
4. Price the American Asian option with no jumps but with the implied volatility σ_{implied} estimated in step 2.

Table 8.8 compares the value of an American Asian fixed strike put option with the value of an American Asian fixed strike put option when the underlying stock follows the jump diffusion process described in [32]. Second-order backward timestepping is used and the initial payoff is smoothed out using (8.2). We observe that quadratic convergence is not recovered, the convergence ratios are ≈ 3.5. It is interesting to note that, at the strike, the price of an American Asian fixed strike put option with jumps is 9% cheaper than the price of the same option without jumps, while at $S = 1.05K$, the jump diffusion price is considerably higher than the no-jump price, as can be seen in Figure 8.3.

REMARK 8.1 (alternative boundary condition). (2.16) $V_{SS} = 0$ [47] M $I - \Delta\tau C - \lambda\Delta\tau B$

TABLE 8.8

Value of a continuously observed fixed strike put American Asian option (under jump diffusion) with constant timestepping. Second-order BDF timestepping is used. The input parameters are defined in Table 8.7. *This table presents convergence rates with and without jumps. Iterations refer to the total (over all timesteps) of the maximum number of iterations required for any value of j (see Algorithm* 7.1*) at each timestep.*

	No jump				Jumps		
Size of	No. of	No. of	$(S = 100)$		No. of	$(S = 100)$	
S and A grids	timesteps	iterations	Value	R	iterations	Value	R
51	25	77	2.220443	n.a.	99	2.044636	n.a.
101	50	160	2.195726	n.a.	167	2.018530	n.a.
201	100	319	2.188555	3.447	340	2.012220	4.138
401	200	692	2.186717	3.903	716	2.010691	4.126
801	400	1397	2.186243	3.874	1609	2.010281	3.728

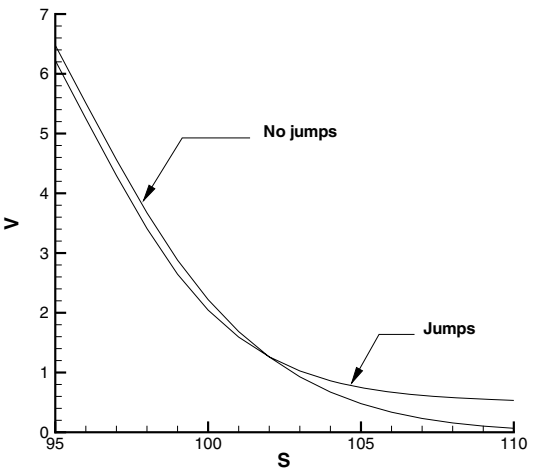


FIG. 8.3. *Comparison between the value of an American Asian fixed strike put option and the value of an American Asian fixed strike put option when the underlying stock follows the jump diffusion process described in* [32]. *The input parameters are defined in Table* 8.7.

M

$S \to \infty$

9. Conclusion. In this paper we have put forward four primary contributions. First we have demonstrated that a semi-Lagrangian method can be used to price continuously observed American Asian options under jump diffusion processes. The implementation suggested here reduces this problem to solving a decoupled set of one-dimensional discrete PIDEs at each timestep.

A second contribution is that since the discretized problem at each timestep reduces to a set of decoupled one-dimensional PIDEs, we can make use of previous techniques developed by the authors to prove certain important properties of the discrete scheme, including convergence of the iterative method used to solve the implicit

discrete equations.

In addition, we have included experimental computations which indicate that, even if second-order timestepping methods are used, observed convergence as the mesh and timestep sizes are refined occurs at a subsecond-order rate. The problem can be traced to the nonsmoothness of the payoff function.

Finally, we have also shown that in the fully implicit case, the semi-Lagrangian method for continuously observed Asian options is algebraically identical to a standard numerical method for pricing discretely observed Asian options, when the observation interval is equal to the discrete timestep.

Appendices.

Appendix A. Discretization of the PDE. In this appendix, we give the details of the discretization of the term $(\mathcal{C}V)_{i,j}$ in (3.15).

Using finite differences, the matrix (5.2) is

$$(C \cdot V)_i = -V_i^{n+1}(\alpha_i + \beta_i + r + \lambda) + \beta_i V_{i+1}^{n+1} + \alpha_i V_{i-1}^{n+1}; \quad i = 2, \ldots, M-1, \tag{A.1}$$

where α_i, β_i depend on the type of approximations used for the derivatives and second derivatives. For $i = 1$, we impose condition (2.10) by setting $\alpha_1 = \beta_1 = \lambda = 0$, and for the row $i = M$, condition (2.16) is imposed by setting $\alpha_M = \beta_M = \lambda = 0$. There are a number of different discretizations of the derivative terms leading to various choices for α_i and β_i.

Discretizing the first derivative term of (3.2) with central differences leads to

$$\begin{aligned}
\alpha_{i,central} &= \frac{\sigma_i^2 S_i^2}{(S_i - S_{i-1})(S_{i+1} - S_{i-1})} - \frac{(r - \lambda\kappa)S_i}{S_{i+1} - S_{i-1}}, \\
\beta_{i,central} &= \frac{\sigma_i^2 S_i^2}{(S_{i+1} - S_i)(S_{i+1} - S_{i-1})} + \frac{(r - \lambda\kappa)S_i}{S_{i+1} - S_{i-1}}.
\end{aligned} \tag{A.2}$$

However if $\alpha_{i,central}$ or $\beta_{i,central}$ is negative, oscillations may appear in the solution. The oscillations can be avoided by using forward or backward differences at the problem nodes, leading to (forward difference)

$$\begin{aligned}
\alpha_{i,forward} &= \frac{\sigma_i^2 S_i^2}{(S_i - S_{i-1})(S_{i+1} - S_{i-1})}, \\
\beta_{i,forward} &= \frac{\sigma_i^2 S_i^2}{(S_{i+1} - S_i)(S_{i+1} - S_{i-1})} + \frac{(r - \lambda\kappa)S_i}{S_{i+1} - S_i},
\end{aligned} \tag{A.3}$$

or, (backward difference)

$$\begin{aligned}
\alpha_{i,backward} &= \frac{\sigma_i^2 S_i^2}{(S_i - S_{i-1})(S_{i+1} - S_{i-1})} - \frac{(r - \lambda\kappa)S_i}{S_{i+1} - S_i}, \\
\beta_{i,backward} &= \frac{\sigma_i^2 S_i^2}{(S_{i+1} - S_i)(S_{i+1} - S_{i-1})}.
\end{aligned} \tag{A.4}$$

Algorithmically, we decide between a central or forward discretization at each node for (A.1) as follows.

Discretization

(A.5)

If $[\alpha_{i,central} \geq 0$ and $\beta_{i,central} \geq 0]$ then
$\alpha_i = \alpha_{i,central}$
$\beta_i = \beta_{i,central}$
ElseIf $[\beta_{i,forward} \geq 0]$ then
$\alpha_i = \alpha_{i,forward}$
$\beta_i = \beta_{i,forward}$
Else
$\alpha_i = \alpha_{i,backward}$
$\beta_i = \beta_{i,backward}$
EndIf

Note that the test condition (A.5) guarantees that α_i and β_i are nonnegative. For typical values of σ, r and grid spacing, forward differencing is rarely required for single factor options. In practice, since this occurs at only a small number of nodes remote from the region of interest, the limited use of a low-order scheme does not result in poor convergence as the mesh is refined. For situations where the low-order method causes excessive numerical diffusion, a flux limiter can be used [49]. As we have seen, requiring that all α_i and β_i are nonnegative has important theoretical ramifications.

Appendix B. Practical monotonicity. In this appendix, we will give a rationale for defining monotonicity using both conditions (5.14) and (5.15).

Writing out (3.12) gives

$$-\frac{DV}{D\tau} + \frac{\sigma^2 S^2}{2} V_{SS} + (r - \lambda\kappa) S V_S - rV + \rho \max(V^* - V, 0) + \mathcal{J}V = 0 \tag{B.1}$$

with $\mathcal{J}V = \lambda \int_0^\infty \{V(S\eta) - V(S)\} g(\eta)\, d\eta$, or

$$\begin{aligned} G\left(V, \frac{DV}{D\tau}, V_S, V_{SS}, \mathcal{J}V\right) &= -\frac{DV}{D\tau} + \frac{\sigma^2 S^2}{2} V_{SS} + (r - \lambda\kappa) S V_S - rV \\ &\quad + \rho \max(V^* - V, 0) + \mathcal{J}V = 0. \end{aligned} \tag{B.2}$$

Let $\zeta > 0$ be a constant, independent of (S, A, τ), then from (B.2) we have

$$\begin{aligned} &G\left(V + \zeta, \frac{D(V+\zeta)}{D\tau}, (V+\zeta)_S, (V+\zeta)_{SS}, \mathcal{J}(V+\zeta)\right) \\ &\quad = G\left(V, \frac{DV}{D\tau}, V_S, V_{SS}, \mathcal{J}V\right) - r\zeta - \rho\left[\max(V^* - V, 0) - \max(V^* - V - \zeta, 0)\right] \\ &\quad < G\left(V, \frac{DV}{D\tau}, V_S, V_{SS}, \mathcal{J}V\right) \quad \text{if} \quad \zeta, r > 0. \end{aligned} \tag{B.3}$$

Recall our notation for the discretized form of (5.13). At each node $(S_i, A_j), \tau = \tau^{n+1}$, the discrete form of G is

$$\left[G(V, \frac{DV}{D\tau}, V_S, V_{SS}, \mathcal{J}V)\right]_{i,j}^{n+1} = g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}) = 0. \tag{B.4}$$

If $g_{i,j}$ is a consistent discretization of (B.2) then we must have ($\zeta = constant > 0$)

$$g_{i,j}(V_{i,j}^{n+1} + \zeta, \{V_{k,j}^{n+1} + \zeta\}_i, \{V^n + \zeta\}) < g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}). \tag{B.5}$$

Assuming that condition (5.14) holds (which is the usual monotonicity condition in the viscosity solution literature), then it follows that

$$g_{i,j}(V_{i,j}^{n+1} + \zeta, \{V_{k,j}^{n+1}\}_i, \{V^n\}) \le g_{i,j}(V_{i,j}^{n+1} + \zeta, \{V_{k,j}^{n+1} + \zeta\}_i, \{V^n + \zeta\}). \tag{B.6}$$

Hence, (B.5)–(B.6) imply that

$$g_{i,j}(V_{i,j}^{n+1} + \zeta, \{V_{k,j}^{n+1}\}_i, \{V^n\}) < g_{i,j}(V_{i,j}^{n+1}, \{V_{k,j}^{n+1}\}_i, \{V^n\}), \tag{B.7}$$

which is condition (5.15).

In summary, condition (5.14) and consistency of the discretization implies condition (5.15). However, as a stand-alone condition, (5.14)–(5.15) are more appealing, in that these conditions have a simple physical interpretation. This definition of monotonicity is also consistent with the definition of monotonicity in the CFD literature.

To see this, consider the case where g_{ij} in (5.13) is differentiable. Then we can restate Definition 5.3 as

$$\begin{aligned} &\frac{\partial g_{i,j}}{\partial V_{i,j}^{n+1}} < 0, \\ &\frac{\partial g_{i,j}}{\partial y} \ge 0; \quad y \in \{V_{k,j}^{n+1}\}_i, \\ &\frac{\partial g_{i,j}}{\partial z} \ge 0; \quad z \in \{V^n\}. \end{aligned} \tag{B.8}$$

If g_{ij} satisfies conditions (B.8), then we have for any $V_{i,j}^{n+1}$ satisfying (B.4)

$$\begin{aligned} &\frac{\partial V_{i,j}^{n+1}}{\partial y} \ge 0; \quad y \in \{V_{k,j}^{n+1}\}_i, \\ &\frac{\partial V_{i,j}^{n+1}}{\partial z} \ge 0; \quad z \in \{V^n\}. \end{aligned} \tag{B.9}$$

In other words, a positive perturbation of any of $\{V_{k,j}^{n+1}\}_i$, $\{V_{k,l}^n\}$ results in a nonnegative perturbation of $V_{i,j}^{n+1}$. This has the intuitive interpretation in terms of fluid or heat flows, that is, discrete heat diffusion should always flow from high-temperature nodes to low-temperature nodes. Hence Definition 5.3 is commonly used [29, 25].

Another advantage of the definitions (5.14)–(5.15) which may not be obvious from condition (5.14) and consistency, is that a consequence of conditions (5.14)–(5.15) is a simple sufficient test for l_∞ stability. Let

$$(V_{min}^n)_{i,j} \le V_{i,j}^n \le (V_{max}^n)_{i,j} \tag{B.10}$$

(these bounds need not be tight) and set

$$\max_{i,j}(V_{max}^n)_{i,j} = V_{max}^n \quad \text{and} \quad \min_{i,j}(V_{min}^n)_{i,j} = V_{min}^n. \tag{B.11}$$

Then if $g_{i,j}$ satisfies conditions (5.14)–(5.15) we have that

$$g_{i,j}((V_{max}^{n+1})_{i,j}, V_{max}^{n+1}, V_{max}^{n}) = 0 \quad \text{and} \quad g_{i,j}((V_{min}^{n+1})_{i,j}, V_{min}^{n+1}, V_{min}^{n}) = 0. \tag{B.12}$$

Equation (B.12) holds for all (i, j), and so in particular, (B.12) holds for (p, q) where $V_{max}^{n+1} = (V_{max}^{n+1})_{p,q}$. Hence V_{max}^{n+1} satisfies

$$g_{p,q}(V_{max}^{n+1}, V_{max}^{n+1}, V_{max}^{n}) = 0. \tag{B.13}$$

Similarly, at node (r, s), where $V_{min}^{n+1} = (V_{min}^{n+1})_{r,s}$, V_{min}^{n+1} satisfies

$$g_{r,s}(V_{min}^{n+1}, V_{min}^{n+1}, V_{min}^{n}) = 0. \tag{B.14}$$

Hence by examining (B.13)–(B.14) for all p, q and for all r, s, we can obtain worst case bounds for $V_{max}^{n+1}, V_{min}^{n+1}$ in terms of V_{max}^{n}, V_{min}^{n}. Equations (B.13)–(B.14) are usually easy to solve by inspection, since most terms will be zero for a consistent discretization.

REFERENCES

[1] A. Allievi and R. Bermejo, *Finite element modified method of characteristics for the Navier-Stokes equation*, Internat. J. Numer. Methods Fluids, 32 (2000), pp. 439–464.

[2] A. L. Amadori, *The Obstacle Problem for Nonlinear Integro-differential Equations Arising in Option Pricing*, Quadem. IACQ 21-000, 2000.

[3] L. Andersen and J. Andreasen, *Jump-diffusion processes: Volatility smile fitting and numerical methods for option pricing*, Rev. Derivatives Research, 4 (2000), pp. 231–262.

[4] J. Andreasen, *The pricing of discretely sampled Asian and lookback options*, J. Comput. Finance, 2 (Fall, 1998), pp. 5–30.

[5] G. Barles, *Convergence of numerical schemes for degenerate parabolic equations arising in finance*, in Numerical Methods in Finance, L. C. G. Rogers and D. Talay, eds., Cambridge University Press, Cambridge, UK, 1997, pp. 1–21.

[6] G. Barles and P. Souganidis, *Convergence of approximation schemes for fully nonlinear second order equations*, Asymptot. Anal., 4 (1991), pp. 271–283.

[7] J. Barraquand and T. Pudet, *Pricing of American path-dependent contingent claims*, Math. Finance, 6 (1996), pp. 17–51.

[8] J. Becker, *A second order backward difference method with variable timesteps for a parabolic problem*, BIT, 38 (1998), pp. 644–662.

[9] A. Belanger, P. Forsyth, and K. Vetzal, *Valuing employee stock options with turnover*, in Proceedings of the Northern Finance Association, St. Johns, Newfoundland, Canada, 2004.

[10] R. Bermejo, *Analysis of a class of quasi-monotone and conservative semi-Lagrangian advection schemes*, Numer. Math., 87 (2001), pp. 597–623.

[11] P. P. Boyle, *New life forms on the option landscape*, J. Financial Engrg., 2 (1996), pp. 217–252.

[12] P. P. Boyle and D. Emanuel, *Options on the General Mean*, manuscript, University of British Columbia, Vancouver, Canada, 1980.

[13] M. Briani, C. L. Chioma, and R. Natalini, *Convergence of numerical schemes for viscosity solutions to integro-differential degenerate parabolic problems arising in financial theory*, Numer. Math., 98 (2004), pp. 607–646.

[14] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*, Chapman & Hall, London, 2004.

[15] R. Cont and E. Voltchkova, *A Finite Difference Scheme for Option Pricing in Jump Diffusion and Exponential Levy Models*, Internal Report 513, CMAP, Ecole Polytechnique, Paris, France, 2003.

[16] M. Dempster, J. Hutton, and D. Richards, *LP valuation of exotic American options exploiting structure*, J. Comput. Finance, 2 (Fall, 1998), pp. 61–84.

[17] J. N. Dewynne and P. Wilmott, *A note on average rate options with discrete sampling*, SIAM J. Appl. Math., 55 (1995), pp. 267–277.

[18] Y. d'Halluin, P. A. Forsyth, and G. Labahn, *A penalty method for American options with jump diffusion processes*, Numer. Math., 97 (2004), pp. 321–352.

[19] Y. d'Halluin, P. A. Forsyth, and K. R. Vetzal, *Robust numerical methods for contingent claims under jump diffusion processes*, IMA J. Numer. Anal., 25 (2005), pp. 87–112.
[20] J. Douglas, Jr., and T. F. Russell, *Numerical methods for convection-dominated diffusion problems based on combining the method of characteritics with finite element or finite difference procedures*, SIAM J. Numer. Anal., 19 (1982), pp. 871–885.
[21] B. Eraker, M. Johannes, and N. Polson, *The impact of jumps in volatility and returns*, J. Finance, 58 (2003), pp. 1269–1300.
[22] M. Falcone and R. Ferretti, *Convergence analysis for a class of high-order semi-Lagrangian advection schemes*, SIAM J. Numer. Anal., 35 (1998), pp. 909–940.
[23] P. A. Forsyth and K. R. Vetzal, *Quadratic convergence for valuing American options using a penalty method*, SIAM J. Sci. Comput., 23 (2002), pp. 2095–2122.
[24] P. A. Forsyth, K. R. Vetzval, and R. Zvan, *Convergence of numerical methods for valuing path-dependent option using interpolation*, Rev. Derivatives Research, 5 (2002), pp. 273–314.
[25] J. Fuhrmann and H. Langmach, *Stability and existence of solutions of time-implicit finite volume schemes for viscous nonlinear conservation laws*, Appl. Numer. Math., 37 (2001), pp. 201–230.
[26] J. Hugger, *Wellposedness of the boundary value formulation of a fixed strike Asian option*, J. Comput. Methods Sci. Engrg., to appear.
[27] J. Hull and A. White, *Efficient procedures for pricing European and American path-dependent options*, J. Derivatives, 1 (1993), pp. 21–31.
[28] L. Jiang and M. Dai, *Convergence of binomial tree methods for European/American path-dependent options*, SIAM J. Numer. Anal., 42 (2004), pp. 1094–1109.
[29] R. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in Mathematics, Birkhäuser Verlag, Basel, 1992.
[30] A. L. Lewis, *Fear of jumps*, Wilmott Magazine, December 2002, pp. 60–67.
[31] M. D. Marcozzi, *On the valuation of Asian options by variational methods*, SIAM J. Sci. Comput., 24 (2003), pp. 1124–1140.
[32] R. C. Merton, *Option pricing when underlying stock returns are discontinuous*, J. Financial Engrg., 3 (1976), pp. 125–144.
[33] G. H. Meyer, *The numerical valuation of options with underlying jumps*, Acta Math. Univ. Comenian. (N.S.), 67 (1998), pp. 69–82.
[34] G. H. Meyer, *On pricing American and Asian options with PDE methods*, Acta Math. Univ. Comenian. (N.S.), 70 (2001), pp. 153–165.
[35] O. A. Oleinik and E. V. Radkevic, *Second Order Equations with Nonnegative Characteristic Form*, AMS, Providence, RI, 1973.
[36] C. Oosterlee, F. Gaspar, and J. Frisch, *WENO and blended BDF discretizations for Asian options*, Comput. Vis. Sci., 6 (2004), pp. 131–138.
[37] K. Parrott and N. Clarke, *A parallel solution of early exercise Asian options with stochastic volatility*, in Proceedings of the 11th Domain Decomposition Conference, Greenwich, 1998.
[38] H. Pham, *Optimal stopping of controlled jump diffusion processes: A viscosity solution approach*, J. Math. Systems Estim. Control, 8 (1998), pp. 1–27.
[39] O. Pironneau, *On the transport diffusion algorithm and its applications to the Navier-Stokes equations*, Numer. Math., 38 (1982), pp. 309–332.
[40] D. Pooley, P. Forsyth, and K. Vetzal, *Numerical convergence properties of option pricing PDEs with uncertain volatility*, IMA J. Numer. Anal., 23 (2003), pp. 241–267.
[41] D. M. Pooley, P. A. Forsyth, and K. R. Vetzal, *Convergence remedies for non-smooth payoffs in option pricing*, J. Comput. Finance, 6 (Summer, 2003), pp. 25–40.
[42] R. Rannacher, *Finite element solution of diffusion problems with irregular data*, Numer. Math., 43 (1984), pp. 309–327.
[43] P. Ritchken and R. Trevor, *Interest rate options with a volatility hump*, Rev. Derivatives Research, 3 (1999), pp. 237–262.
[44] D. Tavella and C. Randall, *Pricing Financial Instruments: The Finite Difference Method*, John Wiley & Sons, New York, 2000.
[45] J. Večeř, *A new PDE approach for pricing arithmetic average Asian options*, J. Comput. Finance, 4 (Summer, 2001), pp. 105–113.
[46] P. Wilmott, *Derivatives: The Theory and Practice of Financial Engineering*, John Wiley & Sons, West Sussex, England, 1998.
[47] H. Windcliff, P. Forsyth, and K. Vetzal, *Analysis of the stability of the linear boundary condition for the Black-Scholes equation*, J. Comput. Finance, 8 (Fall, 2004), pp. 65–92.
[48] R. Zvan, P. Forsyth, and K. Vetzal, *Penalty methods for American options with stochastic volatility*, J. Comput. Appl. Math., 91 (1998), pp. 199–218.

[49] R. Zvan, P. Forsyth, and K. Vetzal, *A finite volume approach for contingent claims valuation*, IMA J. Numer. Anal., 21 (2001), pp. 703–731.

[50] R. Zvan, P. A. Forsyth, and K. R. Vetzal, *Robust numerical methods for PDE models of Asian options*, J. Comput. Finance, 1 (Winter, 1998), pp. 39–78.

[51] R. Zvan, P. A. Forsyth, and K. R. Vetzal, *Discrete Asian barrier options*, J. Comput. Finance, 3 (Fall, 1999), pp. 41–67.

SIAM J. SCI. COMPUT.
Vol. 27, No. 1, pp. 346–367
© 2005 Society for Industrial and Applied Mathematics

HIGHER-ORDER, CARTESIAN GRID BASED FINITE DIFFERENCE SCHEMES FOR ELLIPTIC EQUATIONS ON IRREGULAR DOMAINS*

KAZUFUMI ITO†, ZHILIN LI†, AND YAW KYEI‡

Abstract. Second and fourth order Cartesian grid based finite difference methods are proposed for elliptic and parabolic partial differential equations, and associated eigenvalue problems on irregular domains with general boundary conditions. Our methods are based on the continuation of a solution idea using multivariable Taylor's expansion of the solution about selected boundary points, and the core ideas of the immersed interface method. The methods offer systematic treatment of the general boundary conditions in two- and three-dimensional domains and are directly applied to semi-discretize heat equations on irregular domains. Convergence analysis and numerical examples are presented. The validity and effectiveness of the proposed methods are demonstrated through our numerical results including computations of the eigenvalues of the associated eigenvalue problem.

Key words. fourth order finite difference schemes, elliptic and parabolic equations on irregular domains, immersed interface method

AMS subject classifications. 65N06, 65N12, 65N22, 65F15, 65Y20, 35J67

DOI. 10.1137/03060120X

1. Introduction.

In this paper we develop second and fourth order Cartesian grid based finite difference methods for second order elliptic and parabolic partial differential equations, and the associated eigenvalue problems on irregular domains.

Let Ω be a bounded open domain in R^2 or R^3 with a smooth boundary Γ. Consider the elliptic boundary value problem

$$-\operatorname{div}(\mu(x)\nabla u) + c(x)u = f(x), \quad x \in \Omega \tag{1.1}$$

with a boundary condition

$$\mu(x)\frac{\partial}{\partial \nu}u + \alpha(x)u = g(x), \quad x \in \Gamma, \tag{1.2}$$

where ν is the outward normal to the boundary Γ and we assume that μ, f, c, α, g are sufficiently smooth.

The problems of our present interest in this work have been extensively investigated. Many of the different finite difference methods related approaches in the literature on the problem considered using regular Cartesian grids [1, 7, 11, 19], employ an extension of some sort in the solution algorithm. We describe some of the relevant research. In [23], the authors use Lagrange interpolations of the solution and the given Dirichlet boundary condition along each coordinate direction and obtain a modified finite difference scheme at stencils that have grid points lying in both the inside and outside of the irregular domain. They combined the second order method with a deferred correction or Richardson extrapolation to obtain a fourth order method.

*Received by the editors October 15, 2003; accepted for publication (in revised form) December 2, 2004; published electronically September 12, 2005. The authors are partially supported by ARO grant 43751-MA and by NFS grants DMS-0201094 and DMS-0413183.

http://www.siam.org/journals/sisc/27-1/60120.html

†Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (kito@math.ncsu.edu, zhilin@math.ncsu.edu).

‡Department of Mathematics, North Carolina A & T State University, Greensboro, NC 27411-0001 (ykyei@ncat.edu).

In [17], the author developed a fourth order accurate finite difference scheme for solving Poisson equations on irregular regions. An extension is carried out by a hybrid method incorporating an integral equation technique at the boundary. Recently, the approach has been extended in [18] to evaluate particular solutions of biharmonic/harmonic equations on general regions in two dimensions. The irregular domain is embedded in a larger rectangle, and then the problem is recast as an elliptic interface problem with discontinuity in the source function such that the solution is harmonic outside the irregular domain. In this work, the author uses a Taylor's expansion about irregular grid points in the coordinate directions and the integral representation of the particular solution to complete the equation near the interface.

A second order accurate finite difference scheme using a finite volume discretization has been developed in [7] to solve variable Poisson equations with Dirichlet boundary conditions in one and two dimensions. Their approach embeds the domain in a regular Cartesian grid and uses the flux at the cell boundaries. In two dimensions, the boundary is approximated by straight line segments within the cells. In order to evaluate the flux, the normal direction is determined at the midpoint of each boundary line segment in each cell. Then a quadratic extrapolation of the solution at intersection points of the normal to the boundary segment with two selected Cartesian grid lines and the boundary value on the normal is determined.

The derivative of the extrapolation function at the cell boundary is then the approximated flux. As a result, the coefficient matrix of the linear system of equations obtained is not symmetric.

Based on the fast immersed interface method [12], a second order fast Poisson solver on irregular domains was developed in [15]. The method embeds the irregular domain into a rectangular box or a cube. The problem is then extended to the rectangular domain to become an interface problem with an augmented unknown of the jump in the normal derivative of the solution. The method involves the GMRES iteration for the augmented variable that is only defined along the boundary and a fast Poisson solver on the regular domain. The method has been applied to several problems in two and three dimensions [6, 14].

Then in [16], the basic idea of the ghost fluid method was employed to develop a first order accurate symmetric finite difference scheme based on Cartesian grids to solve a variable Poisson equation in the presence of an irregular interface. Subsequently in [4], the approach in [16] was modified to obtain a second order accurate symmetric finite difference scheme based on Cartesian grids to solve a variable Poisson equation with Dirichlet boundary conditions. The modification used the signed distance level set function to obtain a linear interpolation from the boundary value and the solution values in coordinatewise directions to determine the ghost fluid values.

Our extension employs continuation of the solution u from Ω to the outside of domain Ω. The continuation procedure uses the multivariable Taylor's expansion of the solution u at selected boundary points. The Taylor coefficients are then determined by incorporating the boundary conditions and the equation. This procedure directly follows the core ideas of the immersed interface method (IIM) [2, 11, 13]. For the fourth order, our method is based on the standard compact nine point stencil scheme [21, 22], the continuation of solution, and the fourth order multivariable Taylor's expansion of u in local coordinates. The Taylor coefficients are determined by differentiating the equation twice in the local coordinate and collocating the boundary values at the selected boundary points.

The contribution of this paper includes (i) a systematic treatment by both second order and fourth order methods of the general boundary value problems in two

and three dimensions, (ii) direct application to the heat equation and the associated eigenvalue problem, and (iii) the symmetry and positive definiteness of the coefficient matrix of the resulting linear system of equations. For the fourth order method, differentiations of quantities along the boundary can be avoided by collocating the boundary values at selected boundary points (see section 4). This particular approach is a new one in capturing the geometry, boundary values and their derivatives, and minimizing the effort for implementation.

Specifically, our methods offer the following merits:

- They handle general boundary conditions (see section 3) in a systematic manner.
- They are extended to three-dimensional (3D) cases (see section 4) without any significant effort.
- The fourth order method requires minor modifications (see section 5) from the second order one. This shows the effectiveness of our methodology. Specifically, we demonstrate it for elliptic eigenvalue problems in this paper.
- Only the diagonal entries of the coefficient matrix need to be modified at each grid point near the boundary. Thus, the resulting coefficient matrices of the linear system of equations are symmetric and positive definite and maintain the sparsity of the standard finite difference scheme.
- The extension of the fourth order method to the heat equation is carried out (see section 6) and the method has many potential applications.
- The implementation of the methods are carried out with great ease. Work involved is to solve low (fixed) order linear equations to determine the diagonal entries of the coefficient matrix and the corresponding right-hand side at each grid-point near the boundary. Specifically, we implemented our methods when the boundary is described by the zero level set of a function (we only require the function values at the Cartesian grid points).

For numerical tests we use the sparse matrix solvers in MATLAB. We also tested an iterative method through our eigenvalue computations using the ▪◢ routine in MATLAB which employs an Arnold iteration algorithm. Extensions of the proposed methods to the interface problems discussed in [2, 11] are being considered, and the extension of the fourth order method to the variable coefficient case is also of interest to us.

The paper is organized as follows. In section 2 we introduce the basic idea and the description of the methods by using the Poisson equation on an irregular domain with a Dirichlet boundary condition in R^2 as a benchmark problem. Section 2 provides a general guideline and the scope of our proposed methods without the burden of technical details. We give necessary details and how to apply the method for the following specific cases below. In section 3 we describe the second order method for general boundary conditions including the Neumann boundary condition. In section 4 we describe the second order method for 3D cases. We describe the fourth order method in section 5 and follow up in section 6 with the discussion of time-dependent, (i.e., heat) equations and the corresponding eigenvalue problem. In section 7 we present numerical tests and demonstrate the capability of the proposed algorithms. We also report a stability and convergence rate analysis in section 8 while the stability of methods is clearly demonstrated through our test numerics.

2. The basic ideas and the continuation method. In this section we describe the basic ideas of the proposed method for a Poisson equation $-\Delta u = f$ with a Dirichlet boundary condition $u = g$ on Γ in R^2. We assume that the domain Ω is

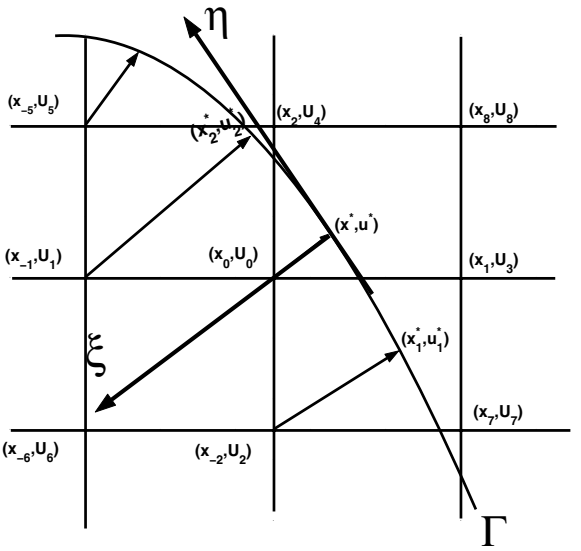


FIG. 1. *A diagram of an irregular grid point x_0, its projection x^* on the interface Γ, and the local coordinates (ξ, η).*

contained in the square $[0,1] \times [0,1]$. We use a uniform Cartesian grid $x_{i,j} = (ih, jh)$, $0 \le i,\ j \le n$ with step size $h = 1/n$. Let $U_{i,j}$ be the finite difference solution at the $x_{i,j}$ grid point. We say (i,j) grid is a [illegible] if all the grid points in the standard five point stencil are in $\overline{\Omega}$. At regular grid points, we use the standard five point central finite difference scheme. At an [illegible] where the standard five point stencil contains grid points outside of $\overline{\Omega}$, we apply the continuation method. We use Figure 1 to illustrate our method and analysis.

Let U_0 denote the value at the irregular grid point x_0, U_1, U_2 be the values at inside grid points x_{-1}, x_{-2} and U_3, U_4 at outside grid points x_1, x_2. If we know the values U_3 and U_4, then we can apply the standard five point central finite difference at x_0,

$$\frac{4U_0 - U_1 - U_2 - U_3 - U_4}{h^2} = f_{i,j}, \tag{2.1}$$

and complete the system for $U_{i,j}$. We represent U_3 and U_4 as a function of U_0, f, and g as follows.

[illegible] 1. We project the irregular grid point x_0 onto the boundary point x^* by the orthogonal projection onto Γ. As a result there is a family of the projected points x_k^* on Γ with the associated family of irregular grid points.

[illegible] 2. Let (η, ξ) be the local orthogonal coordinates in the tangential and normal directions at x^*. We expand the solution u at x^* by

$$u \sim \tilde{u} = \Phi(\eta, \xi) = u^* + u_\xi \xi + u_\eta \eta + \frac{1}{2} u_{\xi\xi} \xi^2 + u_{\xi\eta} \xi\eta + \frac{1}{2} u_{\eta\eta} \eta^2. \tag{2.2}$$

If the solution u is C^3 in a neighborhood of x^*, then the truncation error of (2.2) is of order $O(h^3)$.

[illegible] 3. Among the Taylor coefficients, $u^* = g(x^*)$ and $u_\eta = g_\tau(x^*)$ are known from the boundary condition and thus we have a total of 4 unknowns in (2.2), selected

to be $(u_\xi, u_{\xi\xi}, u_{\xi\eta}, u_{\eta\eta})$. Let x_1^* and x_2^* be the two closest projected points to x^* on Γ as shown in Figure 1. We use the following four conditions to obtain the complete system for the unknown $(u_\xi, u_{\xi\xi}, u_{\xi\eta}, u_{\eta\eta})$:

$$u(x_0) = U_0, \qquad u(x_1^*) = g(x_1^*), \qquad u(x_2^*) = g(x_2^*), \qquad u_{\xi\xi} + u_{\eta\eta} = -f(x^*). \tag{2.3}$$

If we eliminate $u_{\xi\xi}$ by

$$u_{\xi\xi} = -f(x^*) - u_{\eta\eta}, \tag{2.4}$$

and let $u = \tilde{u}$ in (2.4), we then obtain

$$\begin{pmatrix} \xi_0 & \xi_0\eta_0 & .5(\eta_0^2 - \xi_0^2) \\ \xi_1 & \xi_1\eta_1 & .5(\eta_1^2 - \xi_1^2) \\ \xi_2 & \xi_2\eta_2 & .5(\eta_2^2 - \xi_2^2) \end{pmatrix} \begin{pmatrix} u_\xi \\ u_{\xi\eta} \\ u_{\eta\eta} \end{pmatrix} = \begin{pmatrix} U_0 - g(x^*) - g_\tau(x^*)\eta_0 + .5f(x^*)\xi_0^2 \\ g(x_1^*) - g(x^*) - g_\tau(x^*)\eta_1 + .5f(x^*)\xi_1^2 \\ g(x_2^*) - g(x^*) - g_\tau(x^*)\eta_2 + .5f(x^*)\xi_2^2 \end{pmatrix}, \tag{2.5}$$

where (η_i, ξ_i), $i = 0, 1, 2$ are the local coordinates of x_0, x_1^*, x_2^*, respectively. Thus the unknown $(u_\xi, u_{\xi\xi}, u_{\xi\eta}, u_{\eta\eta})$ is a linear function of $(U_0, f(x^*), g(x^*), g(x_1^*), g(x_2^*), g_\tau(x^*))$.

, , 4. Let (ξ_i, η_i), $i = 4, 5$ denote the local coordinates of x_1, x_2, respectively. We complete (2.1) by setting

$$\begin{pmatrix} U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} \Phi(\xi_4, \eta_4) \\ \Phi(\xi_5, \eta_5) \end{pmatrix} = \begin{pmatrix} 1 & \eta_4 & \xi_4 & \xi_4\eta_4 & .5(\eta_4^2 - \xi_4^2) & -.5\xi_4^2 \\ 1 & \eta_5 & \xi_5 & \xi_5\eta_5 & .5(\eta_5^2 - \xi_5^2) & -.5\,\xi_5^2 \end{pmatrix} \begin{pmatrix} g(x^*) \\ g_\tau(x^*) \\ u_\xi \\ u_{\xi\eta} \\ u_{\eta\eta} \\ f(x^*) \end{pmatrix} \tag{2.6}$$

and thus obtain, at the irregular grid point,

$$\frac{c_{i,j}U_0 - U_1 - U_2}{h^2} = f_{i,j} + \tilde{f}_{i,j}, \tag{2.7}$$

where $c_{i,j} \geq 2$ depends only on the local geometry of Γ and $\tilde{f}_{i,j}$ depends on the data $f(x^*)$, $g(x^*)$, $g(x_1^*)$, $g(x_2^*)$, and $g_\tau(x^*)$.

Hence, we obtain the complete system and the resulting system matrix with diagonal $c_{i,j}$ replacing 4 and $f_{i,j} + \tilde{f}_{i,j}$ replacing $f_{i,j}$ at irregular grid points. Work involved in the proposed algorithm is to first, determine the projection point of x^* of each irregular grid point and second, evaluate $c_{i,j}$ and $\tilde{f}_{i,j}$ in (2.7) by (2.5)–(2.6) at each irregular grid point. The resulting system matrix is symmetric and positive definite and has the same sparsity as the original central finite difference scheme. Also, note that the local truncation error of (2.7) at the irregular grid point is of order $O(h)$.

It is necessary to evaluate $g_\tau(x^*)$ better than the first order in the proposed algorithm (2.5)–(2.7). Alternatively we can avoid the evaluation of $g_\tau(x^*)$ as follows. We select $(u_\eta, u_\xi, u_{\xi\eta}, u_{\eta\eta})$ as unknowns in (1.3) and use the five conditions,

$$u(x_0) = U_0, \qquad u(x_1^*) = g(x_1^*), \qquad u(x_2^*) = g(x_2^*), \qquad u(x_3^*) = g(x_3^*),$$
$$u_{\xi\xi} + u_{\eta\eta} = -f(x^*).$$

That is, we drop the condition $u_\tau = g_\tau(x^*)$ but add the condition $u(x_3^*) = g(x_3^*)$ at the next closest projection point x_3^*. Thus we carry out (2.8) instead of (2.5),

$$\begin{pmatrix} \xi_0 & \eta_0 & \xi_0\eta_0 & .5(\eta_0^2-\xi_0^2) \\ \xi_1 & \eta_1 & \xi_1\eta_1 & .5(\eta_1^2-\xi_1^2) \\ \xi_2 & \eta_2 & \xi_2\eta_2 & .5(\eta_2^2-\xi_2^2) \\ \xi_3 & \eta_3 & \xi_3\eta_3 & .5(\eta_3^2-\xi_3^2) \end{pmatrix} \begin{pmatrix} u_\xi \\ u_\eta \\ u_{\xi\eta} \\ u_{\eta\eta} \end{pmatrix} = \begin{pmatrix} U_0 - g(x^*) + .5f(x^*)\xi_0^2 \\ g(x_1^*) - g(x^*) + .5f(x^*)\xi_1^2 \\ g(x_2^*) - g(x^*) + .5f(x^*)\xi_2^2 \\ g(x_3^*) - g(x^*) + .5f(x^*)\xi_3^2 \end{pmatrix} \tag{2.8}$$

to determine the second order Taylor expansion (2.2). In this way we obtain the alternative finite difference scheme at the irregular grid point x^* of the form (2.7) but $\tilde{f}_{i,j}$ now depends on the data $f(x^*)$, $g(x^*)$, $g(x_1^*)$, $g(x_2^*)$, and $g(x_3^*)$ instead. Again, the resulting system matrix is symmetric and positive definite.

REMARK 1. Both options based on (2.5) and (2.8) are tested numerically. The second option (2.8) has the advantage that it does not need to evaluate the tangential derivative $g_\tau(x^*)$ but it uses the additional boundary conditions. For all specific extensions of the proposed method described in sections 3–6, both options will be discussed and the same remark as above applies.

REMARK 2. For the variable coefficient case the second order methods are based on the central finite difference approximation

$$\begin{aligned} &-\frac{b_{i+1/2,j}\frac{U_{i+1,j}-U_{i,j}}{h} - b_{i-1/2,j}\frac{U_{i,j}-U_{i-1,j}}{h}}{h} \\ &-\frac{b_{i,j+1/2}\frac{U_{i,j+1}-U_{i,j}}{h} - b_{i,j-1/2}\frac{U_{i,j}-U_{i,j-1}}{h}}{h} = f_{i,j}, \end{aligned} \tag{2.9}$$

where

$$b_{i\pm1/2,j} = \mu(x_{i\pm1/2}, y_j) \quad \textit{and} \quad b_{i,j\pm1/2} = \mu(x_i, y_{j\pm1/2}),$$

which replaces (2.1). Thus, we will mainly discuss the Poisson equation throughout the paper.

2.1. The level set function and the local coordinate system. In order to carry out Step 1, we represent the boundary Γ as the zero level set of a level set function ϕ,

$$\Gamma = \{x \in R^d : \phi(x) = 0\}, \qquad \Omega = \{x \in R^d : \phi(x) < 0\}.$$

Then x_0 is regular if

$$\phi(x_0 \pm he_i) \le 0 \quad \text{for all } 1 \le i \le d.$$

Using the level set function ϕ we can determine the projection points x^* and the local coordinate system (ξ, η) (see section 2.1). In this way our implementation of the proposed algorithms is very systematic and easy.

We assume that ϕ is given at only the grid points $x_{i,j}$. First we describe our procedure for determining the projection point x^* of the irregular grid point x_0 and the local coordinate (ξ, η) in R^2. We fix the direction d as the central finite difference approximation of the normal vector at $x_0 = x_{i,j}$, i.e.,

$$\tilde{d} = \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2h}\right), \qquad d = \frac{\tilde{d}}{|\tilde{d}|}. \tag{2.10}$$

Then we approximate ϕ by the quadratic form

$$\phi(x_0 - \alpha d) \sim P(\alpha) = \phi(x_0) - \alpha \tilde{d} d + \frac{\alpha^2}{2} d^t H d, \tag{2.11}$$

where H is the central finite difference approximation of the Hessian at x_0,

(2.12)
$$H = \begin{pmatrix} \frac{\phi_{i+1,j} - 2\phi_{i,j} + \phi_{i-1,j}}{h^2} & \frac{\phi_{i+1,j+1} - \phi_{i-1,j+1} - \phi_{i+1,j-1} + \phi_{i-1,j-1}}{4h^2} \\ \frac{\phi_{i+1,j+1} - \phi_{i-1,j+1} - \phi_{i+1,j-1} + \phi_{i-1,j-1}}{4h^2} & \frac{\phi_{i,j+1} - 2\phi_{i,j} + \phi_{i,j-1}}{h^2} \end{pmatrix}.$$

Let $\bar{\alpha} \in R^+$ satisfy $P(\bar{\alpha}) = 0$ and set $x^* = x_0 - \bar{\alpha} d$. Then we set the local coordinate system (ξ, η) by

$$\xi = \left(\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h}, \frac{\phi_{i,j+1} - \phi_{i,j-1}}{2h} \right) / |\tilde{d}|$$
$$\eta = \left(\frac{\phi_{i,j+1} - \phi_{i,j-1}}{2h}, -\frac{\phi_{i+1,j} - \phi_{i-1,j}}{2h} \right) / |\tilde{d}|.$$

We describe the higher order method for determining the projection point x^* and the corresponding local coordinate at x^* in section 5.

3. General boundary conditions. We consider the Poisson equation $-\Delta u = f$ with boundary condition

$$\frac{\partial}{\partial \nu} u + \alpha u = g, \quad x \in \Gamma. \tag{3.1}$$

Using the local coordinate (ξ, η), the boundary Γ is described by

$$\xi = \chi(\eta) \quad \text{with } \chi'(0) = 0,\ \chi''(0) = \kappa, \tag{3.2}$$

where κ is the mean curvature of Γ at x^* and

$$\kappa = \frac{\phi_y^2 \phi_{xx} - 2\phi_x \phi_y + \phi_x^2 \phi_{yy}}{(\phi_x^2 + \phi_y^2)^{3/2}}(x^*). \tag{3.3}$$

Thus we approximate κ by first evaluating $\kappa_{i,j}$ at all grid points $x_{i,j}$ by (3.3) using the central finite difference approximation of the derivatives ϕ_x, ϕ_y, ϕ_{xx}, ϕ_{xy}, and ϕ_{yy} and then using an interpolation of $\kappa_{i,j}$ to get κ at x^*. Moreover, (3.1) is written as

$$u_\xi(\chi(\eta), \eta) - \chi'(\eta) u_\eta(\chi(\eta), \eta) = \sqrt{1 + \chi'(\eta)^2}(g(\eta) - \alpha u(\chi(\eta), \eta)). \tag{3.4}$$

Letting $\eta = 0$ in (3.4), we have

$$u_\xi = g(x^*) - \alpha u^*. \tag{3.5}$$

Moreover, taking the derivative of (2.4) with respect to η and letting $\eta = 0$, we have

$$u_{\xi\eta} - \kappa u_\eta = g'(x^*) - \alpha u_\eta. \tag{3.6}$$

Thus, among the Taylor coefficients in (1.4) we select $(u^*, u_\eta, u_{\eta\eta})$ as unknowns and the remaining can be eliminated by (3.5)–(3.6) and (3.1). Now, we use the three equations

$$u(x_0) = U_0, \qquad \frac{\partial}{\partial \nu} u(x_1^*) + \alpha u(x_1^*) = g(x_1^*), \qquad \frac{\partial}{\partial \nu} u(x_2^*) + \alpha u(x_2^*) = g(x_2^*) \tag{3.7}$$

to determine the unknowns. Let $u = \tilde{u} = \Phi(\xi, \eta)$. Then the second and third equations in (3.7) become

$$(\nu^* \cdot \nu_i^*)\Phi_\xi + (\tau^* \cdot \nu_i^*)\Phi_\eta - \alpha\Phi = g(x_i^*)$$

at (ξ_i, η_i), $i = 1, 2$. Thus, (3.7) is written as

$$\mathbf{A}\begin{pmatrix} u^* \\ u_\eta \\ u_{\eta\eta} \end{pmatrix} = \mathbf{f}, \tag{3.8}$$

where $\mathbf{A}$ is given as

$$\begin{pmatrix} 1-\alpha\xi_0 & \eta_0+(\kappa-\alpha)\xi_0\eta_0 & .5(\eta_0^2-\xi_0^2) \\ \alpha(1-\alpha\xi_1-a_1) & b_1+(\kappa-\alpha)(a_1\eta_1+b_1\xi_1)+\alpha(\eta_1+(\kappa-\alpha)\xi_1\eta_1) & b_1\eta_1-a_1\xi_1+\frac{\alpha}{2}(\eta_1^2-\xi_1^2) \\ \alpha(1-\alpha\xi_2-a_2) & b_2+(\kappa-\alpha)(a_2\eta_2+b_2\xi_2)+\alpha(\eta_2+(\kappa-\alpha)\xi_2\eta_2) & b_2\eta_2-a_2\xi_2+\frac{\alpha}{2}(\eta_2^2-\xi_2^2) \end{pmatrix},$$

$$\mathbf{f} = \begin{pmatrix} u_0-g(x^*)\xi_0-g_\tau(x^*)\xi_0\eta_0+.5f(x^*)\xi_0^2 \\ g(x_1^*)-a_1g(x^*)-(a_1\eta_1+b_1\xi_1)g_\tau(x^*)+a_1f(x^*)\xi_1-\alpha\left(g\xi_1+g_\tau\xi_1\eta_1-\frac{1}{2}f\xi_1^2\right) \\ g(x_1^*)-a_1g(x^*)-(a_2\eta_2+b_2\xi_2)g_\tau(x^*)+a_2f(x^*)\xi_2-\alpha\left(g\xi_2+g_\tau\xi_2\eta_2-\frac{1}{2}f\xi_2^2\right) \end{pmatrix}$$

for $1 \le i \le 2$, and

$$a_i = (\nu^* \cdot \nu_i^*), \qquad b_i = (\tau^* \cdot \nu_i^*).$$

The second option, where (3.6) is replaced by the additional boundary condition at x_3^*, does not require the evaluation of $g_\tau(x^*)$ as well as κ and is carried out by

$$\begin{pmatrix} 1-\alpha\xi_0 & \eta_0 & \xi_0\eta_0 & .5(\eta_0^2-\xi_0^2) \\ \alpha(1-\alpha\xi_1)-\alpha a_1 & b_1+\alpha\eta_1 & (a_1\eta_1+b_1\xi_1)+\alpha\xi_1\eta_1 & b_1\eta_1-a_1\xi_1+\frac{\alpha}{2}(\eta_1^2-\xi_1^2) \\ \alpha(1-\alpha\xi_2)-\alpha a_2 & b_2+\alpha\eta_2 & (a_2\eta_2+b_2\xi_2)+\alpha\xi_2\eta_2 & b_2\eta_2-a_2\xi_2+\frac{\alpha}{2}(\eta_2^2-\xi_2^2) \\ \alpha(1-\alpha\xi_3)-\alpha a_3 & b_3+\alpha\eta_3 & (a_3\eta_3+b_3\xi_3)+\alpha\xi_3\eta_3 & b_3\eta_3-a_3\xi_3+\frac{\alpha}{2}(\eta_3^2-\xi_3^2) \end{pmatrix}\begin{pmatrix} u^* \\ u_\eta \\ u_{\xi\eta} \\ u_{\eta\eta} \end{pmatrix}$$

$$= \begin{pmatrix} u_0-g(x^*)\xi_0+.5f(x^*)\xi_0^2 \\ g(x_1^*)-a_1g(x^*)+a_1f(x^*)\xi_1+\alpha\left(\frac{1}{2}f\xi_1^2-g\xi_1\right) \\ g(x_2^*)-a_2g(x^*)+a_2f(x^*)\xi_1+\alpha\left(\frac{1}{2}f\xi_2^2-g\xi_2\right) \\ g(x_3^*)-a_3g(x^*)+a_3f(x^*)\xi_1+\alpha\left(\frac{1}{2}f\xi_3^2-g\xi_3\right) \end{pmatrix}$$

in order to determine the second order Taylor's expansion (2.2).

4. The 3D case. In this section we discuss the 3D cases. We highlight the necessary steps to extend the method in R^2 to R^3.

We use the standard seven-point central finite difference at $x_0 = x_{i,j,k}$,

$$\frac{6U_{i,j} - U_{i-1,j,k} - U_{i+1,j,k} - U_{i,j-1,k} - U_{i,j+1,k} - U_{i,j,k-1} - U_{i,j,k+1}}{h^2} = f_{i,j}. \tag{4.1}$$

If x_0 is an irregular grid point, then we project x_0 onto the boundary Γ by the formulas corresponding to (2.10)–(2.12). Then we set the local coordinate system (ξ, η, τ) by

$$\xi = \frac{d}{|d|}, \qquad \eta = \frac{(d_2, -d_1, 0)}{|d|}, \qquad \tau = \frac{(d_3d_1, d_3d_2, -d_1^2 - d_2^2)}{\sqrt{(d_3d_1)^2 + (d_3d_2)^2 + (d_1^2 + d_2^2)^2}}$$

if $\sqrt{d_1^2+d_2^2} \geq \sqrt{d_2^2+d_3^2}$; otherwise

$$\xi = \frac{d}{|d|}, \qquad \tau = \frac{(d_3, 0, -d_2)}{|d|}, \qquad \eta = \frac{(-d_1 d_2, d_1^2+d_3^2, -d_2 d_3)}{\sqrt{(d_1 d_2)^2+(d_1^2+d_3^2)^2+(d_2 d_3)^2}}.$$

Here (η, τ) stands for the tangential coordinate and

$$d = (d_1, d_2, d_3) = \left(\frac{\phi_{i+1,j,k} - \phi_{i-1,j}}{2h}, -\frac{\phi_{i,j+1,k} - \phi_{i,j-1,k}}{2h}, \frac{\phi_{i,j,k+1} - \phi_{i,j,k-1}}{2h} \right).$$

We expand the solution u in a neighborhood of x^* by

$$\begin{aligned} u \sim \tilde{u} = \Phi(\xi, \eta, \tau) = u^* + u_\xi \xi + u_\eta \eta + u_\tau \tau + \frac{1}{2}(u_{\xi\xi}\xi^2 + u_{\eta\eta}\eta^2 + u_{\tau\tau}\tau^2) \\ + u_{\xi\eta}\xi\eta + u_{\xi\tau}\xi\tau + u_{\eta\tau}\eta\tau. \end{aligned} \tag{4.2}$$

Among the Taylor coefficients, $u^* = g(x^*)$ and the tangential derivatives u_η, u_τ are known from the boundary condition and thus we have the total of 7 unknowns in (4.2), i.e., u_ξ and the second order derivatives. Let x_i^*, $i = 1, \ldots, 5$ be the five closest projected points to x^* on Γ. We use the following seven conditions to obtain the complete system for the unknowns.

$$\begin{aligned} u(x_0) = U_0, \quad u(x_i^*) = g(x_i^*), \quad 1 \leq i \leq 5 \\ u_{\xi\xi} + u_{\eta\eta} + u_{\tau\tau} = -f(x^*). \end{aligned} \tag{4.3}$$

If we eliminate $u_{\xi\xi}$ by

$$u_{\xi\xi} = -f(x^*) - u_{\eta\eta} - u_{\tau\tau}, \tag{4.4}$$

and let $u = \tilde{u}$ in (3.3), then we obtain the 6×6 equation

$$A(u_\xi, u_{\xi\eta}, u_{\xi\tau}, u_{\eta\tau}, u_{\eta\eta}, u_{\tau\tau}) = b, \tag{4.5}$$

where

$$\text{the } i\text{th row } A = [\xi_i \;\; \xi_i\eta_i \;\; \xi_i\tau_i \;\; \eta_i\tau_i \;\; .5(\eta_i^2 - \xi_i^2) \;\; .5(\eta_i^2 - \xi_i^2)], \quad 0 \leq i \leq 5$$

and

$$\begin{aligned} b_0 &= U_0 - g(x^*) - u_\eta \eta_0 - u_\tau \tau_0 + .5 f(x^*)\xi_0^2 \\ b_i &= g(x_i^*) - g(x^*) - u_\eta \eta_i - u_\tau \tau_i + .5 f(x^*)\xi_i^2, \quad 1 \leq i \leq 5, \end{aligned}$$

where (η_i, ξ_i), $i = 0, 1, \ldots, 5$ are the local coordinates of $x_0, x_1^*, \ldots, x_5^*$, respectively. Thus, all unknowns in (4.2) are determined as a linear function of $(U_0, f(x^*), g(x^*), g_\eta(x^*), g_\tau(x^*))$ $(g(x_1^*), \ldots, g(x_5^*))$. Now, we complete (4.1) at the irregular grid point by expressing the value $u_{i\pm1,j\pm1,k\pm1}$ corresponding to the grid points outside of Ω by (4.2). Again we only modify the diagonal entry $c_{i,j,k}$ and the corresponding right-hand side at each irregular point by (4.2) and (4.5).

. . . . (1) For the second option, in which we avoid the evaluation of the tangential derivatives $(g_\eta(x^*), g_\tau(x^*))$, we carry out (4.6) instead of (4.5)

$$A(u_\xi, u_\eta, u_\tau, u_{\xi\eta}, u_{\xi\tau}, u_{\eta\tau}, u_{\eta\eta}, u_{\tau\tau}) = b, \tag{4.6}$$

where we use the seven (instead of five) additional boundary conditions at x_i^*, $1 \le i \le 7$ and

$$\text{the } i\text{th row } A = [\xi_i \;\; \eta_i \;\; \tau_i \;\; \xi_i\eta_i \;\; \xi_i\tau_i \;\; \eta_i\tau_i \;\; .5(\eta_i^2 - \xi_i^2) \;\; .5(\eta_i^2 - \xi_i^2)], \quad 0 \le i \le 7$$

and

$$b_0 = U_0 - g(x^*) + .5f(x^*)\xi_0^2, \quad b_i = g(x_i^*) + .5f(x^*)\xi_i^2 \quad \text{for } 1 \le i \le 7.$$

(2) For the mixed boundary condition (3.1) we use the following equality (3.5)–(3.6):

$$\begin{aligned} u_\xi &= g(x^*) - \alpha u^* \\ u_{\xi\eta} - \chi_{\eta\eta} u_\eta &= g_\eta(x^*) - \alpha u_\eta \\ u_{\xi\tau} - \chi_{\tau\tau} u_\tau &= g_\tau(x^*) - \alpha u_\tau \end{aligned}$$

or the corresponding second option.

5. The fourth order method. In this section we describe the fourth order method. We apply the nine point standard compact finite difference scheme [22],

$$\begin{aligned} &\frac{1}{h^2}\left(\frac{10}{3}u_{i,j} - \frac{2}{3}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})\right. \\ &\qquad \left. -\frac{1}{6}(u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+,j+1})\right) \\ &\qquad = \frac{8}{12}f_{i,j} + \frac{1}{12}(f_{i-1,j} + f_{i+1,j} + f_{i,j-1} + f_{i,j+1}). \end{aligned} \tag{5.1}$$

The grid point x_0 is irregular if the standard nine point stencil contains grid points outside of $\overline{\Omega}$. At each irregular grid point we complete (5.1) by the fourth order continuation procedure as follows.

(1) The fourth order projection of x_0 onto the boundary Γ is carried out by the fourth order polynomial interpolation of ϕ [5] based on the standard 25-point grid points given by

$$\phi \sim \sum_{m=0}^{4} \sum^{i+j=m} \frac{\phi_{ij}^{(m)}}{m!} \left(\frac{x_1}{h}\right)^i \left(\frac{x_2}{h}\right)^j \tag{5.2}$$

with

$$\begin{aligned} \phi_{10}^{(1)} &= \frac{1}{12}(\phi_{i-2,j} - 8\phi_{i-1,j} + 8\phi_{i+1,j} + \phi_{i+2,j}), \\ \phi_{20}^{(2)} &= \frac{1}{12}(-\phi_{i-2,j} + 16\phi_{i-1,j} - 30\phi_{i,j} + 16\phi_{i+1,j} - \phi_{i+2,j}), \\ \phi_{30}^{(3)} &= \frac{1}{8}(-4\phi_{i-2,j} + 8\phi_{i-1,j} - 8\phi_{i+1,j}4\phi_{i+2,j}), \\ \phi_{40}^{(4)} &= \phi_{i-2,j} - 4\phi_{i-1,j} + 6\phi_{i,j} - 4\phi_{i+1,j} + \phi_{i+2,j}, \end{aligned}$$

and

$$\begin{aligned}
\phi_{11}^{(2)} &= [1\ \ -8\ \ 0\ \ 8\ \ -1\ \ -8\ \ 64\ \ 0\ \ -64\ \ 8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 8\ \ -64\ \ 0\ \ 64\ \ -8 \\
&\quad -1\ \ 8\ \ 0\ \ -8\ \ 1]\phi/144, \\
\phi_{21}^{(3)} &= [-1\ \ 8\ \ 0\ \ -8\ \ 1\ \ 16\ \ -128\ \ 0\ \ 128\ \ -16\ \ -30\ \ 240\ \ 0\ \ -240\ \ 30\ \ 16\ \ -128 \\
&\quad 0\ \ 128\ \ -16\ \ -1\ \ 8\ \ 0\ \ -8\ \ 1]\phi/144, \\
\phi_{31}^{(4)} &= [-4\ \ 32\ \ 0\ \ -32\ \ 4\ \ 8\ \ -64\ \ 0\ \ 64\ \ -8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ -8\ \ 64\ \ 0\ \ -64\ \ 8 \\
&\quad 4\ \ -32\ \ 0\ \ 32\ \ -4]\phi/96, \\
\phi_{22}^{(4)} &= [1\ \ -16\ \ 30\ \ -16\ \ 1\ \ -16\ \ 256\ \ -480\ \ 256\ \ -16\ \ 30\ \ -480\ \ 900\ \ -480\ \ 30 \\
&\quad -16\ \ 256\ \ -480\ \ 256\ \ -16\ \ 1\ \ -16\ \ 30\ \ -16\ \ 1]\phi/144,
\end{aligned}$$

where ϕ is the ordered vector consisting of the function values at the 25 standard grid points centered on x_0. The normal and tangent vectors at the projected point x^* are then computed by using (5.2).

(2)–(4) We expand u in a neighborhood of x^* by the fourth order Taylor polynomial in the local coordinates (ξ, η)

$$u \sim \sum_{m=0}^{4} \sum_{i=0}^{m} \frac{C(m,i) u_{i,j}^{(m)}}{m!} \xi^i \eta^j. \tag{5.3}$$

We use the equality of the equation at x^*

$$\begin{cases}
u_{\xi\xi} + u_{\eta\eta} = -f(x^*), \\
u_{\xi\xi\xi} + u_{\xi\eta\eta} = -f_\xi(x^*), \\
u_{\xi\xi\eta} + u_{\eta\eta\eta} = -f_\eta(x^*), \\
u_{\xi\xi\xi\xi} + u_{\xi\xi\eta\eta} = -f_{\xi\xi}(x^*), \\
u_{\xi\xi\xi\eta} + u_{\xi\eta\eta\eta} = -f_{\xi\eta}(x^*), \\
u_{\xi\xi\eta\eta} + u_{\eta\eta\eta\eta} = -f_{\eta\eta}(x^*).
\end{cases} \tag{5.4}$$

Thus we select $X = (u_\xi, u_\eta, u_{\xi\eta}, u_{\eta\eta})$ and $(u_{\xi\eta\eta}, u_{\eta\eta\eta}, u_{\xi\xi\eta\eta}, u_{\xi\eta\eta\eta})$ as unknowns in R^8. We use the equality $u(x_0) = U_0$ along with the 7 additional Dirichlet boundary conditions $u(x_i^*) = g(x_i^*)$ at the closest projection points x_i^*, $1 \le i \le 7$ to x^* on Γ. Hence we have an 8×8 system

$$AX = b \tag{5.5}$$

with

$$\begin{aligned}
A_i = [&\xi_i\ \ \eta_i\ \ \xi_i\eta_i\ \ (\eta_i^2 - \xi_i^2)/2\ \ (3\xi_i\eta_i^2 - \xi_i^3)/6\ \ (\eta_i^3 - 3\xi_i\eta_i^2)/6 \\
&(\xi_i\eta_i^3 - \xi_i^3\eta_i)/6\ \ (6\xi_i^2\eta_i^2 - \xi_i^4 - \eta_i^4)/24]
\end{aligned}$$

for $0 \le i \le 7$, and

$$\begin{aligned}
b_0 &= U_0 - g(x^*) - f(x^*)\xi_0^2/2 - f_\xi\xi_0^3/6 - f_\eta\eta_0\xi_0^2/2 - f_{\xi\xi}\xi_0^4/24 \\
&\quad -f_{\xi\eta}\eta_0\xi_0^3/6 - f_{\eta\eta}\eta_0^4/24, \\
b_i &= g(x_i^*) - g(x^*) - f(x^*)\xi_i^2/2 - f_\xi\xi_i^3/6 - f_\eta\eta_i\xi_i^2/2 - f_{\xi\xi}\xi_i^4/24 \\
&\quad -f_{\xi\eta}\eta_i\xi_i^3/6 - f_{\eta\eta}\eta_i^4/24
\end{aligned}$$

for $1 \le i \le 7$, to determine the Taylor polynomial (5.3).

At each irregular grid point only the diagonal entry $c_{i,j}$ and the corresponding right-hand side need to be modified and thus the resulting matrix is symmetric and keeps the sparsity.

(1) If we are able to evaluate g_η accurately, then we may use the equality $u_\eta = g_\eta(x^*)$ and (2)–(4) to reduce to the option where only the six boundary conditions $u(x_i^*) = g(x_i^*)$ (instead of seven) are equated and thus a 7×7 system determines completely the Taylor polynomial.

(2) For the mixed boundary condition $u(x) + \alpha\frac{\partial}{\partial\nu}u(x) = g(x)$, $x \in \Gamma$ we have

$$u(x_i^*) + \alpha\frac{\partial}{\partial\nu}u = g(x_i^*)$$

at each projection point x_i^* and

$$\frac{\partial}{\partial\nu}u = \nu_i^* \cdot \nu^* u_\xi + \nu_i^* \cdot \tau^* u_\eta,$$

where (ν^*, τ^*) is the normal-tangent vector pair at x^* and ν_i^* is the normal vector at x_i^*. Thus we select $X = (u^*, u_\eta, u_{\xi\eta}, u_{\eta\eta})$ and $(u_{\xi\eta\eta}, u_{\eta\eta\eta}, u_{\xi\xi\eta\eta}, u_{\xi\eta\eta\eta})$ as the unknown and use $AX = b$ (with (A, b) similar to (3.6) and appropriate additional higher order terms) to determine the Taylor polynomial (5.3).

6. The fourth order method for the heat equation. In this section we describe the fourth order scheme for the heat equation on the irregular domain Ω,

$$u_t(t,x) = \Delta u, \qquad u(0,x) = u_0(x) \tag{6.1}$$

with boundary condition

$$u(t,x) + \frac{\partial}{\partial\nu}u(t,x) = g(t,x). \tag{6.2}$$

For the simplicity of our presentation we only describe the method for the homogeneous (in time) Dirichlet boundary condition $u(t,x) = g(x)$, $x \in \Gamma$. But we indicate the specific changes needed for the general cases.

In order to develop the fourth order method we treat u_t as a forcing function f and the right-hand side of (5.1) $f_{i,j}$, is replaced by $\frac{d}{dt}u_{i,j}$; i.e.,

$$\begin{aligned}\frac{1}{h^2}\Bigg(&\frac{10}{3}u_{i,j} - \frac{2}{3}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})\\ &-\frac{1}{6}(u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+,j+1})\Bigg)\\ &+\frac{d}{dt}\left(\frac{8}{12}u_{i,j} + \frac{1}{12}(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1})\right) = 0.\end{aligned} \tag{6.3}$$

and then we apply the fourth order method as in section 5. Thus we need to be able to evaluate the vector $Y = (u_t(x^*), u_{t\xi}(x^*), u_{t\eta}(x^*), u_{t\xi\xi}(x^*), u_{t\xi\eta}(x^*), u_{t\eta\eta}(x^*))$ at x^*. To this end we expand u_t in a neighborhood of x^* by the second order Taylor polynomial

$$u_t \sim u_{t\xi}(x^*)\xi + \frac{1}{2}u_{t\xi\xi}(x^*)\xi^2 + u_{t\xi\eta}\xi\eta + \frac{1}{2}u_{t\eta\eta}(x^*)\eta^2, \tag{6.4}$$

where $u_t(x^*) = u_{t\eta}(x^*) = 0$ and

$$u_{t\xi\xi} + u_{t\eta\eta} = 0 \quad \text{at } x^*. \tag{6.5}$$

As in Step 2, we determine (6.4) completely by

$$\begin{pmatrix} \xi_0 & \xi_0\eta_0 & .5(\eta_0^2 - \xi_0^2) \\ \xi_1 & \xi_1\eta_1 & .5(\eta_1^2 - \xi_1^2) \\ \xi_2 & \xi_2\eta_2 & .5(\eta_2 - \xi_2^2) \end{pmatrix} \begin{pmatrix} u_{t\xi} \\ u_{t\xi\eta} \\ u_{t\eta\eta} \end{pmatrix} = \begin{pmatrix} (U_0)_t \\ 0 \\ 0 \end{pmatrix}, \tag{6.6}$$

where (η_i, ξ_i), $i = 0, 1, 2$ are the local coordinates of x_0, x_1^*, x_2^*, respectively. Thus the vector Y is determined as $Y = (U_0)_t V$ with $V \in R^7$. Now, we know completely the right-hand side of (5.4) as $-Y$ and we proceed to determine the fourth order Taylor expansion (5.3) completely. In this way we complete (6.3) at each irregular grid point x_0 by substituting expressions (5.3) and (6.4) into (6.3) for the corresponding values outside the domain. The work involved is first, to solve (6.6) to determine the second order Taylor expansion (6.4) for u_t and second, to solve (5.5) to determine the fourth order Taylor expansion (5.3) for u at x^*. As a consequence at each irregular grid point, the diagonal entries $\frac{8}{3}$ and $\frac{8}{12}$ in (6.3) are modified accordingly and also the right-hand side due to the nonhomogeneous boundary condition $u = g(x)$ (otherwise, no changes are required). Thus the resulting system of equations is of the form

$$QU_t = -HU + \tilde{F}, \qquad U(0) = u_0(x_{i,j}), \tag{6.7}$$

where Q and H are symmetric sparse matrices. The positivity of the matrices Q and H is demonstrated and tested numerically for our test examples.

We demonstrate the applicability of the fourth order method by the approximation of eigenvalue problem for (6.1)–(6.2), i.e., finding the eigenpair $(\lambda, u) \in R \times L^2(\Omega)$ satisfying

$$\Delta u = \lambda u, \quad u|_\Gamma = 0. \tag{6.8}$$

We simply consider the generalized eigenvalue problem for (6.7),

$$\lambda QU = -HU \tag{6.9}$$

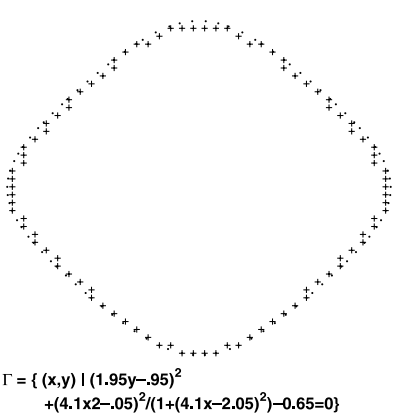
and thus the eigenpairs (λ, U) approximate the ones for (6.8).

Remark 3. For the Dirichlet nonhomogeneous condition $u(t, x) = g(t, x)$, $x \in \Gamma$ we have

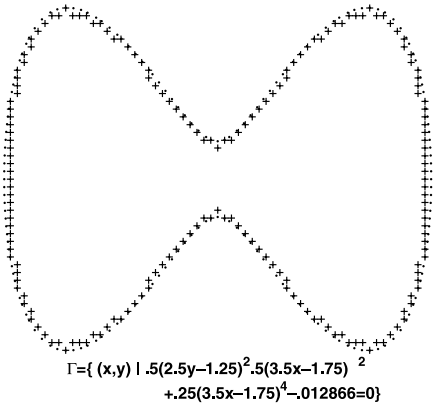
$$u_{t\xi\xi} + u_{t\eta\eta} = u_{tt}, \quad \text{at } x^*.$$

In order to avoid the second order derivatives in time, we do not equate (6.4) as an equality condition for determining the second order Taylor expansion for u_t. Instead we use the additional Dirichlet boundary conditions $u_t(x_3^*) = g_t(x_3^*)$, where x_3^* is the next closest projection point to x^*. Thus (6.6) is replaced by

$$\begin{pmatrix} \xi_0 & \xi_0\eta_0 & \eta_0^2/2 & \xi_0^2/2 \\ \xi_1 & \xi_1\eta_1 & \eta_1^2/2 & \xi_1^2/2 \\ \xi_2 & \xi_2\eta_2 & \eta_2^2/2 & \xi_2^2/2 \\ \xi_3 & \xi_3\eta_3 & \eta_3^2/2 & \xi_3^2/2 \end{pmatrix} \begin{pmatrix} u_{t\xi} \\ u_{t\xi\eta} \\ u_{t\eta\eta} \\ u_{t\xi\xi} \end{pmatrix} = \begin{pmatrix} (U_0)_t - g_t(x^*) - g_{t\tau}(x^*)\eta_0 \\ g_t(x_1^*) - g_t(x^*) - g_{t\tau}(x^*)\eta_1 \\ g_t(x_2^*) - g_t(x^*) - g_{t\tau}(x^*)\eta_2 \\ g_t(x_3^*) - g_t(x^*) - g_{t\tau}(x^*)\eta_3 \end{pmatrix}. \tag{6.10}$$



(a) Diamond looking boundary geometry.



(b) Butterfly looking boundary geometry.

FIG. 2. *Nonrectangular boundary geometries.*

Thus Y is of the form $Y = (U_0)_t V + \tilde{V}(t)$, where $\tilde{V} \in R^7$ depends on $(g_t(x^*), g_t(x_i^*)$, $1 \le i \le 3)$. The step to determine the fourth order Taylor expansion for u remains the same and the resulting system has the form

$$QU_t = -HU + \tilde{F}_1(t) + \frac{d}{dt}\tilde{F}_2(t),$$

where only the diagonal entry of Q at each irregular grid point is modified and depends on the geometry of the boundary Γ and the corresponding right side $\tilde{F}_1(t)$, $\tilde{F}_2(t)$ also depend on the boundary values $g(t, x)$.

For the mixed boundary condition $\frac{\partial}{\partial \nu}u + \alpha u = g$ we use exactly the same procedures in section 3 to determine the second order Taylor expansion of u_t in a neighborhood of x^*. For example, if g is time invariant, the second option uses

$$\begin{pmatrix} 1-\alpha\xi_0 & \eta_0 & \xi_0\eta_0 & .5(\eta_0^2-\xi_0^2) \\ \alpha(1-\alpha\xi_1)-\alpha a_1 & b_1+\alpha\eta_1 & (a_1\eta_1+b_1\xi_1)+\alpha\xi_1\eta_1 & b_1\eta_1-a_1\xi_1+\frac{\alpha}{2}(\eta_1^2-\xi_1^2) \\ \alpha(1-\alpha\xi_2)-\alpha a_2 & b_2+\alpha\eta_2 & (a_2\eta_2+b_2\xi_2)+\alpha\xi_2\eta_2 & b_2\eta_2-a_2\xi_2+\frac{\alpha}{2}(\eta_2^2-\xi_2^2) \\ \alpha(1-\alpha\xi_3)-\alpha a_3 & b_3+\alpha\eta_3 & (a_3\eta_3+b_3\xi_3)+\alpha\xi_3\eta_3 & b_3\eta_3-a_3\xi_3+\frac{\alpha}{2}(\eta_3^2-\xi_3^2) \end{pmatrix}$$
$$\times \begin{pmatrix} u_t^* \\ u_{t\eta} \\ u_{t\xi\eta} \\ u_{t\eta\eta} \end{pmatrix} = \begin{pmatrix} (u_0)_t \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

7. Numerical experiments. We have done a number of numerical tests which confirm the order of accuracy of our proposed second and fourth order methods for different geometries and different boundary conditions. All our computations are done using the sparse matrix routines in **Matlab**. In our numerical tests, we worked on several nonrectangular domains and present some of the results on domains with boundary geometries as shown in Figures 2(a) and 2(b). In Figure 2, the "+"s indicate the irregular grid points and the "."s represent the projected points which collectively approximate the boundary geometry. We tested the schemes by using the following examples.

TABLE 1
Grid refinement using ℓ^∞ for the example in (7.1).

Fig.		Example (7.1)	
	n	Error $\|e\|_{\ell^\infty}$	Order r
2(a)	16	1.013-4	
	32	6.924-6	4.13
	64	1.688-7	5.36

Fig.		Example (7.1)	
	n	Error $\|e\|_{\ell^\infty}$	Order r
2(a)	16	1.068-4	
	32	1.250-6	6.42
	64	3.576-8	5.13

In this example the exact solution is

$$u = \sin(x^2 + y^2) \quad \text{with } f = -4\cos(x^2 + y^2) + 4(x^2 + y^2)\sin(x^2 + y^2), \tag{7.1}$$

and for the second example we have

$$\begin{aligned} &u = \exp(-((x - 0.5)^2 + (y - 0.5)^2)) \quad \text{with} \\ &f = (4 - (1 - 2x)^2 - (1 - 2y)^2)\exp(-((x - 0.5)^2 + (y - 0.5)^2)). \end{aligned} \tag{7.2}$$

The errors $e(h)$ are measured in the ℓ_2 and ℓ^∞ norms of the difference between the finite difference solutions and the exact solutions of the differential equation. The order of convergence r is given by

$$e(h) = Ch^r \tag{7.3}$$

for some C, where the step size or grid spacing is given as $h = \frac{1}{n}$. Therefore, for a constant C the order of accuracy r is calculated by

$$r = \frac{\log\left(\frac{e(h_1)}{e(h_2)}\right)}{\log\left(\frac{h_1}{h_2}\right)}$$

for any two errors due to grid spacings h_1 and h_2 [10, 3, 22].

7.1. Examples for Dirichlet and Neumann boundary value problems in two dimensions. Among the test examples we conducted, our fourth order scheme produced machine accuracy for a fourth degree polynomial for all tested geometries. We present the results of the tests for the fourth order scheme with Dirichlet boundary conditions in Table 1. The right-hand sides of the tables show the results for option two where we avoid the calculation of tangential derivatives at x^* while the left-hand sides show the results for option one where g_n is computed analytically. In the first option, the curvature at the projected points x^* is computed as described in section 3. We use a least squares procedure [10, 3] to better determine the order of convergences of the methods as indicated in (7.3) since C may not uniformly be a constant but may depend on the closeness of the irregular grid points to the boundary (i.e., C depends on h).

In Figure 3, the left graph represents the order of convergence determination for the first option with Dirichlet boundary conditions while the right graph is for the second option. For the first option results reported in the left plot of Figure 3, we plotted the errors for 31 different values of h with $n = 15, 16, 18{-}35, 37, 39, 40, 42, 43, 49, 51, 54, 57{-}60$. Then, we determined the error rate r from (7.3) by

$$\log(\|e\|_{\ell^\infty}) = 4.94\log\left(\frac{1}{n}\right) + 5.09.$$

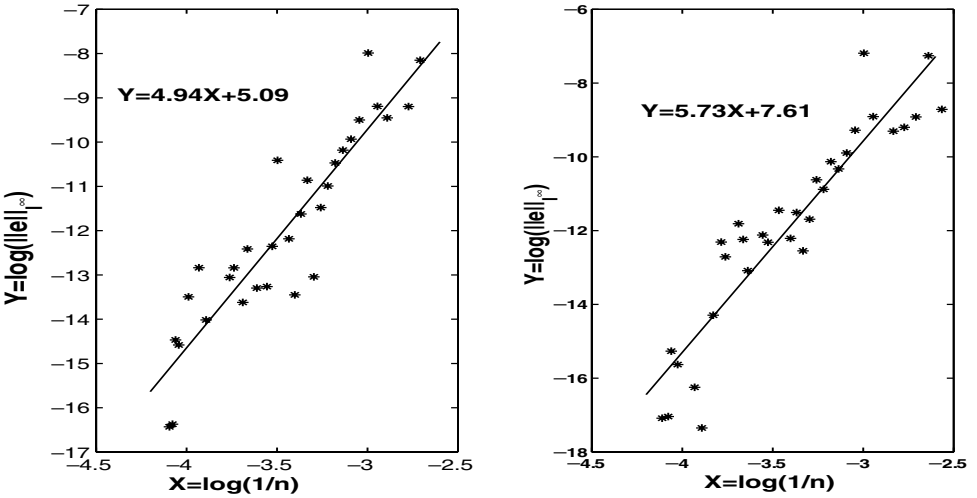


FIG. 3. *Least squares estimates of order of convergence for the first and second options shown in Table* 1.

TABLE 2
Fourth order method grid refinement analysis for Neumann conditions.

Fig.		Example (7.2)	
	n	Error $\|e\|_{\ell_2}$	Order r
2(a)	16	4.355-2	
	32	2.832-3	3.94
	64	1.186-4	4.58

Fig.		Example (7.2)	
	n	Error $\|e\|_{\ell^\infty}$	Order r
2(a)	16	5.559-3	
	32	1.715-4	5.03
	64	4.799-6	5.16

TABLE 3
Second order method grid refinement analysis for Neumann conditions.

Fig.		Example (7.2)	
	n	Error $\|e\|_{\ell_2}$	Order r
2(a)	37	1.056-2	
	74	2.495-3	2.08
	148	4.529-4	2.45

Fig.		Example (7.2)	
	n	Error $\|e\|_{\ell^\infty}$	Order r
2(a)	37	3.639-2	
	74	8.41-3	2.11
	148	4.990-4	2.44

For the second option results reported in the right plot of Figure 3, we plotted the errors for 32 different values of h with $n = 13{-}17, 19{-}30, 32, 34, 35, 38{-}40, 43, 44, 46, 49, 51, 56, 58, 59, 61$. The error rate is therefore determined by

$$\log(\|e\|_{\ell^\infty}) = 5.73 \log\left(\frac{1}{n}\right) + 7.61.$$

The results for the Neumann boundary conditions are shown in Table 2 for the fourth order method and Table 3 for the second order method.

In Figure 4, the left graph is for the fourth order method while the right graph is for the second order method with Neumann boundary conditions. In the least square determination of the rate for the fourth order method, we tried selected values of h with $n = 14{-}20, 23, 24, 27, 29, 30, 32, 34, 36, 38{-}41, 43, 47, 49{-}52, 57, 60, 61, 63$. As a

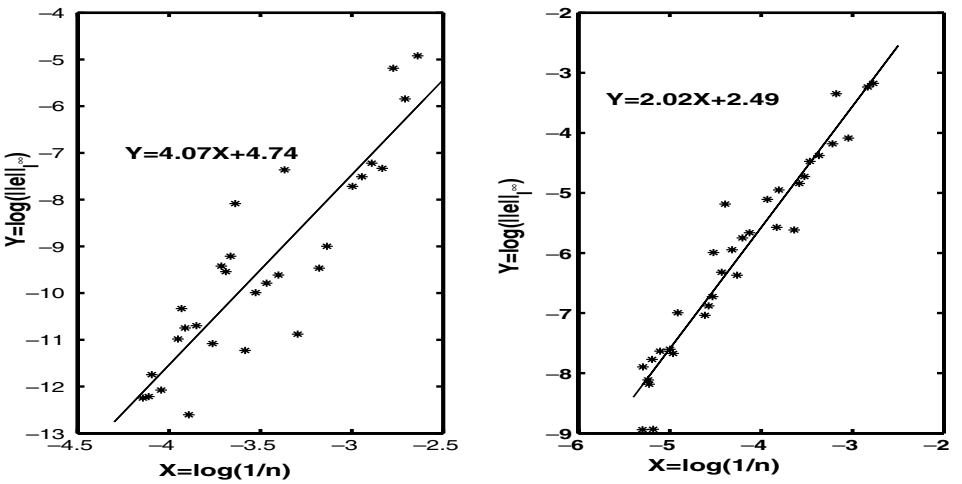


FIG. 4. *Least squares estimates of orders shown in Tables* 3 *and* 2.

result, we obtain the rate as

$$\log(\|e\|_{\ell\infty}) = 4.07 \log\left(\frac{1}{n}\right) + 4.74$$

as shown in the left graph in Figure 4.

For the second order method with Neumann boundary conditions, the different values of h tried have $n = 16, 17, 21, 24, 25, 29, 32, 34, 36, 38, 45, 46, 51, 62, 67, 71, 75, 81, 84, 92, 93, 97, 101, 136, 143, 148, 165, 178, 180, 186, 188, 199, 200$. We therefore obtained the rate as

$$\log(\|e\|_{\ell\infty}) = 2.02 \log\left(\frac{1}{n}\right) + 2.49$$

shown in the right graph in Figure 4.

In the next section, we validate the scheme by solving the associated eigenvalue problem for the circular domain and compare the estimates with exact eigenvalues.

7.2. Examples for eigenvalues and eigenfunctions in a circular domain. In order to validate our fourth order scheme, we use the scheme to solve the associated eigenvalue problem with a circular domain Ω of radius a centered at $(0, 0)$ for homogeneous Dirichlet boundary conditions. Thus we have

$$\begin{aligned} -\Delta u &= \lambda u \quad \text{for } r^2 = x^2 + y^2 < a^2, \\ u &= 0 \quad \text{for } r^2 = x^2 + y^2 = a^2. \end{aligned} \tag{7.4}$$

Through a polar coordinate transformation, we obtain the eigenpair solution as

$$\begin{pmatrix} \lambda_m, & J_m(\sqrt{\lambda} r) & \cos m\theta \\ & & \sin m\theta \end{pmatrix} \quad m = 0, 1, 2, \dots,$$

where $J_m(\cdot)$ is the Bessel function of the first kind [20] with index m and the eigenvalue λ_m satisfies

$$J_m(\sqrt{\lambda} a) = 0. \tag{7.5}$$

TABLE 4

Comparison of the estimates of the first 20 eigenvalues from our fourth order method with the exact eigenvalues for a circular domain.

λ_m^i	Exact eigenvalue	The numbers in () indicate grid points in Ω at that level $\lvert\lambda_{\text{exact}} - \lambda_{\text{estimate}}\rvert$			
	λ	$n = 21(52)$	$n = 41(216)$	$n = 81(840)$	$n = 121(1884)$
λ_0^1	140.71012075	0.00344560	0.00033383	0.00022202	0.00007944
λ_1^1	357.22556306	0.41984796	0.02957536	0.00063390	0.00004888
λ_1^1	357.22556306	0.41984796	0.02957536	0.00063390	0.00004888
λ_1^2	641.71816124	5.52882510	0.28363512	0.01667297	0.00040774
λ_1^2	641.71816124	1.80972635	0.14485952	0.02413351	0.00435833
λ_0^2	741.39324437	6.04472129	0.51548679	0.03500361	0.00839482
λ_3^1	990.42495908	7.32499973	0.14629899	0.01974972	0.01035190
λ_3^1	990.42495908	7.32499973	0.14629899	0.01974972	0.01035190
λ_1^2	1197.52935089	22.14275733	1.74116827	0.11284935	0.02544642
λ_1^2	1197.52935089	22.14275733	1.74116827	0.11284935	0.02544641
λ_4^1	1401.04479083	28.50252033	1.22523225	0.02972824	0.02677208
λ_4^1	1401.04479083	15.58313267	0.42389398	0.06932715	0.02740201
λ_2^2	1723.84425593	110.35200645	7.16848208	0.47841299	0.08676289
λ_2^2	1723.84425593	19.19016496	1.48474518	0.04257484	0.02086816
λ_0^3	1822.06829174	74.19996634	5.90117617	0.38859898	0.08757072
λ_5^1	1871.99339011	59.86580361	1.18701162	0.08585961	0.05461152
λ_5^1	1871.99339011	59.86580361	1.18701162	0.08585961	0.05461153
λ_3^2	2318.18911299	208.09760052	9.59792708	0.52775690	0.09968339
λ_3^2	2318.18911299	208.09760052	9.59792708	0.52775690	0.09968339
λ_6^1	2402.09908703	146.57829637	3.411546756	0.05850641	0.02598372

Thus, with X_m^i representing the ith root of the Bessel's function of index m, we have

$$\lambda_m^i = \left[\frac{X_m^i}{a}\right]^2, \quad i = 1, 2, \ldots . \tag{7.6}$$

We compare the estimates from our fourth order method in Table 4 for different n's with the exact eigenvalues as calculated by (7.6). The "exact" zeros of the Bessel functions X_m are obtained from Maple Software. The last four columns in Table 4 show the difference between the estimates and the exact eigenvalues. The order of convergence of our proposed fourth order method is confirmed to be 4 by employing (7.3). In the following tables, $nnz(\cdot)$ refers to the number of nonzero elements in the stiffness matrix h and the mass matrix q, respectively. [illegible] refers to the number iterations for the Matlab function [illegible] to converge with default tolerance.

In Table 5, we show the efficiency of the fourth order method and we illustrate the complexity level and the conditioning of the resulting system matrix for the fourth order method. Specifically, we use the second order central difference scheme for estimating the eigenvalues as the benchmark. The comparisons are made through the number iterations and the [illegible] required to solve the problem by employing the Matlab function [illegible]. The function [illegible] is an iterative routine for finding a few eigenvalues and eigenvectors of a square matrix based on an implicitly restarted Arnold iteration algorithm [8, 9].

In Table 5, the first row indicates the size of the matrices which are selected almost equal for both methods. Table 5 shows that the complexity level for our fourth order method is comparable to the one for the benchmark.

TABLE 5
Computational results of the first 10 *eigenvalues for a Square and circular domains.*

	λ of Square domain by second order central difference scheme			λ of circular domain by fourth order method		
	225^2	900^2	2025^2	216^2	840^2	1884^2
nnz(h)	1,065	4,380	9,945	1,788	7,244	16,480
nnz(q)	$\mathbf{I}_{225}$	$\mathbf{I}_{900}$	$\mathbf{I}_{2025}$	1,016	4,072	9,220
Iter	9	10	10	9	9	8
cpu time	0.11	0.32	0.64	0.26	0.44	0.81

TABLE 6
First 10 *eigenvalues for a circular domain computed with* eigs.

	$\lvert\lambda_{\text{exact}} - \lambda_{\text{estimate}}\rvert$ at that level					
	Matlab via pdetool		second order method		fourth order method	
			$n = 124$	$n = 250$	$n = 29$	$n = 40$
	1985^2	8065^2	1993^2	8069^2	112×112	213×213
	0.090905	0.022700	0.079863	0.019737	0.001991	0.000492
	0.681314	0.170351	0.516020	0.127375	0.100726	0.020078
	0.654829	0.163679	0.515830	0.127375	0.100724	0.020078
	2.120571	0.530002	2.024477	0.500254	0.998014	0.216279
	1.886004	0.471609	1.302557	0.321602	0.177470	0.066793
	2.899047	0.723868	2.226507	0.549263	1.457512	0.399027
	4.894667	1.224993	3.961473	0.978608	1.400833	0.255192
	4.577428	1.144803	3.961224	0.978608	1.400758	0.255192
	7.788358	1.951316	5.801800	1.432623	5.413804	1.490994
	7.396741	1.852561	5.801164	1.432623	5.413771	1.490994
nnz(h)	13633	55937	9761	39941	892	1757
nnz(q)	1985	8065	$\mathbf{I}_{1993}$	$\mathbf{I}_{8069}$	512	997
Iter	6	7	8	10	7	9
cpu time	0.58	2.91	0.51	2.71	0.17	0.25

Next, in Table 6 we compare the fourth order method with our second order method and the second order finite element method through the toolbox. The comparison is done so that our fourth order method yields similar levels of accuracy. The columns in Table 6 show the absolute error for the eigenvalue estimates. The comparison is described in terms of the and number of iterations for the iterative procedure. By that the efficiency and effectiveness of the fourth order method are demonstrated.

For the sake of completeness, the toolbox uses a similar method (a Matlab function [24]) to compute the eigenvalues in a selected interval.

8. Well-posedness and convergence rate. As we stated, the resulting matrices $-L_h$ are symmetric for our proposed algorithms. The positivity and diagonal dominance of the matrices are confirmed through our extensive numerical tests.

We generated the random geometry by selecting randomly distributed projection points x^* in the local square depicted in Figure 1 and then random graphs for the boundary Γ with the local coordinate determined as

$$\xi = \chi(\eta) = \frac{\kappa}{2}\eta^2 + \frac{\kappa_1}{6}\eta^3 + \frac{\kappa_2}{24}\eta^4,$$

where $\kappa = \chi''(0)$, $\kappa_1 = \chi'''(0)$, and $\kappa_2 = \chi''''(0)$ are selected randomly in the range

$[-5,5]$. Then, we evaluated the corresponding diagonal entry $c_{i,j}$. All the tests conducted in this manner show the resulting matrix to be diagonally dominant.

Next, we examine this analytically for the two-dimensional Dirichlet boundary condition case in (2.5)–(2.7). In (2.5) we have $\eta_0 = 0$ and the asymptotic relations

$$\eta_1 + \eta_2 = 0 + O(h^2), \quad \xi_i = \frac{\kappa}{2}\eta_i^2 + O(h^3), \quad i = 1,2. \tag{8.1}$$

To determine the diagonal entry $c_{i,j}$ in (2.7), we let $f = g = 0$. Then with (8.1) we have $u_{\xi\eta} = O(1)$,

$$\begin{pmatrix} \xi_0 & -.5\xi_0^2 \\ \xi_1 & .5\eta_1^2 \end{pmatrix} \begin{pmatrix} u_\xi \\ u_{\eta\eta} \end{pmatrix} = \begin{pmatrix} U_0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ O(h^3) \end{pmatrix}.$$

Thus

$$u_\xi = \frac{U_0}{\xi_0} + O(h^2), \qquad u_{\eta\eta} = -\frac{2\xi_1}{\eta_1^2}\frac{U_0}{\xi_0} + O(h^2),$$

and

$$c_{i,j} = 4 - \frac{\xi_3 + \xi_4}{\xi_0} + \left(\frac{\eta_3^2 - \xi_3^2}{\eta_1^2} + \frac{\eta_4^2 - \xi_4^2}{\eta_1^2}\right)\frac{\xi_1}{\xi_0} + O(h^2).$$

Since $\frac{\xi_3+\xi_4}{\xi_0} < 0$, it follows from (8.1) that

$$c_{i,j} \sim 4 - \frac{\xi_3 + \xi_4}{\xi_0} > 4.$$

8.1. Convergence rate. Since the central finite difference scheme (2.1) is diagonally dominant and the set P of all regular points is connected by their neighbors, thus the discrete maximum principle holds. If

$$(-L_h U)(p) \geq 0 \quad \text{for all } p \in P,$$

then

$$\max_{p\in P} U_p \leq \max U_q, \quad q \in Q,$$

where Q is the set of all irregular grid points.

Consider the comparison function Φ

$$\Phi(p) = \frac{1}{4}|x_p - x_0|^2.$$

Then

$$(-L_h\Phi)(p) = 1 \quad \text{for all } p \in P. \tag{8.2}$$

Define the error e by $e_p = \bar{U}_p - U_p$ with $\bar{U}_p = u(x_p)$, the exact solution evaluated at the node x_p, and the truncation error T by

$$T = (-L_h\bar{U}) - F \quad \text{since } -L_h U = F.$$

Let $\bar{T} = \max_{p\in P} |T_p|$. Then

$$(-L_h(\bar{T}\Phi + e))(p) = \bar{T} - T_p \geq 0 \quad \text{for all } p \in P.$$

It thus follows from the maximum principle and (8.2) that

$$\max_{p\in P} e_p \leq \max_{p\in P}(\bar{T}\Phi + e_p) \leq \max_{q\in Q}(\bar{T}\Phi_q + e_q). \tag{8.3}$$

At an irregular grid point $q \in Q$ we have the following two cases:

$$c_q e_q \leq \max_{p\in P} e_p + 2\max_{q\in Q} e_q + h^2|T_q|, \quad q \in Q_1$$

or

$$c_q e_q \leq 2\max_{p\in P} e_p + h^2|T_q|, \quad q \in Q_2.$$

Thus,

$$(c_q - 2)\max_{q\in Q} e_q \leq 2\max_{p\in P} e_p + h^2 \max_{q\in Q} |T_q| \tag{8.4}$$

and

$$\max_{q\in Q} e_q \leq \rho \max_{p\in P} e_p + h^2 \max_{q\in Q} \frac{|T_q|}{c_q - 2} \quad q \in Q,$$

where

$$\rho = \max\left(\frac{2}{c_q - 2}\right) < 1 \quad (c_q > 4).$$

Then it follows from (8.3)–(8.4) that

$$(1-\rho)\max_{q\in Q} e_q \leq \bar{T}\max_{q\in Q}\Phi_q + h^2 \max \frac{|T_q|}{c_q - 2} \quad q \in Q.$$

If we apply exactly the same arguments for $\bar{T}\Phi - e_p$ we obtain the same bound for $-e_p$. Hence we conclude the error estimate

$$(1-\rho)\max_{p\in P}|e_p| \leq \bar{T}\max_{q\in Q}\Phi_q + h^2 \max_{q\in Q}\frac{|T_q|}{c_q}$$

and

$$\max_{q\in Q}|e_q| \leq \rho \max_{p\in P}|e_p| + h^2 \max_{q\in Q}\frac{|T_q|}{c_q}.$$

Since $|T_p| \leq C_1 h^2$, $p \in P$, and $|T_q| \leq C_2 h$, $q \in Q$ assuming $u \in C^4(\bar{\Omega})$. The proposed second order method is of order 2 provided the diagonal dominant condition $\rho < 1$.

REFERENCES

[1] G. Birkhoff and R. E. Lynch, *Numerical Solution of Elliptic Problems*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, 1984.

[2] S. Deng, K. Ito, and Z. Li, *Three-dimensional elliptic solvers for interface problems and applications*, J. Comput. Phys., 184 (2003), pp. 215–243.
[3] A. Garbey and W. Shyy, *A least square extrapolation method for improving solution accuracy of PDE computations*, J. Comput. Phys., 186 (2003), pp. 1–23.
[4] F. Gibou, R. P. Fedkiw, L. T. Cheng, and M. Kang, *A second-order-accurate symmetric discretization of the Poisson equation on irregular domains*, J. Comput. Phys., 176 (2002), pp. 205–227.
[5] J. D. Hoffman, *Numerical Methods for Engineers and Scientists*, McGraw-Hill, Highstown, NJ, 1992.
[6] J. Hunter, Z. Li, and H. Zhao, *Autophobic spreading of drops*, J. Comput. Phys., 183 (2001), pp. 335–366.
[7] H. Johansen and P. Colella, *A cartesian grid embedded boundary method Poisson's equation on irregular domains*, J. Comput. Phys., 147 (1998), pp. 60–85.
[8] R. B. Lehoucq and D. C. Sorensen, *Deflation techniques for an implicitly restarted Arnoldi iteration*, SIAM J. Matrix Anal. Appl., 17, pp. 789–821.
[9] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[10] R. J. LeVeque, *Finite Difference Methods for Differential Equations*, DRAFT VERSION for use in courses AMath 585-6, University of Washington, Seattle, WA, 1998.
[11] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.
[12] Z. Li, *A fast iterative algorithm for elliptic interface problems*, SIAM J. Numer. Anal., 35 (1998), pp. 230–254.
[13] Z. Li and K. Ito, *Maximum principle preserving schemes for interface problems with discontinuous coefficients*, SIAM J. Sci. Comput., 23 (2001), pp. 339–361.
[14] Z. Li and C. Wang, *A fast finite difference method for solving Navier-Stokes equations on irregular domains*, Comm. Math. Sci., 1 (2003), pp. 180–196.
[15] Z. Li, H. Zhao, and H. Gao, *A numerical study of electro-migration voiding by evolving level set functions on a fixed Cartesian grid*, J. Comput. Phys., 152 (1999), pp. 281–304.
[16] X. D. Liu, R. P. Fedkiw, and M. Kang, *A boundary condition capturing method for Poisson's equation on irregular domains*, J. Comput. Phys., 160 (2000), pp. 151–178.
[17] A. Mayo, *A decomposition finite difference method for the fourth order accurate solution of Poisson's equation on general regions*, Int. J. High Speed Comput., 3 (1991), pp. 89–106.
[18] A. Mayo, *High order accurate particular solutions of the biharmonic equation on general regions*, Contemp. Math., 323 (2003), pp. 233–243.
[19] P. McCorquodale, P. Colella, and H. Johansen, *A Cartesian grid embedded boundary method for the heat equation on irregular domains*, J. Comput. Phys., 173 (2001), pp. 620–635.
[20] M. A. Pinsky, *Partial Differential Equations and Boundary-Value Problems with Applications*, 3rd ed., International Series in Pure and Applied Mathematics, Waveland Press, Lake Grove, IL, 2003.
[21] W. F. Spotz and G. F. Carey, *A high-order compact scheme for the steady stream-function vorticity equations*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 3497–3512.
[22] J. C. Strikwerda, *Finite Difference and Partial Differential Equations*, Wadsworth & Brooks Cole Advanced Books & Software, Pacific Grove, CA, 1989.
[23] V. Pereyra, W. Proskurowski, and O. Widlund, *High order fast Laplace solvers for the Dirichlet problem on general regions*, Math. Comput., 31 (1977), pp. 1–16.
[24] S. Yusef, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear Algebra Appl., 34 (1980), pp. 269–295.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 369–393
© 2005 Society for Industrial and Applied Mathematics

FAST NUMERICAL SOLUTION OF PARABOLIC INTEGRODIFFERENTIAL EQUATIONS WITH APPLICATIONS IN FINANCE*

ANA-MARIA MATACHE†, CHRISTOPH SCHWAB‡, AND THOMAS P. WIHLER§

Abstract. We numerically solve parabolic problems $u_t + \mathcal{A}u = 0$ in $(0,T) \times \Omega$, $T < \infty$, where $\Omega \subset \mathbb{R}$ is a bounded interval and $\mathcal{A}$ is a strongly elliptic integrodifferential operator of order $\rho \in [0,2]$. A discontinuous Galerkin (dG) discretization in time and a wavelet discretization in space are used. The densely populated matrices in the corresponding linear systems of equations are replaced by sparse ones using appropriate wavelet compression techniques. The linear systems in each time step are solved by an incomplete GMRES iteration. Under these conditions, we show that the complexity of our algorithm is linear (up to some logarithmic terms) in the number of spatial degrees of freedom and present error estimates. Applications to purely discontinuous Lévy processes arising in finance are given.

Key words. parabolic equations, integrodifferential operators, discontinuous Galerkin methods, wavelets, matrix compression, GMRES, Lévy processes, Markov processes

AMS subject classifications. 45K05, 65N30, 60J75, 65M12, 65M60, 65T60

DOI. 10.1137/030602617

1. Introduction.

Parabolic, possibly degenerate, evolution problems arise as Fokker–Planck equations for moments of Markovian diffusion processes X_t [15]. We mention here only micromodels of polymers [30] and asset pricing problems in mathematical finance: the celebrated Black–Scholes equation is a degenerate diffusion equation.

If the Markov process X_t is not a diffusion but a process containing jumps, infinitesimal generator of the X_t appearing in the Fokker–Planck equation is nonlocal. More precisely, if the jump process is of infinite intensity, the associated infinitesimal generators are pseudodifferential operators of order $\rho \in [0,2]$; see, e.g., [18, 19].

The discretization of such operators with standard, Galerkin type schemes, entails densely populated, ill-conditioned stiffness matrices which unduly increase the computational complexity of an implicit time-stepping scheme.

Often, the infinitesimal generators of the Markov processes generate analytic evolution semigroups, even in the case of degenerate spatial operators, so that the solutions are analytic functions of time in the open interval $(0,T]$ (see [18, Chapter 4]). This (nonuniform) time analyticity of the solution is not exploited in the classical, low-order numerical schemes such as backward differencing or implicit Runge–Kutta methods of fixed order.

*Received by the editors February 19, 2003; accepted for publication (in revised form) November 22, 2004; published electronically September 23, 2005.
http://www.siam.org/journals/sisc/27-2/60261.html

†Seminar for Applied Mathematics and RiskLab, ETH Zürich, 8092 Zürich, Switzerland (amatache@math.ethz.ch).

‡Seminar for Applied Mathematics, ETH Zürich, 8092 Zürich, Switzerland (schwab@math.ethz.ch). The research of this author was supported in part by IHP Network Breaking Complexity of the EC (contract HPRN-CT-2002-00286) with support from Swiss Federal Office for Science and Education grant BBW 02.0418.

§School of Mathematics, University of Minnesota, Minneapolis, MN 55455 (wihler@math.umn.edu). The research of this author was supported by Swiss National Science Foundation project PBEZ2-102321.

In the present paper, we analyze and implement a numerical scheme for the efficient numerical solution of Fokker–Planck equations for Markov processes with jumps. The scheme is based on *hp*-discontinuous Galerkin (dG, for short) time-stepping to exploit the time analyticity of the solution for $t > 0$ and to cope with the loss of this analyticity at $t = 0$, and on piecewise polynomial functions (of fixed degree) in space. As in [37], the matrices corresponding to the nonlocal infinitesimal generators are compressed by means of a wavelet basis, thereby reducing complexity of a matrix-vector multiplication from N^2 to $\mathcal{O}(N \log N)$, where N denotes the number of spatial degrees of freedom. At the same time, the condition numbers of the matrices are shown to be bounded independently of the spatial mesh width and of the time step. We prove that this algorithm allows us to compute the whole price vector for European vanillas on Lévy (and even certain classes of Markov) processes to a mean square accuracy of $\mathcal{O}(N^{-1})$ with N degrees of freedom in at most $\mathcal{O}(N(\log N)^8)$ operations.

We discuss in detail some aspects of the implementation, based in part on [36], and numerical examples from the pricing of European contracts.

The outline of the paper is as follows. In section 2, we introduce the parabolic problems under consideration. In section 3, our applications to finance are given. Section 4 deals with the dG and wavelet discretizations in time and space, respectively. The solution of the linear systems is studied in section 5. Section 6 contains the numerical experiments. Finally, in section 7, we summarize our work with some concluding remarks.

2. Abstract parabolic evolution problems. In this section, the parabolic problems studied in this paper are introduced. Moreover, an appropriate functional setting is established.

Let $\Omega = (a, b)$, $-\infty < a < b < \infty$, be a bounded interval in $\mathbb{R}$. For $0 \le \rho \le 2$, define the spaces

$$V = \widetilde{H}^{\rho/2}(\Omega) = \begin{cases} H^{\rho/2}(\Omega) & \text{if } 0 \le \rho < 1, \\ H_{00}^{1/2}(\Omega) & \text{if } \rho = 1, \\ H_0^{\rho/2}(\Omega) & \text{if } 1 < \rho \le 2, \end{cases}$$

where $H^{\rho/2}$, $H_{00}^{1/2}$, and $H_0^{\rho/2}$ are the usual Sobolev spaces; see [1, 24], for example. We have the dense injection $V \overset{d}{\hookrightarrow} L^2(\Omega)$, where $L^2(\Omega)$ is the space of square integrable functions equipped with the usual inner product $(\cdot, \cdot)$, as well as the Gelfand triple $V \overset{d}{\hookrightarrow} L^2(\Omega) \overset{d}{\hookrightarrow} V^*$, where $V^* = H^{-\rho/2}(\Omega)$ denotes the dual space of V. Furthermore, let $\langle \cdot, \cdot \rangle_{V^* \times V}$ be the duality pairing in $V^* \times V$, and $\| \cdot \|$, $\| \cdot \|_V$, $\| \cdot \|_{V^*}$ the norms in $L^2(\Omega)$, V, V^*, respectively.

In the time interval $J = (0, T)$, $0 < T < \infty$, consider the parabolic evolution problem

$$u'(t) + \mathcal{A}u(t) = 0 \qquad \text{in } Q = J \times \Omega, \tag{2.1}$$

$$u(0) = u_0 \tag{2.2}$$

with an initial condition $u_0 \in L^2(\Omega)$, and a possibly nonlocal operator $\mathcal{A} \in \mathcal{L}(V, V^*)$ of order ρ, given in the weak form

$$\langle \mathcal{A}u, v \rangle_{V^* \times V} = a(u, v), \qquad \mathcal{A}u \in V^*,\ u, v \in V.$$

Here, $a : V \times V \to \mathbb{R}$ is a continuous bilinear form; i.e., there exists a constant $\alpha > 0$

such that

$$|a(u,v)| \le \alpha \|u\|_V \|v\|_V \qquad \forall u, v \in V. \tag{2.3}$$

Moreover, we assume that, for some constants $\beta, \gamma > 0$, the form a satisfies a Gårding inequality, i.e.,

$$a(u,u) \ge \beta \|u\|_V^2 - \gamma \|u\|^2 \qquad \forall u \in V. \tag{2.4}$$

Henceforth, without loss of generality, the bilinear form a is supposed to be coercive on $V \times V$. Indeed, substituting $\widetilde{u} = e^{-\gamma t} u$ results in the equation

$$\widetilde{u}'(t) + (\mathcal{A} + \gamma \mathbb{I})\widetilde{u} = 0 \quad \text{in } J \times \Omega, \tag{2.5}$$

$$\widetilde{u}(0) = u_0, \tag{2.6}$$

and by (2.4) the operator $\mathcal{A} + \gamma \mathbb{I}$ is coercive with coercivity constant β.

Problem (2.1)–(2.2) has a unique (weak) solution $u(t)$, and there holds the a priori estimate (see [24])

$$\|u\|_{\mathcal{C}(\overline{J}, L^2(\Omega))} + \|u\|_{L^2(J,V)} + \|u'\|_{L^2(J,V^*)} \le C \|u_0\|_{L^2(\Omega)}.$$

In this paper, the operator $\mathcal{A}$ is assumed to be the sum of a second-order differential operator and of an integrodifferential operator with a distributional kernel $k(x, x-y) \in \mathcal{D}'(\Omega \times \Omega)$, with associated bilinear form given by

$$a(u,v) = \langle \mathcal{A}u, v \rangle_{V^* \times V} = \int_\Omega \int_\Omega k(x, x-y) u(x) v(y)\, dx\, dy.$$

The kernels $k(x,z) \in \mathcal{C}^\infty(\Omega \times \mathbb{R} \setminus \{z=0\})$ are assumed to satisfy [illegible]; i.e., for all $\alpha, \beta \in \mathbb{N}_0$, there holds

$$|D_x^\alpha D_z^\beta k(x,z)| \le C |z|^{-1-\rho-\alpha-\beta}, \qquad z \ne 0, \tag{2.7}$$

uniformly with respect to $x \in \Omega$.

3. Pricing European vanillas under Lévy processes. Parabolic problems of type (2.1)–(2.2) arise in finance in the problem of option pricing. For a general Lévy process X_t [3, 31], rather than the standard Wiener process W_t as price process for the risky underlying, the infinitesimal generator $\mathcal{L}^X$ of the transition semigroup generated by X_t is an integrodifferential operator.

Specifically, we consider the problem of pricing a European option with asset price S_t given by the exponential law

$$S_t = S_0 e^{\mu t + X_t},$$

with $\mu \in \mathbb{R}$ and X_t a Lévy process.

For a given payoff $f(S)$, the value of the European option $V(t,S)$ at $S = S_t$ with maturity T is expressed by the following discounted conditional expectation:

$$V(t,S) = \mathbf{E}^{\mathbb{Q}}[e^{-r(T-t)} f(S_T) | S_t = S], \tag{3.1}$$

where $\mathbb{Q}$ is an equivalent martingale measure and r represents the riskless mean rate of return of S_t under the measure $\mathbb{Q}$; see [5, 20, 23].

3.1. Parabolic integrodifferential equation. Itô's formula for semimartingales and the principle of no-arbitrage imply that V defined in (3.1) solves the parabolic integrodifferential equation (PIDE) (cf. [29]; see also [10, Proposition 1])

$$\begin{aligned}\frac{\partial V}{\partial t}(t,S)+\frac{\sigma^2}{2}S^2\frac{\partial^2 V}{\partial S^2}(t,S)+rS\frac{\partial V}{\partial S}(t,S)-rV(t,S)\\ +\int_{\mathbb{R}}\Big(V(t,Se^y)-V(t,S)-S(e^y-1)\frac{\partial V}{\partial S}(t,S)\Big)\nu_{\mathbb{Q}}(dy)=0\end{aligned}\tag{3.2}$$

in $J\times(0,\infty)$, and

$$V(T,S)=f(S)\tag{3.3}$$

in $(0,\infty)$. Here, by $\sigma\geq 0$, we denote the diffusion component of the Lévy process; i.e., X_t can be uniquely decomposed as $X_t=\sigma W_t+Y_t$ with W_t being a standard Brownian motion and Y_t a quadratic pure jump process independent of W_t. Furthermore, $\nu_{\mathbb{Q}}(dy)$ represents the Lévy measure satisfying

$$\int_{\mathbb{R}}\min(1,y^2)\nu_{\mathbb{Q}}(dy)<\infty.\tag{3.4}$$

The Fourier transform of a Lévy process can be expressed explicitly,

$$E_{\mathbb{Q}}[e^{-iuX_t}]=e^{-t\psi(u)},$$

in terms of the Lévy–Khintchine exponent

$$\psi(u)=\frac{\sigma^2}{2}u^2+i\gamma u+\int_{|x|<1}(1-e^{-iux}-iux)\nu_{\mathbb{Q}}(dx)+\int_{|x|\geq 1}(1-e^{-iux})\nu_{\mathbb{Q}}(dx).\tag{3.5}$$

A Lévy process is characterized by the triplet $(\sigma,\gamma,\nu_{\mathbb{Q}})$, $\sigma,\gamma\in\mathbb{R}$ and the Lévy measure $\nu_{\mathbb{Q}}$.

Henceforth, we assume that $\nu_{\mathbb{Q}}(dy)$ has a density $k_{\mathbb{Q}}$, i.e., $\nu_{\mathbb{Q}}(dy)=k_{\mathbb{Q}}(y)dy$. The Lévy density $k_{\mathbb{Q}}(y)$ describes the activity of jumps of size y in X_t. We say that the Lévy process X_t is of finite activity if $k_{\mathbb{Q}}(y)$ is integrable; otherwise X_t is said to be of infinite activity.

We drop in (3.2) the subscript $\mathbb{Q}$, and switch to logarithmic price $x=\log(S)$ and to time to maturity $t\to T-t$, thereby obtaining

$$\frac{\partial u}{\partial t}(t,x)+\mathcal{L}^X[u](t,x)=0\qquad\text{in } J\times\mathbb{R},\tag{3.6}$$

$$u(0,x)=g(x)\quad\text{in }\mathbb{R},\tag{3.7}$$

where $g(x):=f(e^x)$ and $\mathcal{L}^X=\mathcal{L}^{\mathrm{BS}}+\mathcal{L}^{\mathrm{jump}}$ with

$$\mathcal{L}^{\mathrm{BS}}[v]:=-\frac{\sigma^2}{2}v''+\left(\frac{\sigma^2}{2}-r\right)v'+rv,\tag{3.8}$$

$$\mathcal{L}^{\mathrm{jump}}[v](x):=-\int_{\mathbb{R}}\{v(x+y)-v(x)-(e^y-1)v'(x)\}k(y)dy.\tag{3.9}$$

We shall use some or all of the following assumptions on the Lévy density k [27].

(A1) [illegible]. The characteristic function $\psi_0(u)$ of the pure jump part Y_t of the Lévy process X_t satisfies the following: there exist constants c_1, $C_+ > 0$ and $\alpha < 2$ such that

$$|\psi_0(\xi) - ic_1\xi| \le C_+(1+|\xi|^2)^{\alpha/2} \qquad \forall \xi \in \mathbb{R}. \tag{3.10}$$

(A2) [illegible]. There are constants $C > 0$, $\eta_- > 0$, and $\eta_+ > 1$ such that

$$\forall |z| > 1, \quad k(z) \le C \begin{cases} e^{-\eta_-|z|} & \text{if} \quad z < 0, \\ e^{-\eta_+|z|} & \text{if} \quad z > 0. \end{cases} \tag{3.11}$$

(A3) [illegible]. For all $\beta \in \mathbb{N}_0$, there exists $C(\beta)$ such that

$$|k^{(\beta)}(z)| \le C(\alpha)|z|^{-(1+\alpha+\beta)_+} \qquad \forall z \neq 0. \tag{3.12}$$

In addition, if $\sigma = 0$, we assume that $0 < \alpha < 2$.

(A4) [illegible] $k(z)$. There is $C_- > 0$ such that

$$\forall 0 < |z| < 1, \quad \frac{1}{2}(k(-z) + k(z)) \ge \frac{C_-}{|z|^{1+\alpha}}. \tag{3.13}$$

[illegible] 3.1.

(i) Since $E_{\mathbb{Q}}[S_t] < \infty$, we have that $\int_{|z|>1} e^z k(z)dz < \infty$. This holds for all k satisfying (A1)–(A2) due to $\eta_+ > 1$ which, from now on, is assumed to be fulfilled.

(ii) Assumption (A2) implies that X_t has finite moments of all orders.

(iii) Assumption (A3) is required in the analysis of the wavelet compression of the moment matrix of $k(z)$; however, it only needs to be satisfied for a finite range of β.

(iv) A Lévy process X_t with $\sigma = 0$ is called a [illegible]. If X_t is a pure jump process, we assume that it is of infinite activity, i.e.,

$$k(z) \text{ satisfies (A1)–(A4) with } 0 < \alpha < 2 \qquad (\text{if } \sigma = 0). \tag{3.14}$$

(v) If the Lévy process is of finite activity, we assume $\sigma > 0$ and that $k(z)$ satisfies (A1)–(A3) with $\alpha < 0$.

3.2. Examples of Lévy processes. Different parameterizations of the Lévy measure have been proposed in the literature. We mention here the variance gamma (cf. [25]) (abbreviated as VG) process, with

$$k(y) = C \begin{cases} |y|^{-1}e^{-\eta_-|y|} & \text{if} \quad y < 0, \\ |y|^{-1}e^{-\eta_+|y|} & \text{if} \quad y > 0, \end{cases}$$

the so-called CGMY model [6, 8, 9, 21] (also referred to as tempered or truncated tempered stable processes or "truncated Lévy flights" in physics), with

$$k(y) = C \begin{cases} \dfrac{e^{-\eta_-|y|}}{|y|^{1+\alpha}} & \text{if} \quad y < 0, \\ \dfrac{e^{-\eta_+|y|}}{|y|^{1+\alpha}} & \text{if} \quad y > 0, \end{cases} \tag{3.15}$$

$0 < \alpha < 2$, the normal inverse Gaussian process (NIG) [2], and hyperbolic and generalized hyperbolic processes [12, 13, 14]. These are all examples of Lévy processes of infinite activity.

The classical Merton model [28] is based on a drifted Brownian motion with finitely many jumps, i.e., $X_t = \sigma W_t + \sum_{i=1}^{N_t} Y_i$. Here, $\{Y_i\}$ are independent, identically distributed random variables with distribution function $f(z)$, and $\{N_t\}$ is a Poisson process with intensity λ. The Lévy measure is given by $\nu(dz) = k(z)dz$ with $k(z) = \lambda f(z)$. With $f_M(z) = (\sqrt{2\pi}\sigma_M)^{-1} \exp(-(z-\mu_M)^2/(2\sigma_M^2))$, i.e., f_M is the normal distribution with mean μ_M and standard deviation σ_M, Merton's model is a finite-intensity Lévy process which satisfies (A1)–(A3) with $\alpha = -\infty$.

In addition, we mention here the model proposed by Kou [22] which assumes $f_{Kou}(z) = p_+ M \exp(-Mz)\chi_{\mathbb{R}_+}(z) + p_- G \exp(Gz)\chi_{\mathbb{R}_-}(z)$, $p_+ + p_- = 1$. Here, X_t is a finite activity Lévy process with k satisfying (A1)–(A3) for $\alpha = -1$.

3.3. Variational setting. Here, without loss of generality, we only elaborate on the problem of pricing European vanillas. More precisely, we consider European call, $f(S) = \max(S-K, 0)$, and put, $f(S) = \max(K-S, 0)$, options, with $K > 0$ being the so-called strike price. More general payoffs in (3.3) (with polynomial growth at infinity, for example) could be treated similarly.

Since $(e^x - K)_+, (K - e^x)_+ \notin L^2(\mathbb{R})$, we consider weighted Sobolev spaces accounting for the growth of solutions at infinity in order to obtain an appropriate variational formulation of (3.6), (3.7). To this end, denote, for $w \in L^1_{\mathrm{loc}}(\mathbb{R})$, $w' \in L^\infty(\mathbb{R})$, by $H^1_{\pm w}(\mathbb{R})$ the weighted Sobolev space

$$H^1_{\pm w}(\mathbb{R}) := \{v \in L^1_{\mathrm{loc}}(\mathbb{R}) \mid e^{\pm w} v, e^{\pm w} v' \in L^2(\mathbb{R})\}. \tag{3.16}$$

Similarly, let $L^2_{\pm w}(\mathbb{R}) := \{v \in L^1_{\mathrm{loc}}(\mathbb{R}) \mid e^{\pm w} v \in L^2(\mathbb{R})\}$. To the operator $\mathcal{L}^X$ the following bilinear form is associated:

$$a^{\pm w}(\varphi, \psi) := \int_{\mathbb{R}} \mathcal{L}^X[\varphi](x)\psi(x)e^{\pm 2w(x)}\, dx, \qquad \varphi, \psi \in C_0^\infty(\mathbb{R}).$$

Set $g(x) = (e^x - K)_+$ (the case of a put can be treated analogously), and let

$$w(x) = \begin{cases} \omega_- |x| & \text{if } x < 0, \\ \omega_+ |x| & \text{if } x > 0. \end{cases} \tag{3.17}$$

Furthermore, assume that $0 \le \omega_- < \eta_-$, $1 < \omega_+ < \eta_+$ and that $\sigma > 0$. Then, by [27, Theorem 3.4], the problem of finding $u \in L^2(J; H^1_{-w}(\mathbb{R})) \cap H^1(J; (H^1_{-w}(\mathbb{R}))^*)$ such that

$$\frac{\partial}{\partial t}(u(t), v)_{L^2_{-w}(\mathbb{R})} + a^{-w}(u(t), v) = 0 \quad \forall\, v \in H^1_{-w}(\mathbb{R}),$$
$$u(0) = g,$$

admits a unique solution.

3.4. Localization. The numerical solution of (3.6), (3.7) requires the truncation of $\mathbb{R}$ to a bounded computational domain $\Omega_R = (-R, R)$ (note that barrier-like contracts directly lead to the PIDE on the finite interval). In the classical Black–Scholes model [4], i.e., in the absence of $\mathcal{L}^{\mathrm{jump}}$, the localization can be effected by a simple restriction to Ω_R and the localization error may be estimated by local considerations. However, in the presence of the nonlocal operator $\mathcal{L}^{\mathrm{jump}}$ corresponding to

the jump part of the price process $(X_t)_t$, such local arguments do not apply, since the computation of the solution requires information on the payoff on the whole real line $\mathbb{R}$.

As the transformation $u(t,\cdot) \to e^{-rt}u(t,\cdot + rt)$ reduces (3.6), (3.7) to the case where $r = 0$, we henceforth assume, without loss of generality, that $r = 0$.

Based on the observation that the excess to payoff value $U = u - g$ decays exponentially as $|x| \to \infty$, we restrict the PIDE for the excess to payoff value U to Ω_R. Specifically, the excess to payoff function U solves

$$\frac{\partial U}{\partial t}(t,x) + \mathcal{L}^X[U](t,x) = -\mathcal{L}^X[g] \quad \text{in } J \times \mathbb{R}, \tag{3.18}$$

$$U(0,x) = 0 \qquad \text{in } \mathbb{R}. \tag{3.19}$$

Now, rather than solving (3.18), (3.19), we solve the truncated problem in $(0,T)\times\Omega_R$

$$\frac{\partial U_R}{\partial t} + \mathcal{L}_R[U_R] = -\mathcal{L}^X[g]|_{\Omega_R} \quad \text{in} \quad J \times \Omega_R, \tag{3.20}$$

$$U_R(t,x) = 0 \qquad \text{in} \quad J \times \mathbb{R}\backslash\Omega_R, \tag{3.21}$$

$$U_R(0,x) = 0 \qquad \text{in} \quad \Omega_R, \tag{3.22}$$

with $\mathcal{L}_R$ being the restriction of $\mathcal{L}^X$ to Ω_R. Note that, in contrast to the standard Black–Scholes case ($\mathcal{L}^{\text{jump}} = 0$), the nonlocal operator $\mathcal{L}^{\text{jump}}$ requires the payoff function g to be specified also outside the domain Ω_R.

By [27, Proposition 3.6], the right-hand side $-\mathcal{L}^X[g]$ is given by $-\mathcal{L}^X[g] = \sigma^2/2K\delta_{\log(K)} - \mathcal{L}^{\text{jump}}[g]$ with $\mathcal{L}^{\text{jump}}[g] \in C^\infty(\mathbb{R}\backslash\log(K)) \cap L^1_{\text{loc}}(\mathbb{R})$ decaying exponentially at infinity: $-\mathcal{L}^{\text{jump}}[g] \le Ce^{-\eta_\mp|x|}$ as $\pm x \to \infty$. Choosing the exponent w as in (3.17) with $0 < \omega_+ < \eta_-$ and $0 < \omega_- < \eta_+$ implies that $-\mathcal{L}^X[g] \in (H^1_w(\mathbb{R}))^*$, and by [27, Theorem 3.4], the variational problem of finding $U \in L^2(J; H^1_w(\mathbb{R})) \cap H^1(J; (H^1_w(\mathbb{R}))^*)$ such that

$$\frac{d}{dt}(U(t),v)_{L^2_w(\mathbb{R})} + a^w(U(t),v) = -\langle \mathcal{L}^X[g], v\rangle_{(H^1_w(\mathbb{R}))^*\times H^1_w(\mathbb{R})} \quad \forall\, v \in H^1_w(\mathbb{R}),$$

with

$$U(0) = 0,$$

is uniquely solvable. Consequently, the excess to payoff function U decays exponentially at infinity, and by [27, Theorem 4.1], the error $U_R - U$ caused by the localization decays exponentially as $R \to \infty$.

3.5. Parabolic setting. We now verify that the localized problem (3.20)–(3.22) can be cast in the variational setting (2.3), (2.4). Let $\mathcal{A} := \mathcal{L}_R$ and $a(\cdot,\cdot)$ denote the bilinear form induced by $\mathcal{A}$:

$$a(\varphi,\psi) := \int_{\Omega_R} \mathcal{A}[\varphi](x)\psi(x)dx \qquad \forall\, \varphi,\psi \in C_0^\infty(\Omega_R).$$

If $\sigma > 0$, the price process contains a diffusion component and the order of the operator $\mathcal{L}^X$ is 2. Then $a(\cdot,\cdot)$ can be extended continuously to $V \times V$ with $V := H^1_0(\Omega_R)$ and the following Gårding inequality holds: there exist $c_1, c_2 > 0$ such that

$$a(v,v) \ge c_1\|v\|^2_{H^1(\Omega_R)} - c_2\|v\|^2_{L^2(\Omega_R)} \qquad \forall\, v \in V.$$

In the pure jump case, $\sigma = 0$, the order of $\mathcal{L}^X$ is in general $\max\{1, \alpha\}$ due to the presence of the drift term in $\mathcal{L}^X$.

PROPOSITION 3.2. [illegible] X_t [illegible] $\sigma = 0$ [illegible] k [illegible] (A1)–(A4) [illegible] $0 < \alpha < 2$ [illegible] $c_1 = c_1(R) > 0$ [illegible] $c_2 = c_2(R) > 0$ [illegible]

$$\forall\, u \in \tilde{H}^{\alpha/2}(\Omega_R), \quad a(u,u) \geq c_1 \|u\|^2_{\tilde{H}^{\alpha/2}(\Omega_R)} - c_2 \|u\|^2_{L^2(\Omega_R)}; \tag{3.23}$$

[illegible] $\tilde{H}^{\alpha/2}(\Omega_R)$ [illegible] $a(\cdot,\cdot)$ [illegible] (2.4). [illegible]. By density, we may assume $u \in C_0^\infty(\Omega_R)$. Then $(u', u) = 0$. By $\tilde{u}$, we denote the extension of u by zero to $\mathbb{R}$, and by $a^{\mathrm{sym}}(\varphi, \psi) = (a(\varphi,\psi) + a(\psi,\varphi))/2$ the bilinear form corresponding to the symmetric part $\mathcal{L}^{\mathrm{sym}} := \frac{1}{2}(\mathcal{L}^X + (\mathcal{L}^X)^*) = \frac{1}{2}(\mathcal{L}^{\mathrm{jump}} + (\mathcal{L}^{\mathrm{jump}})^*)$ of $\mathcal{L}^X = \mathcal{L}^{\mathrm{jump}}$. Then there holds

$$a(u,u) = a^{\mathrm{sym}}(u,u) = \langle \mathcal{L}^{\mathrm{sym}} \tilde{u}, \tilde{u} \rangle = \frac{1}{2} \int_{\mathbb{R}} \int_{\mathbb{R}} (k(x-y) + k(y-x))(\tilde{u}(x) - \tilde{u}(y))^2 dx dy.$$

Due to (A4), we obtain

$$a(u,u) \geq C_- \int_{\mathbb{R}} \int_{\mathbb{R}} \frac{(\tilde{u}(x) - \tilde{u}(y))^2}{|x-y|^{1+\alpha}} dy dx = C_- \left(\|u\|^2_{\tilde{H}^{\alpha/2}(\Omega_R)} - \|u\|^2_{L^2(\Omega_R)} \right). \qquad \square$$

[illegible] 3.3. Proposition 3.2 states a Gårding inequality (2.4) in the pure jump case $\sigma = 0$ for all jump intensities $\alpha \in (0,2)$. However, if the term (u', v) is present, the continuity (2.3) on $V = \tilde{H}^{\alpha/2}$ holds only if $1 \leq \alpha \leq 2$ [27]. Nevertheless, our framework is applicable to (3.15), for all $0 < \alpha < 2$, since the term (u', v) can be removed.

In order to prove (2.3) in the case $\sigma = 0$ and $0 < \alpha < 1$, it is important to remove the "drift" term so that the transformed equation satisfies the Gårding inequality (2.4) in $\tilde{H}^{\alpha/2}(\Omega_R)$ as well as the continuity estimate (2.3). Let $\mathcal{A}^{\mathrm{jump}} := \mathcal{L}^{\mathrm{jump}}|_{\Omega_R}$ be the restriction of the integrodifferential operator $\mathcal{L}^{\mathrm{jump}}$ to Ω_R. By reduction to the "transformed" variables, we assume again that $r = 0$, together with $0 < \alpha < 1$. Then

$$\mathcal{A}[u] = -\frac{\sigma^2}{2} u'' + \frac{\sigma^2}{2} u' + \mathcal{A}^{\mathrm{jump}}[u]. \tag{3.24}$$

Next, the operator $\mathcal{A}^{\mathrm{jump}}$ can be written as a sum of a first-order term and an integrodifferential operator $\mathcal{C}$ which is continuous and coercive on $\tilde{H}^{\alpha/2}(\Omega_R)$:

$$\langle \mathcal{A}^{\mathrm{jump}} \varphi, \psi \rangle = c_1 \langle \varphi', \psi \rangle + \langle \mathcal{C}[\varphi], \psi \rangle \quad \forall\, \varphi, \psi \in C_0^\infty(\Omega_R), \tag{3.25}$$

with

$$c_1 := \int_{\mathbb{R}} (e^y - 1) k(y) dy.$$

The operator $\mathcal{C}$ is defined by

$$\langle \mathcal{C} u, v \rangle := - \int_{\mathbb{R}} \int_{\mathbb{R}} k(y-x)(\tilde{u}(y) - \tilde{u}(x)) \tilde{v}(x) dy dx \quad \forall\, u, v \in \tilde{H}^{\alpha/2}(\Omega_R),$$

where, by $\tilde{v}$, we denote the extension by zero of v to $\mathbb{R}$. Then, there exist constants $C, C_1, C_2 > 0$ such that

$$\langle \mathcal{C} u, v \rangle \leq C \|u\|_{\tilde{H}^{\alpha/2}(\Omega_R)} \|v\|_{\tilde{H}^{\alpha/2}(\Omega_R)} \qquad \forall\, u, v \in \tilde{H}^{\alpha/2}(\Omega_R),$$

$$\langle \mathcal{C} u, u \rangle \geq C_1 \|u\|^2_{\tilde{H}^{\alpha/2}(\Omega_R)} - C_2 \|u\|^2_{L^2(\Omega_R)} \qquad \forall\, u \in \tilde{H}^{\alpha/2}(\Omega_R).$$

We have seen that the presence of the drift term $c_1\varphi'$ obstructs the continuity of $\langle \mathcal{A}^{\text{jump}}[\varphi], \psi\rangle$. We remove this term by the transformation

$$U(t,x) = V(t, x - (\sigma^2/2 + c_1)t), \tag{3.26}$$

which yields the parabolic problem

$$\frac{\partial V}{\partial t}(t,x) - \frac{\sigma^2}{2}\frac{\partial^2 V}{\partial x^2} + \mathcal{C}[V](t,x), = \left(-\mathcal{L}^X[g]|_{\Omega_R}\right)(x + (\sigma^2/2 + c_1)t),$$
$$V(0,x) = 0.$$

3.4. We note that the numerical scheme proposed in this paper exploits the parabolic nature of (3.6) and, furthermore, that a dominant first-order term may be a source of instabilities even in the case $\sigma \neq 0$. Therefore, the removal of the "drift" is also of interest from the point of view of numerical approximation.

Due to our previous considerations, $a(\cdot,\cdot) : V \times V \to \mathbb{R}$ satisfies (2.3), (2.4) (cf. Proposition 3.2) in the triple $V \overset{d}{\hookrightarrow} L^2(\Omega_R) \overset{d}{\hookrightarrow} V^*$ with

$$V := \tilde{H}^{\rho/2}(\Omega_R),$$

where the order ρ is given by

$$\rho = \begin{cases} 2 & \text{if } \sigma > 0, \\ \alpha & \text{if } \sigma = 0. \end{cases} \tag{3.27}$$

For any $v \in H^{\rho/2}$, let $P_a^V v$ denote the V-projection of v with respect to $a(\cdot,\cdot)$, i.e., $a(v - P_a^V v, w) = 0$ for all $w \in V$. By the continuity of $a(\cdot,\cdot)$, it holds that

$$\|P_a^V v\|_V \le C\|v\|_{H^{\rho/2}(\Omega_R)} \qquad \forall\, v \in H^{\rho/2}(\Omega_R). \tag{3.28}$$

The variational formulation of (3.20)–(3.22) reads: find $U_R \in L^2(J,V) \cap H^1(J,V^*)$ such that

$$\left(\frac{\partial U_R}{\partial t}(t,\cdot), v\right)_{L^2(\Omega_R)} + a(U_R(t,\cdot), v) = -\langle \mathcal{L}^X[g], v\rangle_{V^*\times V} \qquad \forall\, v \in V, \tag{3.29}$$

$$U_R(0,\cdot) = 0. \tag{3.30}$$

Obviously, (3.29), (3.30) is equivalent to solving

$$\left(\frac{\partial u_R}{\partial t}(t,\cdot), v\right)_{L^2(\Omega_R)} + a(u_R(t,\cdot), v) = 0 \qquad \forall\, v \in V, \tag{3.31}$$

$$u_R(0,\cdot) = u_0 := P_a^V g, \tag{3.32}$$

where, evidently, $u_R = P_a^V g + U_R$.

Hence, dropping the subscript R to simplify notations and referring to (3.31), (3.32), we are now in the setting of section 2 with $u_0 \in V$.

3.6. Non–translation-invariant operators. The numerical method presented in this paper (see sections 4, 5, and 6) applies also to the more general case where

the log-price process X_t is a Markov process with increments $X_t - X_s$ that are no longer independent of X_s for $t > s$. In particular, we allow for volatilities $\sigma(x)$ and singular jump measures that depend on x. Instead of the translation-invariant operator $\mathcal{L}^X$ from (3.8)–(3.9) we now consider a more general operator $\mathcal{L}^X$

$$\mathcal{L}^X[\varphi](x) := -\frac{\sigma(x)^2}{2}\varphi''(x) + \left(\frac{\sigma(x)^2}{2} - r\right)\varphi'(x) + r\varphi + \mathcal{L}^{\text{jump}}[\varphi](x), \tag{3.33}$$

with the integral operator $\mathcal{L}^{\text{jump}}$ given by

$$\mathcal{L}^{\text{jump}}[\phi](x) := -\int_{\mathbb{R}} \{\phi(y) - \phi(x) - \left(e^{y-x} - 1\right)\phi'(x)\}k(x, y-x)\, dy. \tag{3.34}$$

We assume that

$$0 < \sigma_0 \le \sigma(x) \le \sigma_1 \quad \forall\, x \in \Omega_R,$$

and that $k(x, z)$ satisfies the assumptions (A1)–(A2) uniformly with respect to $x \in \Omega_R$. Instead of (A3), we require the Calderón–Zygmund estimates: for all $\beta, \gamma \in \mathbb{N}_0$, there holds

$$|\partial_x^\gamma \partial_z^\beta k(x, z)| \le C(\gamma, \beta)|z|^{-(1+\alpha+\beta+\gamma)}, \qquad z \neq 0. \tag{3.35}$$

In the case $\sigma = 0$, we require (A4) uniformly with respect to $x \in \Omega_R$. If $\sigma > 0$ or $\alpha \ge 1$, the corresponding bilinear form a_R satisfies (2.3), (2.4). If $\sigma = 0$ and $\alpha < 1$, we need to assume that we can transform the problem such that a_R satisfies (2.3).

4. Numerical solution. The aim of this part is to discretize the parabolic equation (2.1), (2.2) in time and space using a dG scheme on $J = (0, T)$, and a wavelet finite element method on $\Omega = (a, b)$. Again, $\mathcal{A} : V \to V^*$ in (2.1) is a possibly nonlocal operator.

In the ensuing analysis, we shall need to consider functions in V with additional regularity. For this reason, we introduce the spaces $\mathcal{H}^s(\Omega)$, $s \ge \rho/2$, given by

$$\mathcal{H}^s(\Omega) = \begin{cases} V = \widetilde{H}^{\rho/2}(\Omega) & \text{for } s = \rho/2, \\ V \cap H^s(\Omega) & \text{for } s > \rho/2. \end{cases}$$

From now on, in order to cope with the assumptions in section 3, the initial condition is supposed to fulfill $u_0 \in V$.

4.1. Spatial semidiscretization. Let $\mathcal{T}^0$ be a fixed coarse partition of Ω. Furthermore, define the mesh $\mathcal{T}^l$, for $l > 0$, by bisection of each interval in $\mathcal{T}^{l-1}$. We assume that the mesh $\mathcal{T}_h$ is obtained in this way as $\mathcal{T}^L$, for some $L > 0$, such that $h = C2^{-L}$.

For $0 \le \rho < 1$, the space V_h is defined as the space of piecewise, possibly discontinuous polynomials of degree $p \ge 0$ on $\mathcal{T}_h$. For $1 \le \rho \le 2$, V_h is the space of all continuous piecewise polynomials of degree $p \ge 1$ on the triangulation $\mathcal{T}_h$ which vanish on the boundary $\partial\Omega$.

In the same way, we define the spaces V^l corresponding to the triangulation $\mathcal{T}^l$, so that we have $V^0 \subset V^1 \subset \cdots \subset V^L = V_h$. Let $N^l = \dim V^l$ and $N = \dim V_h = N^L = C2^L$.

The semidiscrete problem corresponding to (2.1), (2.2) reads as follows: Given $u_0 \in V$, find $u_h \in H^1(J, V_h)$ such that

$$\left(\frac{d}{dt}u_h, v_h\right) + a(u_h, v_h) = 0 \qquad \forall\, v_h \in V_h \tag{4.1}$$

and

$$u_h(0) = P_{L^2}^{V_h} u_0. \tag{4.2}$$

Here, $P_{L^2}^{V_h}$ is the L^2-projection onto V_h.

4.1.1. Wavelet basis. In a fully discrete approximation (in time and space) with the parabolic problem (2.1), (2.2), systems of linear equations have to be solved in each time step. Since $\mathcal{A}$ is nonlocal, the matrices corresponding to these linear systems are fully populated. Therefore, matrix compression techniques, transforming the dense matrices into sparse ones, have to be employed. The compressed systems may then be manipulated in linear complexity.

Using an appropriate basis for the finite-dimensional space V_h, we will be able to represent the bilinear form $a(\cdot,\cdot)$ as a matrix whose entries are mostly negligible, thereby allowing for matrix compression techniques.

More precisely, we consider so-called biorthogonal wavelets

$$\{\psi_j^l\}_{j,l}, \qquad l = 0, 1, \ldots, \quad j = 1, 2, \ldots, M^l, \tag{4.3}$$

which allow for optimal preconditioning. They satisfy the following properties (see, e.g., [7]):

1. $V^L = \operatorname{span}\{\psi_j^l \mid 0 \le l \le L, 1 \le j \le M^l\}$.
2. The function ψ_j^l has support $S_j^l = \operatorname{supp}\psi_j^l$ of diameter bounded by $C2^{-l}$.
3. Wavelets ψ_j^l with $\overline{S_j^l} \cap \partial\Omega = \emptyset$ have vanishing moments up to order p, i.e., $(\psi_j^l, q) = 0$ for all polynomials q of order p or less.
4. The functions ψ_j^l for $l \ge l_0$ are obtained by scaling and translation of the functions $\psi_j^{l_0}$.
5. For all $v_h = \sum_{l=0}^{L}\sum_{j=1}^{M^l} v_j^l \psi_j^l \in V_h$, there holds the norm equivalence

$$|||v_h|||_s^2 := \sum_{l=0}^{L}\sum_{j=1}^{M^l} |v_j^l|^2 2^{2ls} \sim \|v\|_{\widetilde{H}^s(\Omega)}^2 \tag{4.4}$$

for $0 \le s \le 1$.

4.1. Our biorthogonal wavelets in the case $p = 1$ are continuous, piecewise linear spline wavelets vanishing outside $\Omega = (0,1)$ (for more general domains $\Omega = (a,b)$, they are obtained by simple scalings). The interior wavelets have two vanishing moments and are obtained from the mother wavelet $\psi(x)$ which takes the values $(0, -1/2, 1, -1/2, 0)$ at $(0, 1/4, 1/2, 3/4, 1)$ by scaling and translations: $\psi_j^l(x) := \psi(2^{l-1}x - (2j-1)2^{-2})$ for $1 \le j \le 2^l - 2$ and $l \ge 2$. The boundary wavelets are constructed from the continuous, piecewise linear functions ψ_*, with values $(0, 1, -1/2, 0)$ at $(0, 1/4, 1/2, 3/4)$, and ψ^*, taking values $(0, -1/2, 1, 0)$ at $(1/4, 1/2, 3/4, 1)$: $\psi_0^l(x) = \psi_*(2^{l-1}x)$ and $\psi_{2^l-1}^l(x) = \psi^*(2^{l-1}x - 2^{l-1} + 1)$. Higher-order spline wavelets with analogous properties have been given in [11].

4.1.2. Matrix compression. The bilinear form a on $V_h \times V_h$ corresponds to a matrix $\mathbf{A}$ with entries $A_{(l,j),(l',j')} = a(\psi_{j'}^{l'}, \psi_j^l)$ which, due to the Caldéron–Zygmund estimates (2.7), decay with increasing distance of their supports. Hence, we can define a compressed matrix $\tilde{\mathbf{A}}$ and a corresponding bilinear form $\tilde{a}$ by replacing some of the small entries in $\mathbf{A}$ with zero:

$$\tilde{A}_{(j,l),(j',l')} := \begin{cases} A_{(j,l),(j',l')} & \text{if } \operatorname{dist}(S_j^l, S_{j'}^{l'}) \le \delta_{l,l'} \text{ or } S_j^l \cap \partial\Omega \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

Here, the truncation parameters $\delta_{l,l'}$ are given by

$$\delta_{l,l'} := c \max\{2^{-L+\hat{\alpha}(2L-l-l')}, 2^{-l}, 2^{-l'}\}, \tag{4.6}$$

with some parameters $c > 0$ and $1 \ge \hat{\alpha} > 0$, and $S_j^l = \operatorname{supp} \psi_j^l$.

By continuity and coercivity of the bilinear form a, we have the norm equivalence

$$\|u\|_a := (a(u,u))^{1/2} \sim \|u\|_V.$$

In [37, Proposition 3.2], the following was proved.

PROPOSITION 4.2. [illegible] c [illegible] (4.6) [illegible] $0 < \widetilde{\alpha} < \infty$ [illegible] L [illegible]

$$|\widetilde{a}(u_h, v_h)| \le \widetilde{\alpha} \|u_h\|_a \|v_h\|_a, \tag{4.7}$$

[illegible] $0 < \widetilde{\beta} \le \widetilde{\alpha}$ [illegible]

$$\widetilde{a}(u_h, u_h) \ge \widetilde{\beta} \|u_h\|_a^2 \tag{4.8}$$

[illegible] $u_h, v_h \in V_h$

Moreover, we have [35, 37] the following proposition.

PROPOSITION 4.3. [illegible] $0 < \delta < 1$ [illegible] L [illegible] $L > 0$ [illegible]

$$|a(u_h, v_h) - \widetilde{a}(u_h, v_h)| \le \delta \, \|u_h\|_a \, \|v_h\|_a \qquad \forall u_h, v_h \in V_h. \tag{4.9}$$

[illegible]

$$\hat{\alpha} \ge \frac{2p+2}{2p+2+\rho}, \tag{4.10}$$

[illegible] $u_h, v_h \in V_h$ [illegible]

$$|a(u_h, v_h) - \widetilde{a}(u_h, v_h)| \le C h^{p+1-\rho/2} |\log h|^{\nu} |||u_h|||_{p+1} \, \|v_h\|_V, \tag{4.11}$$

[illegible] $\nu = 1$ [illegible] (4.10), [illegible] $\nu = 0$ [illegible]

The matrix compression (4.5) reduces the number of nonzero elements from N^2 in $\mathbf{A}$ to N times a logarithmic term in $\tilde{\mathbf{A}}$; see [32].

PROPOSITION 4.4. [illegible] $\widehat{\alpha} < 1$ [illegible] $\tilde{\mathbf{A}}$ [illegible] $\mathcal{O}(N \log N)$ [illegible] $\widehat{\alpha} = 1$ [illegible] $\tilde{\mathbf{A}}$ [illegible] $\mathcal{O}(N(\log N)^2)$

4.1.3. Perturbed spatial semidiscretization. In accordance with the matrix compression from the previous section, we consider, instead of (4.1), (4.2), the perturbed spatial semidiscretization of (2.1), (2.2): find $\widetilde{u}_h \in V_h$ such that

$$\left(\frac{d}{dt}\widetilde{u}_h, v_h\right) + \widetilde{a}(\widetilde{u}_h, v_h) = 0 \qquad \forall\, v_h \in V_h \tag{4.12}$$

and

$$\widetilde{u}_h(0) = P_{L^2}^{V_h} u_0. \tag{4.13}$$

4.5. We assume that the error between the exact solution u of (2.1), (2.2) and the approximate solution $\widetilde{u}_h$ obtained by (4.12), (4.13) at $t = T$ is of a certain algebraic order; i.e., there exist constants C, ω (depending on p) such that

$$\|u(T) - \widetilde{u}_h(T)\| \le Ch^{\omega}. \tag{4.14}$$

Here, $h = C2^{-L}$ is the mesh width.

4.6. If the operator $\mathcal{A}$ from section 2 is of the form $\mathcal{A} = \mathcal{A}_0 + \mathcal{B}$, where $\mathcal{A}_0 = C_0(-\Delta)^{\frac{p}{2}}$, $C_0 > 0$, is a Riesz potential, and $\mathcal{B} : V \to V^*$ is a compact perturbation, then the bound (4.14) holds true with $\omega = p + 1$; this will be proved in the forthcoming paper [26]. The proof of this result is based on splitting the error $u - \widetilde{u}_h$ of the semidiscrete problem (4.12)–(4.13) into two parts that can be estimated separately. More precisely, we write $u - \widetilde{u}_h = T_1 + T_2$ with $T_1 = u - u_h$ and $T_2 = u_h - \widetilde{u}_h$, where u, u_h, and $\widetilde{u}_h$ are the solutions of (2.1)–(2.2), (4.1)–(4.2), and (4.12)–(4.13), respectively. Then, using techniques similar to those in [16, 17], it can be shown that $T_1 \le Ch^{p+1}$. In addition, applying suitable wavelet compression bounds (such as in Proposition 4.3, for example), it is proved that the term T_2 is bounded analogously.

4.2. Discontinuous Galerkin time discretization. We discretize (4.12), (4.13) in time using a dG method following [34]. For $0 < T < \infty$ and $M \in \mathbb{N}$, let $\mathcal{M} = \{I_m\}_{m=1}^M$ be a partition of $J = (0, T)$ into M subintervals $I_m = (t_{m-1}, t_m)$, $m = 1, 2, \ldots, M$ with $0 = t_0 < t_1 < t_2 < \cdots < t_M = T$. Moreover, denote by $k_m = t_m - t_{m-1}$ the length of I_m.

For $u \in H^1(\mathcal{M}, V_h) := \{v \in L^2(J, V_h) : v|_{I_m} \in H^1(I_m, V_h), m = 1, 2, \ldots, M\}$, define the one-sided limits

$$u_m^+ := \lim_{s \to 0^+} u(t_m + s), \qquad m = 0, 1, \ldots, M-1,$$
$$u_m^- := \lim_{s \to 0^+} u(t_m - s), \qquad m = 1, 2, \ldots, M,$$

and the jumps

$$[u]_m := u_m^+ - u_m^-, \qquad m = 1, 2, \ldots, M-1.$$

In addition, to each time interval I_m, a polynomial degree (approximation order) $r_m \ge 0$ is associated. These numbers are stored in the degree vector $\underline{r} = \{r_m\}_{m=1}^M$. Then, the following space being used for the dG method is introduced:

$$\mathcal{S}^{\underline{r}}(\mathcal{M}, V_h) := \{u \in L^2(J, V_h) : u|_{I_m} \in \mathcal{P}_{r_m}(I_m, V_h), m = 1, 2, \ldots, M\},$$

where $\mathcal{P}_{r_m}(I_m)$ denotes the space of polynomials of degree at most r_m on I_m.

With these definitions, the fully discrete dG scheme for the solution of problem (2.1), (2.2), respectively (4.12), (4.13) reads as follows: find $\widetilde{U}_h^{dG} \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$ such that for all $W \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$, there holds

$$\widetilde{B}_{dG}(\widetilde{U}_h^{dG}, W) = F_{dG}(W), \tag{4.15}$$

where

$$\widetilde{B}_{dG}(U, W) := \sum_{m=1}^{M} \int_{I_m} \big((U', W) + \widetilde{a}(U, W)\big)\, dt + \sum_{m=1}^{M-1} ([U]_m, W_m^+) + (U_0^+, W_0^+) \tag{4.16}$$

and

$$F_{dG}(W) := (P_{L^2}^{V_h} u_0, W_0^+). \tag{4.17}$$

PROPOSITION 4.7 (see [33]).

a. [illegible] $\widetilde{U}_h^{dG} \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$ [illegible] (4.15) [illegible]

b. [illegible] $\widetilde{u}_h$ [illegible] (4.12) (4.13) [illegible] (4.15). [illegible]

[illegible]

$$\widetilde{B}_{dG}(\widetilde{u}_h - \widetilde{U}_h^{dG}, W) = 0 \tag{4.18}$$

[illegible] $W \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$

[illegible] 4.8. The dG method (4.15) can be interpreted as a time-stepping scheme of variable step size k_m and orders r_m. Indeed, assuming that $\widetilde{U}_h^{dG}$ is known on the time intervals $I_m = (t_{m-1}, t_m)$, $m = 1, 2, \ldots, n-1$, we may find $\widetilde{U}_h^{dG} \in \mathcal{P}_{r_n}(I_n, V_h)$, $1 \le n \le M$, by solving the variational problem

$$\int_{I_n} \big((\partial_t \widetilde{U}_h^{dG}, W) + \widetilde{a}(\widetilde{U}_h^{dG}, W)\big)\, dt + (\widetilde{U}_{n-1}^{dG+}, W_{n-1}^+) = (\widetilde{U}_{n-1}^{dG-}, W_{n-1}^+) \tag{4.19}$$

for all $W \in \mathcal{P}_{r_n}(I_n, V_h)$. Here, we set $\widetilde{U}_0^{dG-} = P_{L^2}^{V_h} u_0$.

4.2.1. Stability. For $v_h \in V_h$ and $U \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$, we introduce the norms

$$\|v_h\|_{\widetilde{a}} = \big(\widetilde{a}(v_h, v_h)\big)^{1/2}$$

and

$$\|U\|_{dG}^2 = \sum_{m=1}^{M} \int_{I_m} \|U\|_{\widetilde{a}}^2 dt + \frac{1}{2}\Big(\|U_0^+\|^2 + \sum_{m=1}^{M-1} \|[U]_m\|^2 + \|U_M^-\|^2\Big).$$

The dG time-stepping is stable.

PROPOSITION 4.9. [illegible] $\widetilde{U}_h^{dG} \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$ [illegible] (4.15)

[illegible]

$$\|\widetilde{U}_h^{dG}\|_{dG} \le C \|u_0\|,$$

[illegible] $C > 0$ [illegible] T [illegible] L

[illegible]. Let $U \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h)$ be arbitrary. Then, as in [33, Lemma 2.7], there holds

$$\widetilde{B}_{dG}(U, U) = \|U\|_{dG}^2 \qquad \forall U \in \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h). \tag{4.20}$$

Moreover, we have

$$|F_{dG}(U)| \le |(P_{L^2}^{V_h} u_0, U_0^+)| \le \|P_{L^2}^{V_h} u_0\| \|U_0^+\|.$$

Hence, using the stability of $P_{L^2}^{V_h}$ results in

$$|F_{dG}(U)| \le \|u_0\| \|U_0^+\| \le C\|u_0\| \|U\|_{dG}. \tag{4.21}$$

Now, putting $U := \widetilde{U}_h^{dG}$, recalling the definition of the dG method (4.15), and employing (4.20) and (4.21), we obtain

$$\|\widetilde{U}_h^{dG}\|_{dG}^2 \le |\widetilde{B}_{dG}(\widetilde{U}_h^{dG}, \widetilde{U}_h^{dG})| = |F_{dG}(\widetilde{U}_h^{dG})| \le C\|u_0\| \|\widetilde{U}_h^{dG}\|_{dG}.$$

Dividing both sides by $\|\widetilde{U}_h^{dG}\|_{dG}$ completes the proof. □

4.3. Convergence of the fully discrete scheme. The solution operator of the parabolic problem (2.1), (2.2) generates an analytic semigroup (see, e.g., [33]), and therefore, the solution $u(t)$ becomes analytic with respect to t for $t > 0$. However, due to the nonsmoothness of the initial data, singularities may arise at $t = 0$. In this section, we show how, by the use of so-called geometric time partitions and linearly increasing polynomial degrees in the time discretization, the low regularity of the solution at $t = 0$ can be resolved.

DEFINITION 4.10. [illegible] $\mathcal{M}_{M,\gamma} = \{I_m\}_{m=1}^M$ [illegible] $J = (0,T)$ $T > 0$ [illegible] M [illegible] $I_m = (t_{m-1}, t_m)$ $m = 1, 2, \dots, M$ [illegible] $\gamma \in (0,1)$ [illegible]

$$t_0 = 0, \qquad t_m = T\gamma^{M-m}, \quad 1 \le m \le M.$$

DEFINITION 4.11. [illegible] $\underline{r}$ [illegible] $\mu > 0$ [illegible] $\mathcal{M}_{M,\gamma}$ [illegible] $r_1 = 0$ [illegible] $r_m = \lfloor \mu m \rfloor$ $m = 2, \dots, M$ [illegible] $\lfloor \mu m \rfloor = \max\{q \in \mathbb{N}_0 : q \le \mu m\}$

We have the following a priori error estimate on the fully discrete dG method. Its proof is worked out in the appendix.

THEOREM 4.12. [illegible] $\rho \in [0,2]$ [illegible] u [illegible] (2.1), (2.2) [illegible] $J \times \Omega = (0,T) \times \Omega$ [illegible] $u_0 \in V = \dot{H}^{\rho/2}(\Omega)$ [illegible] (4.14) [illegible] $\mathcal{M}_{M,\gamma}$ [illegible] 4.10 [illegible] $0 < \gamma < 1$ [illegible] $M = c|\log h|$ [illegible] $\underline{r}$ [illegible] $\mu \ge \mu_0$ [illegible] 4.11 [illegible] $c, \mu_0 > 0$ [illegible] (4.15) [illegible] $\mathcal{S}^{\underline{r}}(\mathcal{M}_{M,\gamma}, V_h)$ [illegible]

$$\|u(T) - \widetilde{U}_h^{dG}(T)\| \le Ch^{\omega}. \tag{4.22}$$

h [illegible] $\mathcal{T}_h$ ω [illegible] (4.14), [illegible] $C > 0$ [illegible] h

5. Solution algorithm. We will now study the linear systems resulting from the dG method (4.15). We show that they may be solved iteratively, without causing a loss in the rates of convergence in the error estimate (4.22), by the use of an incomplete GMRES method. Furthermore, we prove that the overall complexity of the fully discrete dG method is linear (up to some logarithmic terms). Since these matters have already been studied in [36, section 5], we restrict ourselves to a brief presentation of the results there.

5.1. Derivation of the linear systems. The dG time-stepping scheme (4.19) corresponds to a linear system of size $(r_m+1)N^L$ to be solved in each time step $m=1,2,\dots,M$. Here and in what follows, in order to clarify the dependence on the refinement level L explicitly, we denote by N^L the dimension of the finite element space $V_h=V^L$.

For $1\le m\le M$, let $\{\phi_j=\sqrt{j+1/2}L_j\}_{j=0}^{r_m}$, where L_j is the jth Legendre polynomial on $(-1,1)$ (normalized such that $L_j(1)=1$), be a basis of the polynomial space $\mathcal{P}_{r_m}(-1,1)$. Then, the temporal shape functions on the time interval I_m are given by $\phi_j\circ F_m^{-1}$, where the mapping $F_m:(-1,1)\to I_m$ is given by

$$t=F_m(\hat t)=\frac{1}{2}(t_{m-1}+t_m)+\frac{1}{2}k_m\hat t,\qquad k_m=t_m-t_{m-1},\qquad \hat t\in(-1,1).$$

Writing $\widetilde U_{h,m}^{dG}(x,t)=\widetilde U_h^{dG}|_{I_m}(x,t)$ and $W_m=W|_{I_m}$ in (4.19) as

$$\widetilde U_{h,m}^{dG}(x,t)=\sum_{j=0}^{r_m}\widetilde U_{h,m,j}^{dG}(x)(\phi_j\circ F_m^{-1})(t),$$
$$W_m(x,t)=\sum_{i=0}^{r_m}W_{m,i}(x)(\phi_i\circ F_m^{-1})(t),$$

the variational formulation (4.19) reads: find $(\widetilde U_{h,m,j}^{dG})_{j=0}^{r_m}\in(V_h)^{\mathrm{r}}$ such that for all $(W_{m,i})_{i=0}^{r_m}\in(V_h)^{\mathrm{r}}$ there holds

$$\sum_{i,j=0}^{r_m}C_{ij}(\widetilde U_{h,m,j}^{dG},W_{m,i})+\frac{k_m}{2}\sum_{i=1}^{r_m}\widetilde a(\widetilde U_{h,m,i}^{dG},W_{m,i})=\sum_{i=0}^{r_m}f_{m,i}(W_{m,i}),\tag{5.1}$$

where, for $i,j=0,1,\dots,r_m$,

$$C_{ij}=\sigma_{ij}\sqrt{(i+1/2)(j+1/2)},\quad \sigma_{ij}=\begin{cases}(-1)^{i+j} & \text{if } j>i,\\ 1 & \text{else,}\end{cases}\tag{5.2}$$

and $f_{m,i}(v)=\phi_i(-1)\big(\widetilde U_{h,m-1}^{dG-}(t_{m-1}),v\big)$.

Henceforth, for the sake of readability, we will drop the subscript m. Denoting by $\mathbf{M}$ and $\widetilde{\mathbf{A}}$ the mass and (compressed) stiffness matrix on $V^h=V^L$ with respect to $(\cdot,\cdot)$ and $\widetilde a(\cdot,\cdot)$, respectively, (4.19) takes the matrix form

$$\mathbf{R}\underline{u}=\underline{f}\quad\text{with}\quad\mathbf{R}=\mathbf{C}\otimes\mathbf{M}+\frac{k}{2}\mathbf{I}\otimes\widetilde{\mathbf{A}},\tag{5.3}$$

where $\underline{u}$ denotes the coefficient vector of $\widetilde U_{h,m}^{dG}=\widetilde U_h^{dG}|_{I_m}\in\mathcal{P}_{r_m}(I_m,V_h)$.

5.2. Decoupling. In [33] it has been found that the system (5.3) of size $(r+1)N^L$ can be reduced to solving $r+1$ linear systems of size N^L. To this end, let $\mathbf{C}=\mathbf{Q}\mathbf{T}\mathbf{Q}^{\mathsf H}$ be the Schur decomposition of the $(r+1)\times(r+1)$ matrix $\mathbf{C}$ with a unitary matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{T}$. Note that the diagonal of $\mathbf{T}$ contains the eigenvalues $\lambda_1,\lambda_2,\dots,\lambda_{r+1}$ of $\mathbf{C}$. Then multiplying (5.3) by $\mathbf{Q}^{\mathsf H}\otimes\mathbf{I}$ from the left results in the linear system

$$\left(\mathbf{T}\otimes\mathbf{M}+\frac{k}{2}\mathbf{I}\otimes\widetilde{\mathbf{A}}\right)\underline{w}=\underline{g}\quad\text{with}\quad\underline{w}=(\mathbf{Q}^{\mathsf H}\otimes\mathbf{I})\underline{u},\quad\underline{g}=(\mathbf{Q}^{\mathsf H}\otimes\mathbf{I})\underline{f}.$$

This system is block-upper-triangular. With $\underline{w} = (\underline{w}_0, \underline{w}_1, \ldots, \underline{w}_r)$, $\underline{w}_j \in \mathbb{C}^{N^L}$, we obtain its solution by solving

$$\left(\lambda_{j+1}\mathbf{M} + \frac{k}{2}\widetilde{\mathbf{A}}\right)\underline{w}_j = \underline{s}_j \tag{5.4}$$

for $j = r, r-1, \ldots, 0$, where $\underline{s}_j = \underline{g}_j - \sum_{l=j+1}^{r} \mathbf{T}_{j+1,l+1}\mathbf{M}\underline{w}_l$.

5.3. Iterative solution of the linear equations. By (5.4), a dG-time step of order r amounts to solving $r+1$ linear systems with a matrix of the form

$$\mathbf{B} = \lambda\mathbf{M} + \frac{k}{2}\widetilde{\mathbf{A}}. \tag{5.5}$$

Here, λ is an eigenvalue of $\mathbf{C}$ in (5.2). The linear equations (5.4) are solved approximately with incomplete GMRES iteration, causing an additional error in the overall scheme which we are going to analyze next.

5.3.1. Eigenvalues of C. The eigenvalues λ_j of $\mathbf{C}$ have a major impact on the choice of the preconditioner (see section 5.3.2) and thus on the convergence of the GMRES method for (5.4). It can be shown that (see [36, Lemma 5.5]) for $r = 1, 2, 3, \ldots$, the eigenvalues $\lambda_j^{(r)}$ of the matrix $\mathbf{C}$ from (5.2) satisfy

$$\mathrm{Re}\lambda_j^{(r)} \geq C_1|\lambda_j^{(r)}|r^{-\alpha}, \quad |\lambda_j^{(r)}| \geq C_2 r^{\widetilde{\alpha}}, \qquad j = 1, 2, \ldots, r+1, \tag{5.6}$$

with $\alpha = 2$, $\widetilde{\alpha} = 0$, and constants $C_1, C_2 > 0$ independent of r. Furthermore, the matrix $\mathbf{T}$ of the Schur decomposition $\mathbf{C} = \mathbf{Q}\mathbf{T}\mathbf{Q}^{\mathsf{H}}$ satisfies $\|\mathbf{T}\|_2 \leq Cr^2$.

Remark 5.1. The estimates (5.6) do not seem to be sharp. Numerical results in [36] suggest that (5.6) holds with $\alpha = 2/3$ and $\widetilde{\alpha} = 1$.

5.3.2. Preconditioning. For the preconditioning of the linear system (5.5), we define the matrix $\mathbf{S}$ and the scaled matrix $\widehat{\mathbf{B}} \in \mathbb{R}^{N^L} \times \mathbb{R}^{N^L}$ by

$$\mathbf{S} := \left(\mathrm{Re}(\lambda)\mathbf{I} + \frac{k}{2}\mathbf{D}_{\mathbf{A}}\right)^{1/2}, \qquad \widehat{\mathbf{B}} := \mathbf{S}^{-1}\mathbf{B}\mathbf{S}^{-1},$$

where $\mathbf{D}_{\mathbf{A}}$ is the diagonal matrix with entries $D_{(l,j),(l,j)} = 2^{l\rho/2}$ corresponding to the basis functions $\psi_j^l \in V_h$; cf. (4.3). Moreover, we introduce the norm

$$\|\underline{w}\|_{\mathbf{D}_{\mathbf{A}}^{-1}} = (\underline{w}^{\mathsf{H}}\mathbf{D}_{\mathbf{A}}^{-1}\underline{w})^{1/2}, \qquad \underline{w} \in \mathbb{R}^{N^L}.$$

Let GMRES(m_0) denote the GMRES method with restart every $m_0 \geq 1$ iterations. For the linear system $\widehat{\mathbf{B}}\underline{\widehat{x}} = \underline{\widehat{b}}$, let $\underline{\widehat{x}}_j$ be the iterates obtained by restarted GMRES(m_0) with initial guess $\underline{\widehat{x}}_0$. Then, there exists $C > 0$ such that (cf. [36, Lemma 5.7])

$$\|\underline{\widehat{b}} - \widehat{\mathbf{B}}\underline{\widehat{x}}_j\| \leq Cq^j\|\underline{\widehat{b}} - \widehat{\mathbf{B}}\underline{\widehat{x}}_0\|, \quad j = 0, 1, 2, \ldots,$$

where $q = 1 - cr^{-2\alpha}$. Moreover, let $\underline{x}_j = \mathbf{S}^{-1}\underline{\widehat{x}}_j$, $\underline{b} = \mathbf{S}\underline{\widehat{b}}$. Then

$$\|\underline{b} - \mathbf{B}\underline{x}_j\|_{\mathbf{D}_{\mathbf{A}}^{-1}} \leq Ch^{-\rho/2}(1 + kr^{\alpha-\widetilde{\alpha}})^{1/2}(1 - cr^{-2\alpha})^j\|\underline{b} - \mathbf{B}\underline{x}_0\|_{\mathbf{D}_{\mathbf{A}}^{-1}}, \tag{5.7}$$

with $\alpha, \widetilde{\alpha}$ from (5.6), and $C, c > 0$ independent of L, k, r.

5.3.3. Convergence of incomplete GMRES. The fully discrete dG scheme with incomplete GMRES yields approximations $\widehat{U}_{h,1}^{dG}, \widehat{U}_{h,2}^{dG}, \dots, \widehat{U}_{h,M}^{dG}$ to $\widetilde{U}_{h,1}^{dG}, \widetilde{U}_{h,2}^{dG}, \dots, \widetilde{U}_{h,M}^{dG}$ (cf. (5.1)). By (5.7) and proceeding as in [36, sections 5.5.3 and 5.5.4], we obtain that

$$\|\widehat{U}_{h,M}^{dG}(T) - \widetilde{U}_{h,M}^{dG}(T)\| \le Ch^{-3\rho/2} k_1^{-1} r_M^3 q^{n_G}. \tag{5.8}$$

Here, n_G denotes the number of GMRES(m_0) iterations which has to be chosen in such a way that the bound in (5.8) is smaller than the right-hand side of (4.22) in the convergence theorem, Theorem 4.12, i.e.,

$$h^{-3\rho/2} k_1^{-1} r_M^3 q^{n_G} \le Ch^{\omega}.$$

In order to fulfill this condition, we recall the definition of q and use, in accordance with the assumptions in Theorem 4.12, that $r_M = \mathcal{O}(|\log h|)$ and $|\log k_1| \le C r_M$. This results in

$$n_G \ge C r_M^{2\alpha} \mathcal{O}(|\log h|) = \mathcal{O}(|\log h|)^{1+2\alpha} \tag{5.9}$$

GMRES iterations per time step. Combining the previous results yields the following theorem.

THEOREM 5.2. [illegible] 4.12 [illegible] $n_G = \mathcal{O}(|\log h|)^{1+2\alpha}$ [illegible]

$$\|u(T) - \widehat{U}_h^{dG}(T)\| \le Ch^{\omega}, \tag{5.10}$$

[illegible] $\widehat{U}_h^{dG}$ [illegible] u [illegible]

5.4. Complexity. Applying the matrix compression techniques from section 4.1.2, the judicious combination of geometric mesh refinement and linear increase of polynomial degrees in the dG time-stepping scheme (Theorem 4.12), and an appropriate number of GMRES iterations (Theorem 5.2), results in a linear (up to some logarithmic terms) overall complexity of the fully discrete scheme (4.15) for the solution of the parabolic problem (2.1), (2.2).

THEOREM 5.3. [illegible] 4.12 [illegible] 5.2, [illegible] (4.15) [illegible] n_G [illegible] (5.9) [illegible] $\widehat{U}_h^{dG}(T)$ [illegible] $\mathcal{O}(N(\log N)^{4+2\alpha})$ [illegible] $N = N^L = \dim V_h$ [illegible] α [illegible] (5.6).

[illegible]. There are $M = \mathcal{O}(|\log h|) = \mathcal{O}(\log N)$ time steps. In each time step $\mathcal{O}(\log N)$ linear systems of type (5.5) have to be solved by the incomplete GMRES method with

$$n_G = \mathcal{O}(|\log h|^{1+2\alpha}) = \mathcal{O}((\log N)^{1+2\alpha})$$

iterations. We note that the mass matrix $\mathbf{M}$ (with respect to the wavelet basis) has $\mathcal{O}(N \log N)$ entries. Hence, recalling Proposition 4.4, each GMRES iteration involves a matrix-vector multiplication of complexity $\mathcal{O}(N \log N)$. This implies the desired result. ∎

6. Numerical examples. In this section, we study the performance of the dG time-stepping scheme. Our wavelet spatial discretization is based on the biorthogonal

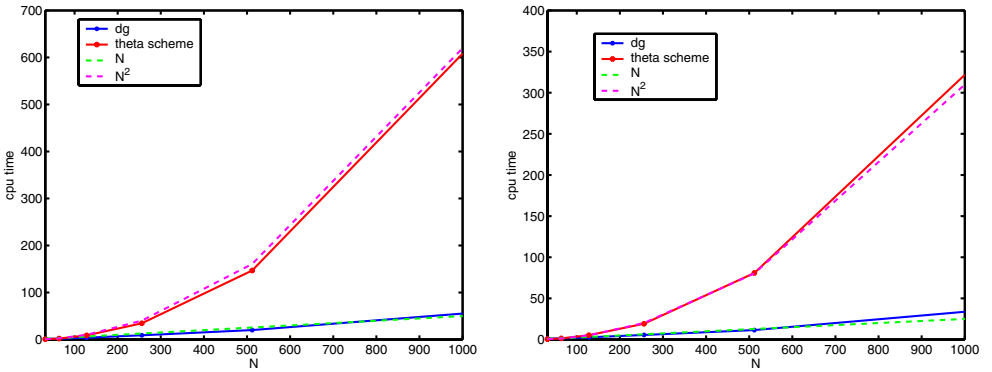


FIG. 1. *Performance comparison for the dG time-stepping versus θ-scheme (both with incomplete GMRES) to advance from $t = 0$ to $t = T$; left: $\sigma > 0$, right: $\sigma = 0$.*

spline wavelets (see Remark 4.1) with wavelet compression according to (4.5). The bandwidth of the lower-right block in the compressed wavelet matrix is 5 and the growth factor is taken to be 0.8, which corresponds to $c = 1$ and $\hat{\alpha} = 0.8$ in (4.5).

Our experiments address the numerical solution of (3.31), (3.32) with $\Omega = \Omega_R = (-6, 6)$, $g(x) = (e^x - K)_+$, $K = 1$, and $T = 1$. The integral kernel k of the nonlocal operator $\mathcal{L}^{\text{jump}}$ in (3.9) is of CGMY-type (3.15) with $\alpha \in [0, 2)$. We recall that, for $\sigma > 0$, the operator $\mathcal{A} : V \to V^*$ with $V = H_0^1(\Omega)$ is given by $\mathcal{A} = (\mathcal{L}^{\text{BS}} + \mathcal{L}^{\text{jump}})|_\Omega$ as in (3.8), (3.9). In the pure jump case, i.e., $\sigma = 0$, there holds $\mathcal{A} = \mathcal{L}^{\text{jump}}|_\Omega$.

Our aim is to illustrate the effect of the choice of the mesh size $h = 2R2^{-L-1}$, the number M of time steps, and the polynomial degree r in the dG method on the error of the numerical approximation at $t = T$.

To do so, let $\widehat{U}_h^{dG}(T)$ be the numerical price obtained by the dG time-stepping scheme using the incomplete GMRES method with wavelet compression and wavelet preconditioning. Furthermore, denote by $\widehat{U}_h^{\theta}(T)$ the corresponding solution obtained by time-stepping with the θ-scheme (also with incomplete GMRES and wavelet preconditioning). In Figure 1 we compare the CPU times for the computation of $\widehat{U}_h^{dG}(T)$ and $\widehat{U}_h^{\theta}(T)$ (with $\theta = 1$, i.e., implicit Euler scheme) with diffusion component (i.e., for $\sigma > 0$, left) and in the quadratic pure jump case (i.e., for $\sigma = 0$, right). The parameters in the stochastic process are $C = 1$, $\eta_- = 1.8$, $\eta_+ = 2.5$, and $\alpha = 1.2$.

We study convergence of the prices obtained from dG time-stepping and by the θ time-stepping schemes, respectively, expressed in the error estimate (5.10) as $h \to 0$. We set $\sigma = 0$ and $\alpha = 1.8$. The other CGMY parameters are $C = 1$, $\eta_- = 1.8$, and $\eta_+ = 2.5$. We have chosen a large fixed value for the number n_G of GMRES iterations, so that all the errors become insensitive with respect to the iteration error. Again, by $U_h^{dG}(T, x)$ and $U_h^{\theta}(T, x)$, we denote the solutions of the fully discrete dG time-stepping and the θ time-stepping schemes, respectively. The number M of time steps is chosen proportionally to L; more precisely, $M = 2|\log(h)|$. Our "exact" solution $U_{\text{ex}}(T, x)$ is computed as the numerical solution of the dG method with $L = 11$, $M = 2|\log(h)|$, and $r = M$ for all time steps. In Figure 2, we plot the relative error with respect to the L^2 norms

$$\frac{\|U_{\text{ex}}(T,x)\|^2_{L^2(\Omega_R)} - \|U_h^{dG}(T,x)\|^2_{L^2(\Omega_R)}}{\|U_{\text{ex}}(T,x)\|^2_{L^2(\Omega_R)}} \tag{6.1}$$

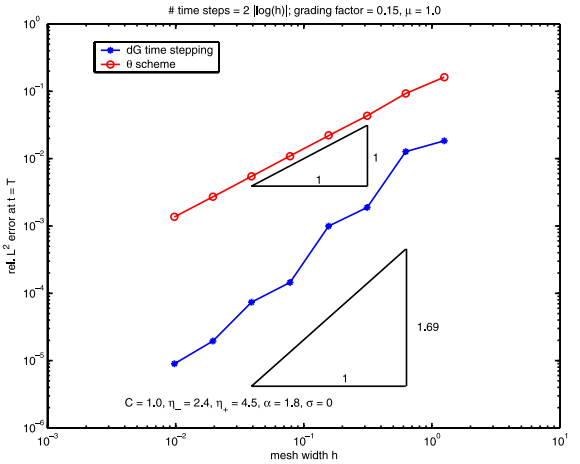


FIG. 2. *Comparison of the L^2 errors at $t = T$.*

for the dG time-stepping scheme, and compare it to the relative L^2 error of the standard (low order) θ-scheme

$$\frac{\|U_{\mathrm{ex}}(T,x)\|^2_{L^2(\Omega_R)} - \|U_h^\theta(T,x)\|^2_{L^2(\Omega_R)}}{\|U_{\mathrm{ex}}(T,x)\|^2_{L^2(\Omega_R)}}.$$

As above, we take $\theta = 1$. We clearly observe linear convergence for the implicit Euler time-stepping and an experimental convergence rate of order 1.69 for the dG time-stepping scheme.

In Figures 3 and 4, we plot the relative L^2 errors and the convergence rates obtained by least square fitting for the dG time-stepping and varying values of α and σ. Again, the number M of time steps is proportional to L, $M = 2|\log(h)|$. The geometric time mesh has a grading factor $\gamma = 0.15$ and linearly increasing polynomial degrees with slope $\mu = 1.0$. The "exact" solution is obtained as before.

Table 1 presents the values of the numerical approximation $U_h^{dG}(T)$ in the pure jump case ($\sigma = 0$) with CGMY parameters $C = 1.0$, $\eta_- = 2.4$, $\eta_+ = 4.5$, and $\alpha = 1.8$. The grading factor of the geometric time mesh has been chosen to be $\gamma = 0.2$ and the slope of the linear polynomial degree vector is given by $\mu = 1.0$. As we increase the number of time steps (for fixed L), i.e., $M \to \infty$, we observe rapidly converging error levels. This indicates that the spatial discretization errors are dominant and that the time-stepping scheme renders exponential convergence.

7. Concluding remarks. In this work, we have studied the (fast) numerical approximation of (possibly degenerate) parabolic equations for general Markov processes arising in finance. Our scheme is based on *hp*-discontinuous Galerkin discretizations in time and wavelet discretizations in space. The densely populated matrices corresponding to the nonlocal infinitesimal generators are compressed using an appropriate wavelet basis. The linear systems are solved by an incomplete GMRES method. Under these conditions, we have shown that the overall complexity of the algorithm is linear up to some logarithmic terms. Moreover, we have proved that if the wavelet spatial semidiscretization error is of order h^ω, then our algorithm gives $\widehat{U}_h^{dG}(T)$ with the same error bound in log-linear complexity. Finally, numerical experiments illustrate the performance of the proposed scheme.

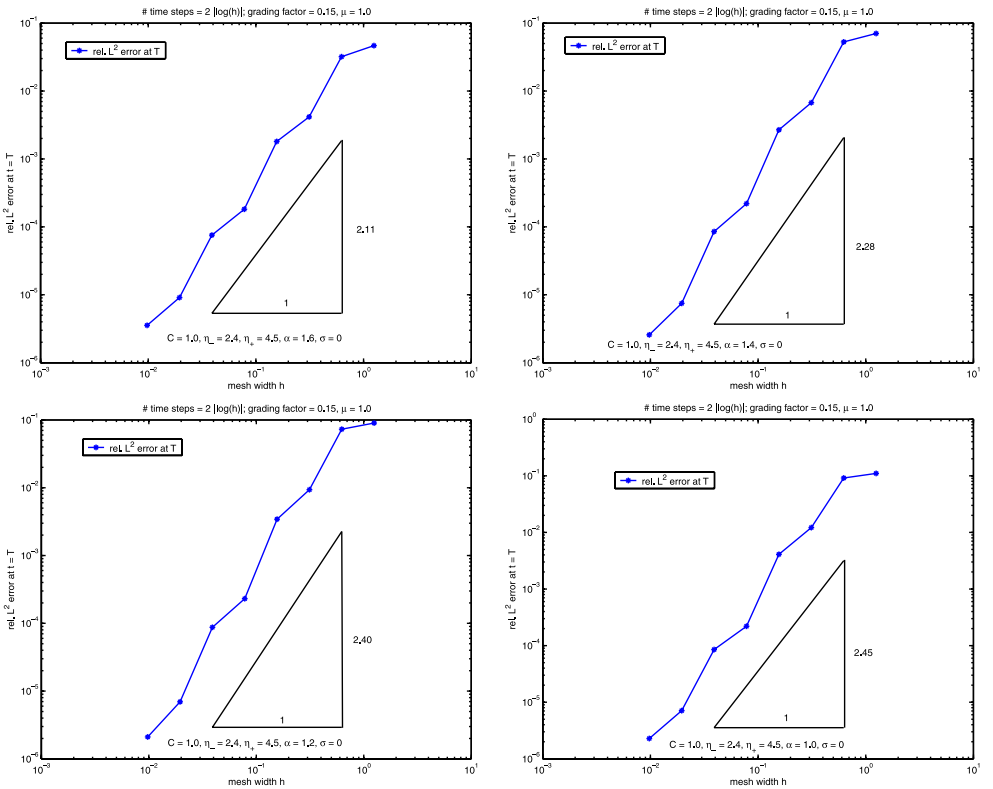


FIG. 3. *Relative L^2 errors* (6.1) *for quadratic pure jump CGMY Lévy processes ($\sigma = 0$), jump intensity $\alpha \in \{1.6, 1.4, 1.2, 1.0\}$.*

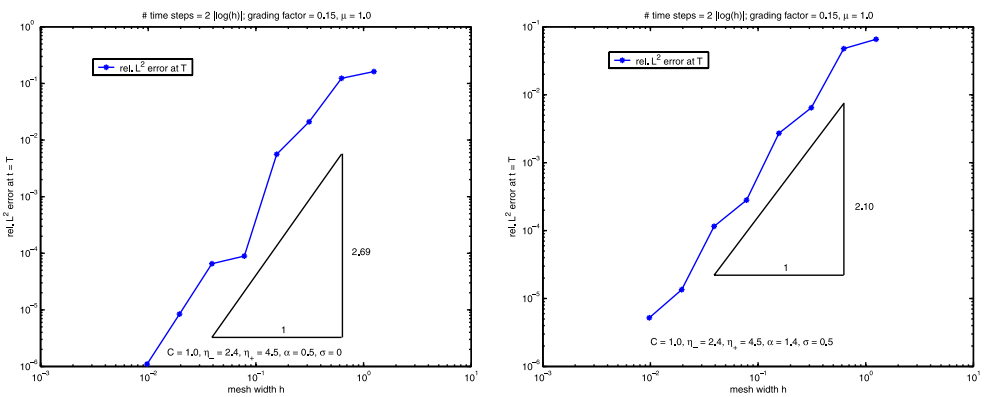


FIG. 4. *Relative L^2 errors at maturity T* (6.1) *for the case of quadratic pure jump ($\sigma = 0$) and finite variation ($\alpha = 0.5 < 1.0$) (left) and for the case of jump diffusion $\sigma > 0$ and $\alpha = 1.4$ (right).*

TABLE 1
Convergence of $\|U_h^{dG}(T)\|_{L^2(\Omega_R)}$ *as* $h \to 0$*;* $\alpha = 1.8$.

Relative L^2 error for $\alpha = 1.8$, $\sigma = 0.0$							
$L\backslash M$	3	4	5	6	7	8	9
2	0.0197202	0.0196413	0.0196417	0.0196431	0.0196427	0.0196428	0.0196428
3	0.0143566	0.0142750	0.0142687	0.0142715	0.0142710	0.0142711	0.0142711
4	0.0019381	0.0018671	0.0018557	0.0018595	0.0018589	0.0018590	0.0018590
5	0.0010473	0.0009786	0.0009663	0.0009702	0.0009696	0.0009697	0.0009697
6	0.0002205	0.0001526	0.0001401	0.0001440	0.0001434	0.0001435	0.0001435
7	0.0001507	0.0000828	0.0000703	0.0000742	0.0000736	0.0000737	0.0000737
8	0.0000964	0.0000287	0.0000161	0.0000201	0.0000195	0.0000195	0.0000195
9	0.0000858	0.0000181	0.0000055	0.0000095	0.0000089	0.0000089	0.0000089
10	0.0000793	0.0000116	0.0000009	0.0000030	0.0000024	0.0000024	0.0000024

Appendix A. Proof of Theorem 4.12. For the proof of Theorem 4.12, we proceed in several steps. Again, we denote by u, $\widetilde{u}_h$, and $\widetilde{U}_h^{dG}$ the solutions of (2.1)–(2.2), (4.12)–(4.13), and (4.15), respectively.

A.1. *hp*-Approximations. We consider an interpolant

$$\Pi^{\underline{r}} : H^1(J, V_h) \to \mathcal{S}^{\underline{r}}(\mathcal{M}, V_h),$$

with $\Pi^{\underline{r}}|_{I_m} = \Pi^{r_m}$ satisfying the conditions

$$\int_{I_m} (u_h - \Pi^{r_m} u_h, q)\, dt = 0 \qquad \forall q \in \mathcal{P}^{r_m - 1}(I_m, V_h), \tag{A.1}$$
$$\Pi^{r_m} u_h(t_m^-) = u_h(t_m^-)$$

for all $1 \le m \le M$, $u_h \in H^1(J, V_h)$. If, for some m, $r_m = 0$, then the first condition is void on I_m. According to [33], the interpolant $\Pi^{\underline{r}}$ is uniquely defined by the above conditions. Moreover, we have the following approximation result.

LEMMA A.1. [illegible] $\mathcal{M}_{M,\gamma}$ [illegible] $0 < \gamma < 1$ [illegible] 4.10 [illegible] $\underline{r}$ [illegible] $\mu \ge \mu_0$ [illegible] 4.11 [illegible] $\mu_0 > 0$ [illegible] $C, b > 0$ [illegible] M [illegible]

$$\|\widetilde{u}_h - \Pi^{\underline{r}} \widetilde{u}_h\|^2_{L^2(J,V)} \le C e^{-b\sqrt{\widehat{N}}}, \tag{A.2}$$

[illegible] $\widehat{N} = \dim \mathcal{S}^{\underline{r}}(\mathcal{M}_{M,\gamma}, V_h) / \dim V_h$ [illegible]

In order to prove this lemma, we follow along the lines of [33]. In particular, we make use of the regularity results in [33, section 4.2] which imply, as in the proof of [33, Theorem 5.4], that the left-hand side in (A.2) is exponentially small on geometrically refined time partitions with linearly increasing polynomial degrees in the time steps.

A.2. Convergence of the fully discrete dG scheme. Next, we split the error of the dG method (4.15) into two parts:

$$\widetilde{u}_h - \widetilde{U}_h^{dG} = \eta + \xi, \tag{A.3}$$

where

$$\eta = \widetilde{u}_h - \Pi^{\underline{r}} \widetilde{u}_h, \qquad \xi = \Pi^{\underline{r}} \widetilde{u}_h - \widetilde{U}_h^{dG}. \tag{A.4}$$

In the following lemma we will prove that the dG error (A.3) at $t = T$ may be estimated in terms of η only.

LEMMA A.2. [illegible] (4.15), [illegible]

$$\|\widetilde{u}_h(T) - \widetilde{U}_h^{dG}(T)\|^2 \le C \int_0^T \|\eta\|_V^2 \, dt.$$

[illegible] η [illegible] (A.4).

[illegible]. The main idea is to bound ξ in terms of η. To do so, we first note that, by the Galerkin orthogonality (4.18), we have

$$\widetilde{B}_{dG}(\xi, \xi) = -\widetilde{B}_{dG}(\eta, \xi). \tag{A.5}$$

Moreover, as in (4.20), there holds that

$$\widetilde{B}_{dG}(\xi, \xi) = \|\xi\|_{dG}^2 \ge \int_0^T \widetilde{a}(\xi, \xi) \, dt. \tag{A.6}$$

Furthermore, making use of the properties (A.1) of the interpolant Π^r, we obtain for all $1 \le m \le M$ that $\int_{I_m} (\eta, \xi') \, dt = 0$ and $\eta(t_m^-) = 0$. Hence, integrating by parts leads to

$$\begin{aligned}\widetilde{B}_{dG}(\eta, \xi) &= \sum_{m=1}^{M} \int_{I_m} \big((\eta', \xi) + \widetilde{a}(\eta, \xi) \big) \, dt + \sum_{m=1}^{M} ([\eta]_m, \xi_m^+) + (\eta_0^+, \xi_0^+) \\ &= \int_0^T \widetilde{a}(\eta, \xi) \, dt \le C \int_0^T \|\eta\|_V \|\xi\|_V \, dt.\end{aligned}$$

In the last step, the continuity of $\widetilde{a}$ was employed. Additionally, using a weighted Cauchy–Schwarz inequality and the coercivity of the bilinear form $\widetilde{a}$ results in

$$|\widetilde{B}_{dG}(\eta, \xi)| \le C \int_0^T \|\eta\|_V^2 \, dt + \frac{1}{2} \int_0^T \widetilde{a}(\xi, \xi) \, dt.$$

Thus, recalling (A.5) and inserting (A.6), we obtain

$$\widetilde{B}_{dG}(\xi, \xi) \le |\widetilde{B}_{dG}(\eta, \xi)| \le C \int_0^T \|\eta\|_V^2 \, dt + \frac{1}{2} \widetilde{B}_{dG}(\xi, \xi).$$

Finally, subtracting $\frac{1}{2} \widetilde{B}_{dG}(\xi, \xi)$ on both sides of the above inequality and noticing that $\eta_M^- = 0$ yields

$$\|\widetilde{u}_h(T) - \widetilde{U}_h^{dG}(T)\|^2 = \|\xi_M^-\|^2 \overset{(4.20)}{\le} C \widetilde{B}_{dG}(\xi, \xi) \le C \int_0^T \|\eta\|_V^2 \, dt.$$

This is the desired estimate. ∎

Combining Lemmas A.1 and A.2 and the assumption (4.14) on the convergence of the spatial semidiscrete problem (4.12), (4.13), we obtain the following.

PROPOSITION A.3. [illegible] $C, b > 0$ [illegible] $u - \widetilde{U}_h^{dG}$ [illegible] (4.15) [illegible]

$$\|u(T) - \widetilde{U}_h^{dG}(T)\|^2 \le C \left(h^{2\omega} + e^{-b\sqrt{\widehat{N}}} \right),$$

[illegible] u $\widetilde{U}_h^{dG}$ [illegible] (2.1), (2.2) [illegible] (4.15), [illegible] $\widehat{N} = \dim \mathcal{S}^r(\mathcal{M}_{M,\gamma}, V_h) / \dim V_h$

. There holds

$$\|u(T) - \widetilde{U}_h^{dG}(T)\|^2 \le 2\|u(T) - \widetilde{u}_h(T)\|^2 + 2\|\widetilde{u}_h(T) - \widetilde{U}_h^{dG}(T)\|^2,$$

where $\widetilde{u}_h$ is the solution of the semidiscrete problem (4.12), (4.13). Then, referring to Lemmas A.1 and A.2 and (4.14) implies

$$\|u(T) - \widetilde{U}_h^{dG}(T)\|^2 \le C\left(h^{2\omega} + e^{-b\sqrt{\widehat{N}}}\right),$$

which completes the proof. ∎

We are now in a position to prove Theorem 4.12. Indeed, since $\sqrt{\widehat{N}} \ge CM$ and $M = c|\log h|$, we have that $e^{-b\sqrt{\widehat{N}}} \le e^{-\widetilde{b}c|\log h|} = h^{\widetilde{b}c}$. Hence, for $c \ge 2\omega\widetilde{b}^{-1}$, the error bound (4.22) follows immediately from Proposition A.3.

REFERENCES

[1] R. A. Adams, *Sobolev Spaces*, Academic Press, New York, 1978.

[2] O. Barndorff-Nielsen, *Processes of normal inverse Gaussian type*, Finance Stoch., 2 (1998), pp. 41–68.

[3] J. Bertoin, *Lévy Processes*, Cambridge University Press, Cambridge, UK, 1996.

[4] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Political Economy, 81 (1973), pp. 637–654.

[5] S. I. Boyarchenko and S. Z. Levendorskii, *Non-Gaussian Merton-Black-Scholes Theory*, Adv. Ser. Stat. Sci. Appl. Prob. 9, World Scientific, River Edge, NJ, 2002.

[6] P. Carr, H. Geman, and D. B. Madan, *The fine structure of asset returns: An empirical investigation*, J. Business, 75 (2002), pp. 305–332.

[7] A. Cohen, *Wavelet methods in numerical analysis*, in Handbook of Numerical Analysis, Vol. 7, North-Holland, Amsterdam, 2000, pp. 417–711.

[8] R. Cont, J. P. Bouchaud, and M. Potters, *Scaling in financial data: Stable laws and beyond*, in Scale Invariance and Beyond, B. Dubrulle, F. Graner, and D. Sornette, eds., Springer, Berlin, 1997.

[9] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*, Chapman and Hall/CRC Press, 2003.

[10] R. Cont and E. Voltchkova, *A Finite-Difference Scheme for Option Pricing in Jump-Diffusion and Exponential Lévy Models*, Rapport Interne 513, CMAP, Palaiseau, France, 2003.

[11] W. Dahmen, A. Kunoth, and K. Urban, *Biorthogonal spline wavelets on the interval—stability and moment conditions*, Appl. Comput. Harmon. Anal., 6 (1999), pp. 132–196.

[12] E. Eberlein, *Application of generalized hyperbolic Lévy motions to finance*, in Lévy Processes: Theory and Applications, O. E. Barndorff-Nielsen, T. Mikosch, and S. Resnick, eds., Birkhäuser, Boston, 2001, pp. 319–337.

[13] E. Eberlein and U. Keller, *Hyperbolic distributions in finance*, Bernoulli, 1 (1995), pp. 281–299.

[14] E. Eberlein, U. Keller, and K. Prause, *New insights into smile, mispricing and value at risk: the hyperbolic model*, J. Business, 71 (1998), pp. 371–405.

[15] M. Freidlin, *Functional Integration and Partial Differential Equations*, Ann. of Math. Stud., Princeton University Press, Princeton, NJ, 1985.

[16] H. Fujita and A. Mizutani, *Remarks on the finite element method for parabolic equations with higher accuracy*, in Functional Analysis and Numerical Analysis (Japan-France Seminar in Tokyo and Kyoto, 1976), Japan Society for the Promotion of Science, Tokyo, 1978.

[17] H. Fujita and T. Suzuki, *Evolution problems*, in Handbook of Numerical Analysis II: Finite Element Methods I, North–Holland, Amsterdam, 1991, pp. 789–928.

[18] N. Jacob, *Pseudo-Differential Operators and Markov Processes*. Vol. I: *Fourier Analysis and Semigroups*, Imperial College Press, London, 2001.

[19] N. Jacob, *Pseudo-Differential Operators and Markov Processes*. Vol. II: *Generators and Their Potential Theory*, Imperial College Press, London, 2002.

[20] I. Karatzas and S. E. Shreve, *Methods of Mathematical Finance*, Springer, Berlin, 1999.

[21] I. Koponen, *Analytic approach to the problem of convergence of truncated Lévy flights towards the Gaussian stochastic process*, Phys. Rev. E, 52 (1995), pp. 1197–1199.
[22] G. Kou, *A jump diffusion model for option pricing*, Management Sci., 48 (2002), pp. 1086–1101.
[23] D. Lamberton and B. Lapeyre, *Introduction to Stochastic Calculus Applied to Finance*, Chapman & Hall, London, 1997.
[24] J. L. Lions and E. Magenes, *Non-Homogeneous Boundary Value Problems and Applications*, Springer, Berlin, 1972.
[25] D. B. Madan, P. Carr, and E. Chang, *The variance gamma process and option pricing*, European Finance Review, 2 (1998), pp. 79–105.
[26] A.-M. Matache, C. Schwab, and T. P. Wihler, *Linear complexity solution of parabolic integro-differential equations*, submitted.
[27] A.-M. Matache, T. von Petersdorff, and C. Schwab, *Fast Deterministic Pricing of Options on Lévy Driven Assets*, M2AN Math. Model. Numer. Anal., to appear.
[28] R. C. Merton, *Option pricing when the underlying stocks are discontinuous*, J. Financ. Econ., 5 (1976), pp. 125–144.
[29] D. Nualart and W. Schoutens, *Backward stochastic differential equations and Feynman-Kac formula for Lévy processes, with applications in finance*, Bernoulli, 7 (2001), pp. 761–776.
[30] H. C. Öttinger, *Stochastic Processes in Polymeric Fluids*, Springer, New York, 1998.
[31] K.-I. Sato, *Lévy Processes and Infinitely Divisible Distributions*, Cambridge University Press, Cambridge, UK, 1999.
[32] R. Schneider, *Multiskalen- und Wavelet-Matrixkompression. Analysisbasierte Methoden zur effizienten Lösung großer vollbesetzter Gleichungssysteme*, Adv. Numer. Math., B. G. Teubner, Stuttgart, 1998.
[33] D. Schötzau and C. Schwab, *Time discretization of parabolic problems by the hp-version of the discontinuous Galerkin finite element method*, SIAM J. Numer. Anal., 38 (2000), pp. 837–875.
[34] D. Schötzau and C. Schwab, *hp-discontinuous Galerkin time-stepping for parabolic problems*, C. R. Acad. Sci. Paris Sér. I Math., 333 (2001), pp. 1121–1126.
[35] T. von Petersdorff and C. Schwab, *Fully discrete multiscale Galerkin BEM*, in Multiresolution Analysis and Partial Differential Equations, Wavelet Analysis and Its Applications 6, W. Dahmen, P. Kurdila, and P. Oswald, eds., Academic Press, New York, 1997, pp. 287–346.
[36] T. von Petersdorff and C. Schwab, *Numerical solution of parabolic equations in high dimensions*, M2AN Math. Model. Numer. Anal., 38 (2004), pp. 93–127.
[37] T. von Petersdorff and C. Schwab, *Wavelet discretizations of parabolic integrodifferential equations*, SIAM J. Numer. Anal., 41 (2003), pp. 159–180.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 394–414
© 2005 Society for Industrial and Applied Mathematics

DISPERSION RELATION ANALYSIS OF THE $P_1^{NC} - P_1$ FINITE-ELEMENT PAIR IN SHALLOW-WATER MODELS*

DANIEL Y. LE ROUX†

Abstract. Shallow-water models typically employ gridpoint, finite, and spectral-element techniques. For most of these methods the coupling between the momentum and continuity equations is a delicate problem and usually leads to spurious solutions in the representation of inertia-gravity waves. The spurious modes have a wide range of characteristics and may take the form of pressure (surface-elevation), velocity, and/or Coriolis modes. The modes usually cause aliasing and an accumulation of energy in the smallest-resolvable scale, leading to noisy solutions. The triangular finite-element pair $P_1^{NC} - P_1$ is shown to "properly" model the dispersion of the inertia-gravity waves, and this is achieved at a reasonable computational cost compared to traditional finite-difference schemes. Two tests are proposed. The first test examines the propagation and dispersion of fast surface gravity waves in a circular basin and their reflection at the lateral boundary. In the second test, results of the Stommel problem are presented using the proposed finite-element pair and are compared with those obtained with the C- and CD-finite-difference grids. They illustrate the promise of the proposed approach.

Key words. partial differential equations, linear waves, finite elements, finite difference method

AMS subject classifications. 35L99, 74J05, 65N30, 65L12

DOI. 10.1137/030602435

1. Introduction.

The shallow-water equations are of considerable importance for a variety of problems of coastal and environmental engineering—including oceanic, atmospheric, and groundwater flows. Numerical methods that are usually employed are typically gridpoint, finite, and spectral-element techniques. One of the main problems encountered with the discretization of the shallow-water equations is a proper simulation of the geostrophic adjustment process, in which the dispersion of the inertia-gravity waves leads to the establishment of a geostrophic balance. For most numerical methods the coupling between the momentum and continuity equations is a delicate problem and may lead to spurious solutions in the representation of inertia-gravity waves [4, 32, 30].

The spurious modes usually take the form of surface-elevation, velocity, and/or Coriolis modes. They are small-scale artifacts introduced by the spatial discretization scheme which do not propagate but are trapped within the model grid, leading to noisy solutions. Their appearance is encountered in both finite-difference and finite-element formulations and is mainly due to an inappropriate placement of variables on the grid and/or a bad choice of approximation function spaces. For example, subject to the no-slip boundary condition, the B-grid [31] usually exhibits surface-elevation modes and hence a checkerboard pattern of noise in both the gravity wave and inertial limits, while the C-grid (shown in Figure 3.1) is prone to Coriolis modes in the inertial limit only, provided the grid resolution is low relative to the Rossby radius of deformation [31]. The finite-element pairs $P_1 - P_1$ and $P_1 - P_0$ suffer from

*Received by the editors December 12, 2003; accepted for publication (in revised form) December 8, 2004; published electronically September 23, 2005. This work is supported by grants from the Natural Sciences and Engineering Research Council (NSERC), and FQRNT (Fonds Québécois de la Recherche sur la Nature et les Technologies).

http://www.siam.org/journals/sisc/27-2/60243.html

†Département de Mathématiques et de Statistique, Université Laval, Québec, QC, Canada G1K 7P4 (Daniel.Leroux@mat.ulaval.ca).

surface-elevation modes in a manner similar to the A- and B-finite-difference grids [30, 19, 21].

Various noise-suppression techniques have been introduced, such as numerical smoothing [6] and enhanced bottom friction [7]. It is, however, difficult to devise a filter or form of dissipation that removes significant high frequency oscillations while leaving the remainder of the spectrum intact. A better approach is to search for the cause of the problem and to find finite-difference and finite-element methods and variable placements that work well.

Attention has been focused almost exclusively on the spurious surface-elevation modes since these were argued to be the most troublesome [30]. This has resulted in several strategies using the following methods: mixed-order finite-element interpolation [29, 33, 28], equal-order elements with variables carried at sets of points staggered in space [9, 32, 16], analogous to staggered finite differences [2], vorticity and divergence instead of velocity in the momentum equations [25, 10], and wave-equation [22] and Petrov–Galerkin [17] formulations.

To circumvent the Coriolis modes of the C-grid in the inertial limit, due to the spatial averaging of the Coriolis terms, the CD-grid has been introduced [1]. It consists of augmenting the C-grid variables with D-grid velocity variables. The resulting grid is free of Coriolis modes and yields satisfying results at both high and coarse resolution [1].

The finite-element method appears to be a promising alternative to classical finite-difference schemes for ocean modelling. Indeed, triangular finite elements offer the enhanced flexibility of using grids of variable sizes, shapes, and orientation for representing the boundaries of complex domains and a natural treatment of boundary conditions. However, the finite-element method has not been widely used in ocean modelling to date. This is because the ocean modelling community has been using finite-difference techniques for a long time and did not invest significant efforts to develop finite-element approaches. It thus seems desirable to search for finite-element formulations free of surface-elevation and Coriolis modes, as for the CD-grid, and at the same time computationally competitive compared to finite-difference discretizations.

Dispersion analysis of discrete solutions to the linear shallow-water equations has been used as a tool to define the relationships between frequency and wave number and eventually to determine the presence of spurious modes. Linear equations have been considered because all the aforementioned problems occur primarily in the context of linear formulations. Two-dimensional dispersion relationships of finite-difference schemes [2] and finite-element discretizations using the same linear basis functions for all variables [3, 12, 30] have been mainly examined.

This paper presents a two-dimensional dispersion relation analysis of the discretized form of the linear inviscid shallow-water equations, using the $P_1^{NC} - P_1$ finite-element pair [16]. To our knowledge, this is the first study analyzing the dispersion relation of a finite-element scheme using different approximation spaces for the velocity and surface-elevation fields on meshes made up of triangles. This paper confirms the satisfying results obtained for the $P_1^{NC} - P_1$ pair in [21], where a linear algebra approach based on the study of the null space of the discrete operators (e.g., gradient, divergence, Coriolis) is employed, instead of the dispersion relation analysis used here, to determine the existence and the behavior of stationary spurious modes. Further, the present study may explain, in part, the accurate results obtained in [15] where the nonlinear shallow-water system is solved using the $P_1^{NC} - P_1$ pair.

The paper is developed as follows: The model equations and the time and Galerkin

finite-element discretization schemes are presented in section 2. The dispersion relation is derived and analyzed in section 3, and the results are compared with those of the C- and CD-grids. In section 4, the propagation and dispersion of fast surface gravity waves and the Stommel problem are simulated by using the $P_1^{NC} - P_1$ pair. In the latter case the results are again compared with those obtained with the C- and CD-finite-difference grids. Some concluding remarks complete the study.

2. Discretization of the inviscid linear shallow-water equations.

2.1. Governing equations. Let Ω be the model domain with boundary Γ. The inviscid linear shallow-water equations are expressed in Cartesian coordinates [18] as

$$\mathbf{u}_t + f\mathbf{k} \times \mathbf{u} + g\nabla\eta = 0, \tag{2.1}$$

$$\eta_t + H\nabla \cdot \mathbf{u} = 0, \tag{2.2}$$

where $\mathbf{u} = (u, v)$ is the horizontal velocity field, η is the surface elevation with respect to the reference level $z = 0$, f and g are the Coriolis parameter and gravitational acceleration, respectively; $\mathbf{k}$ is a unit vector in the vertical direction and the mean depth H is assumed constant. Note that η plays the role that pressure plays in the Navier–Stokes equations. For a contained flow, (2.1) and (2.2) are solved subject to the no-normal flow boundary condition

$$\mathbf{u} \cdot \mathbf{n} = 0 \quad \text{on } \Gamma, \tag{2.3}$$

where $\mathbf{n}$ is the outward pointing normal at the boundary.

2.2. Temporal discretization. For a given time step $\Delta t = t^{n+1} - t^n$ we introduce a general 2-level time discretization of (2.1)–(2.2) of the form

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + f\mathbf{k} \times (\gamma\mathbf{u}^{n+1} + [1-\gamma]\mathbf{u}^n) + g(\alpha\nabla\eta^{n+1} + [1-\alpha]\nabla\eta^n) = 0, \tag{2.4}$$

$$\frac{\eta^{n+1} - \eta^n}{\Delta t} + H(\alpha\nabla \cdot \mathbf{u}^{n+1} + [1-\alpha]\nabla \cdot \mathbf{u}^n) = 0, \tag{2.5}$$

where α and γ are real parameters, such that $0 \le \alpha \le 1$ and $0 \le \gamma \le 1$. Observe that the standard choices $\alpha = \gamma = 0, 1/2, 1$ yield the respective forward Euler, trapezoidal Crank–Nicolson, and backward Euler type schemes.

Equations (2.4)–(2.5) are now spatially discretized using the Galerkin finite-element method with the $P_1^{NC} - P_1$ pair.

2.3. Spatial discretization.

2.3.1. The weak formulation. The Sobolev space $H^1(\Omega)$ is the space of functions in the square-integrable space $L^2(\Omega)$, whose first derivatives belong to $L^2(\Omega)$. Let η be in a subspace V of $H^1(\Omega)$ and let each component of $\mathbf{u}$ be a sufficiently-regular scalar function such that $\mathbf{u} \cdot \mathbf{n} = 0$ on Γ. The weak formulation of (2.4) and (2.5) requires the test functions $\boldsymbol{\varphi}$ (whose x- or y-component is formally denoted by φ) and ψ to, respectively, belong to the same function space as $\mathbf{u}$ and η, such that

$$\begin{aligned} &\int_\Omega (\mathbf{u}^{n+1} - \mathbf{u}^n) \cdot \boldsymbol{\varphi}\, d\Omega + \Delta t \int_\Omega f(\mathbf{k} \times [\gamma\mathbf{u}^{n+1} + (1-\gamma)\mathbf{u}^n]) \cdot \boldsymbol{\varphi}\, d\Omega \\ &\quad + g\Delta t \int_\Omega (\alpha\nabla\eta^{n+1} + [1-\alpha]\nabla\eta^n) \cdot \boldsymbol{\varphi}\, d\Omega = 0, \end{aligned} \tag{2.6}$$

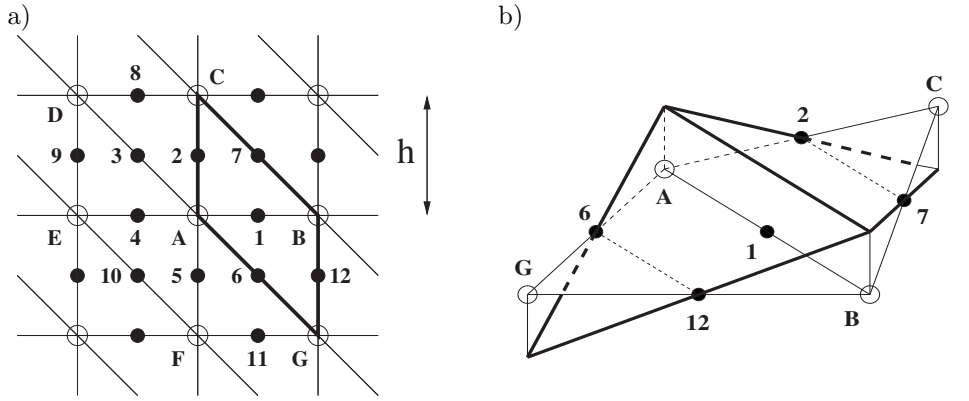


FIG. 2.1. a) *Velocity and surface-elevation node locations are represented by the symbols* ● *and* ○*, respectively, for the* $P_1^{NC} - P_1$ *element pair. The support of the velocity basis function at node* 1 *is in bold.* b) *the* P_1^{NC} *linear basis function at node* 1 *in bold, it takes the values* 1 *at node* 1 *and along the triangle side* AB*,* 0 *at velocity nodes* $2, 6, 7,$ *and* 12*, and* -1 *at surface-elevation nodes* C *and* G*.*

$$\int_\Omega (\eta^{n+1} - \eta^n)\psi \, d\Omega + H\Delta t \int_\Omega (\alpha \nabla \cdot \mathbf{u}^{n+1} + [1-\alpha]\nabla \cdot \mathbf{u}^n)\psi \, d\Omega = 0, \tag{2.7}$$

where $d\Omega$ is the area element.

To avoid computing u and v derivatives, the terms $\nabla \cdot \mathbf{u}^{n+1}$ and $\nabla \cdot \mathbf{u}^n$ appearing in (2.7) are integrated by parts using Green's theorem. In this way, only ψ derivatives are required and no φ derivatives. The former are well defined since ψ belongs to V. The boundary integrals vanish by applying (2.3), and (2.7) is rewritten as

$$\int_\Omega (\eta^{n+1} - \eta^n)\psi \, d\Omega - H\Delta t \int_\Omega (\alpha \mathbf{u}^{n+1} + [1-\alpha]\mathbf{u}^n) \cdot \nabla\psi \, d\Omega = 0. \tag{2.8}$$

2.3.2. The $P_1^{NC} - P_1$ finite-element pair. Conventional finite-element terminology is adopted and the nomenclature $P_m - P_n$ means that velocity components and surface elevation are, respectively, represented as piecewise-defined polynomials of degree m and n. The $P_1^{NC} - P_1$ pair thus approximates velocity components and surface elevation using piecewise linear basis functions on a mesh of triangles.

The P_1^{NC} finite element has velocity nodes at triangle midedge points, and those are represented by the symbol ● in Figure 2.1. The shape of the linear basis function that is used to approximate the velocity components u and v on the element's two-triangle support is given in bold at node 1 in Figure 2.1b. The basis function is zero outside the support triangles ABC and AGB and discontinuous along sides AC, AG, BC, and BG. It takes the values 1 at node 1 and along the triangle side AB, 0 at velocity nodes $2, 6, 7,$ and 12, and -1 at surface-elevation nodes C and G. Since this particular representation of velocity is only continuous across triangle boundaries at midedge points, and discontinuous everywhere else around a triangle boundary, this element is termed nonconforming (NC) in the finite-element literature.

Standard piecewise linear continuous basis functions (P_1) are used to approximate the surface elevation at triangle vertices (represented by the symbol ○ in Figure 2.1).

2.3.3. Galerkin finite-element discretization. The Galerkin method approximates the solution of (2.6) and (2.8) in a finite-dimensional subspace. Consider a finite-element triangulation $\mathcal{T}_h$, of the polygonal domain Ω, where h is a representative meshlength parameter that measures resolution. For triangle $K \in \mathcal{T}_h$, let $P_1(K)$ denote the space of linear polynomials on K.

The discrete solution $\mathbf{u}_h$ sought belongs to a finite-dimensional space $\mathbf{W}_h$ defined to be the set of functions $\mathbf{u}_h$ whose restriction on K belongs to $P_1(K) \times P_1(K)$, with $\mathbf{u}_h$ being continuous only at the midpoints of each face of $\mathcal{T}_h$, and $\mathbf{u}_h \cdot \mathbf{n} = 0$ on Γ.

Each of the two components of the discrete velocity vector $\mathbf{u}_h$ is represented over a triangle K_i by a linear interpolating function of the form $\varphi(x, y) = a + bx + cy$. Let the vector $\mathbf{u}_h$ be expanded over K_i using the linear basis functions φ_p, with $\boldsymbol{\varphi}_p$ belonging to $\mathbf{W}_h$, so that

$$\mathbf{u}_h = \sum_{p \in I_{K_i}} \mathbf{u}_p \varphi_p + \sum_{p \in B_{K_i}} \sigma_p \mathbf{t}_p \varphi_p, \tag{2.9}$$

where p represents the midside nodes of K_i. In (2.9), I_{K_i} and B_{K_i} denote the set of midside nodes of K_i belonging to the interior of Ω and to the boundary Γ, respectively. At a boundary node, because of the no-normal flow boundary condition, there is only one degree of freedom per boundary velocity node, denoted by $\sigma_p = \mathbf{u}_p \cdot \mathbf{t}_p$, where $\mathbf{t}_p$ (whose x and y components are denoted by t_p^x and t_p^y, respectively) designates the local tangential vector of the boundary.

Since velocity nodes are located at triangle midedge points, the direction of the local tangential vector $\mathbf{t}_p$ is uniquely defined along the boundary. This crucial property circumvents the difficulty of satisfying exactly the no-normal flow condition at boundary vertices [11, 14, 23], a problem encountered with the usual conforming elements. It is important to satisfy as exactly as possible the boundary condition since for many applications shallow-water solutions are strongly trapped near the boundary.

A similar linear approximation is used for the Coriolis term, rather than the quadratic expansion that usually arises from the product of $\mathbf{u}_h$ with f. This approach simplifies the algebra while having a negligible impact on the accuracy of the solution, since f varies very smoothly, and so over triangle K_i we have

$$f\mathbf{u}_h = \sum_{p \in I_{K_i}} f_p \mathbf{u}_p \varphi_p + \sum_{p \in B_{K_i}} f_p \sigma_p \mathbf{t}_p \varphi_p. \tag{2.10}$$

The discrete solution η_h is sought in a finite-dimensional subspace V_h of V, where V_h is defined to be the set of functions η_h whose restriction on K belongs to $P_1(K)$, with η_h being continuous at each vertex of $\mathcal{T}_h$. Hence, η_h is represented over a triangle K_i by a linear interpolating function $\psi(x, y)$. The surface elevation is thus expanded over triangle K_i using the linear basis functions ψ_s of V_h, so that

$$\eta_h = \sum_{s \in K_i} \eta_s \psi_s, \tag{2.11}$$

where the sum is over the three vertices of K_i.

The expansions of $\mathbf{u}_h$, $f\mathbf{u}_h$, and η_h over the whole domain Ω are then obtained by summing (2.9), (2.10), and (2.11) over all triangles K_i of the triangulation $\mathcal{T}_h$.

The basis functions φ_p and ψ_s are linked over K_i by the relation $\varphi_p = 1 - 2\psi_s$ if p is the midedge point of K_i facing the vertex s of K_i. For example, $\varphi_1 = 1 - 2\psi_C$ over triangle (A, B, C) of Figure 2.1.

Introducing the finite-element basis leads to a finite-element statement as in (2.6) and (2.8) but with $\mathbf{u}$, η replaced by the finite element trial functions $\mathbf{u}_h$, η_h and φ, ψ replaced by the corresponding finite-element test functions. We then decompose the integrals in (2.6) and (2.8) into triangle contributions.

At a velocity node q, for $\boldsymbol{\varphi} = \boldsymbol{\varphi}_q = (\varphi_q, 0)$, (2.6) leads to

$$\begin{aligned}
&\sum_{K_i} \sum_{p \in I_{K_i}} ([u_p^{n+1} - u_p^n] - \Delta t f_p [\gamma v_p^{n+1} + (1-\gamma) v_p^n]) \int_{K_i} \varphi_p \varphi_q \, d\Omega \\
&+ \sum_{K_i} \sum_{p \in B_{K_i}} (t_p^x [\sigma_p^{n+1} - \sigma_p^n] - \Delta t f_p t_p^y [\gamma \sigma_p^{n+1} + (1-\gamma)\sigma_p^n]) \int_{K_i} \varphi_p \varphi_q \, d\Omega \\
&+ g \Delta t \sum_{K_i} \sum_{s \in K_i} (\alpha \eta_s^{n+1} + [1-\alpha] \eta_s^n) \int_{K_i} \frac{\partial \psi_s}{\partial x} \varphi_q \, d\Omega = 0,
\end{aligned} \tag{2.12}$$

and similarly, for $\boldsymbol{\varphi}_q = (0, \varphi_q)$, we obtain

$$\begin{aligned}
&\sum_{K_i} \sum_{p \in I_{K_i}} ([v_p^{n+1} - v_p^n] + \Delta t f_p [\gamma u_p^{n+1} + (1-\gamma) u_p^n]) \int_{K_i} \varphi_p \varphi_q \, d\Omega \\
&+ \sum_{K_i} \sum_{p \in B_{K_i}} (t_p^y [\sigma_p^{n+1} - \sigma_p^n] + \Delta t f_p t_p^x [\gamma \sigma_p^{n+1} + (1-\gamma)\sigma_p^n]) \int_{K_i} \varphi_p \varphi_q \, d\Omega \\
&+ g \Delta t \sum_{K_i} \sum_{s \in K_i} (\alpha \eta_s^{n+1} + [1-\alpha] \eta_s^n) \int_{K_i} \frac{\partial \psi_s}{\partial y} \varphi_q \, d\Omega = 0.
\end{aligned} \tag{2.13}$$

At a surface-elevation node r, for $\psi = \psi_r$, (2.8) becomes

$$\begin{aligned}
&\sum_{K_i} \sum_{s \in K_i} (\eta_s^{n+1} - \eta_s^n) \int_{K_i} \psi_s \psi_r \, d\Omega - H \Delta t \\
&\quad \times \left(\sum_{K_i} \sum_{p \in I_{K_i}} (\alpha u_p^{n+1} + [1-\alpha] u_p^n) \int_{K_i} \frac{\partial \psi_r}{\partial x} \varphi_p \, d\Omega \right. \\
&\quad + \sum_{K_i} \sum_{p \in I_{K_i}} (\alpha v_p^{n+1} + [1-\alpha] v_p^n) \int_{K_i} \frac{\partial \psi_r}{\partial y} \varphi_p \, d\Omega \\
&\quad + \sum_{K_i} \sum_{p \in B_{K_i}} (\alpha \sigma_p^{n+1} + [1-\alpha] \sigma_p^n) \\
&\quad \left. \times \left(t_p^x \int_{K_i} \frac{\partial \psi_r}{\partial x} \varphi_p \, d\Omega + t_p^y \int_{K_i} \frac{\partial \psi_r}{\partial y} \varphi_p \, d\Omega \right) \right) = 0.
\end{aligned} \tag{2.14}$$

The orthogonality property of the NC velocity basis functions [27] leads to

$$\int_\Omega \varphi_p \varphi_q \, d\Omega = \frac{A_q}{3}\delta_{pq}, \tag{2.15}$$

where A_q is the area of the support of φ_q and δ_{pq} is the Kronecker delta. The velocity and Coriolis mass matrices in (2.12) and (2.13) are thus diagonal, a desirable and unusual property of the finite-element method that greatly enhances computational efficiency. Further, the x- and y-derivatives of the piecewise-linear basis functions ψ involved in (2.12)–(2.14) are constant over each triangle K_i of $\mathcal{T}_h$ and may therefore be written out of the integral symbols. Finally, another property of the NC velocity basis functions

$$\int_\Omega \varphi_q \, d\Omega = \frac{A_q}{3}, \tag{2.16}$$

simplifes the calculation of the integrals in (2.12)–(2.14).

For the purposes of the following analysis, it suffices that we consider a uniform mesh made up of biased right triangles as in Figure 2.1a, and h is thus taken as a constant in the x- and y-directions. Midedge velocity nodes may be located on three possible types of faces, i.e., horizontal, vertical, and diagonal, and for each type the corresponding nodes are distributed on a regular grid of size h. Selected discrete pairs of equations are then obtained from (2.12) and (2.13) at the velocity nodes $q = 1, 2, 3$, of Figure 2.1a:

$$u_1^t - f v_1^\gamma + \frac{g}{h}(\eta_B^\alpha - \eta_A^\alpha) = 0, \tag{2.17}$$

$$v_1^t + f u_1^\gamma + \frac{g}{2h}(\eta_B^\alpha + \eta_C^\alpha - \eta_A^\alpha - \eta_G^\alpha) = 0, \tag{2.18}$$

$$u_2^t - f v_2^\gamma + \frac{g}{2h}(\eta_B^\alpha + \eta_C^\alpha - \eta_A^\alpha - \eta_D^\alpha) = 0, \tag{2.19}$$

$$v_2^t + f u_2^\gamma + \frac{g}{h}(\eta_C^\alpha - \eta_A^\alpha) = 0, \tag{2.20}$$

$$u_3^t - f v_3^\gamma + \frac{g}{2h}(\eta_A^\alpha + \eta_C^\alpha - \eta_D^\alpha - \eta_E^\alpha) = 0, \tag{2.21}$$

$$v_3^t + f u_3^\gamma + \frac{g}{2h}(\eta_C^\alpha + \eta_D^\alpha - \eta_A^\alpha - \eta_E^\alpha) = 0, \tag{2.22}$$

respectively, while at the surface-elevation node $r = A$ of Figure 2.1a (2.14) leads to

$$\begin{aligned} &6\eta_A^t + \eta_B^t + \eta_C^t + \eta_D^t + \eta_E^t + \eta_F^t + \eta_G^t \\ &\quad + \frac{2H}{h}(2u_1^\alpha + u_2^\alpha + u_7^\alpha + u_6^\alpha + u_{12}^\alpha - 2u_4^\alpha - u_3^\alpha - u_9^\alpha - u_5^\alpha - u_{10}^\alpha \\ &\quad + 2v_2^\alpha + v_3^\alpha + v_8^\alpha + v_1^\alpha + v_7^\alpha - 2v_5^\alpha - v_4^\alpha - v_{10}^\alpha - v_6^\alpha - v_{11}^\alpha) = 0, \end{aligned} \tag{2.23}$$

where

$$\Phi_i^t = \frac{\Phi_i^{n+1} - \Phi_i^n}{\Delta t}, \qquad \Phi_i^\gamma = \gamma\Phi_i^{n+1} + (1-\gamma)\Phi_i^n, \qquad \Phi_i^\alpha = \alpha\Phi_i^{n+1} + (1-\alpha)\Phi_i^n,$$

and $\Phi = u, v$, or η.

The contribution of the time derivative in (2.2) leads, at least theoretically, to a problem that is always stable without an inf-sup (or LBB) condition [13, 8]. However, an important result obtained in [24], in the context of the incompressible Navier–Stokes equations, is generalized in [30] to include the shallow-water and the compressible Navier–Stokes equations by examining their numerical dispersion relations. It is found that spurious solutions may arise from the coupling of the momentum and continuity equations. The spurious modes are small-scale artifacts introduced by the spatial discretization scheme which do not propagate but are trapped within the model grid.

In [30] attention is focused almost exclusively on the spurious pressure or surface-elevation modes, since these are argued to be the most troublesome. For these modes the velocity field is zero and nonconstant pressure functions lie in the null space of the discrete gradient operator. Solution uniqueness is then lost since any multiple of a spurious mode can be added to any solution of the discrete equations and still satisfy them. The occurrence of such spurious modes has been observed in a variety of finite-difference [31] and finite-element [19, 20, 21, 30] approximations to the shallow-water equations.

By setting the velocity field to zero in (2.17)–(2.22) we obtain $\eta_A = \eta_B = \eta_C$ from (2.17) and (2.20), $\eta_C = \eta_G = \eta_D$ from (2.18) and (2.19), and finally $\eta_C = \eta_E$ from (2.21) or (2.22). By taking into account the continuity of η, we deduce that the null space of the discrete gradient operator is one-dimensional. This proves that the $P_1^{NC} - P_1$ finite-element pair is free of spurious pressure modes.

3. Study of the dispersion relation. In the continuum case the free modes of (2.1)–(2.2) are examined by perturbing about the basic sate $u = v = \eta = 0$. We seek solutions of (2.1)–(2.2) of the form $(u, v, \eta) = (\tilde{u}, \tilde{v}, \tilde{\eta})e^{i(kx+ly+\omega t)}$, where k and l are the wave numbers in the x- and y-directions, respectively, and ω is the angular frequency. Substitution into (2.1)–(2.2) then leads to a square matrix system for the Fourier amplitudes $\tilde{u}$, $\tilde{v}$, $\tilde{\eta}$. For a nontrivial solution to exist, the determinant of the matrix must equal zero, and this constraint leads to the following dispersion relation for the frequency:

$$\omega(\omega^2 - f^2 - gH(k^2 + l^2)) = 0. \tag{3.1}$$

The first solution $\omega = 0$ is the geostrophic mode and would correspond to the slow Rossby mode on a β-plane, while the other two solutions $\omega = \pm(f^2 + gH(k^2 + l^2))^{1/2}$ correspond to the free-surface gravitational modes with rotational correction. Since ω is purely real, all modes are neutrally stable and neither amplify nor decay.

In the discrete case, solutions of (2.17)–(2.23) corresponding to $(u, v, \eta) = (\tilde{u}, \tilde{v}, \tilde{\eta})$ $e^{i(kx+ly+\omega t)}$ are sought. Setting $E = e^{i\omega\Delta t}$ and substituting in (2.17)–(2.23) we obtain

$$(E-1)\tilde{u}_q - f\Delta t(E\gamma + 1 - \gamma)\tilde{v}_q + ig\Delta t(E\alpha + 1 - \alpha)a_{q,1}\tilde{\eta}_A = 0, \tag{3.2}$$

$$(E-1)\tilde{v}_q + f\Delta t(E\gamma + 1 - \gamma)\tilde{u}_q + ig\Delta t(E\alpha + 1 - \alpha)a_{q,2}\tilde{\eta}_A = 0, \tag{3.3}$$

$$(E-1)a\tilde{\eta}_A + iH\Delta t(E\alpha + 1 - \alpha)\sum_{q=1}^{3}(a_{q,1}\tilde{u}_q + a_{q,2}\tilde{v}_q) = 0, \tag{3.4}$$

for $q = 1, 2, 3$ in (3.2) and (3.3), with

$$a_{1,1} = \frac{2}{h} \sin \frac{kh}{2}, \qquad a_{1,2} = \frac{2}{h} \sin \frac{lh}{2} \cos \frac{(k-l)h}{2},$$

$$a_{2,1} = \frac{2}{h} \sin \frac{kh}{2} \cos \frac{(k-l)h}{2}, \qquad a_{2,2} = \frac{2}{h} \sin \frac{lh}{2},$$

$$a_{3,1} = \frac{2}{h} \sin \frac{kh}{2} \cos \frac{lh}{2}, \qquad a_{3,2} = \frac{2}{h} \sin \frac{lh}{2} \cos \frac{kh}{2},$$

$$a = 1 + 2 \cos \frac{kh}{2} \cos \frac{lh}{2} \cos \frac{(k-l)h}{2} \quad \text{with } a \geq \frac{3}{4} \text{ for all } k \text{ and } l.$$

A nontrivial solution $(\tilde{u}_1, \tilde{v}_1, \tilde{u}_2, \tilde{v}_2, \tilde{u}_3, \tilde{v}_3, \tilde{\eta}_A)$ exists, if the 7×7 determinant of the square matrix system (3.2)–(3.4) vanishes, which requires solving a polynomial of degree seven in E. This leads to

$$E_1 = 1, \tag{3.5}$$

$$E_{2,3} = 1 - \frac{\Delta t^2 \Big(f^2 \gamma + \frac{gH}{a} \alpha \sum_{q=1}^{3} (a_{q,1}^2 + a_{q,2}^2) \Big)}{1 + \Delta t^2 \Big(f^2 \gamma^2 + \frac{gH}{a} \alpha^2 \sum_{q=1}^{3} (a_{q,1}^2 + a_{q,2}^2) \Big)} \tag{3.6}$$

$$\pm i \frac{\Delta t \sqrt{f^2 + \frac{gH}{a} (1 + f^2 \Delta t^2 (\alpha - \gamma)^2) \sum_{q=1}^{3} (a_{q,1}^2 + a_{q,2}^2)}}{1 + \Delta t^2 \Big(f^2 \gamma^2 + \frac{gH}{a} \alpha^2 \sum_{q=1}^{3} (a_{q,1}^2 + a_{q,2}^2) \Big)},$$

$$E_{4,5} = 1 - \frac{f^2 \Delta t^2 \gamma}{1 + f^2 \Delta t^2 \gamma^2} \pm i \frac{f \Delta t}{1 + f^2 \Delta t^2 \gamma^2} \quad \text{double roots}, \tag{3.7}$$

where $E_j = e^{i \omega_j \Delta t}$, $j = 1, 2, \ldots, 5$.

The first mode E_1 in (3.5) is the geostrophic mode, on a β-plane it would be a planetary Rossby wave. The modes $E_{2,3}$ in (3.6) are inertia-gravity waves and the modes $E_{4,5}$ in (3.7) are computational modes.

The latter modes, as for the CD-grid [1], take the form of propagating inertial oscillations and have no particular spatial characteristics; they are a consequence of using three times more velocity nodes than surface-elevation nodes. For the computational modes to be stable ($|E_{4,5}| \leq 1$), it is sufficient that $\gamma \geq 1/2$. Following [1], the computational modes can effectively be filtered by using an implicit backward scheme with $\gamma = 1$. As shown in [1], for all the timescales of interest and relevance to large-scale oceanography, inertial oscillations will, in practice, be fully filtered. Implicit dissipation would dissipate energy from the balanced state, whereas here, only the transients about the balance state are damped. Indeed, implicit filtering extracts energy only from inertial oscillations and faithfully sustains the balance state. Furthermore, this procedure applies equally well to regular as well as unstructured meshes with irregular geometry.

TABLE 3.1

Dispersion relations for various schemes: AN, analytical (continuum); FE, the finite-element pair $P_1^{NC} - P_1$; CD, the CD-grid, and C, the C-grid, shown in Figure 3.1. In each case, the normalized frequency squared $(\omega / f)^2$ has been written as the sum of its inertia and gravity parts.

		Inertia part		Gravity part
AN	$\left(\frac{\omega}{f}\right)^2 =$	1	$+$	$\left(\frac{\lambda}{h}\right)^2 ([kh]^2 + [lh]^2)$
FE	$\left(\frac{\omega}{f}\right)^2 =$	1	$+$	$4\left(\frac{\lambda}{h}\right)^2 \left(\sin^2\frac{kh}{2} + \sin^2\frac{lh}{2} + \frac{1}{a}\left[\sin^4\frac{kh}{2} + \sin^4\frac{lh}{2}\right]\right)$
CD	$\left(\frac{\omega}{f}\right)^2 =$	1	$+$	$4\left(\frac{\lambda}{h}\right)^2 \left(\sin^2\frac{kh}{2} + \sin^2\frac{lh}{2}\right)$
C	$\left(\frac{\omega}{f}\right)^2 =$	$\cos^2\frac{kh}{2}\cos^2\frac{lh}{2}$	$+$	$4\left(\frac{\lambda}{h}\right)^2 \left(\sin^2\frac{kh}{2} + \sin^2\frac{lh}{2}\right)$

In the limit as time step $\Delta t \to 0$ and mesh spacing $h \to 0$, we obtain $a_{q,1} = k$ and $a_{q,2} = l$ for $q = 1, 2, 3$, and $a = 3$. Hence, the discrete inertia-gravity modes (3.6) asymptote to the more familiar continuum modes $\omega_{2,3} = \pm(f^2 + gH(k^2 + l^2))^{1/2}$, while the frequencies corresponding to the computational modes give $\omega_{4,5} = \pm f$.

In the limit as time step $\Delta t \to 0$ only, we still have $\omega_{4,5} = \pm f$, while (3.6) gives

$$\left(\frac{\omega_{2,3}}{f}\right)^2 = 1 + \frac{gH}{f^2}\frac{1}{a}\sum_{q=1}^{3}(a_{q,1}^2 + a_{q,2}^2), \tag{3.8}$$

for the inertia-gravity modes. After some long and tedious algebra (3.8) reduces to

$$\left(\frac{\omega_{2,3}}{f}\right)^2 = 1 + 4\left(\frac{\lambda}{h}\right)^2 \left(\sin^2\frac{kh}{2} + \sin^2\frac{lh}{2} + \frac{1}{a}\left[\sin^4\frac{kh}{2} + \sin^4\frac{lh}{2}\right]\right), \tag{3.9}$$

where $\lambda \equiv \sqrt{gH} / f$ is the radius of deformation (Rossby radius) and we recall that $a \geq 3/4$ for all k and l.

Table 3.1 compares the dispersion relation obtained in (3.9) for the $P_1^{NC} - P_1$ pair with those derived for the continuous case and the CD- and D-grids. Observe that the dispersion relations obtained for the finite-element pair and the CD-grid only differ from one another in the representation of their gravity parts, due to the presence of the extra term in brackets [ ] in (3.9). Furthermore, the inertia part of the dispersion relation for the C-grid (resulting from the spatial averaging of the Coriolis terms) vanishes in the case $kh = lh = \pi$ for small values of λ / h, and spurious Coriolis modes may thus exist. Other finite-difference schemes are not mentioned here as they generate spurious surface-elevation modes.

The nondimensional frequency (ω / f) corresponding to the discrete inertia-gravity modes is plotted in Figure 3.1 as a function of kh and lh. Two cases are considered, namely $\lambda / h = 2$ (high resolution) and $\lambda / h = 0.1$ (coarse resolution), and the results for the $P_1^{NC} - P_1$ pair are compared with those obtained for the continuous case and for the CD- and C-finite-difference grids. When $\lambda / h = 2$ and $\lambda / h = 0.1$, the nondimensional frequency (ω / f) is monotonic in k and l, as in the continuous case, for both the $P_1^{NC} - P_1$ pair and the CD-grid. However, the maximum value of ω / f is much closer to the continuous case for the $P_1^{NC} - P_1$ pair than

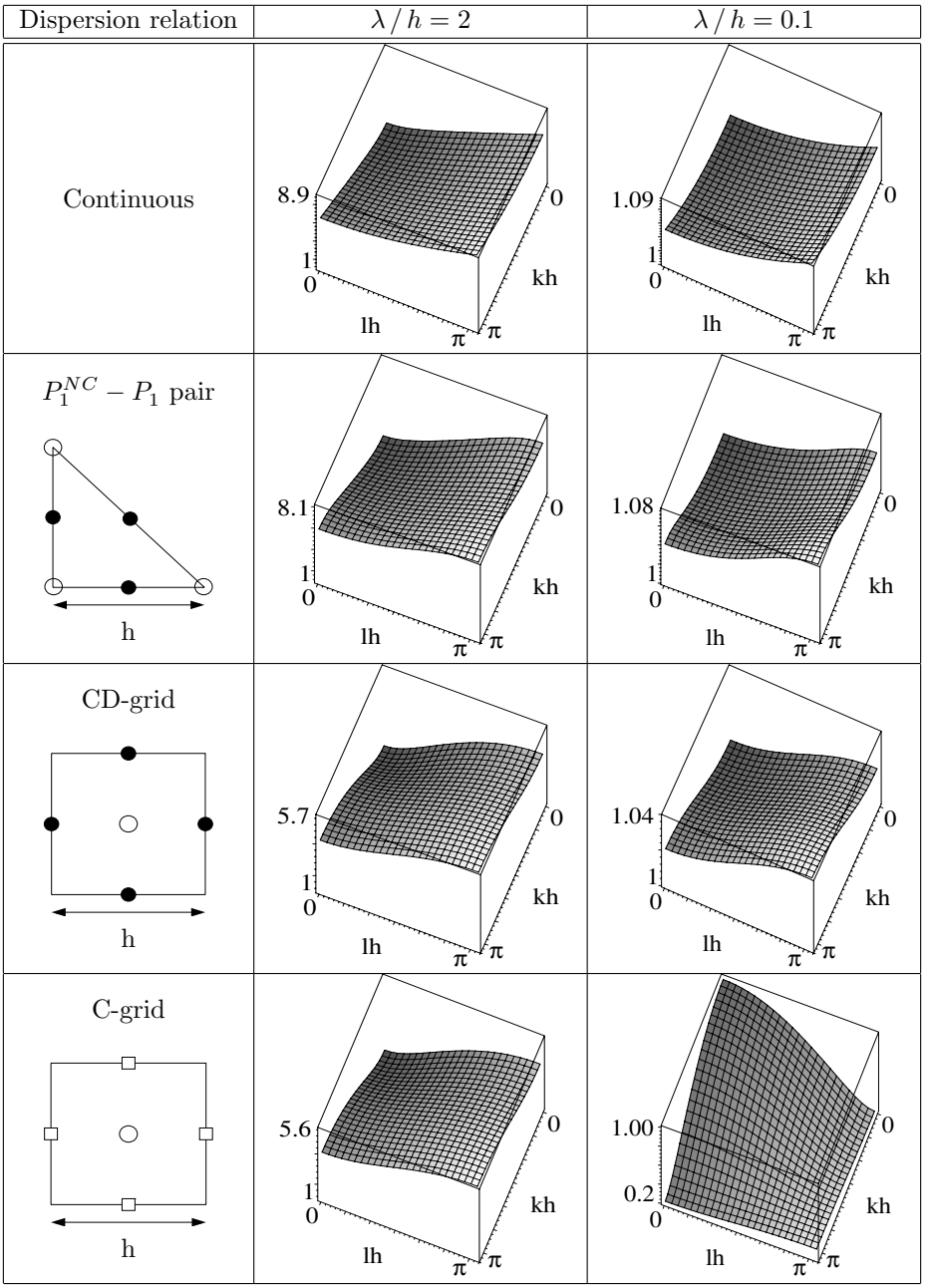


FIG. 3.1. *Dispersion relation results for several grids and* $\lambda / h = 2$ *(high resolution) and* $\lambda / h = 0.1$ *(coarse resolution). The symbols* ● *and* ○ *indicate velocity and surface-elevation nodes, respectively, while the symbol* □ *indicates normal velocity nodes. The horizontal coordinates in the plots are* kh *and* lh*, respectively.*

for the CD-grid, and this is due to the presence of the extra term in brackets [ ] in (3.9). For the C-grid, the presence of the Coriolis mode is clearly shown at low resolution ($\lambda / h = 0.1$) when $kh = lh = \pi$, while at high resolution the nondimensional frequency is monotonic in k and l, yielding accurate results. A cross section ($k = l$)

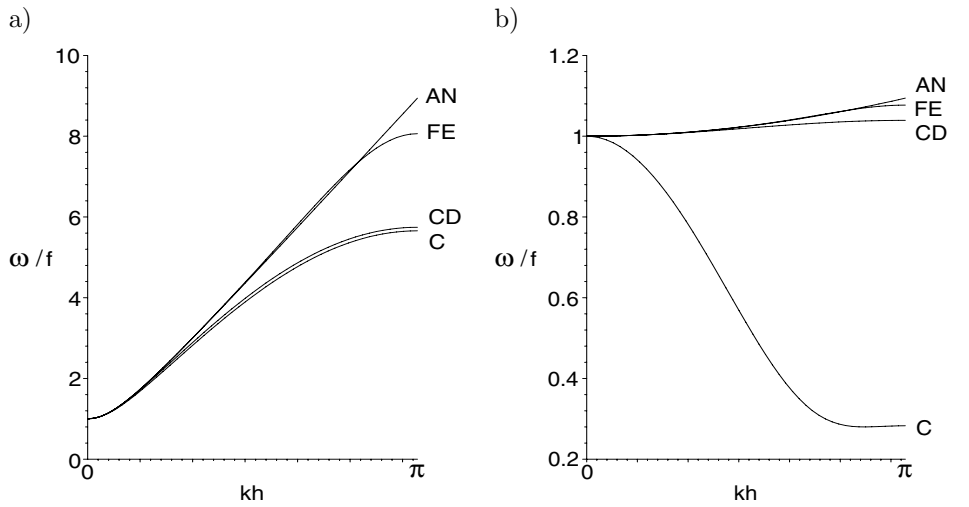


FIG. 3.2. *Cross section ($k = l$) of the numerical results shown in Figure* 3.1 *in the case* a) $\lambda / h = 2$ *and* b) $\lambda / h = 0.1$ *for various schemes: AN, analytical (continuum); FE, the finite-element pair ($P_1^{NC} - P_1$); and the CD- and C-grids. The vertical coordinate is the normalized frequency ω / f.*

of the numerical results shown in Figure 3.1 is presented in Figure 3.2 in the cases $\lambda / h = 2$ and $\lambda / h = 0.1$.

The computational cost of the $P_1^{NC} - P_1$ scheme will be discussed in the next section and we shall see that this finite-element pair defines a reasonably cheap approximation of the shallow-water system.

4. Solving the discrete shallow-water system. Equations (2.12)–(2.14) lead to a set of linear equations of the form

$$\mathsf{M}^{\mathbf{u}}\mathbf{u}_p^{n+1} + \alpha g \Delta t \mathsf{G} \eta_s^{n+1} = \mathsf{M}^{\mathbf{u}}\mathbf{u}_p^n - (1-\alpha) g \Delta t \mathsf{G} \eta_s^n \equiv \mathsf{R}^{\mathbf{u}}, \tag{4.1}$$

$$\mathsf{M}^{\eta}\eta_s^{n+1} - \alpha H \Delta t \mathsf{G}^t \mathbf{u}_p^{n+1} = \mathsf{M}^{\eta}\eta_s^n + (1-\alpha) H \Delta t \mathsf{G}^t \mathbf{u}_p^n \equiv \mathsf{R}^{\eta}, \tag{4.2}$$

for $1 \le p \le N_{\mathbf{u}}$ and $1 \le s \le N_\eta$, where $N_{\mathbf{u}}$ and N_η are the number of velocity and surface-elevation unknowns, respectively. In (4.1) and (4.2), $\mathsf{M}^{\mathbf{u}}$ and M^{η} denote the velocity (including the Coriolis part) and surface-elevation mass matrices, respectively, G is the gradient matrix, and $\mathsf{R}^{\mathbf{u}}$ and R^{η} are the right-hand sides.

Due to the orthogonality property of the velocity basis functions [27], $\mathsf{M}^{\mathbf{u}}$ is a diagonal matrix with 2×2 blocks along the diagonal. Equation (4.1) is rewritten as

$$\mathbf{u}_p^{n+1} = -\alpha g \Delta t (\mathsf{M}^{\mathbf{u}})^{-1} \mathsf{G} \eta_s^{n+1} + (\mathsf{M}^{\mathbf{u}})^{-1} \mathsf{R}^{\mathbf{u}}, \tag{4.3}$$

and (4.3) is then used to eliminate $\mathbf{u}_p^{n+1}$, $1 \le p \le N_{\mathbf{u}}$, from (4.2). The substitution results in

$$(\mathsf{M}^{\eta} + \alpha^2 g H \Delta t^2 \mathsf{G}^t (\mathsf{M}^{\mathbf{u}})^{-1} \mathsf{G}) \eta_s^{n+1} = \mathsf{R}^{\eta} + \alpha H \Delta t \mathsf{G}^t (\mathsf{M}^{\mathbf{u}})^{-1} \mathsf{R}^{\mathbf{u}}. \tag{4.4}$$

The elimination of $\mathbf{u}_p^{n+1}$ thus leads to N_η discrete Helmholtz equations in η, and hence to solve for a linear system $N_\eta \times N_\eta$ for the surface-elevation only, where N_η is the

number of vertices of $\mathcal{T}_h$, which greatly enhances computational efficiency. The matrix on the left-hand side (4.4) is a very sparse matrix, with an average of 13 nonzero elements per row. Finally, once η is obtained from (4.4) $\mathbf{u}$ is computed explicitly from (4.3).

For most finite-element discretizations $\mathsf{M}^{\mathbf{u}}$ is not diagonal, and hence the substitution procedure leading to (4.4) likely involves a full $(\mathsf{M}^{\mathbf{u}})^{-1}$ matrix, even if $\mathsf{M}^{\mathbf{u}}$ is sparse. The system (4.4) is then too large to be stored and handled. The orthogonality property of the P_1^{NC} basis functions, leading to the diagonal matrix $\mathsf{M}^{\mathbf{u}}$, implies a tremendous savings in the computational cost since a system of only N_η equations is solved instead of $N_{\mathbf{u}}+N_\eta$, i.e., approximately $7N_\eta$ equations. In the case of the popular P_1 iso P_2-P_1 finite-element pair [5], which also approximates velocity components and surface elevation using piecewise linear basis functions, we would have to solve a linear system of approximately $9N_\eta$ equations. Finally, the use of the CD-grid, in its semi-implicit formulation, leads to N_η discrete Helmholtz equations of the form (4.4), as the matrix $\mathsf{M}^{\mathbf{u}}$ is "naturally" diagonal, and the Helmholtz matrix has an average of 9 nonzero elements per row. By taking advantage of the increased flexibility of unstructured meshes, the computational cost of the $P_1^{NC}-P_1$ and CD schemes for solving (4.4) should thus be comparable in the semi-implicit case.

5. Numerical results. The results of two tests using the $P_1^{NC}-P_1$ element pair are now presented. The first test examines the propagation and dispersion of fast surface gravity waves in a circular basin and their reflection at the lateral boundary. In the second test, the Stommel problem is computed and the results are compared with those obtained with the C- and CD-finite-difference grids.

5.1. Gravity wave propagation and dispersion. For many applications very little energy is carried by the small-amplitude fast-moving surface gravity waves, and this justifies slowing them down via a semi-implicit time discretization. Nevertheless, when an accurate representation of the fast gravity modes is important, the numerical solution of (2.1)–(2.3), at small values of the gravitational Courant number C_g, should be expected to reasonably well approximate the analytical one in the semi-implicit case. We then use a small value for C_g and $\alpha = 1/2$ in the present test.

The analytical solution of the problem is obtained from (2.1) and (2.2) by exploiting the circular symmetry. In order to do so, the Coriolis term is set to zero, leading to a second-order wave equation in polar coordinates for η, viz

$$\eta_{tt} - c^2 \frac{1}{r}(r\eta_r)_r = 0 \tag{5.1}$$

subject to the boundary conditions $\eta_r|_{r=R} = 0$ and no singularity at the origin, where $c \equiv \sqrt{gH}$ is the phase speed of the surface gravity waves. The initial conditions are $\eta(r,0) = ae^{-br^2}$ and $\eta_t(r,0) = 0$; the latter condition follows from setting the initial velocity to zero. The exact solution of (5.1) is the Bessel-function expansion

$$\eta(r,t) = \sum_{n=1}^{\infty} \mu_n J_0(\sqrt{\lambda_n} r) \cos c\sqrt{\lambda_n} t,$$

where $\mu_n = \dfrac{\int_0^R \eta(r,0) J_0(\sqrt{\lambda_n} r) r\, dr}{\int_0^R J_0^2(\sqrt{\lambda_n} r) r\, dr}$ and λ_n are the roots of $J_1(\sqrt{\lambda_n} R) = 0$.

The circular domain has a radius of $R = 1000$ km and is discretized using a 25 km surface-elevation node spacing unstructured triangulation $\mathcal{T}_h$, as shown in Figure 5.1a.

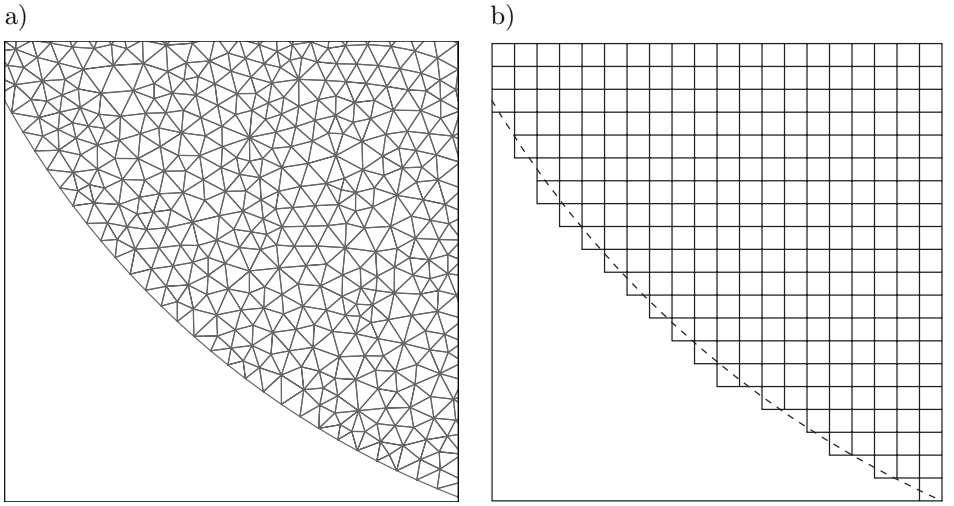


FIG. 5.1. *A window (close to the boundary) of the circular domain of radius* $R = 1000$ *km, discretized using a* 25 *km vertex node spacing in the case of* a) *an unstructured triangulation,* b) *a uniform grid (the dash line corresponds to the domain boundary).*

The corresponding uniform grid, shown in Figure 5.1b, crudely represents the domain boundary by a series of artificial "steps." It is therefore not appropriate for evaluating the performance of the CD scheme in the experiment conducted here. Consequently, only the $P_1^{NC} - P_1$ pair is used in this test.

A flat bottom with mean depth $H = 2000\,\mathrm{m}$ is assumed, leading to $c \approx 140\,\mathrm{m\ s^{-1}}$. By taking a timestep of 20 s, C_g is approximately 0.2. The Gaussian distribution parameters that define η at initial time are set to $a = 100\,\mathrm{m}$ and $b = 6.4 \times 10^{-11}\,\mathrm{m^{-2}}$, hence the e-folding of the Gaussian at initial time is 125 km and is resolved by about 5 surface-elevation nodes.

Time sequences for η are shown in Figure 5.2. The surface elevation is first shown at stage 1, after a single timestep. At stage 3 it is being reflected by the basin wall, and by stages $5, 6, 7$, and 8 it has returned one, two, three, and four times, respectively, to its starting point. Comparing panels 1 and 5 of Figure 5.2 we can see that very little dispersion has occurred after a single cycle, but after two and three cycles the dispersion effect is quite obvious. The dispersion is due to the individual Bessel modes of the exact solution propagating with different phase speeds.

Good agreement is obtained between the analytical and computed solutions for the $P_1^{NC} - P_1$ pair, as shown in Table 5.1, since the error is about 1%. Further, the radial symmetry of the exact solution is very well reproduced by the numerical one.

5.2. The Stommel problem. In the second experiment the wind-driven large-scale circulation test case, proposed by Stommel [26], is performed in a homogeneous square ocean under the influence of surface winds, linearized bottom friction, flat bottom, and Coriolis force. The solution is an intense crowding of streamlines towards the western boundary caused by the variation of the Coriolis parameter with latitude.

The linear shallow-water equations (2.1)–(2.2) now include forcing ($\boldsymbol{\tau}$) and bottom friction ($-\varepsilon\mathbf{u}$), hence we solve for

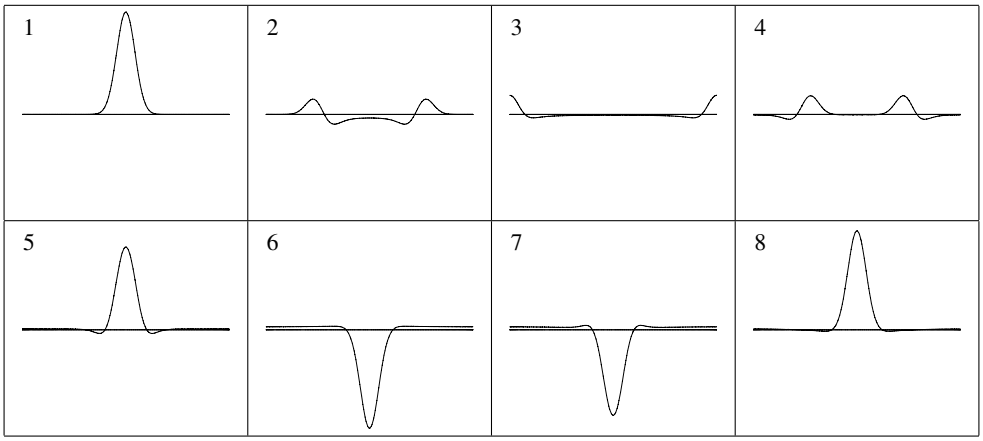


FIG. 5.2. *Vertical cross sections in the x, z plane of the surface elevation η at different stages of gravity wave propagation and dispersion. The initial Gaussian distribution is shown in panel* 1. *Reflection occurs at the boundary in panel* 3, *and in panels* 5, 6, 7, *and* 8 *the disturbance has returned to its starting point one, two, three, and four times, respectively.*

TABLE 5.1
Maximum (max) and minimum (min) values of the analytical and computed surface-elevation field, for the $P_1^{NC} - P_1$ pair at the stages corresponding to the plots of Figure 5.2.

Stage		1	2	3	4	5	6	7	8
Timestep		1	179	357	536	683	1430	2113	2860
Analytical	max	99.55	14.84	18.53	18.12	80.76	3.40	4.26	95.91
	min	0.00	−9.47	−3.33	−4.66	−3.58	−95.57	−83.92	−1.33
$P_1^{NC} - P_1$	max	99.44	14.94	18.64	18.23	80.79	3.45	4.16	94.23
	min	0.00	−9.50	−3.36	4.67	−3.56	−94.61	−84.76	−1.43

$$\mathbf{u}_t + f\mathbf{k} \times \mathbf{u} + g\nabla\eta = \boldsymbol{\tau} - \varepsilon\mathbf{u}, \tag{5.2}$$

$$\eta_t + H\nabla \cdot \mathbf{u} = 0, \tag{5.3}$$

subject to the no-normal flow boundary condition (2.3).

The domain is an idealized 4000 km × 4000 km square basin with solid boundaries on all sides. The β-plane approximation, $f = f_0 + \beta y$ is used, with $f_0 = 10^{-4}\,\mathrm{s}^{-1}$ and $\beta = 10^{-11}\,\mathrm{m}^{-1}\,\mathrm{s}^{-1}$. The reduced-gravity acceleration and the mean depth are taken as $g' = 10^{-2}\,\mathrm{m\ s}^{-2}$ and $H = 400\,\mathrm{m}$, respectively, leading to $c = 2\,\mathrm{m\ s}^{-1}$. The radius of deformation at midbasin is thus $R_d \equiv \sqrt{g'H}\,/\,f_0 = 20\,\mathrm{km}$.

A zonal wind applies a surface stress of the form $\boldsymbol{\tau} = (\tau_0 \sin(\pi y\,/\,L), 0)$, where $-L\,/\,2 < y < L\,/\,2$ and $\tau_0 = 2 \times 10^{-7}\,\mathrm{m\ s}^{-2}$ spinning up a single anticyclonic gyre. By setting $\varepsilon = 10^{-6}\,\mathrm{s}^{-1}$ the bottom drag leads to a Stommel layer width of $\delta = \varepsilon\,/\,\beta = 100\,\mathrm{km}$. Two uniform grids are used (made up of biased right triangles for the $P_1^{NC} - P_1$ pair), from a high resolution grid (201 × 201 points) with $h = 20\,\mathrm{km}$ so that the Stommel layer is well resolved, and a low resolution grid (21 × 21 points) with $h = 200\,\mathrm{km}$ so that the Stommel layer is barely resolved. The timestep is 1728 s (50 steps per day).

An implicit backward ($\gamma = 1$) and an Euler forward scheme are used for the Coriolis term and the bottom friction, respectively. Because the value of the gravitational Courant number C_g is small for the two grids, choosing either the explicit

($\alpha = 0$) or the trapezoidal Crank–Nicolson ($\alpha = 1/2$) scheme has negligible impact on the results. Note that $\mathsf{R}^{\mathbf{u}}$ in (5.2) now includes forcing and bottom friction.

The goal of the present experiment is to test the ability of the shallow-water finite-element model described herein to simulate the evolution of the divergence of the velocity field for the Stommel problem. Such an experiment has already been performed in [1] to test the behavior of the CD-grid at both high and low resolutions. In [1], the results obtained for the CD-grid are compared with those of the C-grid at high resolution, where the latter are free of Coriolis modes and are believed to be quite accurate. A precise calculation of the horizontal divergence of the velocity field is crucial in order to well represent the vertical velocity field in general circulation models. Hence, all the aforementioned parameters have been chosen to correspond to those employed in [1].

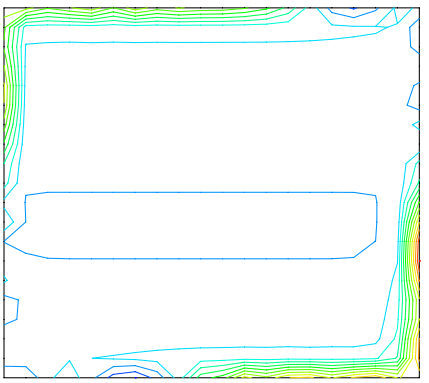
The evolution of the divergence of the velocity field is shown in Figure 5.3 for the CD-grid and the $P_1^{NC} - P_1$ pair after 100 days of simulation. Both schemes mainly differ with respect to the low resolution calculation (Figure 5.3 panels a and b). At high resolution the finite-element solution better matches the C-grid (Figure 5.4 panel e) result than the CD-grid one. Indeed, the eddy appearing at the bottom right corner of the domain for the C-grid and the $P_1^{NC} - P_1$ pair (Figure 5.3 panel d) is not present for the CD-grid (Figure 5.3 panel c). Further, it seems that some numerical noise appears in the CD-grid solution close to the boundary (Figure 5.3 panels c and e), which is not the case for the C-grid (Figure 5.4 panel e) and the $P_1^{NC} - P_1$ pair (Figure 5.3 panels d and f) at this stage of the propagation.

In Figure 5.4 the evolution of the divergence of the velocity field is again shown for the CD-grid and the $P_1^{NC} - P_1$ pair, but after 500 days of simulation. Both schemes again considerably differ with respect to the low resolution calculation. Indeed, the finite-element solution well captures the bottom right eddy (Figure 5.4 panel b) while the CD-grid does not represent such a behavior (Figure 5.4 panel a). The eddy is, however, fully developed for all schemes at high resolution (Figure 5.4 panels c, d, f), which suggests that even at low resolution the finite-element pair performs well. The results obtained for the CD scheme on a 26×26 grid, where the number of unknows is approximately the same than for the $P_1^{NC} - P_1$ pair on a 21×21 grid, are very close to those obtained on a 21×21 grid.

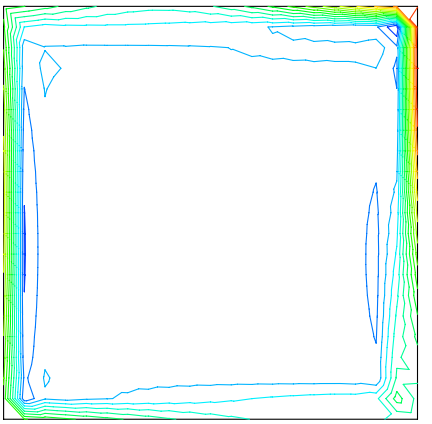
Even on a 51×51 grid (Figure 5.5 panel a), the CD scheme barely captures the bottom right eddy after 500 days of simulation (Figure 5.5 panel b) while the finite-element solution (Figure 5.5 panel d), obtained on a nonuniform triangular mesh having only 441 points (Figure 5.5 panel c), shows an improved behavior compared to the solution of Figure 5.4 panel b obtained on a uniform grid having the same number of points (441). Further, in the case of Figure 5.5, the use of the CD-grid results in having 5.7 and 4.1 times more surface-elevation and velocity unknowns, respectively, than for the $P_1^{NC} - P_1$ pair.

Consequently, the finite-element scheme is shown to be more accurate than the CD one at low resolution. On the fine grid the $P_1^{NC} - P_1$ pair and the C-grid give nearly indistinguishable results (Figure 5.4 panels d and f) close to those obtained for the CD-grid (Figure 5.4 panel c). The $P_1^{NC} - P_1$ pair thus yields equally favorable results in both the low and high resolution regimes, and those are significantly better than for the CD-grid. Further, the fact that the finite-element scheme applies equally well to structured as well as unstructured meshes demonstrates the excellent flexibility of the method.

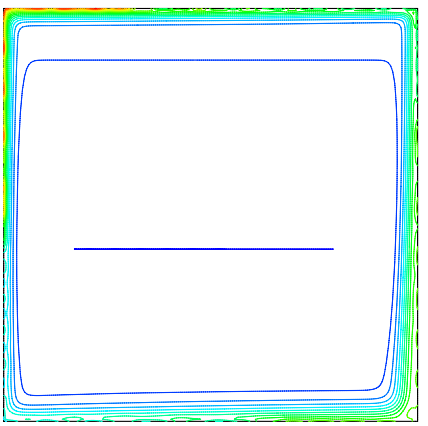
a) CD-grid (21 × 21) after 100 days



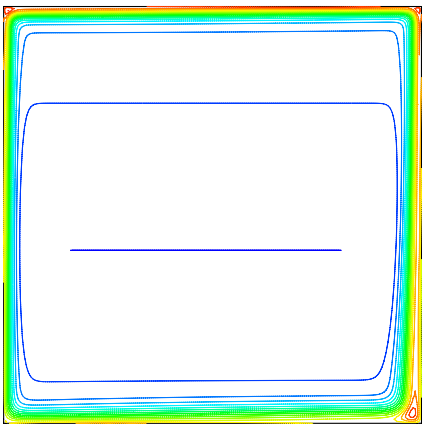
b) FE grid (21 × 21) after 100 days



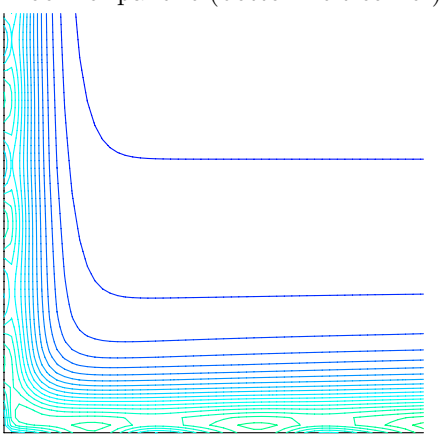
c) CD-grid (201 × 201) after 100 days



d) FE grid (201 × 201) after 100 days



e) Zoom of panel c (bottom left corner)



f) Zoom of panel d (bottom left corner)

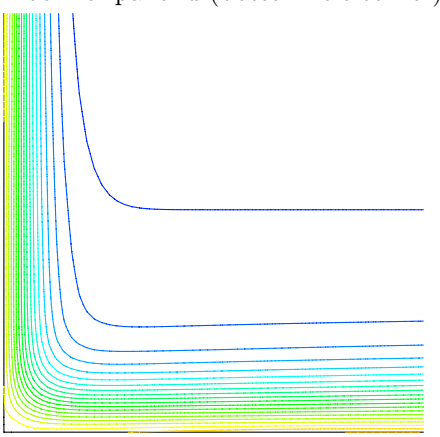


FIG. 5.3. *The divergence field for the CD-grid and the finite element* $P_1^{NC} - P_1$ *pair (FE) at both low (*21×21*) and high (*201×201*) resolution. Contour interval is* 1.3×10^{-9} s^{-1} *for panels a, b, c, d, and* $0.5 \times 10^{-9}\,\mathrm{s}^{-1}$ *for panels e and f.*

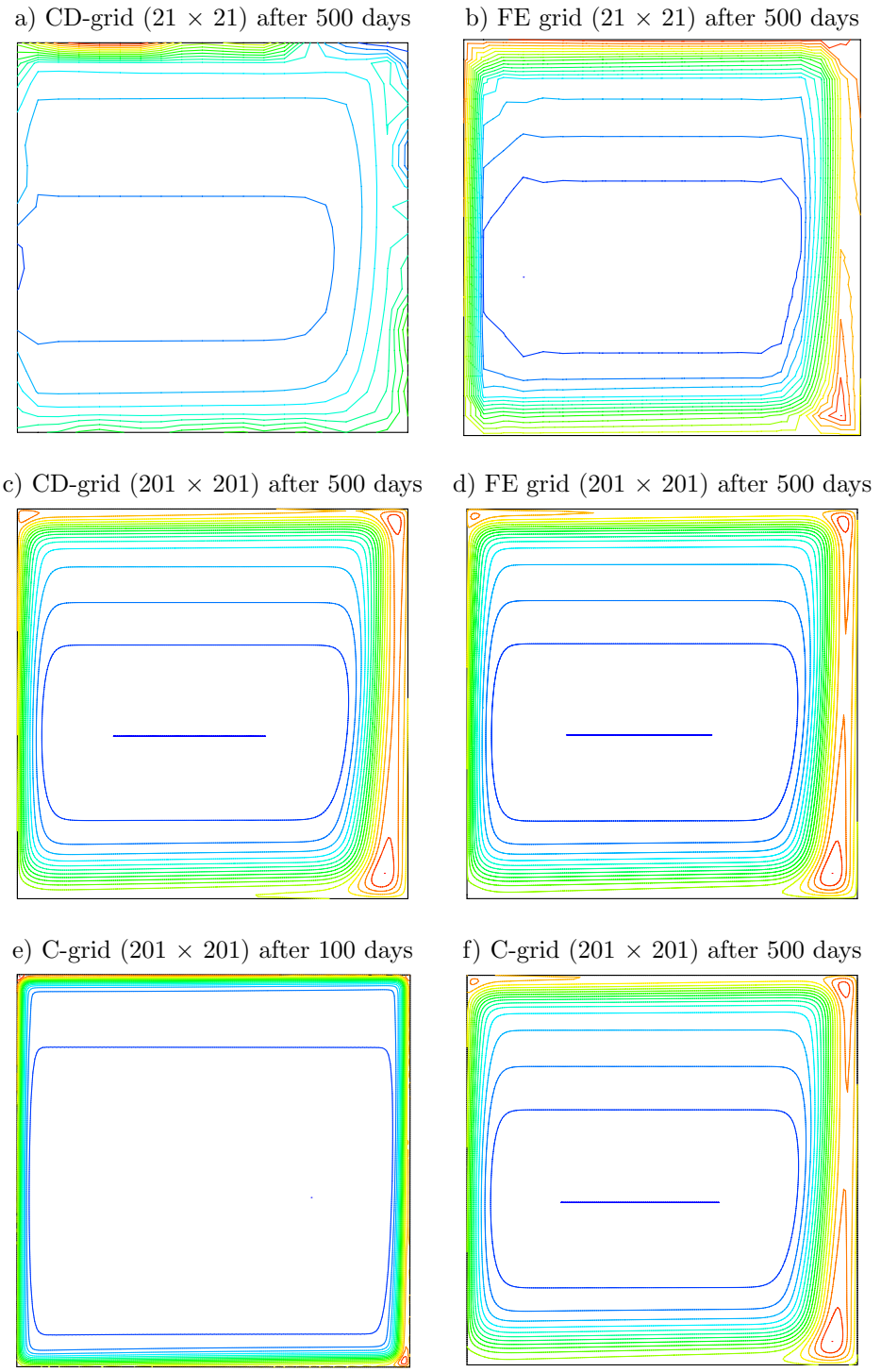


FIG. 5.4. *The divergence field for the CD-grid, the finite element* $P_1^{NC} - P_1$ *pair (FE) at both low (*21×21*) and high (*201×201*) resolution, and the C-grid at high resolution. Contour interval is* $0.9 \times 10^{-9}\,\mathrm{s}^{-1}$ *for panel a and* $0.4 \times 10^{-9}\,\mathrm{s}^{-1}$ *for panels b, c, d, and f.*

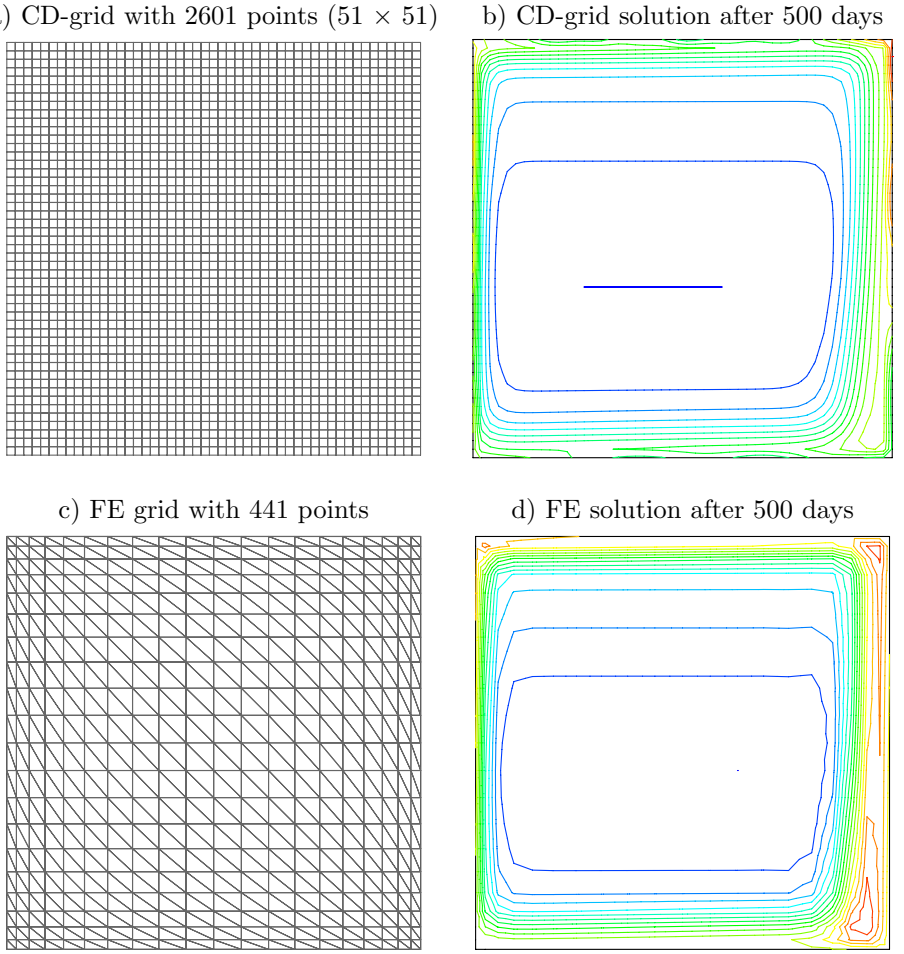


FIG. 5.5. *The divergence field for the CD-grid and the finite element* $P_1^{NC} - P_1$ *pair (FE) obtained on a uniform grid (panel a) and an unstructured triangular mesh (panel c), respectively. Contour interval is* $0.4 \times 10^{-9}\,\mathrm{s}^{-1}$ *for panels b and d.*

6. Conclusions. Most of the numerical methods that are used to solve the shallow-water equations poorly approximate the representation of inertia-gravity waves and usually lead to spurious solutions. In this paper, the triangular finite-element pair $P_1^{NC} - P_1$ is shown to "properly" model the dispersion of the inertia-gravity waves, and this is achieved at a reasonable computational cost compared to traditional finite-difference schemes.

Two tests are performed: simulation of propagation and dispersion of gravity waves in a circular basin using an unstructured triangular mesh, and simulation of the wind-driven large-scale circulation test case of Stommel. In the first experiment good agreement is obtained between the analytical and computed solutions for the finite-element pair. In the second test the $P_1^{NC} - P_1$ pair yields significantly better results than for the finite-difference CD-grid in both the low and high resolution regimes. The fact that the method applies equally well to structured as well as unstructured meshes demonstrates its excellent flexibility.

In this paper linear equations have been considered. However, the nonlinear shallow-water equations have been successfully solved using the $P_1^{NC} - P_1$ finite-element pair [15] and an accurate representation of the slow Rossby modes has been obtained with a small amount of numerical diffusion. It is thus concluded that the $P_1^{NC} - P_1$ pair is a very promising choice for the discretization of a variety of coastal and environmental problems.

REFERENCES

[1] A. J. Adcroft, C. N. Hill, and J. C. Marshall, *A new treatment of the Coriolis terms in C-grid models at both high and low resolutions*, Mon. Wea. Rev., 127 (1999), pp. 1928–1936.

[2] A. Arakawa and V. R. Lamb, *Computational design of the basic dynamical processes of the UCLA general circulation model*, Methods Comput. Phys., 17 (1977), pp. 173–265.

[3] J. H. Atkinson, J. J. Westerink, and R. A. Luettich Jr, *Two dimensional dispersion analyses of finite element approximations to the shallow water equations*, Internat. J. Numer. Methods Fluids, 45 (2004), pp. 715–749.

[4] M. L. Batteen and Y. J. Han, *On the computational noise of finite-difference schemes used in ocean models*, Tellus, 33 (1981), pp. 387–396.

[5] M. Bercovier and O. Pironneau, *Error estimates for the finite element method solution of the Stokes problem in the primitive variables*, Numer. Math., 33 (1979), pp. 211–224.

[6] C. A. Brebbia and P. W. Partridge, *Mathematical Models for Environmental Problems*, C. A. Brebbia, ed., Wiley, New York, 1976.

[7] C. A. Brebbia and P. W. Partridge, *Finite element simulation of water circulation in the North Sea*, Appl. Math. Modelling, 1 (1976), pp. 101–107.

[8] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Series in Computational Mathematics, 15, Springer-Verlag, Berlin, 1991.

[9] M. J. P. Cullen, *On the use of artificial smoothing in Galerkin and finite-difference solutions of the primitive equations*, Quart. J. Roy. Meteor. Soc., 102 (1976), pp. 77–93.

[10] M. J. P. Cullen and C. D. Hall, *Forecasting and general circulation results from finite-element models*, Quart. J. Roy. Meteor. Soc., 105 (1979), pp. 571–592.

[11] M. S. Engelman, R. L. Sani, and P. M. Gresho, *The implementation of normal and/or tangential boundary conditions in finite-element codes for incompressible fluid flow*, Internat. J. Numer. Methods Fluids, 2 (1982), pp. 225–238.

[12] M. G. G. Foreman, *A two-dimensional dispersion analysis of selected methods for solving the linearized shallow water equations*, J. Comput. Phys., 56 (1984), pp. 287–323.

[13] M. Fortin and R. Pierre, *Stability analysis of discrete generalized Stokes problems*, Numer. Methods Partial Differential Equations, 8 (1992), pp. 303–323.

[14] W. G. Gray, *On normal flow boundary conditions in finite-element codes for two-dimensional shallow-water flow*, Internat. J. Numer. Methods Fluids, 4 (1984), pp. 99–104.

[15] E. Hanert, D. Y. Le Roux, V. Legat, and E. Deleersnijder, *An efficient Eulerian finite-element method for the shallow water equations*, Ocean Modelling, 10 (2005), pp. 115–136.

[16] B. L. Hua and F. Thomasset, *A noise-free finite element scheme for the two-layer shallow water equations*, Tellus Ser. A, 36 (1984), pp. 157–165.

[17] T. J. R. Hughes, L. P. Franca, and M. Balestra, *A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accomodating equal-order interpolations*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 85–99.

[18] P. H. LeBlond and L. A. Mysak, *Waves in the Ocean*, Elsevier, Amsterdam, 1978.

[19] D. Y. Le Roux, A. Staniforth, and C. A. Lin, *Finite elements for shallow-water equation ocean models*, Mon. Wea. Rev., 126 (1998), pp. 1931–1951.

[20] D. Y. Le Roux, *A new triangular finite-element with optimum constraint ratio for compressible fluids*, SIAM J. Sci. Comput., 23 (2001), pp. 66–80.

[21] D. Y. Le Roux, A. Sène, V. Rostand, and E. Hanert, *On some spurious mode issues in shallow-water models using a linear algebra approach*, Ocean Modelling, 10 (2005), pp. 83–94.

[22] D. R. Lynch and W. G. Gray, *A wave-equation model for finite-element tidal computations*, Comput. & Fluids, 7 (1979), pp. 207–228.

[23] G. F. Pinder and W. G. Gray, *Finite Element Simulation in Surface and Subsurface Hydrology*, Academic Press, New York, 1977.

[24] R. L. Sani, P. M. Gresho, R. L. Lee, and D. Griffiths, *The cause and cure (!) of the spurious pressures generated by certain FEM solutions of the incompressible Navier-Stokes equations*, Internat. J. Numer. Methods Fluids, 1 (1981), Part 1 pp. 17–43, Part 2 pp. 171–204.

[25] A. Staniforth and H. L. Mitchell, *A semi-implicit finite-element barotropic model*, Mon. Wea. Rev., 105 (1977), pp. 154–169.

[26] H. M. Stommel, *The westward intensification of wind-driven ocean currents*, Trans. Amer. Geophys. Union, 29 (1948), pp. 202–206.

[27] F. Thomasset, *Implementation of Finite Element Methods for Navier-Stokes Equations*, Springer-Verlag, Berlin, 1981.

[28] R. A. Walters, *Numerically induced oscillations in finite-element approximations to the shallow-water equations*, Internat. J. Numer. Methods Fluids, 3 (1983), pp. 591–604.

[29] R. A. Walters and R. T. Cheng, *Accuracy of an estuarine hydrodynamic model using smooth elements*, Water Resour. Res., 16 (1980), pp. 187–195.

[30] R. A. Walters and G. F. Carey, *Analysis of spurious oscillation modes for the shallow water and Navier-Stokes equations*, Comput. & Fluids, 11 (1983), pp. 51–68.

[31] R. A. Walters and G. F. Carey, *Numerical noise in ocean and estuarine models*, Adv. Water Res., 7 (1984), pp. 15–20.

[32] R. T. Williams, *On the formulation of the finite-element prediction models*, Mon. Wea. Rev., 109 (1981), pp. 463–466.

[33] R. T. Williams and O. C. Zienkiewicz, *Improved finite-element forms for the shallow-water wave equations*, Internat. J. Numer. Methods Fluids, 1 (1981), pp. 81–97.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 415–439
© 2005 Society for Industrial and Applied Mathematics

OVERLAPPING SCHWARZ WAVEFORM RELAXATION FOR CONVECTION-DOMINATED NONLINEAR CONSERVATION LAWS*

MARTIN J. GANDER† AND CHRISTIAN ROHDE‡

Abstract. We analyze the convergence of the overlapping Schwarz waveform relaxation algorithm applied to convection-dominated *nonlinear* conservation laws in one spatial dimension. For two subdomains and bounded time intervals we prove superlinear asymptotic convergence of the algorithm in the parabolic case and convergence in a finite number of steps in the hyperbolic limit. The convergence results depend on the overlap, the viscosity, and the length of the time interval under consideration, but they are independent of the number of subdomains, as a generalization of the results to many subdomains shows. To investigate the behavior of the algorithm for a long time, we apply it to the Burgers equation and use a steady state argument to prove that the algorithm converges linearly over long time intervals. This result reveals an interesting paradox: while for the superlinear convergence rate on bounded time intervals decreasing the viscosity improves the performance, in the linear convergence regime decreasing the viscosity slows down the convergence rate and the algorithm can converge arbitrarily slowly, if there is a standing shock wave in the overlap. We illustrate our theoretical results with numerical experiments.

Key words. domain decomposition, waveform relaxation, Schwarz methods, nonlinear conservation laws

AMS subject classifications. 65M55, 35K55, 35L65

DOI. 10.1137/030601090

1. Introduction.

Overlapping Schwarz waveform relaxation is a class of domain decomposition algorithms to solve evolution problems in parallel. The classical way of applying domain decomposition methods to evolution problems is to discretize the time dimension first uniformly over the whole domain by an implicit scheme, and then to apply domain decomposition at each time step separately to solve the sequence of steady state problems obtained from the implicit time discretization. Numerical experiments for this approach with an overlapping Schwarz method for the two-dimensional heat equation can be found in [Meu91] and the application of the additive and multiplicative Schwarz preconditioners to the convection diffusion equation have been analyzed in [Cai91] and [Cai94], respectively. A nonoverlapping domain decomposition in this context using a Neumann–Dirichlet preconditioner at each time step was proposed in [Dry91], and an interesting variant, which uses an explicit method to advance the interface values in time and then an implicit method on each subdomain and thus avoids a subdomain iteration, is proposed in [DD91]; see also [RZ94] and [CL96]. For hyperbolic problems, the case of the wave equation with discontinuous bulk modulus and density fields per subdomain was analyzed in [BGT97] and the advantage of different grids in space due to the domain decomposition was emphasized ("the possibility of assigning to each subdomain its own space step makes numerical simulations much less expensive"), but due to the uniform time discretization, one cannot have an optimal space-time discretization close to the

*Received by the editors October 3, 2003; accepted for publication (in revised form) August 17, 2004; published electronically September 23, 2005.
http://www.siam.org/journals/sisc/27-2/60109.html
†Section de Mathématiques, Université de Genève, CH-1211 Genève (martin.gander@math.unige.ch).
‡Fakultát für Mathmatik, Universität Bielefeld, D-33615 Bielefeld, Germany (crohde@math.uni-bielefeld,de).

CFL condition in each subdomain in the classical approach. For a linear first-order hyperbolic equation in two dimensions, numerical results of the additive Schwarz preconditioner together with GMRES acceleration were reported in [WCK98] and a first analysis for the case of the Euler equations can be found in [DLN00]. As mentioned above, the main disadvantage of this classical approach is that one needs to use a uniform time discretization over the entire domain and thus loses one of the main features of domain decomposition methods, namely, to treat the problem on each subdomain numerically differently, with an appropriate discretization both in time and space adapted to the subdomain problems.

Overlapping Schwarz waveform relaxation does not have this disadvantage; the algorithm also uses an overlapping domain decomposition in space, like the classical Schwarz algorithm for steady state problems (see [Sch70]), but then the algorithm solves evolution problems on the subdomains and uses an iteration to converge to the solution of the original problem posed on the entire domain, like in waveform relaxation methods (see [LRSV82]), which explains the name of the algorithm. Since subdomain problems are solved both in space and time on subdomains, appropriate discretizations in space and time can be applied per subdomain. In addition, communication is not required at each time step; the computation can be performed over several time steps in a time window before information is exchanged with neighboring subdomains. This can be beneficial in a parallel environment where the startup cost of a connection with another processor is significant.

A very early use of this type of algorithm can be found in the research report [MS87], where it was applied to the one-dimensional wave equation, and it was shown that the algorithm converges in a finite number of steps in this case; see also [Gan97]. The algorithm has been analyzed in [Bjø95] for more general hyperbolic problems. The algorithm applied to parabolic problems was first analyzed for the heat equation in [GZ97, GZ02], for a semilinear model problem in [Gan98], and independently for convection diffusion problems in [GK02]. Applied to parabolic problems, the algorithm has the interesting property of having two different convergence regimes, depending on the length of the time evolution on the subdomains before information is exchanged with neighboring subdomains. The algorithm's performance can be further improved by using better transmission conditions; see, for example, [GHN99].

We analyze in this paper the convergence properties of overlapping Schwarz waveform relaxation applied to convection-dominated viscous conservation laws with nonlinear fluxes. Let $T > 0$ be the end of the time interval of interest, $t \in [0, T)$, and $\Omega \subseteq \mathbb{R}$ be the bounded or unbounded spatial domain. We analyze the overlapping Schwarz waveform relaxation method to compute solutions $u = u(x, t) : \Omega \times (0, T) \to \mathbb{R}$ of the corresponding initial and initial boundary value problems

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = \varepsilon \frac{\partial^2 u}{\partial x^2} \quad \text{in } \Omega \times (0, T), \tag{1.1}$$

where $\varepsilon \geq 0$ and $f \in C^2(\mathbb{R})$ denotes a function which in general depends in a nonlinear way on u.

The initial boundary value problem for (1.1) is a scalar model problem for the physically important equations of viscous compressible flow: the compressible Navier–Stokes equations. Flow problems impose considerable difficulties if the Reynolds number becomes big. In our case we are therefore interested especially in the case $0 < \varepsilon \ll 1$. To understand this case it is essential to consider also the singular hyperbolic limit problem with $\varepsilon = 0$. We start by collecting in section 2 all the analytical background results we need in our analysis. In section 3 we analyze the behavior

of the overlapping Schwarz waveform relaxation algorithm for two subdomains over bounded time intervals, $T < \infty$. For $\varepsilon > 0$ we prove convergence of the algorithm at a superlinear rate. The analysis allows us to track exactly how the rate depends on ε when ε tends to zero. In particular, we show the connection to the results of the Schwarz waveform relaxation algorithm in the case $\varepsilon = 0$ which is analyzed separately. The convergence rate of the Schwarz waveform relaxation algorithm depends on the size of the overlap, the length of the time interval, and the viscosity term, and the algorithm becomes faster as the viscosity term becomes smaller, which leads in the limit to convergence in a finite number of steps. In section 4 we then generalize the superlinear convergence result to $I > 2$ subdomains and show that the convergence rate is independent of the number of subdomains. To understand the long-time behavior of the algorithm ($T \longrightarrow \infty$), we study in section 5 the overlapping Schwarz algorithm for the steady Burgers equation, which can be considered as the long-time behavior of the evolution case. We show that the algorithm converges linearly for the two subdomain cases and that the convergence rate depends again on the size of the overlap and the viscosity term. In this case, however, when there is a standing shock in the overlap, we find that the smaller the viscosity is, the slower the convergence becomes, and the dependence is exponential. Our analytical results agree with the observations for long time computations for the Burgers equation that have been reported in [GK97, GK00]. There, it is shown that the solution depends in a [illegible] way on the boundary conditions, if an internal layer is present. Moreover, an asymptotic analysis is presented and a two-dimensional version of the Burgers problem is considered. General analytical results using the Cole–Hopf transformation can be found in [LO95]. In section 6 we illustrate our results with numerical experiments.

2. Initial and initial boundary value problem. We consider for a given function $f \in C^2(\mathbb{R})$ the scalar conservation law

$$\frac{\partial u^\varepsilon}{\partial t}(x,t) + \frac{\partial}{\partial x} f(u^\varepsilon(x,t)) = \varepsilon \frac{\partial^2 u^\varepsilon}{\partial x^2}(x,t), \quad (x,t) \in \Omega \times (0,T), \tag{2.1}$$

where $\Omega \subseteq \mathbb{R}$ is an open set, $T > 0$, and $\varepsilon \geq 0$ is the viscosity parameter. We start with a review of basic analytical results for the parabolic case, $\varepsilon > 0$, and the (singular) hyperbolic case, $\varepsilon = 0$. In the first case we focus on the behavior of solutions for $0 < \varepsilon \ll 1$, which is important for the analysis of the Schwarz waveform relaxation algorithm later in section 3.

2.1. The parabolic case, $\varepsilon > 0$. In our analysis we need results both for the initial value problem (2.1) on the infinite domain $\Omega = \mathbb{R}$ with the initial condition

$$u^\varepsilon(x,0) = u_0(x), \quad x \in \Omega, \tag{2.2}$$

where we assume that $u_0 \in C^2(\mathbb{R}) \cap L^\infty(\mathbb{R})$, and for the initial boundary value problem (2.1) on the half line $\Omega = (-\infty, 0)$ with the initial and boundary conditions

$$u^0(x,0) = u_0(x),\ x \in \Omega, \quad u^0(0,t) = g(t),\ t \in [0,T], \tag{2.3}$$

where we suppose that $u_0 \in C^2([-\infty,0]) \cap L^\infty((-\infty,0])$, $g \in C^2([0,T])$, and that the appropriate compatibility condition holds for u_0 and g at $x = t = 0$. For both problems we consider classical solutions, i.e., functions that are smooth enough to satisfy the conservation law and the constraints pointwise.

THEOREM 2.1. $\varepsilon > 0$

(i) (2.1), (2.2), $\underline{u} = \inf_{x\in\mathbb{R}}\{u_0(x)\}$ $\overline{u} = \sup_{x\in\mathbb{R}}\{u_0(x)\}$ $u^\varepsilon \in C^1(0,T;C^2(\mathbb{R}))$ (2.1), (2.2)

$$\underline{u} \le u^\varepsilon(x,t) \le \overline{u}, \quad (x,t) \in \mathbb{R}\times[0,T], \qquad \varepsilon\|u^\varepsilon_x\|_{L^\infty(\mathbb{R}\times[0,T])} \le C_{IVP}.$$

$u^0 \in L^\infty(\mathbb{R}\times[0,T])$ $Q \subset \mathbb{R}$

$$\lim_{\varepsilon\to 0}\|u^0 - u^\varepsilon\|_{L^1(Q\times[0,T])} = 0.$$

(ii) (2.1), (2.3),

$$\underline{u} = \min\{\inf_{x\le 0}\{u_0(x)\}, \inf_{t\in[0,T]}\{g(t)\}\},$$

$$\overline{u} = \max\{\sup_{x\le 0}\{u_0(x)\}, \sup_{t\in[0,T]}\{g(t)\}\}.$$

$u^\varepsilon \in C^1(0,T;C^2((-\infty,0]))$ (2.1), (2.3)

$$\underline{u} \le u^\varepsilon(x,t) \le \overline{u}, \quad (x,t) \in (-\infty,0]\times[0,T]$$

$$\varepsilon\|u^\varepsilon_x\|_{L^\infty((-\infty,0]\times[0,T])} \le C_{IBVP}.$$

$u^0 \in L^\infty((-\infty,0]\times[0,T])$ $Q \subset (-\infty,0]$

$$\lim_{\varepsilon\to 0}\|u^0 - u^\varepsilon\|_{L^1(Q\times[0,T])} = 0.$$

$C_{IVP}, C_{IBVP} > 0$ f u_0 g T ε . For the existence part in (i) and (ii) we refer the reader to, e.g., [LSU68]. The L^∞-estimates follow from an application of the maximum principle.

For the proof of the vanishing viscosity limits we refer the reader to [MNRR96, Chapter 2]. □

2.2. The hyperbolic limit $\varepsilon = 0$. We now consider for the limiting case $\varepsilon = 0$ the initial value problem (2.1), (2.2) on $\Omega = \mathbb{R}$ with $u_0 \in L^\infty(\mathbb{R})$ and the initial boundary value problem (2.1), (2.3) on $\Omega = (-\infty, 0)$ with $u_0 \in L^\infty((-\infty,0])$ and $g \in L^\infty([0,T])$. It is well known that initial (boundary) value problems for nonlinear functions f do not have time-global classical solutions for arbitrary, even smooth functions u_0, g: within finite time singularities, i.e., shock waves, develop. The well-posedness theory for the singular problems (2.1), (2.2) and (2.1), (2.3) relies therefore on a weaker notion of a solution: . It can be proved that in both cases there exists a unique entropy solution which satisfies the boundary data in an appropriate sense. These are the solutions we consider throughout the paper, although we do not introduce the precise definitions here, since they are not needed in an essential way later. It is important that the vanishing viscosity limits u^0 from Theorem 2.1 coincide with the entropy solutions for (2.1), (2.2) and (2.1), (2.3) in the case $\varepsilon = 0$. We refer the reader interested in the background on nonlinear

conservation laws and the precise definition of entropy solutions to [MNRR96]. The properties of the entropy solutions we need in our analysis are summarized in the following theorem.

THEOREM 2.2. $\varepsilon = 0$

(i) (2.1), (2.2), $\underline{u} = \operatorname{essinf}_{x\in\mathbb{R}}\{u_0(x)\}$ $\overline{u} = \operatorname{esssup}_{x\in\mathbb{R}}\{u_0(x)\}$ $u^0 \in L^\infty(\mathbb{R}\times[0,T])$ (i) 2.1 (2.1), (2.2). $(x,t) \in \mathbb{R}\times[0,T]$

$$\underline{u} \le u^0(x,t) \le \overline{u}. \tag{2.4}$$

(ii) (2.1), (2.3),

$$\underline{u} = \min\left\{\operatorname{essinf}_{x\le 0}\{u_0(x)\}, \operatorname{essinf}_{t\in[0,T]}\{g(t)\}\right\},$$
$$\overline{u} = \max\left\{\operatorname{esssup}_{x\le 0}\{u_0(x)\}, \operatorname{esssup}_{t\in[0,T]}\{g(t)\}\right\}$$

$u^0 \in L^\infty(\mathbb{R}\times[0,T])$ (ii) 2.1 (2.1), (2.3). $(x,t) \in (-\infty,0]\times[0,T]$

$$\underline{u} \le u^0(x,t) \le \overline{u}. \tag{2.5}$$

(iii) $u_{01}, u_{02} \in L^\infty(-\infty,0])$ $g_1, g_2 \in L^\infty([0,T])$, $I = [a,b] \subset (-\infty,0]$ $u_1^0, u_2^0 \in L^\infty((-\infty,0]\times[0,T])$ (2.1), (2.3) $u_0 = u_{01}$ $u_0 = u_{02}$ $g = g_1$, $g = g_2$ $t \in [0,T]$

$$\int_a^b |u_1(x,t) - u_2(x,t)|\,dt \le \int_{a-\lambda t}^{\min\{0,b+\lambda t\}} |u_{01}(x) - u_{02}(x)|\,dx + \lambda \int_0^{b/\lambda+t} |g_1(s) - g_2(s)|\,ds.$$

$\lambda > 0$

$$\lambda = \max_{\underline{u}\le v\le\overline{u}}\{|f'(v)|\},$$

$$\underline{u} = \min\left\{\operatorname*{ess\,inf}_{x\le 0}\{u_{01}(x)\}, \operatorname*{ess\,inf}_{t\in[0,T]}\{g_1(t)\}, \operatorname*{ess\,inf}_{x\le 0}\{u_{02}(x)\}, \operatorname*{ess\,inf}_{t\in[0,T]}\{g_2(t)\}\right\}$$

$\overline{u}$. All statements can be found in Chapter 2 of [MNRR96]. □

Theorem 2.1 gives the basic existence results for classical solutions and states that u^ε converges for $\varepsilon \to 0$ pointwise almost everywhere to the entropy solutions u^0 of the corresponding hyperbolic problems given in Theorem 2.2. Thus, in the particularly interesting singularly perturbed cases $0 < \varepsilon \ll 1$ we have to take into account the viscous counterparts of shock waves, i.e., internal layers of width $\mathcal{O}(\varepsilon)$ and amplitude $\mathcal{O}(1)$. Precisely the L^∞-estimates on the derivative u_x^ε in Theorem 2.1 allow us to compute the dependency on ε explicitly in all our estimates.

We note that statement (ii) in Theorem 2.1 and statements (ii), (iii) in Theorem 2.2—appropriately adapted—hold true for more general domains, particularly for all kinds of quarter spaces.

3. Overlapping Schwarz waveform relaxation for two subdomains. We analyze now the overlapping Schwarz waveform relaxation algorithm for the initial value problem (2.1), (2.2) and two subdomains. We start with the parabolic case, $\varepsilon > 0$, in subsection 3.1. We pay close attention in this analysis to the dependence of the convergence results on ε, since we are also interested in the convergence behavior of the algorithm in the limit when $\varepsilon \to 0$. In section 3.2 we then analyze directly the behavior of the algorithm in the hyperbolic limit when $\varepsilon = 0$, and we show how [illegible] flux functions lead to convergence properties of the algorithm which are not present when it is applied to linear problems. To simplify the notation we will skip the index ε in u^ε in what follows.

3.1. The parabolic case, $\varepsilon > 0$. The overlapping Schwarz waveform relaxation algorithm applied to the initial value problem (2.1), (2.2) with the two subdomains $\Omega_1 = (-\infty, L)$ and $\Omega_2 = (0, \infty)$, $L > 0$, is given for iteration index $n \in \mathbb{N}$ by

$$\begin{aligned} \frac{\partial u_1^n}{\partial t} + f'(u_1^n)\frac{\partial u_1^n}{\partial x} &= \varepsilon \frac{\partial^2 u_1^n}{\partial x^2} && \text{in } \Omega_1 \times (0,T), \\ u_1^n(\cdot,0) &= u_0 && \text{in } \Omega_1, \\ u_1^n(L,\cdot) &= u_2^{n-1}(L,\cdot) && \text{on } [0,T], \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} \frac{\partial u_2^n}{\partial t} + f'(u_2^n)\frac{\partial u_2^n}{\partial x} &= \varepsilon \frac{\partial^2 u_2^n}{\partial x^2} && \text{in } \Omega_2 \times (0,T), \\ u_2^n(\cdot,0) &= u_0 && \text{in } \Omega_2, \\ u_2^n(0,\cdot) &= u_1^{n-1}(0,\cdot) && \text{on } [0,T]. \end{aligned} \tag{3.2}$$

There is also a more sequential variant of this algorithm, where the interface values on the second domain are taken from the newer iterate u_1^n on the first subdomain, like in a Gauss–Seidel iteration, but we analyze only the Jacobi version given in (3.1) and (3.2); the Gauss–Seidel case can be analyzed similarly. For the analysis we also require that the iteration starts with the initial guess

$$\begin{aligned} u_1^0(x,t) &= \inf_{x' \in (-\infty, L]} \{u_0(x')\}, \quad (x,t) \in \Omega_1 \times (0,T), \\ u_2^0(x,t) &= \inf_{x' \in [0, \infty)} \{u_0(x')\}, \quad (x,t) \in \Omega_2 \times (0,T). \end{aligned} \tag{3.3}$$

We define the errors in the Schwarz waveform relaxation iteration by $e_1^n := u - u_1^n$ on the left subdomain and $e_2^n := u - u_2^n$ on the right subdomain for $n \in \mathbb{N}_0$. For $n \in \mathbb{N}$, we find with (3.1), (3.2) that the errors satisfy the equations

$$\begin{aligned} \frac{\partial e_1^n}{\partial t} + f'(u)\frac{\partial e_1^n}{\partial x} + \theta_1^n e_1^n &= \varepsilon \frac{\partial^2 e_1^n}{\partial x^2} && \text{in } \Omega_1 \times (0,T), \\ e_1^n(\cdot,0) &= 0 && \text{in } \Omega_1, \\ e_1^n(L,\cdot) &= e_2^{n-1}(L,\cdot) && \text{on } [0,T], \end{aligned} \tag{3.4}$$

and

$$\begin{aligned} \frac{\partial e_2^n}{\partial t} + f'(u)\frac{\partial e_2^n}{\partial x} + \theta_2^n e_2^n &= \varepsilon \frac{\partial^2 e_2^n}{\partial x^2} && \text{in } \Omega_2 \times (0,T), \\ e_2^n(\cdot,0) &= 0 && \text{in } \Omega_2, \\ e_2^n(0,\cdot) &= e_2^{n-1}(0,\cdot) && \text{on } [0,T], \end{aligned} \tag{3.5}$$

where the functions $\theta_i^n : \Omega_i \to \mathbb{R}$ are given by

(3.6)
$$\theta_i^n(x,t) = \frac{\partial}{\partial x} u_i^n(x,t) \int_0^1 f''\Big(u_i^n(x,t) - s\big(u(x,t) - u_i^n(x,t)\big)\Big)\, ds, \qquad i = 1,2.$$

For later use we also define here the functions $K_{1,x}$, $K_{2,x}$ by

$$K_{1,x}(x,t) = -\frac{1}{2\sqrt{\pi}} \frac{x-L}{\varepsilon^{1/2} t^{3/2}} \exp\Big(-\frac{(x-L)^2}{4\varepsilon t}\Big), \tag{3.7}$$

$$K_{2,x}(x,t) = \frac{1}{2\sqrt{\pi}} \frac{x}{\varepsilon^{1/2} t^{3/2}} \exp\Big(-\frac{x^2}{4\varepsilon t}\Big). \tag{3.8}$$

Because of our particular choice of the starting values (3.3) and the comparison principle for parabolic differential equations (see [Fri64]), the errors on both subdomains stay nonnegative for all iterations $n \in \mathbb{N}_0$,

$$e_1^n(x,t) \geq 0, \quad (x,t) \in \Omega_1 \times (0,T), \qquad e_2^n(x,t) \geq 0, \quad (x,t) \in \Omega_2 \times (0,T). \tag{3.9}$$

It suffices therefore to derive upper bounds for the errors to obtain a bound on the convergence rate of the overlapping Schwarz waveform relaxation algorithm.

LEMMA 3.1 (supersolutions). [illegible] $\{e_1^n\}_{n\in\mathbb{N}}$ $\{e_2^n\}_{n\in\mathbb{N}}$ [illegible] (3.4), (3.5). [illegible] $n \in \mathbb{N}$ [illegible]

$$0 \leq e_1^n(x,t) \leq \bar{e}_1^n(x,t) \qquad \forall\, (x,t) \in \Omega_1 \times (0,T),$$
$$0 \leq e_2^n(x,t) \leq \bar{e}_2^n(x,t) \qquad \forall\, (x,t) \in \Omega_2 \times (0,T),$$

[illegible] $\bar{e}_1^n$ [illegible]

$$\begin{aligned} \frac{\partial \bar{e}_1^n}{\partial t} + a_1 \frac{\partial \bar{e}_1^n}{\partial x} + b_1 \bar{e}_1^n &= \varepsilon \frac{\partial^2 \bar{e}_1^n}{\partial x^2} && \text{[illegible]}\ \Omega_1 \times (0,T) \\ \bar{e}_1^n(\cdot,0) &= 0 && \text{[illegible]}\ \Omega_1 \\ \bar{e}_1^n(L,t) &= \exp(\sigma_1 t) \sup_{0\leq\tau\leq t} e_2^{n-1}(L,\tau), && t \in [0,T], \end{aligned} \tag{3.10}$$

[illegible] $a_1, b_1, \sigma_1 \in \mathbb{R}$ [illegible]

$$\begin{aligned} a_1 &:= \inf_{(x,t)\in\Omega_1} f'(u(x,t)), \\ b_1 &:= \inf_{(x,t)\in\Omega_2} \Big\{\theta_1^n(x,t) + (f'(u(x,t)) - a_1)\frac{a_1}{2\varepsilon}\Big\}, \\ \sigma_1 &:= \begin{cases} -\frac{a_1^2}{4\varepsilon} - b_1 & \text{[illegible]}\ -\frac{a_1^2}{4\varepsilon} - b_1 \geq 0 \\ 0 & \text{[illegible]}, \end{cases} \end{aligned}$$

[illegible] $\bar{e}_2^n$ [illegible]

$$\begin{aligned} \frac{\partial \bar{e}_2^n}{\partial t} + a_2 \frac{\partial \bar{e}_2^n}{\partial x} + b_2 \bar{e}_2^n &= \varepsilon \frac{\partial^2 \bar{e}_2^n}{\partial x^2} && \text{[illegible]}\ \Omega_2 \times (0,T) \\ \bar{e}_2^n(x,0) &= 0 && \text{[illegible]}\ \Omega_2 \\ \bar{e}_2^n(0,t) &= \exp(\sigma_2 t) \sup_{0\leq\tau\leq t} e_1^n(0,\tau), && t \in [0,T], \end{aligned} \tag{3.11}$$

$a_2, b_2, \sigma_2 \in \mathbb{R}$

$$a_2 := \sup_{(x,t)\in\Omega_1} f'(u(x,t)),$$
$$b_2 := \inf_{(x,t)\in\Omega_2} \Big\{\theta_2^n(x,t) + (f'(u(x,t)) - a_2)\frac{a_2}{2\varepsilon}\Big\},$$
$$\sigma_2 := \begin{cases} -\frac{a_2^2}{4\varepsilon} - b_2 & -\frac{a_2^2}{4\varepsilon} - b_2 \geq 0 \\ 0 & \end{cases}$$

σ_1 σ_2 $\mathcal{O}(\varepsilon^{-1})$ (ii) 2.1.

. The proof is constructive. We first define for $i = 1, 2$ the constants

$$p_i = \frac{a_i}{2\varepsilon}, \quad q_i = -\frac{a_i^2}{4\varepsilon} - b_i, \tag{3.12}$$

and the function

$$g_1 = g_1(t) = \exp(-p_1 L + (\sigma_1 - q_1)t) \sup_{0\leq\tau\leq t} e_2^{n-1}(L,\tau),$$

which is nonnegative due to (3.9), and monotonically increasing because of our choice of σ_1. For the linear, constant coefficient problem (3.10) satisfied by the supersolution we have the closed form solution formula

$$\bar{e}_1^n(x,t) = \exp(p_1 x + q_1 t) \int_0^t K_{1,x}(x, t-\tau) g_1(\tau)\, d\tau, \tag{3.13}$$

where the kernel $K_{1,x}(x,t)$ is given in (3.7). To show that $\bar{e}_1^n$ is indeed a supersolution, we have to show that

$$d_1^n := \bar{e}_1^n - e_1^n \geq 0. \tag{3.14}$$

Now the difference function d_1^n satisfies the linear convection diffusion equation

$$\frac{\partial d_1^n}{\partial t} + f'(u)\frac{\partial d_1^n}{\partial x} + \theta_1^n d_1^n - \varepsilon\frac{\partial^2 d_1^n}{\partial x^2} = Q_1(x,t), \tag{3.15}$$

where the source term $Q_1(x,t)$ is given by

$$\begin{aligned}
Q_1(x,t) &= (f'(u(x,t)) - a_1)\frac{\partial \bar{e}_1^n}{\partial x} + (\theta_1^n(x,t) - b_1)\,\bar{e}_1^n(x,t) \\
&= (f'(u(x,t)) - a_1)\frac{e^{(p_1 x + q_1 t)}}{2\sqrt{\pi}} \int_0^t \frac{e^{\left(-\frac{(x-L)^2}{4\varepsilon(t-\tau)}\right)}}{\varepsilon^{1/2}(t-\tau)^{3/2}} \left[\frac{(x-L)^2}{2\varepsilon(t-\tau)} - 1\right] g_1(\tau)\, d\tau \\
&\quad -\Big((f'(u(x,t)) - a_1)p_1 + \theta_1^n(x,t) - b_1\Big)(x-L) \\
&\qquad \times \frac{e^{(p_1 x + q_1 t)}}{2\sqrt{\pi}} \int_0^t \frac{e^{\left(-\frac{(x-L)^2}{4\varepsilon(t-\tau)}\right)}}{\varepsilon^{1/2}(t-\tau)^{3/2}} g_1(\tau)\, d\tau \\
&=: (f'(u(x,t)) - a_1)e^{(p_1 x + q_1 t)} Q_{11}(x,t) \\
&\quad + ((f'(u(x,t)) - a_1)p_1 + \theta_1^n(x,t) - b_1)\, e^{(p_1 x + q_1 t)}(L-x) Q_{12}(x,t).
\end{aligned}$$

If we can show that $Q_{11}(x,t)$ and $Q_{12}(x,t)$ are nonnegative for all $(x,t) \in \Omega_1 \times (0,T)$, we obtain $Q_1(x,t) \geq 0$ for all $(x,t) \in \Omega_1 \times (0,T)$ by the definition of a_1, b_1 which implies (3.14) by the maximum principle for (3.15) with zero initial and boundary data. But Q_{12} is nonnegative since g_1 from (3.13) is nonnegative by (3.9), and for Q_{11} we observe that it is the x-derivative of the solution w of the heat equation $w_t = \varepsilon w_{xx}$ in $\Omega_1 \times (0,T)$ which satisfies $w(L,\cdot) = g_1$ and $w(\cdot,0) \equiv 0$. Since g_1 is nonnegative and monotonically increasing, Q_{11} must also be nonnegative, which concludes the proof that $\bar{e}_1^n$ is a supersolution of e_1^n. Similarly one can also show that $\bar{e}_2^n$ is a supersolution of e_2^n. □

THEOREM 3.2 (superlinear convergence). [illegible] (3.1), (3.2) [illegible] (2.1), (2.2) [illegible] $t > 0$ [illegible]

$$\sup_{x\in\Omega_1,0\leq\tau\leq t}\{e_1^{2n}(x,\tau)\} \leq e^{\frac{C(t+L)}{\varepsilon}n}\operatorname{erfc}\left(\frac{nL}{\sqrt{\varepsilon t}}\right)\sup_{0\leq\tau\leq t}\{e_1^0(0,\tau)\}, \tag{3.16}$$

$$\sup_{x\in\Omega_2,0\leq\tau\leq t}\{e_2^{2n}(x,\tau)\} \leq e^{\frac{C(t+L)}{\varepsilon}n}\operatorname{erfc}\left(\frac{nL}{\sqrt{\varepsilon t}}\right)\sup_{0\leq\tau\leq t}\{e_2^0(0,\tau)\}, \tag{3.17}$$

[illegible] C [illegible] ε L t [illegible] n [illegible]. Using Lemma 3.1 and the explicit formula for the supersolutions, we obtain the estimates

$$\begin{aligned} e_1^n(x,t) &\leq e^{p_1(x-L)+q_1 t}\int_0^t K_{1,x}(x,t-\tau)\sup_{0\leq s\leq\tau}\{e_2^{n-1}(L,s)\}e^{(\sigma_1-q_1)\tau}\,d\tau, \\ e_2^n(x,t) &\leq e^{p_2 x+q_2 t}\int_0^t K_{2,x}(x,t-\tau)\sup_{0\leq s\leq\tau}\{e_1^{n-1}(0,s)\}e^{(\sigma_2-q_2)\tau}\,d\tau, \end{aligned} \tag{3.18}$$

where $K_{1,x}$, $K_{2,x}$ are defined in (3.7), (3.8), respectively, and the constants p_1, q_1, p_2, and q_2 are defined in (3.12). We evaluate the second inequality in (3.18) at $x = L$ and insert it into the first one to obtain

$$\begin{aligned} e_1^n(x,t) \leq{}& e^{p_1(x-L)+q_1 t}\int_0^t K_{1,x}(x,t-\tau)e^{(\sigma_1-q_1)\tau} \\ &\times\sup_{0\leq s\leq\tau}\left\{e^{p_2 L+q_2 s}\int_0^s K_{2,x}(L,s-\tilde{s})\sup_{0\leq\tilde{\tilde{s}}\leq\tilde{s}}\{e_1^{n-2}(0,\tilde{\tilde{s}})\}e^{(\sigma_2-q_2)\tilde{s}}d\tilde{s}\right\}d\tau. \end{aligned}$$

Evaluating this inequality at $x = 0$ we get

$$\begin{aligned} e_1^n(0,t) \leq{}& e^{-p_1 L+q_1 t}\int_0^t K_{1,x}(0,t-\tau)e^{(\sigma_1-q_1)\tau} \\ &\times\sup_{0\leq s\leq\tau}\left\{e^{p_2 L+q_2 s}\int_0^s K_{2,x}(L,s-\tilde{s})\sup_{0\leq\tilde{\tilde{s}}\leq\tilde{s}}\{e_1^{n-2}(0,\tilde{\tilde{s}})\}e^{(\sigma_2-q_2)\tilde{s}}d\tilde{s}\right\}d\tau \\ \leq{}& \widetilde{C}(t)\int_0^t K_{1,x}(0,t-\tau)\int_0^\tau K_{2,x}(L,\tau-\tilde{s})\sup_{0\leq\bar{s}\leq\tilde{s}}\{e_1^{n-2}(0,\bar{s})\}\,d\tilde{s}\,d\tau \\ ={}& \widetilde{C}(t)\int_0^t K_{2,x}(L,t-\tau)\int_0^\tau K_{2,x}(L,\tau-\tilde{s})\sup_{0\leq\bar{s}\leq\tilde{s}}\{e_1^{n-2}(0,\bar{s})\}\,d\tilde{s}\,d\tau, \end{aligned}$$

where we used that

$$K_{2,x}(L,t) = K_{1,x}(0,t) = \frac{L}{2\sqrt{\pi}\varepsilon^{1/2}t^{3/2}} e^{-\frac{L^2}{4\varepsilon t}}$$

and the function $\widetilde{C}$ given by

$$\widetilde{C}(t) := \max\{1, e^{q_2 t}, e^{q_1 t}, e^{(q_1+q_2)t}\} e^{(\sigma_1+\sigma_2-q_1-q_2)t} e^{(p_2-p_1)L}.$$

Note that $\sigma_1+\sigma_2-q_1-q_2 \geq 0$. The definition of $p_1, p_2, q_1, q_2, \sigma_1, \sigma_2$ and Theorem 2.1 imply that there is a constant $C > 0$ independent of ε, t, L such that

$$\widetilde{C}(p_1, p_2, q_1, q_2, \sigma_1, \sigma_2, t) \leq e^{\frac{C(t+L)}{\varepsilon}}, \qquad t > 0. \tag{3.19}$$

By induction we obtain

$$\sup_{0\leq\tau\leq t} e_1^{2n}(0,\tau) \leq e^{\frac{C(t+L)}{\varepsilon} n} \sup_{0\leq\tau\leq t} \{e_1^0(0,\tau)\} K^{2n}(t),$$

where K^{2n} is the $2n$-fold convolution of $K_{2,x}(L,.)$, that is, the integral term

$$\int_0^t K_{2,x}(L, t-s_1) \int_0^{s_1} K_{2,x}(L, s_1-s_2)$$
$$\cdots \int_0^{s_{2n-2}} K_{2,x}(L, s_{2n-2}-s_{2n-1}) \int_0^{s_{2n-1}} K_{2,x}(L, s_{2n-1}-s_{2n}) ds_{2n} ds_{2n-1} \cdots ds_2 ds_1.$$

Since the Laplace transform $\mathcal{L}[K_{2,x}(L,.)]$ is $e^{-L\sqrt{s/\varepsilon}}$, we obtain for the $2n$-fold convolution

$$\mathcal{L}[K^{2n}](s) = \frac{1}{s}\Big(\mathcal{L}[K_{2,x}(L,.)](s)\Big)^{2n} = \frac{1}{s} e^{-2nL\sqrt{s/\varepsilon}}.$$

The inverse Laplace transformation then yields

$$\sup_{0\leq\tau\leq t} e_1^{2n}(0,\tau) \leq e^{\frac{C(t+L)}{\varepsilon} n} \sup_{0\leq\tau\leq t} \{e_1^0(0,\tau)\} \int_0^t K_{2,x}(2nL, t-s)\, ds,$$

and the right-hand side of the last inequality can be simplified by a transform of variables to

$$\sup_{0\leq\tau\leq t} e_1^{2n}(0,\tau) \leq e^{\frac{C(t+L)}{\varepsilon} n} \sup_{0\leq\tau\leq t} \{e_1^0(0,\tau)\} \operatorname{erfc}\left(\frac{nL}{\sqrt{\varepsilon t}}\right),$$

where the error function complement erfc is defined by

$$\operatorname{erfc}(\mathrm{x}) = \frac{1}{\sqrt{\pi}} \int_x^\infty e^{-y^2}\, dy.$$

We thus have proved convergence of the algorithm on the interface $x = 0$ and with a similar argument convergence also follows on the interface $x = L$. Applying the maximum principle for problem (3.4) then shows that the error e_1^{2n} in the interior of the domain $\Omega_1 \times (0,t)$ is amplified at most by a factor $\exp(\bar{\theta}t)$ with

$\bar{\theta} = \sup_{(x,\tau)\in(-\infty,L)\times(0,t)}\{|\theta_1(x,\tau)|\}$. Theorem 2.1, the definition of θ_1 in (3.6), and (3.19) show that there exists a constant $C \geq 0$, independent of ε, n, t, and L, such that (3.16) holds. The estimate (3.17) follows along the same lines when starting with the second inequality in (3.18). ∎

3.3. If we apply the expansion $\sqrt{\pi}\mathrm{erfc}(z) = e^{-z^2}(z^{-1} + \mathcal{O}(z^{-3}))$ for large values $z > 0$ in the estimate (3.16), we obtain

$$\sup_{0\leq\tau\leq t}\{e_1^{2n}(x,\tau)\} \approx \frac{1}{\sqrt{\pi}} e^{\frac{Cn(t+L)}{\varepsilon} - \frac{n^2L^2}{\varepsilon t}} \frac{\sqrt{\varepsilon t}}{nL} \sup_{0\leq\tau\leq t}\{e_1^0(0,\tau)\}. \tag{3.20}$$

For fixed $T, L, \varepsilon > 0$ we observe that the algorithm converges superlinearly for $n \to \infty$ and $t \leq T$. In other words, we obtain the same asymptotic result as for the heat equation with linear convection and linear or nonlinear source terms; see [GZ97, Gan98].

3.4. Let n in (3.20) be fixed. Then there exists a $T = T(n)$ such that (a) the algorithm converges for $\varepsilon \to 0$ and $t < T(n)$, and (b) the estimate for the error e_1^{2n} does not converge to 0 for $\varepsilon \to 0$ and $t > T(n)$. This scenario does not happen for the heat equation with linear convection. In section 3.2 below we apply the Schwarz waveform relaxation algorithm to the hyperbolic case, $\varepsilon = 0$, and we will see that in this case it can happen that for all $n > 0$ there exists a time $T^* = T^*(n) > 0$ such that the error of the iteration vanishes for $t < T^*$ and is nonzero for $t > T^*$. It turns out that this behavior can only occur for nonlinear convection. Thus the scenario described in (a) and (b) is not an artifact of the proof.

3.2. The hyperbolic case $\varepsilon = 0$. The Jacobi version of the overlapping Schwarz waveform relaxation algorithm with two subdomains $\Omega_1 = (-\infty, L)$ and $\Omega_2 = (0, \infty)$ for the initial value problem (2.1), (2.2) in the hyperbolic limit, $\varepsilon = 0$, is given by

$$\begin{aligned} \frac{\partial u_1^n}{\partial t} + \frac{\partial}{\partial x} f(u_1^n) &= 0 && \text{in } \Omega_1 \times (0,T), \\ u_1^n(\cdot,0) &= u_0 && \text{in } \Omega_1, \\ u_1^n(L,\cdot) &= u_2^{n-1}(L,\cdot) && \text{on } [0,T], \end{aligned} \tag{3.21}$$

and

$$\begin{aligned} \frac{\partial u_2^n}{\partial t} + \frac{\partial}{\partial x} f(u_2^n) &= 0 && \text{in } \Omega_2 \times (0,T), \\ u_2^n(\cdot,0) &= u_0 && \text{in } \Omega_2, \\ u_2^n(0,\cdot) &= u_1^{n-1}(0,\cdot) && \text{on } [0,T], \end{aligned} \tag{3.22}$$

for iteration index $n \in \mathbb{N}$ and we are denoting by u_i^n the entropy solution in each iteration. The initial iterates u_1^0 and u_2^0 are chosen as in (3.3) taking the essential infimum.

The iterative algorithm (3.21), (3.22) is a priori not well defined. Note that problems (3.21), (3.22) are uniquely solvable by Theorem 2.2 for $u_0 \in L^\infty(\mathbb{R})$ and $u_2^{n-1}(L,.), u_1^{n-1}(0,.) \in L^\infty((0,T])$. However it is not clear whether the traces $u_2^{n-1}(L,.)$ and $u_1^{n-1}(0,.)$ exist in an appropriate way as L^∞-functions. This is assumed subsequently. Note that the local regularity of entropy solutions is a delicate issue which is out of the scope of this paper (cf. [Daf00, Chapter 11.3], for instance).

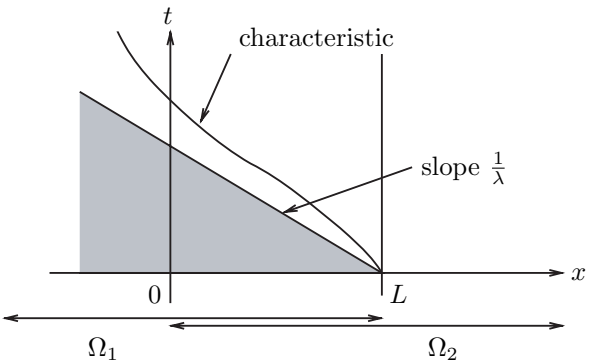


FIG. 1. *Estimated region in space-time where the error vanishes in the first iteration on the subdomain* Ω_1.

Finally we note that the definition of the trace is no problem after discretization of the problem and that is how the algorithm is used in practice.

We define as in the viscous case the errors in the Schwarz iteration by $e_1^n := u - u_1^n$ on the left subdomain and $e_2^n := u - u_2^n$ on the right subdomain for $n \in \mathbb{N}_0$, where u is the entropy solution of the initial value problem (2.1), (2.2) in the hyperbolic limit $\varepsilon = 0$.

THEOREM 3.5. $\underline{u} = \operatorname{essinf}_{x\in\mathbb{R}}\{u_0(x)\}$ $\overline{u} = \operatorname{esssup}_{x\in\mathbb{R}}\{u_0(x)\}$ $n \in \mathbb{N}$ $T^* = T^*(n, L) = nL/\lambda$

$$\lambda = \sup_{\underline{u}\le v\le\overline{u}} \{|f'(v)|\}.$$

$t \in [0, T] \cap [0, T^*(n, L)]$

$$\|e_1^{n+1}\|_{L^1((-\infty,L]\times[0,t])} = \|e_2^{n+1}\|_{L^1([0,\infty)\times[0,t])} = 0. \tag{3.23}$$

The proof relies on the finite speed of propagation property of entropy solutions which manifests itself in the statement (iii) of Theorem 2.2. To apply this statement for $n = 1$ in $\Omega_1 \times (0, T)$, we compare the subdomain solution at the first iteration on $\Omega_1 \times (0, T)$ with the exact solution restricted to $\Omega_1 \times (0, T)$. With the notation of Theorem 2.2, let

$$u_{01} = u_{02} = u_0|_{(-\infty,L]}, \quad g_1 = u_2^0(., L), \quad g_2 = u(., L).$$

The unique entropy solutions of the corresponding initial boundary value problems are u_1^1 and $u|_{\Omega_1\times(0,T)}$. Using statement (iii) of Theorem 2.2 and varying the interval I, we get in particular that $e_1^1 = u_1^1 - u|_{\Omega_1\times(0,T)} = 0$ almost everywhere in $\mathcal{C}_1^1$, where

$$\mathcal{C}_1^n = \{(x, s) \in \Omega_1 \times (0, T) \,|\, \lambda s \le nL - x\}, \quad n \in \mathbb{N};$$

see Figure 1. The analogous argument for Ω_2 gives $e_2^1 = 0$ almost everywhere in $\mathcal{C}_2^1$ given by

$$\mathcal{C}_2^n = \{(x, s) \in \Omega_2 \times (0, T) \,|\, \lambda s \le nL + x\}, \quad n \in \mathbb{N}.$$

Thus the errors e_1^1 and e_2^1 vanish on the interfaces in the time interval $[0, \frac{L}{\lambda}]$ (almost everywhere). Doing the next iteration we find for $t \le T^*(1, L)$

$$\|e_1^2\|_{L^1((-\infty,L]\times[0,t])} = \|e_2^2\|_{L^1([0,\infty),\times[0,t])} = 0,$$

and the general result for arbitrary $n \in \mathbb{N}$ follows by induction. □

REMARK 3.6. We consider the linear case, $f(u) = au$, for a convection speed $a > 0$. Taking any initial data and any initial guess for the iterates u_1^0 and u_2^0, one can easily see that the algorithm (3.21), (3.22) converges in two steps: for $n = 1$ the error e_1^1 vanishes in $\Omega_1 \times (0, T)$ since the solution is completely determined by the (correct) initial data, and no boundary data for $x = L$ can be prescribed for $a > 0$. Thus the third equation in (3.21) is ignored. On the second subdomain the error e_2^1 will in general not vanish in the whole of $\Omega_2 \times (0, T)$, since the boundary data u_2^0 is not correct in general. For $n = 2$, again e_1^2 vanishes in $\Omega_1 \times (0, T)$, but now also e_2^2 vanishes in $\Omega_2 \times (0, T)$, since $u_1^1(., L)$ provides the correct boundary data at the interface. The same result also holds for all functions f with $f'(v) \neq 0$ for $\underline{u} \le v \le \overline{u}$.

REMARK 3.7. For $L > 0$ we consider the case $f(u) = u^2/2$ together with the initial condition

$$u_0(x) = \begin{cases} 1 & : \quad x < L/2, \\ -1 & : \quad x > L/2. \end{cases} \tag{3.24}$$

The entropy solution is given by the discontinuous time-independent function $u(x, t) = u_0(x)$ for $t \in [0, T]$ and we do not have only one propagation direction in this case. Starting the overlapping Schwarz waveform relaxation algorithm with the initial guess $u_1^0 = a$ and $u_2^0 = -a$ for $a > 1$, we obtain the restriction of the entropy solution u to $(-\infty, L] \times [0, T]$ or $[0, \infty) \times [0, T]$ when computing the first iterates u_1^1 and u_2^1 on regions whose shape and size depend on a; a sketch of the iterates is displayed in Figure 2. For example, on the left subdomain Ω_1, the speed of the shock separating 1 from $-a$ and -1 from $-a$ increases when a increases. Thus the part of the boundary $x = 0$ where the correct data is obtained for the next iteration decreases when a increases and tends to zero when $a \to \infty$. The same argument also holds for the right subdomain (see Figure 2), which shows that the overlapping Schwarz waveform relaxation algorithm becomes slower as a increases. Stationary shocks are a generically nonlinear phenomenon which is not observed for linear equations and this example shows that the performance of the overlapping Schwarz waveform relaxation algorithm is affected by shocks.

One might argue that the choice for the initial guess does not agree with the choice for the first iterates u_1^0, u_2^0 as in (3.3). However, one can easily change the initial data u_0 far outside such that the solution in, say, $[-L, 2L] \times [0, T]$ is not altered due to finite propagation speeds and takes values less than -1. Then the flux function outside $[-1, 1]$ is altered such that the wave speeds $f'(u_1^0)$ and $f'(u_2^0)$ take the values $-a$ and a. This results in the same effect as described above. But the nonconvex flux function would introduce complicated shock-rarefaction patterns, so we omit a detailed construction here.

4. Overlapping Schwarz waveform relaxation for more than two subdomains. We now extend the results we obtained for two subdomains of $\Omega = \mathbb{R}$ in section 3 to the case of $I > 2$ subdomains, $I \in \mathbb{N}$, of a bounded domain $\Omega = (0, 1)$. Skipping the index ε, we search for the classical solution u of the initial boundary

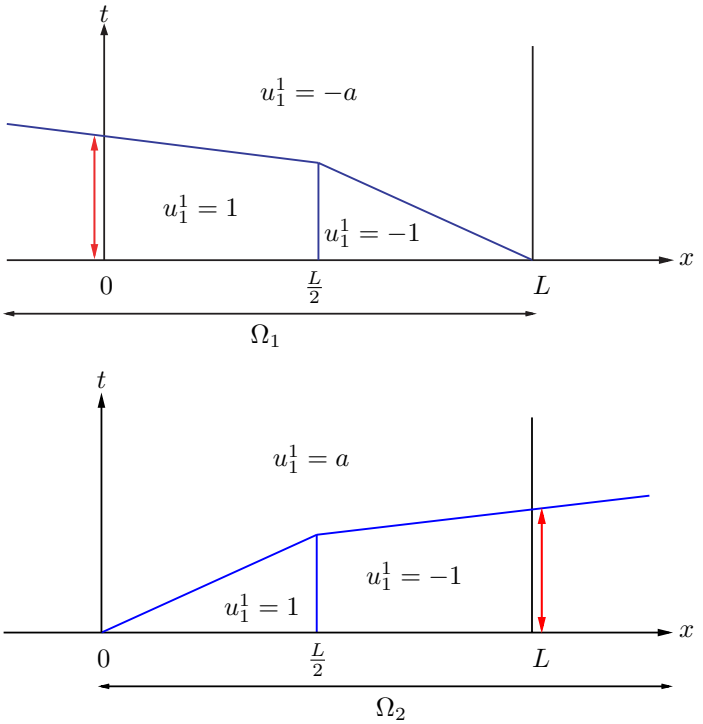


FIG. 2. *The top figure displays the entropy solution u_1^1 in Ω_1 taking the states 1, -1, and $-a$ separated by shock waves. The double arrow indicates the part of the line $x = 0$ where the correct boundary data for the next iteration is obtained. The bottom figure displays the corresponding situation for u_2^1 in Ω_2 and the part of the line $x = L$ where the correct boundary data for the next iteration is obtained.*

value problem

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) &= \varepsilon \frac{\partial^2 u}{\partial x^2} && \text{in } \Omega \times (0,T), \\ u(\cdot,0) &= u_0 && \text{in } \Omega, \\ u(0,\cdot) &= g_0 && \text{on } [0,T], \\ u(1,\cdot) &= g_1 && \text{on } [0,T]. \end{aligned} \tag{4.1}$$

Here $u_0 : \Omega \to \mathbb{R}$ and $g_1, g_2 : [0,T] \to \mathbb{R}$ are given smooth functions. To represent the subdomains, we introduce the numbers $L_1, \ldots, L_I \in \mathbb{R}$ and $R_1, \ldots, R_I \in \mathbb{R}$ such that

$$0 = L_1 < L_2 < R_1 < L_3 < \cdots < L_I < R_{I-1} < R_I = 1,$$

which leads to a decomposition of Ω into overlapping subdomains $\Omega_i = (L_i, R_i)$, $i = 1, 2, \ldots, I$, as shown in Figure 3. The overlapping Schwarz waveform relaxation algorithm is then given by solving for $i = 1, \ldots, I$ and $n = 1, 2, \ldots$ the problems

$$\begin{aligned} \frac{\partial u_i^n}{\partial t} + \frac{\partial}{\partial x} f(u_i^n) &= \varepsilon \frac{\partial^2 u_i^n}{\partial x^2} && \text{in } \Omega_i \times (0,T), \\ u_i^n(\cdot,0) &= u_0 && \text{in } \Omega_i, \\ u_i^n(L_i,\cdot) &= u_{i-1}^{n-1}(L_i,.) && \text{on } [0,T], \\ u_i^n(R_i,\cdot) &= u_{i+1}^{n-1}(R_i,.) && \text{on } [0,T], \end{aligned} \tag{4.2}$$

where we identify $u_0^{n-1}(L_1,.)$ with g_0 and $u_{I+1}^{n-1}(R_I,.)$ with g_1. In each Ω_i the iteration is started with the constant initial guess

$$u_i^0(x,t) = \min\Big\{ \inf_{x'\in\Omega}\{u_0(x')\}, \min_{t'\in[0,T]}\{g_0(t')\}, \min_{t'\in[0,T]}\{g_1(t')\}\Big\}. \tag{4.3}$$

For $i = 1,\dots,I$ we define the error $e_i^n(x,t) := u(x,t) - u_i^n(x,t)$ in $\Omega_i \times (0,T)$. Note that $e_0^n(0,t) = e_{I+1}^n(1,t) = 0$ and also $e_i^n(x,0) = 0$. The choice (4.3) ensures the nonnegativity of the error for all iterations and all subdomains. We also introduce the maximum error e^n defined by

$$e^n = \max_{i=1,\dots,I}\Big\{ \sup_{0\le t\le T}\{e_{i+1}^n(R_i,t)\}, \sup_{0\le t\le T}\{e_{i-1}^n(L_i,t)\}\Big\}.$$

THEOREM 4.1. [illegible] $L > 0$

$$L := \min_{i=1,\dots,I-1}(R_i - L_{i+1}). \tag{4.4}$$

[illegible] (4.2) [illegible] $I > 2$ [illegible] (4.1) [illegible] I [illegible] $T > 0$ [illegible]

$$e^{2n} \le e^{\frac{Cn(T+L)}{\varepsilon}} \operatorname{erfc}\left(\frac{nL}{\sqrt{\varepsilon T}}\right) e^0. \tag{4.5}$$

[illegible] C [illegible] ε T I [illegible] n [illegible]. We first derive an explicit bound for the error by successive construction of supersolutions. The error e_i^n is a solution of the problem

$$\begin{aligned} \frac{\partial e_i^n}{\partial t} + f'(u)\frac{\partial e_i^n}{\partial x} + \theta_i^n e_i^n &= \varepsilon \frac{\partial^2 e_i^n}{\partial x^2} && \text{in } \Omega_i \times (0,T),\\ e_i^n(\cdot,0) &= 0 && \text{in } \Omega_i,\\ e_i^n(L_i,\cdot) &= e_{i-1}^{n-1}(L_i,\cdot) && \text{on } [0,T],\\ e_i^n(R_i,\cdot) &= e_{i+1}^{n-1}(R_i,\cdot) && \text{on } [0,T], \end{aligned} \tag{4.6}$$

where the function θ_i^n is defined analogously to (3.6). One can check that $e_i^n = e_{iL}^n + e_{iR}^n$, where e_{iL}^n satisfies

$$\begin{aligned} \frac{\partial e_{iL}^n}{\partial t} + f'(u)\frac{\partial e_{iL}^n}{\partial x} + \theta_i^n e_{iL}^n &= \varepsilon \frac{\partial^2 e_{iL}^n}{\partial x^2} && \text{in } \Omega_i \times (0,T),\\ e_{iL}^n(\cdot,0) &= 0 && \text{in } \Omega_i,\\ e_{iL}^n(L_i,\cdot) &= 0 && \text{on } [0,T],\\ e_{iL}^n(R_i,\cdot) &= e_{i+1}^{n-1}(R_i,\cdot) && \text{on } [0,T], \end{aligned}$$

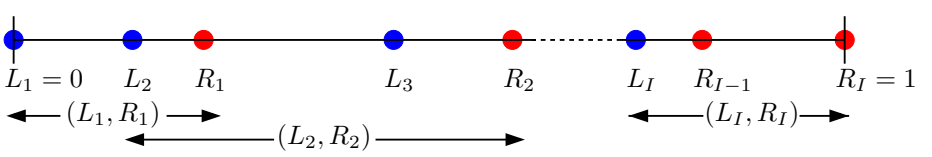


FIG. 3. *Sketch of the multidomain decomposition for the interval* $(0,1)$.

and e_{iR}^n satisfies

$$\begin{aligned} \frac{\partial e_{iR}^n}{\partial t} + f'(u)\frac{\partial e_{iR}^n}{\partial x} + \theta_i^n e_{iR}^n &= \varepsilon \frac{\partial^2 e_{iR}^n}{\partial x^2} && \text{in } \Omega_i \times (0,T), \\ e_{iR}^n(\cdot,0) &= 0 && \text{in } \Omega_i, \\ e_{iR}^n(L_i,\cdot) &= e_{i-1}^{n-1}(L_i,\cdot) && \text{in } [0,T], \\ e_{iR}^n(R_i,\cdot) &= 0 && \text{in } [0,T]. \end{aligned}$$

Furthermore, the classical solution $\tilde{e}_{iL}^n : (-\infty, R_i] \times \mathbb{R} \to \mathbb{R}$ of the quarter-space problem

$$\begin{aligned} \frac{\partial \tilde{e}_{iL}^n}{\partial t} + f'(u)\frac{\partial \tilde{e}_{iL}^n}{\partial x} + \theta_i^n \tilde{e}_{iL}^n &= \varepsilon \frac{\partial^2 \tilde{e}_{iL}^n}{\partial x^2} && \text{in } (-\infty, R_i] \times (0,T), \\ \tilde{e}_{iL}^n(\cdot,0) &= 0 && \text{in } (-\infty, R_i), \\ \tilde{e}_{iL}^n(R_i,\cdot) &= e_{i+1}^{n-1}(R_i,t) && \text{on } [0,T], \end{aligned} \tag{4.7}$$

with solutions decaying at infinity, $\lim_{x\to-\infty} \tilde{e}_{iL}^n(x,t) = 0$ for $t \in [0,T]$, satisfies

$$\tilde{e}_{iL}^n(x,t) \geq e_{iL}^n(x,t), \quad x \in \Omega_i,\ t \in (0,T).$$

The functions $f(u)$ and θ_i^n are so far defined in $\Omega \times (0,T)$; outside this set $f(u)$ and θ_i^n in the quarter-plane problem (4.7) have to be understood as smooth extensions taking only values from the range of $f(u)$ and θ_i^n in $[0,1]\times[0,T]$. Similarly we obtain a function $\tilde{e}_{iR}^n$ with $\tilde{e}_{iR}^n \geq e_{iR}^n$ in Ω_i. In Lemma 3.1 we have constructed supersolutions for the classical solutions of quarter-space problems. Using the same technique here, we obtain the function

$$\bar{e}_{iL}^n(x,t) = \exp(p_{iL}x + q_{iL}t) \int_0^t K_{iL}(x,t-\tau) g_{iL}(\tau)\, d\tau,$$

which satisfies

$$\bar{e}_{iL}^n(x,t) \geq \tilde{e}_{iL}^n(x,t), \quad x \in \Omega_i,\ t \in (0,T), \tag{4.8}$$

and solves the constant coefficient problem

$$\begin{aligned} \frac{\partial \bar{e}_{iL}^n}{\partial t} + a_{iL}\frac{\partial \bar{e}_{iL}^n}{\partial x} + b_{iL}\bar{e}_{iL}^n &= \varepsilon \frac{\partial^2 \bar{e}_{iL}^n}{\partial x^2} && \text{in } (-\infty, R_i) \times (0,T), \\ \bar{e}_{iL}^n(\cdot,0) &= 0 && \text{in } (-\infty, R_i), \\ \bar{e}_{iR}^n(R_i,t) &= \exp(\sigma_{iL}t) \sup_{0\leq\tau\leq t} e_{i+1}^{n-1}(R_i,\tau), && t \in [0,T], \end{aligned}$$

where the constants are defined as in Lemma 3.1 by

$$\begin{gathered} p_{iL} := \frac{a_{iL}}{2\varepsilon}, \quad q_i := -\frac{a_{iL}^2}{4\varepsilon} - b_{iL}, \\ a_{iL} := \inf_{(x,t)\in\Omega_i} \{f'(u(x,t))\}, \quad b_{iL} := \inf_{(x,t)\in\Omega_i} \left\{\theta_i^n(x,t) + (f'(u(x,t)) - a_{iL})\frac{a_{iL}}{2\varepsilon}\right\}, \\ \sigma_{iL} := \begin{cases} q_{iL} & : \quad q_{iL} \geq 0, \\ 0 & : \quad \text{otherwise}, \end{cases} \end{gathered}$$

and the kernel function and g_{iL} in the integral are given by

$$\begin{aligned} K_{iL,x}(x,t) &:= -\frac{1}{2\sqrt{\pi}} \frac{x-R_i}{\varepsilon^{1/2} t^{3/2}} \exp\left(-\frac{(x-R_i)^2}{4\varepsilon t}\right), \\ g_{iL}(t) &:= \exp(-p_{iL}R_i + (\sigma_{iL} - q_{iL})t) \sup_{0\leq\tau\leq t} \{e_{i+1}^{n-1}(R_i,\tau)\}. \end{aligned}$$

With the analogous definitions for p_{iR}, q_{iR}, a_{iR}, b_{iR}, σ_{iR}, and

$$K_{iR,x}(x,t) := \frac{1}{2\sqrt{\pi}} \frac{x-L_i}{\varepsilon^{1/2}t^{3/2}} \exp\left(-\frac{(x-L_i)^2}{4\varepsilon t}\right),$$
$$g_{iR}(t) := \exp(-p_{iR}L_i + (\sigma_{iR} - q_{iR})t) \sup_{0\le\tau\le t}\{e_{i-1}^{n-1}(L_i,\tau)\},$$

we define the function

$$\bar{e}_{iR}^n(x,t) = \exp(p_{iR}x + q_{iR}t) \int_0^t K_{iR}(x,t-\tau) g_{iR}(\tau)\, d\tau,$$

which is an upper bound on the quarter-plane problem solution,

$$\bar{e}_{iR}^n(x,t) \ge \tilde{e}_{iR}^n(x,t), \quad x \in \Omega_i,\ t \in (0,T). \tag{4.9}$$

From (4.8), (4.9), and the explicit formulas for the supersolutions we derive

$$\begin{aligned} e_i^n(x,t) &\le \bar{e}_{iL}^n(x,t) + \bar{e}_{iR}^n(x,t) \\ &= \exp(p_{iL}x + q_{iL}t) \int_0^t K_{iL,x}(x,t-\tau) g_{iL}(\tau) d\tau \\ &\quad + \exp(p_{iR}x + q_{iR}t) \int_0^t K_{iR,x}(x,t-\tau) g_{iR}(\tau) d\tau. \end{aligned}$$

Then there exists a constant $\widetilde{C} \ge 0$ that depends on $p_{iL/R}$, $q_{iL/R}$, $a_{iL/R}$, $b_{iL/R}$, $\sigma_{iL/R}$, the time T, and the decomposition L_1, R_1, …, L_I, R_I such that

$$\begin{aligned} &\max\{e_i^n(R_{i-1},t), e_i^n(L_{i+1},t)\} \\ &\le \widetilde{C} \left(\int_0^t K_{iL,x}(R_{i-1}, t-\tau)\, d\tau + \int_0^t K_{iR,x}(R_{i-1}, t-\tau)\, d\tau \right. \\ &\quad \left. + \int_0^t K_{iL,x}(L_{i+1}, t-\tau)\, d\tau + \int_0^t K_{iR,x}(L_{i+1}, t-\tau)\, d\tau \right) e^{n-1} \\ &\le 2\widetilde{C} \left(\int_0^t K_{iR,x}(R_{i-1}, t-\tau)\, d\tau + \int_0^t K_{iL,x}(L_{i+1}, t-\tau)\, d\tau \right) e^{n-1}. \end{aligned}$$

The last inequality is a consequence of the fact that the integrals are solutions of quarter-space problems for the heat equations where the boundary function is monotonically increasing. Therefore the solutions decay monotonically in space if one moves in the direction where the domain is unbounded. Thus the long-range error contribution involving $K_{iL,x}(R_{i-1}, t-\tau)$ and $K_{iR,x}(L_{i+1}, t-\tau)$ can be estimated by the short-range error contributions. Similarly one can now also estimate the contributions from the various overlap sizes using the minimum of the overlaps: letting $K_{L,x}(t) := \frac{1}{2\sqrt{\pi}} \frac{L}{\varepsilon^{1/2}t^{3/2}} \exp(-\frac{L^2}{4\varepsilon t})$, we obtain

$$\max\{e_i^n(R_{i-1},t), e_i^n(L_{i+1},t)\} \le 4\widetilde{C} \int_0^t K_{L,x}(t-\tau) d\tau e^{n-1}.$$

As in the proof of Lemma 3.1 we denote now by $K_{L,x}^{2n}(t)$ the $2n$-fold convolution of the kernel $K_{L,x}(t)$ and obtain

$$e^{2n} \le (4\widetilde{C})^{2n} K_{L,x}^{2n}(t) e^0 \le (4\widetilde{C})^{2n} \operatorname{erfc}\left(\frac{nL}{\sqrt{\varepsilon t}}\right) e^0,$$

which is formula (4.5). ∎

5. Behavior of the algorithm over long time intervals. The superlinear convergence estimates for the overlapping Schwarz waveform relaxation algorithm derived in the previous sections depend on the time interval under consideration. To get more insight into the behavior of the algorithm over long time intervals, we study in this section the limiting case of a steady state solution and we apply an overlapping Schwarz method to this steady state case. We study here only the Burgers equation to illustrate that the convergence rate can be arbitrarily slow in this case, and convergence can be lost if the viscosity goes to zero. Similar behavior has been in fact observed for long time calculations for the Burgers equation in [GK00].

5.1. Boundary value problem. We consider the boundary value problem obtained for the steady state case of the Burgers equation on the domain $\Omega = (-1, 1)$,

$$\varepsilon \frac{\partial^2 u^\varepsilon}{\partial x^2} - \frac{1}{2}\frac{\partial (u^\varepsilon)^2}{\partial x} = 0, \qquad u^\varepsilon(-1) = 1, \quad u^\varepsilon(1) = -1. \tag{5.1}$$

The general solution for the differential equation (5.1) can be computed in closed form,

$$u^\varepsilon(x) = -\sqrt{2}C \tanh\left(C\frac{x+D}{\sqrt{2}\varepsilon}\right), \quad C, D \in \mathbb{R}. \tag{5.2}$$

To satisfy the boundary conditions, the two constants $C, D \in \mathbb{R}$ have to satisfy the nonlinear system of equations

$$\frac{-1}{\sqrt{2}C} = \tanh\left(C\frac{D-1}{\sqrt{2}\varepsilon}\right), \qquad \frac{1}{\sqrt{2}C} = \tanh\left(C\frac{D+1}{\sqrt{2}\varepsilon}\right).$$

This implies in particular $D = 0$. In the limit as $\varepsilon \to 0$ the constant $C = C(\varepsilon)$ converges to $\frac{1}{\sqrt{2}}$; thus in this regime it is bounded from above and below independently of ε. Therefore, the solution u^ε contains an [illegible] layer for ε small. This type of solution is typical for [illegible] flux functions f with extrema. It does not exist for linear advection-diffusion problems which allow for (ordinary) boundary layers only. Note that we have for almost all $x \in [-1, 1]$

$$\lim_{\varepsilon \to 0} u^\varepsilon(x) = u^0(x) \equiv \begin{cases} 1 & : \quad x < 0, \\ -1 & : \quad x > 0. \end{cases}$$

We consider now the corresponding singular boundary value problem

$$\frac{\partial}{\partial x}\left(\frac{1}{2}u^2\right) = 0, \qquad u(-1) = 1, \quad u(1) = -1, \tag{5.3}$$

where the function u^0 satisfies the boundary conditions and is a distributional solution of the differential equation (5.3). Such solutions are not unique; in fact all piecewise constant functions taking only the values ± 1 and satisfying the boundary conditions at $x = \pm 1$ are solutions in that sense. This failure of uniqueness is crucial for the behavior of the overlapping Schwarz algorithm for (5.1) that we will analyze now.

5.2. Overlapping Schwarz algorithm. We first investigate the influence of the internal layer within the overlap for $\varepsilon \ll 1$. The overlapping Schwarz algorithm with the two subdomains $\Omega_1 = (-1, L)$ and $\Omega_2 = (-L, 1)$ for problem (5.1) is given

by

$$
\begin{aligned}
\varepsilon\frac{\partial u_1^n}{\partial x^2}+u_1^n\frac{\partial u_1^n}{\partial x}&=0 && \text{in } \Omega_1,\\
u_1^n(-1)&=1,\\
u_1^n(L)&=u_2^{n-1}(L),\\
\varepsilon\frac{\partial u_2^n}{\partial x^2}+u_2^n\frac{\partial u_2^n}{\partial x}&=0 && \text{in } \Omega_2,\\
u_2^n(1)&=-1,\\
u_2^n(-L)&=u_1^{n-1}(-L),
\end{aligned}
\tag{5.4}
$$

where $L>0$ is a constant that defines the overlap $2L$. The iteration is started with $u_1^0=1$, $u_2^0=-1$, and we have skipped the index ε to simplify the notation.

THEOREM 5.1 (linear convergence). [illegible] (5.4) [illegible] $0<L<1$ [illegible] (5.1). [illegible] $\varepsilon\to 0$

$$
1-2e^{-\frac{1-L}{\varepsilon}}+o(e^{-\frac{1}{\varepsilon}}).
\tag{5.5}
$$

[illegible]. From (5.2) we obtain

$$
u_1^n(x)=-\sqrt{2}C_1\tanh\left(C_1\tfrac{x+D_1}{\sqrt{2}\varepsilon}\right),
$$

where the two constants of integration C_1 and D_1 are determined by the boundary conditions. This leads to the system of nonlinear equations for C_1 and D_1,

$$
\begin{aligned}
D_1&=1-\tfrac{\sqrt{2}\varepsilon}{C_1}\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_1}\right),\\
u_2^{n-1}(L)&=-\sqrt{2}C_1\tanh\left(C_1\tfrac{L+D_1}{\sqrt{2}\varepsilon}\right).
\end{aligned}
$$

Inserting the first equation into the second one, we find the transcendental equation

$$
u_2^{n-1}(L)=\sqrt{2}C_1\tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_1}\right)-\tfrac{C_1(1+L)}{\sqrt{2}\varepsilon}\right),
\tag{5.6}
$$

which defines the constant C_1 implicitly. Denoting its solution by $C_1(u_2^{n-1}(L))$, we find over one iteration

$$
u_1^n(x)=\sqrt{2}C_1(u_2^{n-1}(L))\tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_1(u_2^{n-1}(L))}\right)-\tfrac{C_1(u_2^{n-1}(L))(1+x)}{\sqrt{2}\varepsilon}\right).
\tag{5.7}
$$

Similarly we find for the solution on the second subdomain

$$
u_2^n(x)=\sqrt{2}C_2(u_1^{n-1}(-L))\tanh\left(\operatorname{arctanh}\left(\tfrac{C_2(u_1^{n-1}(-L))(1-x)}{\sqrt{2}\varepsilon}-\tfrac{1}{\sqrt{2}C_2(u_1^{n-1}(-L))}\right)\right),
$$

where the function $C_2(u_1^{n-1}(-L))$ is implicitly defined by the transcendental equation

$$
u_1^{n-1}(-L)=-\sqrt{2}C_2\tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_2}\right)-\tfrac{C_2(1+L)}{\sqrt{2}\varepsilon}\right).
\tag{5.8}
$$

We are interested in the asymptotic convergence of this fixed point iteration on the interfaces $x=-L$ and $x=L$. We could evaluate $u_2^{n-1}(x)$ at $x=L$ and insert the result into the relation for $u_1^n(x)$ to find directly a nonlinear relation between $u_1^n(-L)$ and $u_1^{n-2}(-L)$ and analyze the fixed point of this iteration. But this would lead to

formulas of twice the complexity of the formulas of the single iteration. To avoid this, we define the two auxiliary functions $g(C_1) := u_1^n(-L, C_1)$ and $h(C_2) := u_2^{n-1}(L, C_2)$. With those, the above double step can be written as

$$u_1^n(-L) = g(C_1(u_2^{n-1}(L))) = g\big(C_1(h(C_2(u_1^{n-2}(-L))))\big). \tag{5.9}$$

Now note that $g(y) = -h(y)$ and $C_1(y) = C_2(-y)$, which leads to

$$\begin{aligned} u_1^n(-L) &= g\big(C_1(h(C_2(u_1^{n-2}(-L))))\big) \\ &= g\big(C_1(-g(C_2(u_1^{n-2}(-L))))\big) \\ &= g\big(C_2(g(C_2(u_1^{n-2}(-L))))\big), \end{aligned}$$

and hence it suffices to analyze the fixed point iteration

$$y^n = G(y^n) := g(C_2(y^n)), \tag{5.10}$$

where by (5.7) the function $g(C_2)$ is defined by

$$g(C_2) = \sqrt{2}C_2 \tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_2}\right) - \tfrac{C_2(1-L)}{\sqrt{2}\varepsilon}\right), \tag{5.11}$$

and the function $C_2(y)$ is defined implicitly by

$$y = -\sqrt{2}C_2 \tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_2}\right) - \tfrac{C_2(1+L)}{\sqrt{2}\varepsilon}\right); \tag{5.12}$$

see (5.8). To analyze the convergence of this fixed point iteration, we need to study the derivative

$$G'(y) = \frac{d}{dy}g(C_2(y)) = \frac{dg}{dC_2}\frac{dC_2(y)}{dy} = \frac{dg}{dC_2}\left(\frac{dy}{dC_2}\right)^{-1} =: \widetilde{G}'(C_2) \tag{5.13}$$

at the fixed point y^* on the left or for the expression on the right at the corresponding C_2^*. Note that the fixed point iteration depends on ε; we have $\widetilde{G}'(C_2, \varepsilon)$, and we want to study this function at the fixed point $C_2^*(\varepsilon)$ which also depends on ε. We thus need to expand $\widetilde{G}'(C_2^*(\varepsilon), \varepsilon)$ for small ε. Unfortunately the expression $C_2^*(\varepsilon)$ is not available in closed form, but we have the inverse, $\varepsilon(C_2^*)$ from the equation of the solution $u(-1, C_2^*(\varepsilon)) = 1$, which gives

$$1 = -\sqrt{2}C_2^* \tanh\left(\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_2^*}\right) - \tfrac{2C_2^*}{\sqrt{2}\varepsilon}\right)$$

or

$$\varepsilon = \varepsilon(C_2^*) = \frac{C_2^*}{\sqrt{2}\operatorname{arctanh}\left(\tfrac{1}{\sqrt{2}C_2^*}\right)}. \tag{5.14}$$

Hence we expand $\widetilde{G}'(C_2^*, \varepsilon(C_2^*))$ about $C_2^* = \frac{1}{\sqrt{2}}$ which corresponds to $\varepsilon = 0$. To simplify the expansion, we perform a further change of variables,

$$d = C_2^* - 1/\sqrt{2}, \tag{5.15}$$

with which we find, after simplifying, the expression

$$\widetilde{G}'(d) = \frac{2d(\sqrt{2}+d)(T^2-1)(L-1)A - (\sqrt{2}Td + T - 1)(T + \sqrt{2}d + 1)}{2d(\sqrt{2}+d)(T^2-1)(L+1)A + (\sqrt{2}Td + T + 1)(T - \sqrt{2}d - 1)}, \tag{5.16}$$

where the functions $A = A(d)$ and $T = T(d, L)$ are given by

$$A(d) = \operatorname{arctanh}\left(\frac{1}{\sqrt{2}d+1}\right), \quad T(d,L) = \tanh(LA(d)). \tag{5.17}$$

The difficulty in the expansion process lies in the fact that L is an arbitrary real number in the hyperbolic tangent in the function $T(d, L)$. We first expand $A(d)$ for d small, taking into account the singularity,

$$A(d) = -\ln\left(\frac{\sqrt{d}}{2^{\frac{1}{4}}}\right) + \frac{1}{2\sqrt{2}}d - \frac{1}{8}d^2 + \mathcal{O}(d^3). \tag{5.18}$$

Hence $A(d)$ tends to infinity as d goes to zero. We now use an asymptotic expansion of

$$\tanh(\ln(z)) = 1 - \frac{2}{z^2} + \frac{2}{z^4} - \frac{2}{z^6} + \mathcal{O}\left(\frac{1}{z^8}\right)$$

and insert $z := \exp(LA)$ into this expansion. This leads to

$$\begin{aligned} T(d,L) = 1 - 2^{1-\frac{L}{2}}d^L + 2^{1-L}d^{2L} - 2^{1-\frac{3}{2L}}d^{3L} \\ + \cdots + L2^{\frac{1-L}{2}}d^{1+L} + o(d^{1+L}). \end{aligned} \tag{5.19}$$

Using the structure of (5.16), we see that the numerator and the denominator contain the same terms, but some with the sign changed. Denoting by $p = 2d(\sqrt{2}+d)(T^2-1)$ the factor in front of A and rearranging, we find

$$\widetilde{G}'(d) = -\frac{(1+\sqrt{2}d)(1-T^2) - 2Td(\sqrt{2}+d) + p(L-1)A}{(1+\sqrt{2}d)(1-T^2) + 2Td(\sqrt{2}+d) - p(L+1)A},$$

where in both the numerator and the denominator the first term is $O(d^L)$, the second is $O(d)$, and the third is $O(d^{1+L}\log(d))$. Hence for d small, the last term can be neglected. Inserting the expressions (5.19) of T and (5.18) of A and expanding again for d small, we get for the leading order term

$$\widetilde{G}'(d) \approx -1 + \sqrt{2}2^{L/2}d^{1-L}.$$

To find the asymptotic convergence rate as a function of ε, we expand the implicit relation between d and ε given in (5.14) and (5.15), which leads to

$$\varepsilon = \frac{1}{\ln(2) - \ln(\sqrt{2}d)} + \mathcal{O}(d),$$

and thus $d \approx \sqrt{2}e^{-\frac{1}{\varepsilon}}$, which inserted into the asymptotic convergence rate gives the result (5.5). □

Theorem 5.1 shows that the convergence rate of the Schwarz algorithm applied to this model problem tends exponentially fast to 1 in modulus as ε goes to zero. This indicates that the overlapping Schwarz waveform relaxation algorithm applied to nonlinear convection-dominated conservation laws over long time intervals can be slow if there are steady shock waves in the overlap.

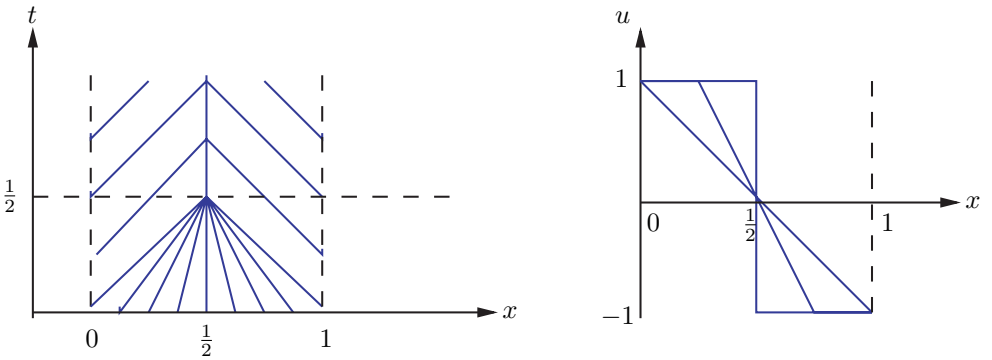


FIG. 4. *The left picture displays the characteristics for the entropy solution* u^0 *of* (6.1) *colliding at* $(1/2, t)$ *for* $t \geq 1/2$. *The right graph shows* u^0 *for* $t = 0, 1/4$ *and* $t \geq 1/2$.

6. Numerical experiments. We first illustrate the superlinear convergence rate of the overlapping Schwarz waveform relaxation algorithm given in Theorem 3.2. We solve the following initial boundary value problem for the Burgers equation with $\varepsilon \geq 0$ on the bounded domain $\Omega = (0, 1)$:

$$
\begin{aligned}
u_t^\varepsilon + \Big(\frac{(u^\varepsilon)^2}{2}\Big)_x &= \varepsilon u_{xx}^\varepsilon && \text{in } \Omega \times (0,T),\\
u^\varepsilon(\cdot,0) &= 1 - 2x && \text{in } \Omega,\\
u^\varepsilon(0,\cdot) &= 1 && \text{on } [0,T],\\
u^\varepsilon(1,\cdot) &= -1 && \text{on } [0,T].
\end{aligned}
\tag{6.1}
$$

For the case $\varepsilon = 0$ we can compute the entropy solution of (6.1) explicitly: for $(x, t) \in [0, 1] \times [0, 1/2)$ we find

$$
u^0(x,t) = \begin{cases} 1 & : \quad x < t, \\ \dfrac{2}{-1+2t}\Big(x - \dfrac{1}{2}\Big) & : \quad t \leq x \leq 1-t, \\ -1 & : \quad x > 1-t, \end{cases}
$$

and for $(x, t) \in [0, 1] \times [1/2, \infty)$ the entropy solution is

$$
u^0(x,t) = \begin{cases} 1 & : \quad x < 1/2, \\ -1 & : \quad x > 1/2. \end{cases}
$$

The solution is continuous (but not classical) up to $t = 1/2$ and then produces a stationary shock for all times; see Figure 4. We use the overlapping Schwarz waveform relaxation algorithm with the two subdomains $\Omega_1 = (0, \frac{1}{2} + L)$ and $\Omega_2 = (\frac{1}{2} - L, 1)$ with the overlap parameter $L = 0.1$. We choose $T = 0.6$ and a centered finite difference scheme in space, explicit for the nonlinear term and implicit for the Laplacian. The discretization parameters are $\Delta x = 0.01$ and $\Delta t = 0.003$. Figure 5 shows the superlinear convergence behavior of the overlapping Schwarz waveform relaxation algorithm for various values of the viscosity parameter ε. This numerical experiment shows that the smaller the viscosity is, the faster the algorithm converges, as predicted by the analysis, when the algorithm is in the superlinear convergence regime.

In the next numerical experiment, we solve the steady viscous Burgers equation

on the domain $\Omega = (-1, 1)$,

$$\left(\frac{(u^\varepsilon)^2}{2}\right)_x = \varepsilon u_{xx} \qquad \text{in } \Omega,$$
$$u(-1) = 1,$$
$$u(1) = -1,$$

using an overlapping Schwarz method with the two subdomains $\Omega_1 = (-1, L)$ and $\Omega_2 = (-L, 1)$ and the overlap parameter $L = 0.2$. The solution of this problem consists of a shock centered at $x = 0$. We discretize the problem with the same centered finite difference scheme as in the evolution case and solve the nonlinear system of equations with iteration in time. Figure 6 shows the convergence behavior of the overlapping Schwarz method for various values of the viscosity parameter ε. This experiment shows that the overlapping Schwarz algorithm converges linearly, and one can see that the smaller the viscosity parameter ε is, the slower the algorithm becomes, as predicted by Theorem 5.1. Since the convergence rate is exponential in the viscosity, already for a moderate value of the viscosity parameter ε, the algorithm seems to stand still.

7. Conclusions. We have analyzed the performance of the overlapping Schwarz waveform relaxation algorithm applied to convection-dominated nonlinear conservation laws. We have proved that the algorithm's asymptotic convergence rate is superlinear and independent of the number of subdomains. The convergence rate depends strongly on the viscosity parameter: the smaller the viscosity is, the faster the algorithm becomes. To learn more about the algorithm's performance over long time intervals, we have also analyzed the convergence rate of the overlapping Schwarz method applied to a special steady case of the Burgers equation. This analysis revealed that the asymptotic convergence rate is linear and depends exponentially on the viscosity

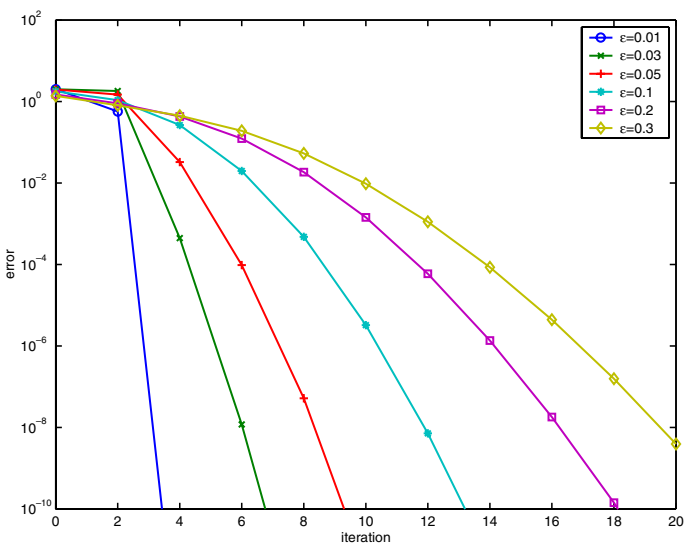


FIG. 5. *Superlinear convergence behavior of the overlapping Schwarz waveform relaxation algorithm for the viscous Burgers equation with various values for the viscosity parameter ε.*

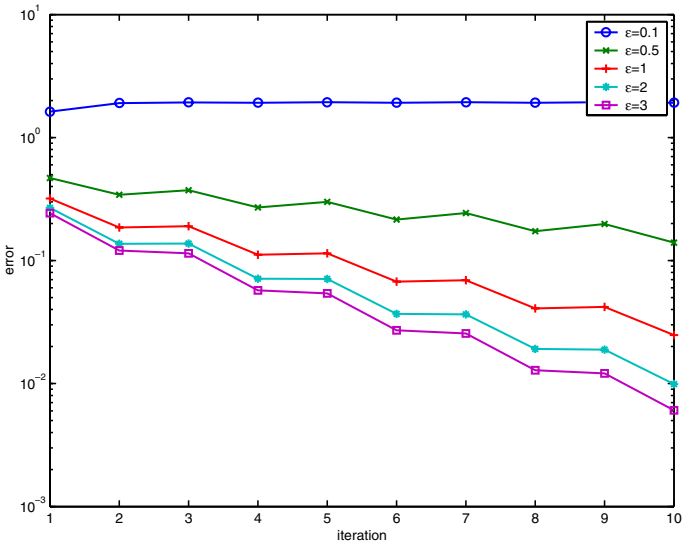


FIG. 6. *Linear convergence behavior of the overlapping Schwarz method for the steady viscous Burgers equation for various values of the viscosity parameter ε.*

parameter. In contrast to the superlinear case, however, we showed that the smaller the viscosity is, the slower the algorithm becomes.

REFERENCES

[BGT97] A. BAMBERGER, R. GLOWINSKI, AND Q. H. TRAN, *A domain decomposition method for the acoustic wave equation with discontinuous coefficients and grid change*, SIAM J. Numer. Anal., 34 (1997), pp. 603–639.

[Bjø95] M. BJØRHUS, *On Domain Decomposition, Subdomain Iteration and Waveform Relaxation*, Ph.D. thesis, University of Trondheim, Norway, 1995.

[Cai91] X.-C. CAI, *Additive Schwarz algorithms for parabolic convection-diffusion equations*, Numer. Math., 60 (1991), pp. 41–61.

[Cai94] X.-C. CAI, *Multiplicative Schwarz methods for parabolic problems*, SIAM J. Sci. Comput., 15 (1994), pp. 587–603.

[CL96] H. CHEN AND R. D. LAZAROV, *Domain splitting algorithm for mixed finite element approximations to parabolic problems*, East-West J. Numer. Math., 4 (1996), pp. 121–135.

[Daf00] C. M. DAFERMOS, *Hyperbolic Conservation Laws in Continuum Physics*, 1st ed., Springer-Verlag, New York, 2000.

[DD91] C. N. DAWSON AND Q. DU, *A domain decomposition method for parabolic equations based on finite elements*, in Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, 1991, pp. 255–263.

[DLN00] V. DOLEAN, S. LANTERI, AND F. NATAF, *Convergence analysis of a Schwarz type domain decomposition method for the solution of the Euler equations*, Technical report 3916, INRIA, Le Chesnay, France, 2000.

[Dry91] M. DRYJA, *Substructuring methods for parabolic problems*, in Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, 1991, pp. 264–271.

[Fri64] A. FRIEDMAN, *Partial Differential Equations of Parabolic Type*, Prentice–Hall, Englewood Cliffs, NY, 1964.

[Gan97] M. J. Gander, *Analysis of Parallel Algorithms for Time Dependent Partial Differential Equations*, Ph.D. thesis, Stanford University, Stanford, CA, 1997.

[Gan98] M. J. Gander, *A waveform relaxation algorithm with overlapping splitting for reaction diffusion equations*, Numer. Linear Algebra Appl., 6 (1998), pp. 125–145.

[GHN99] M. J. Gander, L. Halpern, and F. Nataf, *Optimal convergence for overlapping and non-overlapping Schwarz waveform relaxation*, in Eleventh International Conference of Domain Decomposition Methods, C.-H. Lai, P. Bjørstad, M. Cross, and O. Widlund, eds., DDM.org, Augsburg, 1999, pp. 27–36.

[GK97] M. Garbey and H. G. Kaper, *Heterogeneous domain decomposition for singularly perturbed elliptic boundary value problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1513–1544.

[GK00] M. Garbey and H. G. Kaper, *Asymptotic-numerical study of supersensitivity for generalized Burgers' equation*, SIAM J. Sci. Comput., 22 (2000), pp. 368–385.

[GK02] E. Giladi and H. B. Keller, *Space time domain decomposition for parabolic problems*, Numer. Math., 93 (2002), pp. 279–313.

[GZ97] M. J. Gander and H. Zhao, *Overlapping Schwarz waveform relaxation for parabolic problems in higher dimension*, in Proceedings of Algoritmy 14, A. Handlovičová, M. Komorníkova, and K. Mikula, eds., Slovak Technical University, 1997, pp. 42–51.

[GZ02] M. J. Gander and H. Zhao, *Overlapping Schwarz waveform relaxation for the heat equation in n-dimensions*, BIT, 42 (2002), pp. 779–795.

[LO95] J. G. L. Laforgue and R. E. O'Malley, Jr., *Shock layer movement for Burgers' equation*, SIAM J. Appl. Math., 55 (1995), pp. 332–347.

[LRSV82] E. Lelarasmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, *The waveform relaxation method for time-domain analysis of large scale integrated circuits*, IEEE Trans. CAD IC Syst., 1 (1982), pp. 131–145.

[LSU68] O. A. Ladyzenskaja, V. A. Solonnikov, and N. N. Ural'ceva, *Linear and Quasilinear Equations of Parabolic Type*, AMS, Providence, RI, 1968.

[Meu91] G. A. Meurant, *Numerical experiments with a domain decomposition method for parabolic problems on parallel computers*, in Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, Y. A. Kuznetsov, G. A. Meurant, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, 1991, pp. 394–408.

[MNRR96] J. Malek, J. Necas, M. Rokyta, and M. Ruzicka, *Weak and Measure-valued Solutions to Evolutionary PDEs*, 1st ed., Chapman & Hall, London, 1996.

[MS87] J. C. Meza and W. W. Symes, *Domain Decomposition Algorithms for Linear Hyperbolic Equations*, Technical report 20, Rice University, Houston, TX, 1987.

[RZ94] R. Rannacher and G. Zhou, *Analysis of a domain-splitting method for nonstationary convection-diffusion problems*, East-West J. Numer. Math., 2 (1994), pp. 151–174.

[Sch70] H. A. Schwarz, *Über einen Grenzübergang durch alternierendes Verfahren*, Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, 15 (May 1870), pp. 272–286.

[WCK98] Y. Wu, X.-C. Cai, and D. E. Keyes, *Additive Schwarz methods for hyperbolic equations*, in Tenth International Conference on Domain Decomposition Methods, Contemp. Math. 218, J. Mandel, C. Farhat, and X.-C. Cai, eds., AMS, Providence, RI, 1998, pp. 468–476.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 440–457
© 2005 Society for Industrial and Applied Mathematics

DOMAIN DECOMPOSITION SOLUTION OF ELLIPTIC BOUNDARY-VALUE PROBLEMS VIA MONTE CARLO AND QUASI-MONTE CARLO METHODS*

JUAN A. ACEBRÓN†, MARIA PIA BUSICO‡, PIERO LANUCARA‡, AND RENATO SPIGLER§

Abstract. Domain decomposition of two-dimensional domains on which boundary-value elliptic problems are formulated is accomplished by probabilistic (Monte Carlo) as well as by quasi-Monte Carlo methods, generating only a few interfacial values and interpolating on them. Continuous approximations for the trace of solution are thus obtained, to be used as boundary data for the subproblems. The numerical treatment can then proceed by standard deterministic algorithms, separately in each of the so obtained subdomains. Monte Carlo and quasi-Monte Carlo simulations may naturally exploit multiprocessor architectures, leading to *parallel computing*, as well as the ensuing domain decomposition does. The advantages such as *scalability* obtained by increasing the number of processors are shown, both theoretically and experimentally, in a number of test examples, and the possibility of using *clusters* of computers (grid computing) is emphasized.

Key words. Monte Carlo methods, quasi-Monte Carlo methods, domain decomposition, parallel computing

AMS subject classifications. 65C05, 65C30, 65N55

DOI. 10.1137/030600692

1. Introduction.

The probabilistic representation of solutions to elliptic boundary-value problems, e.g., the Dirichlet problem for the Laplace equation,

$$\Delta u = 0 \quad \text{in } \Omega, \qquad u|_{\partial\Omega} = g, \tag{1.1}$$

with boundary data, g, and boundary $\partial\Omega$ of $\Omega \subset \mathbf{R}^d$ in suitable classes, has been known for a long time [10, 17], as has the possibility of using it to obtain numerical approximations of solutions. Such representation is given by

$$u(x) = E_x \left[g(W(\tau_{\partial\Omega})) \right], \tag{1.2}$$

where $W(\cdot)$ represents a path of the standard Brownian motion (also called Wiener process) starting at point $x \in \Omega$, and $\tau_{\partial\Omega}$ is the first passage (hitting) time of the path $W(\cdot)$ started at x to $\partial\Omega$. This approach, which can be called a Monte Carlo approach, is, however, considered very inefficient, at least in low dimension, unless a

*Received by the editors August 13, 2003; accepted for publication (in revised form) November 4, 2004; published electronically September 23, 2005. This work was supported, in part, by the Italian National Project in Numerical Analysis "Scientific Computing: Innovative Models and Numerical Methods" (2001-2002), GNFM-INdAM, CNR International Short-Term Mobility Program for Scientists/Researchers, CASPUR, and UNESCO contracts UVO-ROSTE n. 875.704.0 (2000) and n. 875.646.2 (2001-2002).

http://www.siam.org/journals/sisc/27-2/60069.html

†Departamento de Automatica, Escuela Politécnica, Universidad de Alcala, Crta. Madrid-Barcelona, Km.31,600, 28871 Alcala de Henares, Spain, and Dipartimento di Ingegneria dell' Informazione, Università di Padova, Via Gradenigo 6/B, 35131 Padova, Italy (juan.acebron@uah.es).

‡CASPUR, Via dei Tizii 6 b, 00185 Rome, Italy (MariaPia.Busico@caspur.it, lanucara@caspur.it).

§Dipartimento di Matematica, Università di "Roma Tre," Largo S.L. Murialdo 1, 00146 Rome, Italy (spigler@mat.uniroma3.it).

very complicated geometry of the boundary, $\partial\Omega$, of the domain Ω, rules out any other deterministic algorithm; see, e.g., [22].

In this paper we propose a domain decomposition method for general elliptic problems, accomplished by generating values inside the domain, Ω, in two dimensions, and interpolating on the points where the values above have been computed, taken as nodes. These nodes are considered as placed on suitable interfaces, internal to the domain; thus continuous approximations of the trace of solutions to be used as subdomain boundary data can be obtained. In this way, full decoupling into as many subdomains as we wish can be realized. The idea of using a probabilistic representation of solutions to elliptic problems to accomplish a preliminary domain decomposition was first put forth in [36]; the same was also suggested later in [38]. Recently, a probabilistic numerical treatment of elliptic and parabolic problems was presented in [29]. The results obtained there, however, did not show any improvement with respect to fully deterministic methods, but this was due to the very small time-steps used, lacking a suitable boundary treatment. We shall refer to these methods as "probabilistic domain decomposition" (PDD) methods, or "domain decomposition by Monte Carlo" methods. We stress that the approach followed here fully exploits in that (a) it implements a domain decomposition algorithm and (b) realization (or path) of the stochastic processes starting at every point can be simulated independently (if we generate N sample paths at m points, we can use up to mN independent processors). Clearly, such a degree of parallelization is compatible with the use of possibly different and even geographically distant computers (grid computing) and/or clusters of them, and a balance among them is also feasible.

In order to improve the performance of the "classical" Monte Carlo method [16] (based on the so-called numbers, which mimic the ideal random numbers), we explored the possibility of using, instead, sequences of numbers [5, 25, 27, 28]. The corresponding strategy is called the quasi-Monte Carlo, and using such sequences in our approach, the algorithm will be called the "quasi-probabilistic domain decomposition" (quasi-PDD) method. Such sequences are actually deterministic, and their elements are uniformly distributed, though subject to some degree of correlation. They have been successfully applied to the numerical evaluation of high dimensional integrals (cf. [26]), in particular to problems of financial mathematics [6]. Applications have been made to the generation of quasi-random paths of stochastic processes in [23, 24], to the Boltzmann equation [19], and to a simple system of diffusion equations in $\mathbf{R}^d$, subject to purely initial values on all space [20].

In section 2, some preliminaries are given, and in section 3 the numerical method is presented along with the various sources of error. In section 4, the performance of the numerical algorithm is discussed. Numerical examples are shown in section 5, where the overall numerical error, as well as speed-up results, is given to substantiate the efficiency of the PDD and quasi-PDD methods. Conclusions are drawn in the final section.

2. Generalities on domain decomposition and on Monte Carlo methods. The basic idea of solving elliptic boundary-value problems (say the Dirichlet problem, for instance) by domain decomposition is to assign the numerical solution on each subdomain into which the domain Ω is partitioned to a separate processor; see, e.g., [7, 33]. Unfortunately, since the boundary-value problems above are global in nature, the trace of the solution on the interfaces internal to Ω, to be used as

boundary values for the subproblems, cannot be obtained before solving the entire problem. In the framework of the [illegible] domain decomposition methods, it is possible to compute approximations of such values prior to passing to the implementation on several separate processors, imposing suitable continuity conditions on the interfaces. This procedure, however, requires solving, usually iteratively, certain related linear algebraic subproblems. The latter represents the algebraic formulation of the so-called discrete Steklov–Poincaré operator, which is usually unbounded. This leads to a capacitance matrix (which is the finite dimensional counterpart of the Steklov–Poincaré operator [32] and also has the form of a Schur complement matrix), that is typically ill-conditioned [33]. Preconditioning it is therefore mandatory, and finding preconditioners that are at the same time optimal and efficient is a nontrivial though fundamental problem. Optimality refers to the possibility of constructing, for a given problem, preconditioners yielding a spectral condition number (the ratio between the maximum and the minimum eigenvalue) bounded independently of the mesh size, h, as well as the subdomain (average) diameter, H. In fact, the Schur complement matrix, Σ_h, has a condition number $\kappa(\Sigma_h) = O(h^{-1}H^{-1})$. Note that the number of subdomains, M, scales as $M \sim H^{-1}$; cf. [33].

In practice, in the domain decomposition methods with overlapping (Schwarz type), the condition number above would actually be independent of h and H, provided that the overlapping is wide enough [8]; the bounds found in [8] were shown to be optimal in [3]. In the domain decomposition methods without overlap, also referred to as "iterative substructuring" methods, things are worse, and conditioning leads to bounds for condition numbers like $C\,(1 + \log(H/h))^a$, where C is a constant and a may be 1 or 2 [9]. This logarithmic estimate is considered to be not improvable; however, it is of little harm, because it grows slowly when h and/or H are reduced in size. Sometimes h is reduced at the same rate as H is reduced, in order to keep the same ratio H/h and thus the condition estimate. This operation, however, pushes to unnecessarily smaller mesh sizes, h, preventing one from taking advantage of solving smaller-size problems on separate processors. In any case, it seems that, at least with parallel computers with a few hundred processors, and facing problems with $2 \cdot 10^6$ to 10^7 variables, the overall cost of the method is always dominated by the cost of solving the local problems, for instance, the cost of factorizing into triangular factors (Choleski factorization).

Adopting a [illegible] representation of solutions, one can generate numerical approximations of the sought solution [illegible] Ω, at every single point, [illegible] solving the full problem. Of course, since such a value is an average of a number of Brownian motion paths (for the Laplace equation) starting at that point, some "globality" is felt somehow, in that the numerous Brownian paths explore the entire domain, Ω. Here the idea is to generate [illegible] values of the sought solution, on some interfaces, then interpolate on each interface to obtain continuous approximations of the "boundary values" needed to split the problem into subproblems, and finally solve numerically on each subdomain on separate processors. This procedure will require no communication among the processors involved, no iteration across the interfaces, and of course no initial guess for it.

It should be observed that, for the boundary-value problems for elliptic equations, both on the entire domain, Ω, and on each subdomain, the maximum principle holds, and this remains true for the related discretized problems formulated for their numerical treatment. As a consequence, the error made inside each domain is estimated by the error on the boundary data plus the residual error due to the local solver.

When the latter is made negligible due to a sufficiently accurate scheme, the maximum principle essentially provides an estimate for the solution inside each subdomain in terms of the boundary error. This fact holds true for both the deterministic and the probabilistic domain decomposition scenario, but it turns out to be especially favorable for the latter approach, since due to the poor performance of the Monte Carlo simulations one expects to start with rather poor approximations of the interfacial boundary data.

This method retains the features of the deterministic domain decomposition methods, that is, of being a "divide and conquer" technique, where, however, no iterations on the boundary values inside Ω are needed, while it entails an enhanced [illegible]. In fact, now there is the possibility to split the problem into several fully decoupled subproblems, each assigned to a separate processor. Moreover, irregular or complex geometries and possible singularities as well as anomalous regions can be handled; see [7, 33]. Sometimes, a reduction of the computational complexity can be achieved by the use of a domain decomposition strategy even on sequential architectures. In fact, smaller-size problems can be managed more efficiently, even sequentially. Finally, note that our approach is of the nonoverlapping type, [illegible], and possesses an additional degree of parallelism, since Monte Carlo sequences can be generated in parallel.

Consider, more generally than in section 1 (see (1.1), (1.2)), the elliptic boundary-value problem

$$Lu - c(x)u = f(x) \quad \text{in } \Omega, \qquad u|_{\partial\Omega} = g, \tag{2.1}$$

where $\Omega \subset \mathbf{R}^d$, and L denotes a linear elliptic operator, say $L := a_{ij}(x)\partial_i\partial_j + b_i(x)\partial_i$ (using the summation convention), with continuous bounded coefficients, as well as $c(x)$, $c(x) \geq 0$, continuous boundary data g, source term f in $L^2(\Omega)$, and $\partial\Omega$ Lipschitz continuous. The probabilistic representation of the solutions is now given by

$$u(x) = E_x^L \left[g(\beta(\tau_{\partial\Omega}))e^{-\int_0^{\tau_{\partial\Omega}} c(\beta(s))\,ds} - \int_0^{\tau_{\partial\Omega}} f(\beta(t))\, e^{-\int_0^t c(\beta(s))\,ds}\, dt \right], \tag{2.2}$$

where $\tau_{\partial\Omega}$ is as above, $\beta(\cdot)$ is the stochastic process associated to the operator L, and the expected values are taken with respect to the corresponding measure (see, e.g., [10, 17]). When $L \equiv \Delta$, $\beta(\cdot)$ reduces to the standard d-dimensional Brownian motion, and the measure to the Gaussian measure. The process $\beta(\cdot)$ is the solution of a stochastic differential equation (SDE) of the Ito type, related to the elliptic partial differential equation in (2.1), namely,

$$d\beta = b(x)\, dt + \sigma(x)\, dW(t). \tag{2.3}$$

Here $W(t)$ represents the d-dimensional standard Brownian motion (also called Wiener process); see, e.g., [1, 17] for generalities and [12, 18, 30] for the related numerical treatment. As is known, the solution to (2.3) is a stochastic process, $\beta(t, \omega)$, where ω, which is usually not indicated explicitly in probability theory, denotes the "chance variable," ranging on a suitable abstract probability space. The drift, b, and the diffusion, σ, in (2.3) are related to the coefficients of the elliptic operator in (2.1) by $b = (b_i)^T$ and $\sigma\sigma^T = a$, with $\sigma = (\sigma_{ij})$, $a = (a_{ij})$.

Confining ourselves to 2-dimensional problems, in the examples in section 5 we write (x, y) instead of the vector "x" above.

3. The numerical method. Consider the simpler case $f \equiv c \equiv 0$ in (2.2), so that

$$u(x) = E_x^L \left[g(\beta(\tau_{\partial\Omega})) \right], \tag{3.1}$$

where $\beta(\cdot)$ is obtained solving the SDE in (2.3).

3.1. Evaluating the solution at a single internal point by probabilistic methods. In evaluating interfacial values by a Monte Carlo or quasi-Monte Carlo method, one should consider, in practice, three sources of numerical error. In fact, the expected value in (3.1) must be replaced necessarily by a finite sum, and, moreover, the stochastic paths are actually simulated by resorting to suitable numerical schemes. Finally, the exit time is also affected by numerical errors and has to be replaced by an appropriate estimate. To be more precise, the global error made in simulating interfacial values can be evaluated as

$$\begin{aligned} \epsilon_N = u(x) - \frac{1}{N}\sum_{j=1}^{N} g(x_j(t_j)) &\equiv E_x^L \left[g(\beta(\tau_{\partial\Omega})) \right] - \frac{1}{N}\sum_{j=1}^{N} g(x_j(t_j)) \\ &= \epsilon_N^{(1)} + \epsilon_N^{(2)} + \epsilon_N^{(3)}, \end{aligned} \tag{3.2}$$

where

$$\epsilon_N^{(1)} = E_x^L \left[g(\beta(\tau_{\partial\Omega})) \right] - \frac{1}{N}\sum_{j=1}^{N} g(\beta_j(\tau_{\partial\Omega}^j)), \tag{3.3}$$

$$\epsilon_N^{(2)} = \frac{1}{N}\sum_{j=1}^{N} g(\beta_j(\tau_{\partial\Omega}^j)) - \frac{1}{N}\sum_{j=1}^{N} g(x_j(\tau_{\partial\Omega}^j)), \tag{3.4}$$

$$\epsilon_N^{(3)} = \frac{1}{N}\sum_{j=1}^{N} g(x_j(\tau_{\partial\Omega}^j)) - \frac{1}{N}\sum_{j=1}^{N} g(x_j(t_j)). \tag{3.5}$$

Here $x_j(t)$ is a numerical approximation of $\beta_j(t) := \beta(t, \omega_j)$, which is the jth realization of the stochastic process $\beta(t, \omega)$, solution to (2.3); $\tau_{\partial\Omega}^j$ is the first exit time of the path $\beta_j(t)$; and t_j is the first exit time of the approximating path $x_j(t)$.

The first error, $\epsilon_N^{(1)}$, is the pure Monte Carlo (or quasi-Monte Carlo) statistical error, and

$$\epsilon_N^{(1)} = \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) \tag{3.6}$$

for the Monte Carlo method. In fact, it is well known that the arithmetic mean appearing in $\epsilon_N^{(1)}$ (actually with the almost inessential modification of having $1/(N-1)$ instead of $1/N$, since $N \gg 1$) provides the best unbiased estimator for the expected value in (3.1); see, e.g., [5]. In practice, one should simulate on a computer several random variables, based on generating random numbers. The latter are necessarily not ideally random, for which reason they are called more precisely pseudo-random numbers. Doing that, the error made in replacing the expected value in (3.1) with the average over a finite size sample is statistical in nature and of the order of $1/\sqrt{N}$.

More precisely, $\epsilon_N^{(1)}$ turns out to be, for N large, approximately a random Gaussian variable with standard deviation proportional to $N^{-1/2}$, i.e.,

$$\epsilon_N^{(1)} \approx \sigma N^{-1/2} \nu, \tag{3.7}$$

where σ denotes the square root of the variance of the integrand, g, and ν is a standard normal (i.e., $N(0,1)$) random variable; see, e.g., [5]. It follows that

$$\lim_{N\to\infty} \text{Prob}\left(a < \frac{\sqrt{N}}{\sigma} < b\right) = \text{Prob}(a < \nu < b) = (2\pi)^{-1/2} \int_a^b e^{-x^2/2}\, dx. \tag{3.8}$$

All this clearly shows that Monte Carlo has in principle a very poor numerical performance (because, due to the dependence on the sample size, N, through its square root, the convergence above when $N \to \infty$ is very slow) and also that the error is merely statistical, so it can only be bounded by some quantity with a certain degree of confidence.

An alternative method to the standard Monte Carlo approach is provided by the quasi-Monte Carlo method. Examples of [illegible] sequences (also called low-discrepancy sequences) are those named after Faure, Halton, and Sobol'; see, e.g., [28]. Following this approach, the error is

$$\epsilon_N^{(1)} = \mathcal{O}\left(\frac{1}{N} \log^{d^*-1} N\right). \tag{3.9}$$

When $N \to \infty$, the advantage over Monte Carlo is clear, but one should notice that even when solving elliptic equations in two dimensions, the [illegible] dimension, d^*, is much higher than two. In fact, $d^* = Md$, where d is the dimension of the space, $\mathbf{R}^d$, in which the domain Ω is embedded, and M is the number of steps in the numerical integration of the SDEs (2.3). In practice, one somehow needs to break the correlation in the sequences of quasi-random numbers, and this can be done, for instance, by a "reordering" or by other scrambling strategies. The idea is to contrast the correlations by using [illegible] sequences of uncorrelated quasi-random numbers (which are, however, correlated within each sequence), picking up at each time-step pairs of numbers out of them, and relabeling them according to their distances from the starting point. This idea was applied to the one-dimensional (1D) heat equation in [24] and then to some higher-dimensional problems in [20]. Consequently, we can take $d^* = 2$ in (3.9). When reordering was not implemented, the performance achieved using quasi-random sequences in the numerical solution of SDEs turned out to be poor, as shown in [13] and pointed out in [20].

The second error is due to the fact that the ideal stochastic path, $\beta_j(\cdot)$, in practice has to be approximated by some numerical scheme yielding the paths $x_j(\cdot)$, having discretized the time. Hence, the estimate

$$|\epsilon_N^{(2)}| \le \frac{1}{N} \sum_{j=1}^{N} \left| g(\beta_j(\tau_{\partial\Omega}^j)) - g(x_j(\tau_{\partial\Omega}^j)) \right| \le \frac{L_g}{N} \sum_{j=1}^{N} \left\| \beta_j(\tau_{\partial\Omega}^j) - x_j(\tau_{\partial\Omega}^j) \right\| \tag{3.10}$$

holds, where L_g is the Lipschitz constant of $g(\cdot)$. The truncation error on the right-hand side of (3.10), made in numerically solving the SDE (2.3), obviously depends on the specific scheme that is chosen; see, e.g., [18]. Among these are the Euler scheme, a number of Taylor schemes, and also schemes where the time-step is chosen randomly,

for instance, the exponential timestepping method; see [14, 15]. In the latter case, the size of the time-step, Δt, is picked up from a given probability distribution, since Δt is a random variable itself. When such a distribution is the exponential distribution, with the probability density $exp(-\lambda \Delta t)$, the mean value of Δt equals λ^{-1}. The Euler scheme has a truncation error of order $\mathcal{O}(\Delta t^{\alpha})$, where $\alpha = 1/2$ or $\alpha = 1$ depending on whether the scheme is of the "strong" or "weak" type, respectively; see [18]. Taylor schemes with α equal to 3/2 or 3, respectively, can be considered as well.

Finally,

$$|\epsilon_N^{(3)}| \leq \frac{L_g}{N} \sum_{j=1}^{N} \left\| x_j(\tau_{\partial\Omega}^j) - x_j(t_j) \right\| \leq \frac{L_g}{N} \sum_{j=1}^{N} M_j \left| \tau_{\partial\Omega}^j - t_j \right|$$

$$\leq \frac{L_g M}{N} \sum_{j=1}^{N} \left| \tau_{\partial\Omega}^j - t_j \right|, \tag{3.11}$$

where M_j is the Lipschitz constant of $x_j(\cdot)$, and $M := \max_{1 \leq j \leq N} M_j$. The latter quantity takes into account the error made in numerically evaluating the exact first exit time, $\tau_{\partial\Omega}^j$, of the jth path, $x_j(\cdot)$, from the boundary $\partial\Omega$. Indeed, numerical experiments show that the error made estimating the first exit time may be dominant over the other sources of numerical errors and is therefore of paramount importance in providing an accurate value of such a quantity. In fact, the probability that an approximate path, such as $x_j(\cdot)$, exits the boundary between two consecutive time-steps is nonzero, and it is quite possible that the true exit point is missed. This circumstance was pointed out first in [37] and later in [2, 4], and taken into account in simple cases of numerical integration of 1D SDEs in [14, 21].

Below, we have adopted the strategy put forth in [14, 15], based on exponential timestepping. The advantage of such a method rests on the fact that, at the price of adopting an approximate distribution of the underlying stochastic process, analytical results can be obtained for the hitting probability (i.e., the probability distribution of hitting the boundary for the first time). To solve two-dimensional (2D) problems on the square, $\Omega = (0,1) \times (0,1)$, as done in this paper, the hitting probability has been taken as the maximum among the four hitting probability values that a path first exits the four possible boundary sides.

3.2. Interpolation on the internal nodes. After obtaining approximations of a few internal values by Monte Carlo (or quasi-Monte Carlo) simulations, we use them to interpolate. We chose Chebyshev interpolation because of its global (quasi-) optimality properties [34, 35]. The error made in interpolating a given smooth function, say $f(x)$, by the nth degree Chebyshev polynomial of the first kind, $I_n f(x) \equiv T_n(x)$, between any two nodes when the nodal values are exactly known, is given by $\|f - I_n f\|_\infty \leq C n^{-k} \|f^{(k)}\|_\infty$ for every fixed k. However, we should account for the additional errors affecting the nodal values $f(x_i)$ themselves, which are obtained, in fact, by the Monte Carlo or quasi-Monte Carlo simulation. Such a problem (stability of the interpolation) leads to the estimate $\|I_n f - I_n \tilde{f}\| \leq \Lambda_n \cdot \max_{i=1,2,\ldots,n} |f(x_i) - \tilde{f}(x_i)|$, where $\tilde{f}(x_i)$ are the values of $f(x_i)$ affected by errors, and the Lebesgue constant, Λ_n, grows only logarithmically with the number of nodes and thus will be completely negligible here. Indeed, the number of nodes will be kept low, since two or three internal nodes on each interval will suffice within the numerical approximations made in our algorithm. Hence, Chebyshev polynomials of degree 3 or 4 will be used.

3.3. Local solvers. Once approximate values of the solution on all interfaces have been found, any deterministic algorithm can be used to complete the numerical solution on each subdomain. For instance, finite differences (FDs) of various kinds can be implemented, and then there are a number of methods to solve the ensuing linear algebraic systems. When the dimension of such systems is low (which occurs when the number of subdomains is high), direct methods can be used. In general, and especially when the dimension is high, iterative methods may perform better. Below, we chose to use Gauss–Seidel iterations, with an exit criterion that had to be tested according to the specific model example and the number of subdomains.

4. Performance of the numerical method. Let us now turn our attention to assessing, at least qualitatively, the performance of our method in two dimensions. The aim here is to estimate the , , which is a measure of the gain of the present method as a function of the number of processors involved. We assume that the domain Ω is the square $(0,1) \times (0,1)$, subdivided into a number of subdomains. These are nonoverlapping rectangles, with sides parallel to the axes x and y, singled out by some subdivision points, say n_x and n_y, of the sides $0 \le x \le 1$ and $0 \le y \le 1$, respectively. Therefore, the domain Ω is partitioned into $(n_x+1)\cdot(n_y+1)$ subdomains. Suppose that as many processors, p, as we wish, are available and assume $p = (n_x+1)(n_y+1)$. We compute by Monte Carlo or quasi-Monte Carlo simulations only a few interfacial values, on the lines $x = i/(n_x+1)$, $i = 1,\dots,n_x$, and $y = j/(n_y+1)$, $j = 1,\dots,n_y$, taking, say, k points on each such line (usually $k = 2$ or 3), hence, in total, $k(n_x+n_y)$ points.

The speed-up is the ratio between the time T_1 spent solving sequentially the problem on the entire domain, and the time T_p required for solving it in parallel with p processors. The numerical solution of each subproblem requires at most a fraction $1/p$ of the time T_1. We say "at most," because with the integration domain being smaller, some amount of time can be saved even proceeding sequentially. Therefore, if T_{MC} denotes the time required for computing by Monte Carlo or quasi-Monte Carlo simulations , interfacial value, neglecting the time spent for interpolation on the interfaces, we have, approximately,

$$T_p = \frac{k(n_x+n_y)T_{MC}}{p} + \frac{T_1}{p}, \tag{4.1}$$

and thus the , of our method is

$$S_p := \frac{T_1}{T_p} = \frac{p}{1+k(n_x+n_y)\frac{T_{MC}}{T_1}}. \tag{4.2}$$

Note that if T_{MC} could be made negligible, the speed-up would attain its ideal theoretical value, $S_p = p$. In practice, T_{MC} cannot be arbitrarily small, since it will be proportional to the number of realizations used in simulations, whose value directly affects the achieved accuracy.

To gain a better insight, take $n_x = n_y =: n$. Using as many processors as subdomains, we have $p = (n+1)^2$, while the total number of interfacial points is $2kn$. Note that the latter scales linearly with the parameter n, while p scales quadratically with it. Thus

$$S_p = \frac{(n+1)^2}{1+2kn\frac{T_{MC}}{T_1}}, \tag{4.3}$$

and hence, in the limit for $n \to \infty$ (or, equivalently, for $p \to \infty$), we obtain

$$S_p \sim \frac{T_1}{2kT_{MC}} \sqrt{p}, \quad p \to \infty. \tag{4.4}$$

This formula shows the kind of scalability that can be achieved by means of the PDD. We stress that the speed-up relation in (4.4) is fully general in that it is independent of the choice of the specific local solver, as well as whether Monte Carlo or quasi-Monte Carlo is adopted. In case of domains $\Omega \in \mathbf{R}^d$, formulae (4.3) and (4.4) are generalized into

$$S_p = \frac{(n+1)^d}{1 + dkn\frac{T_{MC}}{T_1}} \sim \frac{T_1}{dkT_{MC}}\, p^{1-1/d}, \quad p \to \infty, \tag{4.5}$$

since $p \sim n^d$ as $p \to \infty$. Consider, at this point, the dependence of T_{MC} and T_1 on the various parameters characterizing the numerical treatment.

4.1. Estimating T_{MC}. The time T_{MC} required for computing a single value of the solution by Monte Carlo or quasi-Monte Carlo can be estimated as follows. The computation of a single stochastic path, originating at some given point internal to the domain, requires, on the average, a number of operations proportional to the number of steps, say N_s. The latter quantity is about the ratio between the mean first exit time and the time-step size, Δt, involved in the numerical integration of the underlying SDEs. When using an exponential timestepping, we take for Δt its mean value, λ^{-1}.

To estimate how the mean first exit time depends on dimension, recall that for the special case of Brownian motion going out from the hypersphere B_r of radius r, starting from the point x, this quantity is $E_x\{\tau_{B_r}\} \le r^2/d$; see [17]. If an $\mathcal{O}(1)$ drift is added, such a value becomes of order r. When B_r is replaced by a hypercube whose diagonal length is 1, these values become of order $1/\sqrt{d}$ and of order 1, respectively. Therefore, taking N realizations, all starting from a given point, the cost is proportional to $NN_s \approx Nf(d)(\Delta t)^{-1}$, where $f(d)$ represents the dependence of the mean first exit time on dimension.

The statistical error is expected to dominate; thus we choose a time-step size such that the truncation error is of the same order, i.e., $1/N^\beta = (\Delta t)^\alpha$. Here $\beta = 1/2$ for pseudorandom sequences, or (a little worse than) $\beta = 1$ using quasi-random sequences, and α is the order of the scheme adopted to solve the SDEs. This links the two parameters, N and Δt, so that $(\Delta t)^{-1} \approx N^{\beta/\alpha}$. Hence, the cost for computing N realizations will be $\approx N^{1+\beta/\alpha} f(d)$. In d dimensions, d SDEs should be solved, and thus such a cost should be multiplied by d. Therefore,

$$T_{MC} \approx N^{1+\beta/\alpha}\, d\, f(d). \tag{4.6}$$

4.2. Estimating T_1. In order to estimate T_1, we choose an FD solver whose truncation error equals the statistical error in our approach, that is, $h^2 \sim 1/N^\beta$, where $h := \Delta x = \Delta y$ is the mesh size. T_1 will be proportional to the total number of the grid points, h^{-d}, and to the number of iterations, n_{iter}, needed to solve iteratively the underlying linear algebraic system. To be concrete, it is found that for the 2D Laplacian such a number is given by [31]

$$n_{iter} \sim c\, q\, h^{-\xi}, \tag{4.7}$$

taking an iteration error equal to 10^{-q} as an exit criterion. Here $\xi = 2$ for the Jacobi and the Gauss–Seidel methods, and 1 for the successive overrelaxation (SOR) method with optimal parameter. The constant c is a fraction of 1, and q is typically of order 10 in our experiments. Therefore,

$$T_1 \sim h^{-d}\, n_{iter} = c\, q\, N^{\beta(d+\xi)/2}. \tag{4.8}$$

The speed-up in (4.5) then becomes

$$S_p^{PDD} \sim \frac{c\, q}{k\, d^2 f(d)}\, N^{\gamma}\, p^{1-1/d}, \quad p \to \infty, \tag{4.9}$$

where we set

$$\gamma := \beta\left(\frac{d+\xi}{2} - \frac{1}{\alpha}\right) - 1. \tag{4.10}$$

In order to compare with the case when the entire problem (on the entire domain) is solved by [illegible] (PFD), which represents a deterministic domain decomposition strategy, we now have $T_p = T_1/p + T_{com}$. Here T_{com} is the [illegible] time among the processors, which affects the algorithm [illegible] iteration step. Therefore, the speed-up relation takes on the form

$$S_p^{PFD} := \frac{T_1}{T_p} = \frac{p}{1 + \frac{T_{com}}{T_1} p} \sim \frac{T_1}{T_{com}} \quad \text{as } p \to \infty. \tag{4.11}$$

Note that this shows the nonscalability ("saturation") of such a method, since $S_p^{PFD} \sim T_1/T_{com}$ remains bounded as $p \to \infty$. The quantity T_{com}, which is assumed to be essentially independent of p, can be modeled as

$$T_{com} = 2N^{\beta(d-1)/2}\, n_{iter} B(d), \tag{4.12}$$

where $B(d)$ takes into account the bandwidth, and the factor 2 is due to both sending and receiving communications. In the exponent of N, $d-1$ replaces d because the points involved in intercommunications across the interfaces lie on a $(d-1)$-dimensional manifold. Hence, (4.11) yields

$$S_p^{PFD} \sim \frac{1}{2B(d)}\, N^{\beta/2} \quad \text{as } p \to \infty. \tag{4.13}$$

The speed-up of the PDD method in (4.9) wins over the speed-up of the PFD algorithm (4.13), that is, $S_p^{PDD} > S_p^{PFD}$, whenever, approximately,

$$p > \left(\frac{k\, d^2 f(d)}{2\, c\, q\, B(d)}\right)^{d/(d-1)} N^{\delta}, \tag{4.14}$$

where

$$\delta := \frac{d}{d-1}\left[\frac{\beta}{2}\left(1 - d - \xi + \frac{2}{\alpha}\right) + 1\right]. \tag{4.15}$$

The factor on the right-hand side of (4.14) that multiplies N^{δ} is assumed to depend weakly on dimension and is typically of order 1. In view of the fact that N should

be allowed to be arbitrarily large to achieve a sufficiently high accuracy in both algorithms, PDD and PFD, the interesting case is given by $\delta \leq 0$. This condition is verified if and only if the dimension is larger than or equal to a certain [illegible], d_{crit}, given by

$$d_{crit} := \begin{cases} \tilde{d} & \text{for } \tilde{d} \text{ integer,} \\ \left[\tilde{d}\right] + 1 & \text{for } \tilde{d} \text{ not integer,} \end{cases} \qquad \tilde{d} := 1 - \xi + 2\left(\frac{1}{\alpha} + \frac{1}{\beta}\right). \tag{4.16}$$

When $d \geq d_{crit}$, the PDD method wins over the PFD even with very few processors (or, equivalently, subdomains). The smallest critical dimension is clearly achieved when the order α of the scheme used to solve the SDEs is the highest, and quasi-random sequences ($\beta = 1$) and slower iterative solvers ($\xi = 2$, Gauss–Seidel method) are used. For instance, when $\beta = 1$, $\xi = 2$, and $\alpha = 2$, the value $d_{crit} = 2$ is obtained. For a general set of parameters, the critical dimension will be greater than 2. For instance, setting $\beta = 1$, $\xi = 2$, and $\alpha = 1$, we get $d_{crit} = 3$.

Even when $\delta > 0$, however, the condition (4.14) may also be satisfied with a reasonably low number of processors. As an example, when choosing $N = 10^4$ realizations (corresponding to an error of order 10^{-4} using quasi-random sequences), such an inequality requires about $p > 20$, whenever $\delta \leq 1/3$: Take, e.g., $d = 2$, $\xi = 2$, and $\alpha = 3/2$.

5. Numerical examples. In this section, we show the performance of the parallel code we designed to exploit a probabilistically induced domain decomposition method, by means of a few numerical examples. By simplicity, the code has been implemented using OpenMP, a standard parallelization library designed for shared memory architectures. The parallel machine used to conduct the numerical tests is a 16 processor IBM Power 3, running at a clock speed of 375 MHz, with a theoretical peak performance of 24 Giga Flops. Even though our algorithm allows for using an arbitrary number, p, of processors, the implementation of the code by OpenMP is restricted to run on a single node, which usually supports, so far, a relatively low number of processors. Applications based on message passing interface (MPI), which is characterized by the possibility of internodal communication, however, could be considered.

In the numerical examples below, our method is compared with a PFD solver, a parallel version of a standard finite difference solver, which is the same as that used in the local solver within the PDD method, and is also parallelized using OpenMP. It should be observed that such a comparison is in fact made between our PDD algorithm and a kind of "deterministic domain decomposition method." Such a method actually consists of distributing the overall computational load among the available processors.

It is important to stress that in the PDD method the nodal values on the interfaces are generated by classical Monte Carlo simulations, which can take advantage of massively parallel computation. After that, interpolation on the interfaces allows for full decoupling into subdomains, and hence the numerical solution of the given problem can be obtained also computing in parallel. In the quasi-PDD method (where we used Halton sequences of quasi-random numbers), however, there is no clear way to compute in parallel the nodal values above. In fact, due to the existing correlations among the elements within each sequence of quasi-random numbers, it is mandatory to adopt a suitable scrambling strategy, such as reordering, which contrasts with the possibility of computing independently the various realizations. The possibility of computing in parallel with quasi-random sequences deserves further investigation. The second source of parallelization, however, still holds true.

In the examples below, we compare the efficiency of PDD with that of PFD and observe the overall error reduction attained when passing from PDD to quasi-PDD.

A. Consider first the Dirichlet problem for the Laplace equation in two dimensions:

$$u_{xx} + u_{yy} = 0 \qquad \text{in } \Omega := (0,1) \times (0,1),$$

$$u(x,y)|_{\partial\Omega} = g(x,y), \tag{5.1}$$

where $g(x,y) := \left(x^2 - y^2\right)_{\partial\Omega}$. This choice of g corresponds to the fact that the analytical solution we wish to recover is the function $u(x,y) = x^2 - y^2$, harmonic in $\overline{\Omega} = [0,1] \times [0,1]$. In Figures 5.1(a) and 5.1(b), the pointwise numerical error, made when running the PDD and the quasi-PDD code, based on the PDD and the quasi-PDD methods, respectively, is shown on the whole domain, Ω, using $p = 4$ subdomains. The L^∞ error can be read from the color bar at the side.

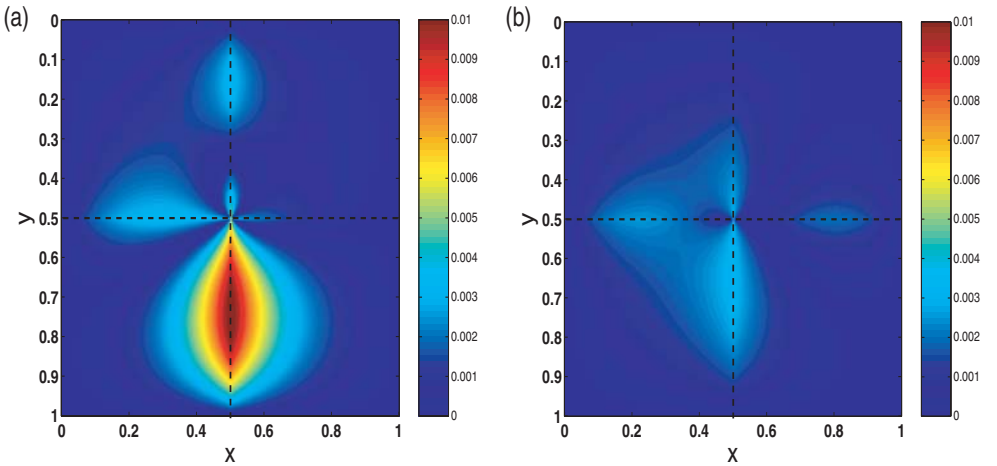


FIG. 5.1. *Example* A. *Pointwise numerical error in* (a) *the PDD algorithm and* (b) *the quasi-PDD algorithm. Parameters are* $N = 10^4$, $\Delta x = \Delta y = 2 \times 10^{-3}$, $\lambda = 10^2$.

In both pictures in Figure 5.1, it clearly appears that the maximum error on the entire domain Ω is achieved on the interfaces, and more precisely on the interpolation nodes. This occurs according to the observation made in section 2 concerning the maximum principle. Indeed, the error due to the Monte Carlo evaluation of the nodal values dominates over both the interpolation error and on the FD errors due to the local solver. This is true for both PDD and quasi-PDD methods. Moreover, it is also clear that the quasi-PDD algorithm outperforms the PDD algorithm. The parameters used here are $N = 10^4$ realizations, $\Delta x = \Delta y = 2 \times 10^{-3}$ grid size, and $\lambda = 10^2$ timestepping parameter (and thus an average time-step $\langle \Delta t \rangle = 10^{-2}$ to integrate SDEs in (2.3)). In this example, concerning the Laplace equation, the hitting probability, which is the probability that a given trajectory hits the boundary between two consecutive time-steps, can be evaluated exactly; see [15]. Consequently, a relatively large mean time-step could be used.

Numerical experiments show that two nodes suffice on each of the two interfaces. The local solver is based on Gauss–Seidel iteration, where the exit tolerance has been found experimentally ($q = 14$).

TABLE 5.1
CPU time in seconds for example A.

Processors	*PFD*	PDD_{Total}	$PDD_{Monte\,Carlo}$	PDD_{FD}
4	121.237	106.079	24.315	81.764
9	56.539	35.833	17.625	18.208
16	42.752	20.992	14.676	6.316

In Table 5.1, the second column, labeled by PFD, shows the overall computational time (in seconds) spent by the PFD algorithm for solving the problem in Example A, using $p = 4$, 9, and 16 processors. This corresponds to 4, 9, and 16 subdomains. In the third column the same is done, when the probabilistic domain decomposition is used. On the last two columns, the time needed in the latter algorithm is split into two parts, that concerning the Monte Carlo simulation, and that needed by the local solvers. This comparison between the two methods is made for about the same maximum error, 10^{-3}. It clearly appears that in both algorithms the CPU time decreases as p increases, and this behavior is more dramatic in the PDD algorithm. Also the CPU time decreases for each given number of processors passing from PFD to PDD. One can observe that the ratio between the CPU time spent in total by PDD (PDD_{Total}) and that spent by PFD favorably decreases as p increases; see section 4.

B. Consider the 2D elliptic equation with variable coefficients

$$u_{xx} + (6x^2 + 1)u_{yy} = 0 \qquad \text{in } \Omega := (0,1) \times (0,1), \tag{5.2}$$

subject to the boundary data

$$u(x,y)|_{\partial\Omega} = g(x,y), \tag{5.3}$$

where $g(x,y) := \left(x^4 + x^2 - y^2\right)_{\partial\Omega}$. Similarly to Example A, we wrote the boundary data in this way because the analytical solution of this problem is $u(x,y) = x^4 + x^2 - y^2$ in $\overline{\Omega}$.

In Figures 5.2(a), and 5.2(b), the same plots as in Example A are shown, and the same comments hold. The only difference is that here three nodal points are needed in

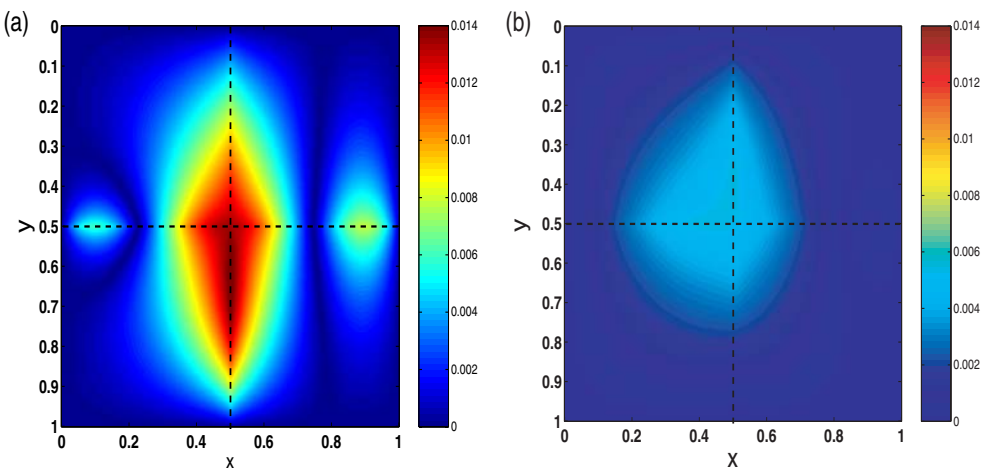


FIG. 5.2. *Example* B. *Pointwise numerical error in* (a) *the PDD algorithm and* (b) *the quasi-PDD algorithm. Parameters are* $N = 10^4$, $\Delta x = \Delta y = 2 \times 10^{-3}$, $\lambda = 10^3$.

TABLE 5.2
CPU time in seconds for example B.

Processors	*PFD*	PDD_{Total}	$PDD_{Monte\,Carlo}$	PDD_{FD}
4	599.066	317.778	11.042	306.736
9	271.991	75.018	9.144	65.874
16	174.784	30.101	7.710	22.391

the interpolation on each interface, which choice is determined upon experimentation. Parameters are also the same used in Example A, with the only exception being the timestepping parameter λ, which has been chosen equal to 10^3. Other than in Example A, here the hitting probability has to be evaluated approximately in powers of λ^{-1} [15]. A higher value of λ is needed here in order to ensure a good accuracy in approximating such a hitting probability.

Table 5.2 shows the same comparison as in Table 5.1, and the comments are similar. The PFD method here requires more CPU time because the present problem is more elaborate, due to the variable coefficients entering the equation. Thus more iterations are needed in the linear algebraic part of the code. On the other hand, the Monte Carlo part in the PDD algorithm is faster than in Example A, because the variable diffusion implies reaching the boundary in fewer steps.

C. Consider the elliptic equation with constant diffusion and (variable) drift terms:

$$\Delta u + \frac{1}{x+1}\,u_x + \frac{2}{y+1}\,u_y = 0 \qquad \text{in } \Omega := (0,1)\times(0,1), \tag{5.4}$$

with the boundary data

$$u(x,y)|_{\partial\Omega} = \left[(x+1)^2 + \frac{2}{3}(y+1)^2\right]_{\partial\Omega}, \tag{5.5}$$

the solution being given by $u(x,y) = (x+1)^2 + 2(y+1)^2/3$.

Figure 5.3 is analogous to the previous ones, and the parameters used are the

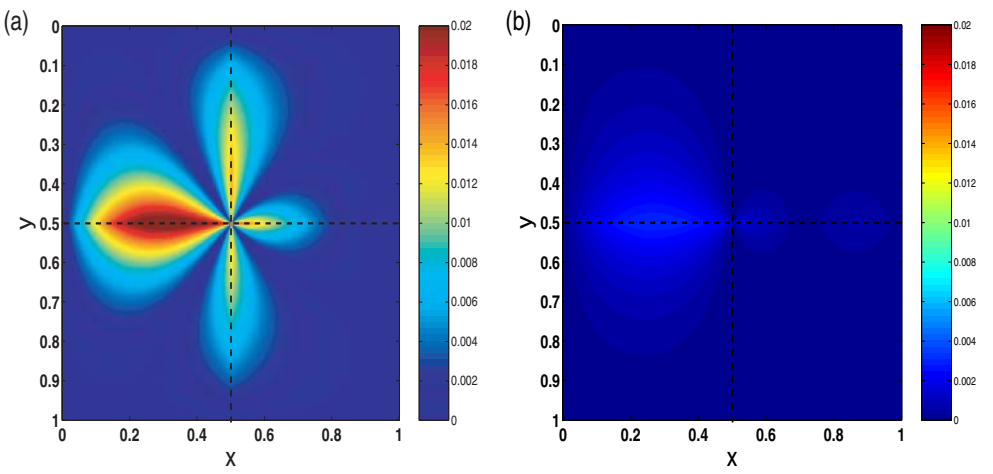


FIG. 5.3. *Example* C. *Pointwise numerical error in* (a) *the PDD algorithm and* (b) *the quasi-PDD algorithm. Parameters are the same as in Example* B.

TABLE 5.3
CPU time in seconds for example C.

$Processors$	PFD	PDD_{Total}	$PDD_{Monte\,Carlo}$	PDD_{FD}
4	350.989	166.728	151.640	15.088
9	160.351	48.374	34.577	13.797
16	105.862	22.585	12.104	10.481

same as in Example A. In particular, only two nodes on each interface have been used, but $\lambda = 10^3$ as in Example B. The general comments made above also apply.

In Table 5.3, as in the previous tables, one can see that now the Monte Carlo part of the PDD algorithm takes longer than in the previous examples. This fact can be attributed to the asymmetry introduced by the drift in both directions, which ultimately requires more time-steps to exit the boundary.

D. Consider finally the Dirichlet problem for the 2D elliptic equation

$$u_{xx} + u_{yy} - 5u = 0 \qquad \text{in } \Omega := (0,1) \times (0,1), \tag{5.6}$$

$$u(x,y)|_{\partial\Omega} = g(x,y), \tag{5.7}$$

where $g(x,y) := \left(e^{2x+y}\right)_{\partial\Omega}$. The analytical solution of such a problem is $u(x,y) = e^{2x+y}$ in $\overline{\Omega} = [0,1] \times [0,1]$. In Figures 5.4(a) and 5.4(b), similarly to the previous cases, the pointwise numerical error, made when running the PDD and the quasi-PDD code, based on the PDD and the quasi-PDD methods, is shown. Here only two nodes on each interface have been used. Figure 5.4(c) shows the result for three nodes on each interface (that is, six nodal points in total). Note that increasing the number of nodal points yields an overall reduction of the numerical error on the whole domain. When pseudorandom numbers are used, the statistical error dominates, making irrelevant the improvement achieved by interpolating on three (instead of two) nodes on each interface. On the other hand, the time required obviously increases; see Tables 5.4 and 5.5. Using quasi-random numbers, instead, which are characterized by smaller errors in computing the nodal values, an overall error reduction due to a better interpolation can be observed when switching from two to three nodes.

In Tables 5.4 and 5.5, the same features observed in the previous examples can be seen. In Table 5.5, the CPU times related to PFD have been omitted since they coincide with those reported in Table 5.4. In fact, the comparison between PFD

TABLE 5.4
CPU time in seconds for Example D with two nodes per interface.

$Processors$	PFD	PDD_{Total}	$PDD_{Monte\,Carlo}$	PDD_{FD}
4	556.427	242.453	25.327	217.126
9	252.911	67.190	17.890	49.300
16	163.941	33.496	16.306	17.190

TABLE 5.5
CPU time in seconds for example D with three nodes per interface.

$Processors$	PDD_{Total}	$PDD_{Monte\,Carlo}$	PDD_{FD}
4	242.406	25.391	217.015
9	72.634	23.377	49.257
16	36.877	19.702	17.175

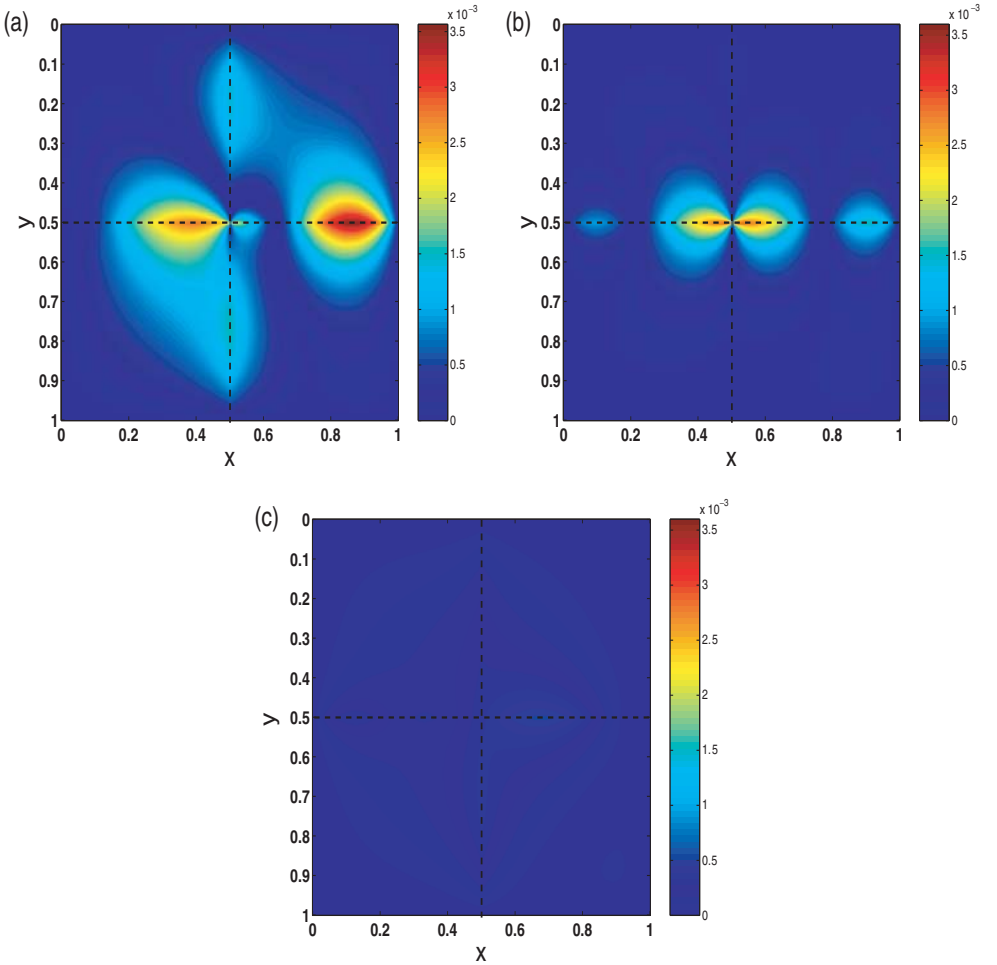


FIG. 5.4. *Example* D. *Pointwise numerical error in* (a) *the PDD algorithm,* (b) *the quasi-PDD algorithm with two nodal points on each interface, evaluated by quasi-Monte Carlo, and* (c) *the quasi-PDD algorithm with three nodal points on each interface. Parameters are* $N = 10^4$, $\Delta x = \Delta y = 2 \times 10^{-3}$, $\lambda = 10^3$.

and PDD has been made by keeping fixed the overall numerical error, which does not change when passing from two to three nodes per interface, as pointed out above. Note also that the CPU times spent here in $PDD_{Monte\,Carlo}$ are about the same as those in Example A (see Table 5.1). In fact, in both examples, A and D, the numerical solution of the same system of SDEs is involved, and only a short additional time is required in Example D. This is due to the quadrature corresponding to the presence of the potential term in (5.6); see (2.2). Finally, it appears that the CPU time appreciably increases when switching from two to three nodes per interface, for a higher number of processors, since this corresponds to increasing the total number of nodal points.

6. Concluding remarks. Probabilistic representations of solutions to elliptic boundary-value problems and the ensuing possibility of using them for numerical approximations have been known for a long time. In spite of this, in the literature there is apparently only a modest body of work addressing the latter possibility; see,

e.g., [11, 20, 24]. One reason certainly can be found in the poor performance of the Monte Carlo method. In this paper, we have first improved the overall performance of such a probabilistic numerical approach to solving linear elliptic boundary value problems. This has been done even for [illegible] equations, including both cases of variable drift and diffusion. A suitable boundary treatment, realized by means of the exponential timestepping method, was recognized as an essential ingredient and implemented. Moreover, quasi-random sequences have been successfully used, resulting in a faster convergence rate in the Monte Carlo simulations. Finally, we have adopted a "divide and conquer" strategy, represented by a domain decomposition method, where only a few values are computed by Monte Carlo or quasi-Monte Carlo, thus reducing the overall inherent computational cost. Such values are used as nodes for interpolating the sought solution, then providing continuous approximations for the interfacial values, to be exploited subsequently to split the problem into several subproblems. We were then able to exploit as many processors, p, as subdomains. Speed-up and scalability with p are shown, both theoretically and in a few practical examples.

In closing, notice that even a relatively coarse solution obtained by the PDD algorithm (through a Monte Carlo or quasi-Monte Carlo strategy with a moderate value of N) could be used as the [illegible] for the FD solver on the whole domain. Indeed, the exit criterion in the linear algebraic solver depends both on the spectral radius of the iteration matrix (and thus on its spectral condition number) [illegible] on the initial guess of solution. While the former can be reduced by preconditioning, which is essential in certain iterative methods, such as the iterative substructuring (deterministic domain decomposition) methods, the PDD algorithm might provide a good initial guess.

Acknowledgments. We are indebted to P. Dai Prà, L. Pavarino, A. Quarteroni, M. Vianello, and O. Widlund for several useful and enlightening discussions.

REFERENCES

[1] L. Arnold, *Stochastic Differential Equations: Theory and Applications*, Wiley, New York, 1974.

[2] P. Baldi, *Exact asymptotics for the probability of exit from a domain and applications to simulation*, Ann. Probab., 23 (1995), pp. 1644–1670.

[3] S. C. Brenner, *Lower bounds for two-level additive Schwarz preconditioners with small overlap*, SIAM J. Sci. Comput., 21 (2000), pp. 1657–1669.

[4] F. M. Buchmann and W. Petersen, *Solving Dirichlet problems numerically using the Feynman-Kac representation*, BIT, 43 (2003), pp. 519–540.

[5] R. E. Caflisch, *Monte Carlo and quasi-Monte Carlo methods*, in Acta Numerica, Acta Numer. 7, Cambridge University Press, Cambridge, UK, 1998, pp. 1–49.

[6] R. E. Caflisch, W. Morokoff, and A. B. Owen, *Valuation of mortgage-backed securities using Brownian bridges to reduce effective dimension*, J. Comput. Finance, 1 (1997), pp. 27–46.

[7] T. F. Chan and T. P. Mathew, *Domain decomposition algorithms*, in Acta Numerica, Acta Numer., Cambridge University Press, Cambridge, UK, 1994, pp. 61–143.

[8] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comput., 15 (1994), pp. 604–620.

[9] M. Dryja, B. F. Smith, and O. B. Widlund, *Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions*, SIAM J. Numer. Anal., 31 (1994), pp. 1662–1694.

[10] M. Freidlin, *Functional Integration and Partial Differential Equations*, Ann. Math. Stud. 109, Princeton University Press, Princeton, NJ, 1985.

[11] E. Hausenblas, *A Monte-Carlo method with inherent parallelism for numerical solving partial differential equations with boundary conditions*, in Parallel Computation, Lecture Notes in Comput. Sci. 1557, Springer-Verlag, Berlin, 1999, pp. 117–126.

[12] D. J. Higham, *An algorithmic introduction to numerical simulation of stochastic differential equations*, SIAM Rev., 43 (2001), pp. 525–546.
[13] N. Hofmann and P. Mathé, *On quasi-Monte Carlo simulation of stochastic differential equations*, Math. Comp., 66 (1997), pp. 573–589.
[14] K. M. Jansons and G. D. Lythe, *Efficient numerical solution of stochastic differential equations using exponential timestepping*, J. Statist. Phys., 100 (2000), pp. 1097–1109.
[15] K. M. Jansons and G. D. Lythe, *Exponential timestepping with boundary test for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2003) pp. 1809–1822.
[16] M. H. Kalos and P. A. Withlock, *Monte Carlo Methods, Vol.* I: *Basics*, Wiley, New York, 1986.
[17] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, 2nd ed., Springer-Verlag, Berlin, 1991.
[18] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1992.
[19] C. Lécot, *A quasi-Monte Carlo method for the Boltzmann equation*, Math. Comp., 56 (1991), pp. 621–644.
[20] C. Lécot and F. El Khettabi, *Quasi-Monte Carlo simulation of diffusion*, J. Complexity, 15 (1999), pp. 342–359.
[21] R. Mannella, *Absorbing boundaries and optimal stopping in a stochastic differential equation*, Phys. Lett. A, 254 (1999), pp. 257–262.
[22] M. Mascagni, *Parallel Wiener integral methods for elliptic BVPs: A tale of two architectures*, in Applications on Advanced Architecture Computers, Software Environ. Tools, G. Astfalk, ed., SIAM, Philadelphia, 1996, pp. 27–33.
[23] W. J. Morokoff, *Generating quasi-random paths for stochastic processes*, SIAM Rev., 40 (1998), pp. 765–788.
[24] W. J. Morokoff and R. E. Caflisch, *A quasi-Monte Carlo approach to particle simulation of the heat equation*, SIAM J. Numer. Anal., 30 (1993), pp. 1558–1573.
[25] W. J. Morokoff and R. E. Caflisch, *Quasi-random sequences and their discrepancies*, SIAM J. Sci. Comput., 15 (1994), pp. 1251–1279.
[26] W. J. Morokoff and R. E. Caflisch, *Quasi-Monte Carlo integration*, J. Comput. Phys., 122 (1995), pp. 218–230.
[27] B. Moskowitz and R. E. Caflisch, *Smoothness and dimension reduction in quasi-Monte Carlo methods*, Math. Comput. Modelling, 23 (1996), pp. 37–54.
[28] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, CBMS-NSF Regional Conf. Ser. in Appl. Math. 63, SIAM, Philadelphia, PA, 1992.
[29] E. Peirano and D. Talay, *Domain decomposition by stochastic methods*, in Proceedings of the 14th International Conference on Domain Decomposition Methods, I. Herrera, D. Keyes, O. Widlund, and R. Yates, eds., National Autonomous University of Mexico, Mexico City, 2003, pp. 131–147.
[30] E. Platen, *An introduction to numerical methods for stochastic differential equations*, in Acta Numerica, Acta Numer. 8, Cambridge University Press, Cambridge, UK, 1999, pp. 197–246.
[31] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1996.
[32] A. Quarteroni and A. Valli, *Theory and application of Steklov–Poincaré operators for boundary-value problems*, in Applied and Industrial Mathematics, Math. Appl. 56, R. Spigler, ed., Kluwer, Dordrecht, The Netherlands, 1991, pp. 179–203.
[33] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Numer. Math. Sci. Comput., Oxford Science Publications, Clarendon Press, Oxford, UK, 1999.
[34] T. J. Rivlin, *An Introduction to the Approximation of Functions*, Dover, New York, 1981.
[35] T. J. Rivlin, *The Chebyshev Polynomials*, Wiley, New York, 1974.
[36] R. Spigler, *A probabilistic approach to the solution of PDE problems via domain decomposition methods*, in Proceedings of the Second International Conference on Industrial and Applied Mathematics, Washington, DC, 1991, p. 12 (Book of Abstracts).
[37] W. Strittmatter, *Numerical simulation of the mean first passage time*, Technical report THEP 87/12, University of Freiburg, Freiburg, Germany, 1987.
[38] D. Talay, *Probabilistic numerical methods for partial differential equations: Elements of analysis*, in Probabilistic Models for Nonlinear Partial Differential Equations, Lecture Notes in Math. 1627, Springer-Verlag, Berlin, 1996, pp. 148–196.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 458–481
© 2005 Society for Industrial and Applied Mathematics

COMPUTATION OF SMALLEST EIGENVALUES USING SPECTRAL SCHUR COMPLEMENTS*

CONSTANTINE BEKAS† AND YOUSEF SAAD†

Abstract. The automated multilevel substructuring method (AMLS) was recently presented as an alternative to well-established methods for computing eigenvalues of large matrices in the context of structural engineering. This technique is based on exploiting a high level of dimensional reduction via domain decomposition and projection methods. This paper takes a purely algebraic look at the method and explains that it can be viewed as a combination of three ingredients: (a) A first order expansion to a nonlinear eigenvalue problem that approximates the restriction of the original eigenproblem on the interface between the subdomains, (b) judicious projections on partial eigenbases that correspond to the interior of the subdomains, (c) recursivity. This viewpoint leads us to explore variants of the method which use Krylov subspaces instead of eigenbases to construct subspaces of approximants. The nonlinear eigenvalue problem viewpoint yields a second order approximation as an enhancement to the first order technique inherent to AMLS. Numerical experiments are reported to validate the approaches presented.

Key words. AMLS, eigenvalues, substructuring, spectral Schur complements, Krylov subspaces

AMS subject classifications. 65F15

DOI. 10.1137/040603528

1. Introduction.

The numerical solution of large sparse symmetric eigenvalue problems continues to be at the forefront of current research in scientific computing. In the last few decades, projection methods such as the Lanczos algorithm and its variants have dominated the scene. For example, a block version of this method combined with shift-and-invert [11], whereby the problem $Ku = \lambda Mu$ is replaced by $(K - \sigma M)u = (\lambda - \sigma)Mu$, is used in major commercial structural engineering packages such as MSC.NASTRAN [15]. ARPACK [17], a package based on an implicitly restarted Arnoldi/Lanczos process, is currently the best known public-domain eigenvalue package for large eigenvalue problems.

When computing a large number of eigenvalues the Lanczos process faces difficulties associated with orthogonalizing large Krylov bases. The method scales poorly with the number of eigenvalues. In recent years, an alternative approach has emerged in structural engineering which has been reported to be superior to the standard shift-and-invert Lanczos approach. The algorithm, called automated multilevel substructuring method (AMLS), is rooted in a domain decomposition framework. It has been reported as being capable of computing thousands of the smallest normal modes of dynamic structures on commodity workstations and of being orders of magnitude faster than the standard approach [16].

The paper [5] presents a theoretical framework for the algorithm from the point of domain decomposition, using adequate functional spaces and operators on them. The goal of our paper is to present a different, yet complementary viewpoint, which is entirely algebraic. AMLS is essentially a Schur complement method. Schur complement techniques are well understood for solving linear systems and play a major role in

*Received by the editors January 25, 2004; accepted for publication (in revised form) December 20, 2004; published electronically September 23, 2005. This work was supported by the NSF under grants NSF/ITR-0082094 and NSF ACI-0305120 and by the Minnesota Supercomputing Institute.
http://www.siam.org/journals/sisc/27-2/60352.html
†Computer Science & Engineering Dept., University of Minnesota, 200 Union St. SE, Minneapolis, MN, 55455 (bekas@cs.umn.edu, saad@cs.umn.edu).

domain decomposition techniques [20, 21]. Relatively speaking, the formulation of this method for eigenvalue problems has been essentially neglected so far. One could of course extend the approach used for linear systems for eigenvalues by formulating a Schur complement problem for each different eigenvalue, (e.g., by solving the eigenvalue problem as a sequence of linear systems through shift-and-invert). A more complete framework was suggested in early work by Abramov [1, 2] and Chichov [7]. These authors presented what may be termed a spectral Schur complement method. It can easily be verified that a scalar λ is an eigenvalue of a matrix A partitioned as

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

if and only if it is an eigenvalue of $S(\lambda) = C - E(B - \lambda I)^{-1}F$ (this is clearly restricted to those λ's that are not in the spectrum of B). This nonlinear eigenvalue problem may be solved by a Newton-type approach. Alternatively, one can also devise special iterative schemes based on the above observation. An approach of this type is clearly limited by the fact that a Schur complement (or several consecutive ones in an iterative process) is required for each different eigenvalue. It can, however, work well for computing one or a few eigenvalues or in some other special situations. For example, this nonlinear viewpoint led to the development of effective shifts of origin for the QR algorithm for tridiagonal matrices [19] (see also Parlett [18]).

The fundamental premise of AMLS, and its attraction, is that it is capable of extracting very good approximations [illegible]. To achieve this, AMLS relies heavily on projection techniques. It builds good bases from one Schur complement, and expands them in an effective way to bigger and bigger domains.

This paper will begin with a brief overview of AMLS and an introduction of the notation. It will then introduce spectral (or nonlinear) Schur complements and explore their relations to AMLS. With this relation established, two enhancements of the basic AMLS scheme are presented. The first introduces Krylov subspaces to the technique and the second considers a more accurate (second order instead of first order) scheme. Combinations of these improvements are investigated. Finally, numerical experiments illustrate the performance of the suggested approaches.

2. The AMLS approach. Let $A \in \mathbb{C}^{n\times n}$ be a Hermitian matrix

$$A = \begin{pmatrix} B & E \\ E^* & C \end{pmatrix}, \tag{1}$$

such that $B \in \mathbb{C}^{(n-p)\times(n-p)}$, $C \in \mathbb{C}^{p\times p}$, and $E \in \mathbb{C}^{(n-p)\times p}$. Consider the linear eigenvalue problem $A\tilde{u} = \lambda\tilde{u}$, which can be written as

$$\begin{pmatrix} B & E \\ E^* & C \end{pmatrix} \begin{pmatrix} \tilde{u}^B \\ \tilde{u}^S \end{pmatrix} = \lambda \begin{pmatrix} \tilde{u}^B \\ \tilde{u}^S \end{pmatrix}, \tag{2}$$

where $\tilde{u}^B \in \mathbb{C}^{n-p}$ and $\tilde{u}^S \in \mathbb{C}^p$. Component mode synthesis (CMS) is a classical technique in structural dynamics for the computation of the smallest normal modes of a structure [10, 12]; see also [3] for a comparison of iterative algorithms for modal analysis.

A structure is typically approximated by a discrete domain Ω. In CMS the original structure is decomposed into several substructures separated by means of an interface Γ. Each of the substructures is approximated by a subdomain $\Omega_i \subset \Omega$. Notice

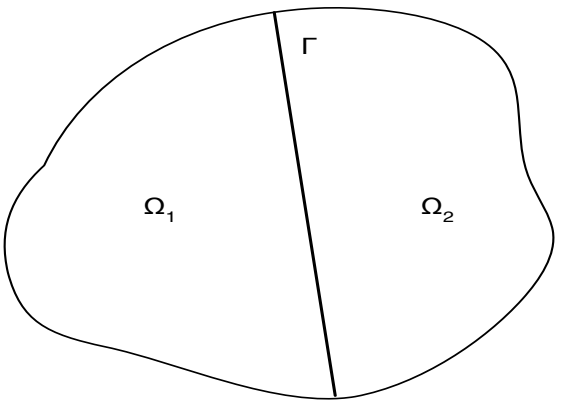


FIG. 1. *The simple case of two subdomains* Ω_1, Ω_2 *and an interface* Γ.

that the subdomains Ω_i can either overlap or not. In this paper we consider only the latter case. Thus, the unknowns in the interior of each subdomain Ω_i are completely decoupled from the unknowns in all other subdomains. Coupling among substructures is represented by the unknowns on the interface Γ and the unknowns in each Ω_i that are adjacent to Γ. Figure 1 provides a graphical illustration in the simplest case of two subdomains.

Consider now the simple model problem

$$-\nabla^2 \tilde{u} = \lambda \tilde{u} \tag{3}$$

on a rectangular domain Ω. If we employ a standard 5-point stencil, centered differences discretization of (3) and label the unknowns in the subdomains Ω_i first, followed by those on Γ, then the algebraic eigenvalue problem that arises from the discretization of (3) will be of the form (2). Notice that the matrix B acts on the unknowns of the domains Ω_i, while C corresponds to the unknowns on the interface Γ and E corresponds to the coupling between the unknowns in B and C. Furthermore, since the unknowns in the domains Ω_i are decoupled, B will be block diagonal.

The idea of `CMS` is to begin by solving the problem $Bv_j = \mu_j v_j$ for a number of the smallest eigenvalues. In particular, since B is block diagonal, this is equivalent to solving each of the decoupled smaller eigenvalue problems $B^i v_j^i = \mu_j^i v_j^i$ corresponding to each subdomain Ω_i (in parallel). The eigenvectors v_j^i are extended by zero outside the subdomain Ω_i. Then, `CMS` injects additional vectors v_Γ^k, that are not identically zero on the interface Γ, to account for the coupling among subdomains. This is done by invoking a carefully selected operator for the interface nodes (cf [5]). The complete eigenvectors (and eigenvalues) of the original eigenproblem are then approximated by Rayleigh–Ritz projection on the subspace

$$\texttt{span}\left\{v_j^i, i = 1, 2, \ldots, j = 1, 2, \ldots, \quad v_\Gamma^k, k = 1, 2, \ldots\right\}.$$

Recently, Bennighof and Lehoucq [5] have presented `AMLS` that extends the basic framework of `CMS` into a multilevel method. `AMLS` was reported to be orders of magnitude faster than competitive commercial software based on the traditional shift-and-invert Lanczos approach [16]. It was shown to be capable of computing thousands of the smallest normal modes of structures on commodity workstations.

Consider the matrix

$$U = \begin{pmatrix} I & -B^{-1}E \\ 0 & I \end{pmatrix}, \tag{4}$$

which is a block Gaussian eliminator for matrix (1), selected so that

$$U^*AU = \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix},$$

where S is the Schur complement

$$S = C - E^*B^{-1}E. \tag{5}$$

We now consider the application of `AMLS` for only a single level of subdivisions. The original problem (2) is replaced by the equivalent generalized eigenvalue problem $U^*AUu = \lambda U^*Uu$, which can be written as

$$\begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} u^B \\ u^S \end{pmatrix} = \lambda \begin{pmatrix} I & -B^{-1}E \\ -E^*B^{-1} & M_S \end{pmatrix} \begin{pmatrix} u^B \\ u^S \end{pmatrix}, \tag{6}$$

where $M_S = I + E^*B^{-2}E$. Observe that the eigenvectors $\tilde{u}$ of the original problem (2) are connected with the eigenvectors of the generalized problem (6) by

$$\begin{bmatrix} \tilde{u}^B \\ \tilde{u}^S \end{bmatrix} = U \begin{bmatrix} u^B \\ u^S \end{bmatrix},$$

because of the congruence transformation of (2) to (6).

`AMLS` computes approximations to the eigenvalues and eigenvectors of the original problem by means of a projection method applied to the generalized problem (6). The subspace of approximants is constructed from eigenvectors of a version of (6) in which the coupling in the right-hand side matrix (i.e., off-diagonal blocks in U^*U) is ignored. This gives the two decoupled eigenvalue problems:

$$Bv = \mu\, v, \tag{7}$$

$$Sw = \eta\, M_S w. \tag{8}$$

Once the desirable eigenpairs have been obtained from (7)–(8), they are utilized in a projection method (Rayleigh–Ritz) applied to the original problem (6). The basis used for this projection is of the form

$$\left\{ \hat{v}_i = \begin{pmatrix} v_i \\ 0 \end{pmatrix} \quad i = 1, \ldots, m_B, \qquad \hat{w}_j = \begin{pmatrix} 0 \\ w_j \end{pmatrix} \quad j = 1, \ldots, m_S \right\},$$

where $m_B < (n-p)$ and $m_S < p$. It is important to note that the projection is applied to (6) rather than to the original problem (2). There is an inherent change of basis between the two and, for reasons that will become clear later, the basis $\{\hat{v}_i\}_i, \{\hat{w}_j\}_j$, is well suited for the transformed problem rather than the original one.

The essence of `AMLS` is to "solve the problem (6) without the coupling and then use some eigenvectors obtained to solve the problem with its coupling." It is interesting to note that a very similar idea is utilized with great success in structural engineering: to solve a problem related to a dynamical system, one first obtains the modes of the

problem without damping, then uses these modes to define a subspace in which the full problem (with damping) is solved. We refer to Algorithm 2.1 for an algorithmic description of `AMLS` for one level of subdivisions (SPD stands for symmetric positive definite matrices). In the multilevel case, this algorithmic step is applied recursively for every subdomain of the original domain. Observe that in practice the block Gaussian eliminator matrix U (line 4) is not formed. Instead, Gaussian elimination is applied to compute the Schur complement S and the matrix M_S, since

$$U^*U = \begin{pmatrix} I & -B^{-1}E \\ E^*B^{-1} & M_S \end{pmatrix}.$$

In this paper we are interested in an analysis of the basic approximation mechanism of `AMLS`. This analysis will serve to devise a few improvements to the basic method. For simplicity, our study will be restricted to the single level of subdomains.

ALGORITHM 2.1. **`AMLS` for a single level of subdomains**

$A \in \mathbb{C}^{n\times n}$, $m_B < n_B < n$, $m_S < n$, n_e n_e A.

1. $\mathtt{B} = \mathtt{A}(1:\mathtt{n_B}, 1:\mathtt{n_B})$, $\mathtt{C} = \mathtt{A}(\mathtt{n_B}+1:\mathtt{end}, \mathtt{n_B}+1:\mathtt{end})$, $\mathtt{E} = \mathtt{A}(1:\mathtt{n_B}, \mathtt{n_B}+1:\mathtt{end})$.
2. $R^*R = B$.
3. $S = C - E^*B^{-1}E$ (R).
4. $U = [I, -B^{-1}E; 0, I]$.
5. $M_S = I + (E^*B^{-1})(B^{-1}E)$ R .
6. m_S W_S (S, M_S).
7. m_B V_B B.
8. $Z = [V_B, 0; 0, W_S]$.
9. n_e $(Z^*U^*AUZ)x = \lambda(Z^*U^*UZ)x$.

---

3. Spectral Schur complements. The following equations result from (6):

$$Bu^B = \lambda(u^B - B^{-1}Eu^S), \tag{9}$$

$$Su^S = \lambda(-E^*B^{-1}u^B + M_Su^S). \tag{10}$$

Assume that E is of full rank and B is nonsingular. In this situation, it is easy to see that (λ, u^B) cannot be an eigenpair of B. Then, any pair (λ, u^B), where (λ, u^B, u^S) is a solution of (6), cannot be an eigenpair for B. Indeed, if the contrary were to be true, then (9) would imply $u^S = 0$, from which (10) would imply that $\lambda = 0$. This is impossible since B is nonsingular. A consequence of this is that when λ is an eigenvalue of B, then (9) always has a solution in the orthogonal of the eigenspace of B associated with this eigenvalue. We can therefore always, theoretically, solve the system $(B - \lambda I)u_B = -B^{-1}u^S$ derived from (9) provided the inverse of B is interpreted as a pseudo-inverse. In what follows, $(B - \lambda I)^{-1}$ will always mean the pseudo-inverse $(B - \lambda I)^\dagger$ when λ is an eigenvalue of B. The notation is not changed because this is a rare occurrence in practice.

We can substitute (9) into (10) to obtain

$$Su^S = \lambda \left(\lambda E^* B^{-1}(B - \lambda I)^{-1} B^{-1} E u^S + M_s u^S\right),$$

which results in the equivalent nonlinear eigenvalue problem

$$\left[S - \lambda \left(E^* B^{-2} E\right) - \lambda^2 E^* B^{-1}(B - \lambda I)^{-1} B^{-1} E\right] \; u^S \; = \; \lambda u^S. \tag{11}$$

Rewriting the above problem in an expanded notation we obtain the following nonlinear eigenvalue problem

$$\left[C - E^* B^{-1}(B + \lambda I + \lambda^2 (B - \lambda I)^{-1}) B^{-1} E\right] u^S = \lambda u^S. \tag{12}$$

We can show that the above problem is equivalent to a nonlinear eigenvalue problem involving the spectral Schur complement

$$S(\lambda) = C - E^*(B - \lambda I)^{-1} E. \tag{13}$$

The first resolvent equality (see [14]),

$$(B - \lambda I)^{-1} - B^{-1} = \lambda (B - \lambda I)^{-1} B^{-1},$$

can be substituted in the expression of the left-hand matrix in (12) which we denote by $\hat{S}(\lambda)$,

$$\begin{aligned}
\hat{S}(\lambda) &= C - E^* B^{-1} \left(I + \lambda B^{-1} + \lambda (B - \lambda I)^{-1} - \lambda B^{-1}\right) E \\
&= C - E^* \left(B^{-1} + \lambda B^{-1} (B - \lambda I)^{-1}\right) E \\
&= C - E^* \left(B^{-1} - B^{-1} + (B - \lambda I)^{-1}\right) E \\
&= C - E^* (B - \lambda I)^{-1} E \\
&= S(\lambda).
\end{aligned}$$

In fact, the Schur complement S can be viewed as the first term of the Taylor series expansion of $S(\lambda)$ with respect to λ around $\lambda = 0$. The standard expansion of the resolvent (see [14])

$$(B - \lambda I)^{-1} = B^{-1} \sum_{k=0}^{\infty} (\lambda B^{-1})^k = \sum_{k=0}^{\infty} \lambda^k B^{-k-1}, \tag{14}$$

leads to the following series expansion for $S(\lambda)$:

$$\begin{aligned}
S(\lambda) &= C - E^* \sum_{k=0}^{\infty} (\lambda^k B^{-k-1}) E \\
&= C - E^* \left(B^{-1} + \lambda B^{-2} + \lambda^2 B^{-3} + \cdots\right) E. \qquad (15)
\end{aligned}$$

Observe that in `AMLS` the lower part of the projection basis Z (see Algorithm 2.1) is comprised of eigenvectors associated with the smallest eigenvalues of the generalized eigenproblem

$$Sw = \lambda M_S w, \tag{16}$$

which, after some algebraic manipulation, leads to the equation

$$\left(C - E^*(B^{-1} + \lambda B^{-2})E\right) w = \lambda w. \tag{17}$$

In light of relation (15), the prevously stated eigenproblem can clearly be considered as a truncated version of the original nonlinear problem (12), where we ignore the terms $\lambda^k B^{-k-1}$ for $k \geq 2$ in the expansion of the resolvent $(B - \lambda I)^{-1}$. Thus, the eigenvector w can be seen as a direct approximation to the bottom part u^S of the exact eigenvector of (6). In what follows, when w is considered to be a good enough approximation to u^S, we will often not distinguish between u^S and w.

The above observations immediately lead to some possible suggestions on how to improve the approximation by including additional terms of the infinite expansion. Section 5 will describe a second order approximation to (12) obtained by adding the term $\lambda^2 B^{-3}$. We begin by analyzing how `AMLS` expands the approximation of the lower part u^S to an approximation $[u^B; u^S]^*$ of an eigenvector of the complete problem (6).

4. The projection viewpoint. Consider again the nonlinear Schur complement (13). The eigenvalues of the original problem, which do not belong to the spectrum of B, can be obtained from those of the nonlinear eigenvalue problem

$$S(\lambda)x = \lambda x.$$

PROPOSITION 4.1. λ, u^S (11)

$$S(\lambda)u^S = \lambda u^S.$$

λ (2)

$$\begin{pmatrix} -(B - \lambda I)^{-1} E u^S \\ u^S \end{pmatrix}. \tag{18}$$

. The proof consists of a simple verification. □

Now assume that we have a good approximation to the nonlinear Schur complement problem, i.e., to a solution λ and u^S of the nonlinear problem (11). It is clear that the best we can do to retrieve the corresponding eigenvector of (2) is to use substitution, i.e., to compute the top part of (18)

$$u^B = -(B - \lambda I)^{-1} E u^S, \tag{19}$$

which will give us the top part of the exact eigenvector. This entails factoring the matrix $(B - \lambda I)$ for each different eigenvalue λ, which is not practical. As was seen in the previous section, `AMLS` extracts an approximation to the nonlinear problem (11) by solving the generalized eigenvalue problem (17) and then it replaces the substitution step (19) by a projection step. Specifically, once an approximate pair λ, u^S is obtained, `AMLS` computes approximate eigenvectors to the original problem by a projection process using the space spanned by the family of vectors

$$\left\{ \begin{pmatrix} v_i^B \\ 0 \end{pmatrix} \right\}, \qquad \left\{ \begin{pmatrix} -B^{-1} E u_j^S \\ u_j^S \end{pmatrix} \right\}, \tag{20}$$

in which v_i^B are eigenvectors of B associated with its smallest eigenvalues. Note that these two sets of vectors are of the form $U\begin{pmatrix} v_i^B \\ 0 \end{pmatrix}$ for the first, and $U\begin{pmatrix} 0 \\ u_j^S \end{pmatrix}$ for the second,

where U was defined earlier by (4). The question is: why is this a good way to replace the substitution step (19)? Another issue is the quality we might expect from this process.

Ideally, the prolongation matrix U should be replaced by one which depends on the eigenvalue λ, namely

$$U(\lambda) = \begin{pmatrix} I & -(B-\lambda I)^{-1}E \\ 0 & I \end{pmatrix}.$$

Indeed, if we were to use the prolongator $U(\lambda)$ instead of U, then $U(\lambda)\binom{0}{u_j^S}$ would be an exact eigenvector (if the approximation to u^S that we use were exact). Of course, it is not practical to use a different prolongator $U(\lambda)$ for each different eigenvalue λ. What is interesting and important to note is that $U(\lambda)$ and U are likely to be close to each other for small (in modulus) eigenvalues λ. Furthermore, [illegible] B [illegible] λ. It is helpful to examine this difference,

$$[U(\lambda) - U]\begin{pmatrix} 0 \\ u^S \end{pmatrix} = \begin{pmatrix} 0 & -((B-\lambda I)^{-1} - B^{-1})E \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 \\ u^S \end{pmatrix}$$
$$= \begin{pmatrix} -\lambda\ (B-\lambda I)^{-1}B^{-1}Eu^S \\ 0 \end{pmatrix}.$$

In order to compensate for this difference, it is natural to add to the subspace eigenvectors of B in which this difference will have large components, i.e., eigenvectors corresponding to the smallest eigenvalues of B. This is at the heart of the approximation exploited by `AMLS` which incorporates the first set in (20). Let us assume that we have the expansion in the eigenbasis of B

$$Eu^S = \sum_{k=1}^{N_B} \alpha_k v_k, \tag{21}$$

where $N_B = n - p$ is the dimension of B. Let $\hat{u}^B$ be the B-part (top part) of the approximation provided by applying the prolongation matrix U on vector $\binom{0}{u^S}$ and u^B be the top part of the exact eigenvector. Then

$$u^B - \hat{u}^B = -\lambda\ (B-\lambda I)^{-1}B^{-1}\ Eu^S = \sum_{k=1}^{N_B} \frac{\lambda\alpha_k}{(\mu_k - \lambda)\mu_k} v_k.$$

As expected, when $\lambda = 0$, the projection process will yield exact eigenvectors. The error in the eigenvector of (2) will be of the order of the distance between the above vector and the space spanned by the added eigenvectors of B. If we call X^B the restriction to the B-part of the space of approximants, then

$$dist(u^B, X^B) = \min_{v \in \ \mathtt{span}\ (v_1,\ldots,v_{m_B})} \|u^B - \hat{u}^B - v\|_2$$
$$= |\lambda| \left(\sum_{k>m_B} \frac{|\alpha_k|^2}{|\mu_k - \lambda|^2\ |\mu_k|^2} \right)^{1/2},$$

and therefore

$$dist(u^B, X^B) \leq \max_{k>m_B} \frac{|\lambda|}{|\mu_k - \lambda|\ |\mu_k|} \|Eu^S\|_2. \tag{22}$$

Under the mild assumption that $\lambda < \mu_{m_B}$, it is clear that

$$dist(u^B, X^B) \leq \frac{|\lambda|}{|\mu_{m_B} - \lambda| \ |\mu_{m_B}|} \|Eu^S\|_2. \tag{23}$$

Note that for $\lambda \ll \mu_{M_B}$, we will have $dist(u^B, X^B) \approx O(|\lambda/\mu_{M_B}^2|)$.

It is clear that we have three sources of error for the whole eigenvector $\left[\, u^B; u^S \,\right]$. For the bottom part u^S, the error comes from approximating it by solving the generalized eigenvalue problem (17). For the upper part u^B we have an error from using the projection described above, instead of using the prolongator $U(\lambda)$ and of course the error in using an approximate u^S in this projection. Notice, however, that our analysis has shown that it is beneficial to inject in the projection eigenvectors of B associated with eigenvalues that are closest to the (exact) smallest eigenvalues of the original problem. This is true regardless of the error in the approximation to u^S. In particular, let u^B be exact and assume that instead we use an approximation $u^B + \delta$, where $\|\delta\|$ is small. Then, using the expansion $E\delta = \sum_{k=1}^{N_B} \xi_k v_k$, where v_k are the eigenvectors of B, we now have

$$u^B - \hat{u}^B = -\lambda \ (B - \lambda I)^{-1} B^{-1} \ Eu^S - B^{-1} E\delta = \sum_{k=1}^{N_B} \left(\frac{\lambda \alpha_k}{(\mu_k - \lambda)\mu_k} - \frac{\xi_k}{\mu_k} \right) v_k,$$

and thus we can similarly show for the revised distance $dist(u^B, X^B)$

$$dist(u^B, X^B) \leq \left(\sum_{k>m_B}^{N_B} \frac{|\lambda|^2 |\alpha_k|^2 + |\xi_k|^2 |\mu_k - \lambda|^2 - 2|\lambda \alpha_k \xi_k (\mu_k - \lambda)|}{|\mu_k - \lambda|^2 \ |\mu_k|^2} \right)^{1/2}, \tag{24}$$

which readily reduces to (23) for small enough $\|\delta\|$.

5. Higher degree approximations. The analysis in previous sections has demonstrated that `AMLS` approximates the nonlinear eigenvalue problem (12) by the generalized eigenvalue problem (16). In particular, this problem can be seen as a truncated, first order approximation of (12). Let us now consider a second order approximation obtained by solving the quadratic eigenvalue problem

$$[\lambda^2 E^* B^{-3} E + \lambda (I + E^* B^{-2} E) - S] q \ = \ 0. \tag{25}$$

This problem can be solved by a linearization which leads to an equivalent generalized eigenvalue problem (see for example [22])

$$\begin{pmatrix} 0 & I \\ S & -K \end{pmatrix} \begin{pmatrix} q \\ \lambda q \end{pmatrix} = \lambda \begin{pmatrix} I & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} q \\ \lambda q \end{pmatrix}, \tag{26}$$

where $K = I + E^* B^{-2} E$ and $H = E^* B^{-3} E$. As in the previous sections, when q is an accurate approximation to the exact bottom part of the eigenvector, i.e., u^S, we will not distinguish between these vectors.

In section 4 we showed that adding eigenvectors of B to the space of approximants can improve the basic approximation provided by the nonlinear Schur complement. In order for the approximate eigenvectors provided by the solution of (26) to fit in well with the projection step, we need to modify the latter so that the terms $-(B - \lambda I)^{-1} Eu^S$ are well approximated. One way to achieve this is to add more

eigenvectors of B. However, a more relevant alternative in this case is to add a third set of vectors to the basis of approximants (20), namely the set:

$$\left\{ \begin{pmatrix} -B^{-2}Eu_j^S \\ u_j^S \end{pmatrix} \right\}, \tag{27}$$

which corresponds to the truncation of the resolvent expansion (14) from the third term on, rather than from the second term. Observe that including the approximate eigenvalue λ in the term $-\lambda B^{-2}$ (see (14)) will not affect the direction of the added vectors, and thus is avoided.

In order to apply the projection, we construct the matrix

$$Z = \begin{pmatrix} V_B & -B^{-1}EU_S & -B^{-2}EU_S \\ 0 & U_S & 0 \end{pmatrix}, \tag{28}$$

where V_B is a matrix of eigenvectors corresponding to the smallest eigenvalues of B and U_S is a matrix of eigenvectors associated with the smallest eigenvalues of (26), in modulus. We then solve the projected generalized eigenvalue problem

$$Z^* \begin{pmatrix} B & E \\ E^* & C \end{pmatrix} Zx = \lambda Z^* Zx. \tag{29}$$

Algorithm 5.1 illustrates a second order version of `AMLS`, based on this approach.

ALGORITHM 5.1. **Second order** `AMLS`

[illegible] $A \in \mathbb{C}^{n\times n}$, [illegible] $n_B < n$, $m_S < n$,
[illegible] n_e [illegible]
[illegible] n_e [illegible] A.

1. [illegible] $\mathtt{B} = \mathtt{A}(1:\mathtt{n_B}, 1:\mathtt{n_B})$, $\mathtt{C} = \mathtt{A}(\mathtt{n_B}+1:\mathtt{end}, \mathtt{n_B}+1:\mathtt{end})$,
   $\mathtt{E} = \mathtt{A}(1:\mathtt{n_B}, \mathtt{n_B}+1:\mathtt{end})$.
2. [illegible] $R^*R = B$.
3. [illegible] $S = C - E^*B^{-1}E$ [illegible] R.
4. [illegible] $K = I + (E^*B^{-1})(B^{-1}E)$ $R = (E^*B^{-1})B^{-1}(B^{-1}E)$
   [illegible] L.
5. [illegible] m_S [illegible] U_S
   [illegible] (25) [illegible] (26).
6. [illegible] m_B [illegible] V_B *of* B.
7. [illegible] Z [illegible] (28).
8. [illegible] n_e [illegible] $(Z^*AZ)x = (Z^*Z)x$.

Consider now the effect of adding the vectors (27). We are looking again at the distance between u^B, which is the top part of the exact eigenvector, and the closest vector from the space of approximants X^B. Let $\hat{u}^B$ be as in the previous section and consider the vector $u^{(2)} = -B^{-2}Eu^S$. Assuming that the expansion (21) is available, then

$$u^B - \hat{u}^B - \lambda u^{(2)} = -\lambda^2 \, (B - \lambda I)^{-1} B^{-2} \, Eu^S = \sum_{k=1}^{N_B} \frac{\lambda^2 \alpha_k}{(\mu_k - \lambda)\mu_k^2} v_k.$$

Therefore, the distance of u^B from the space of approximants X^B is such that

$$dist(u^B, X^B) = \min_{v \in \mathtt{span}\{V_B\}} \|u^B - \hat{u}^B - \lambda u^{(2)} - v\|_2$$
$$\leq |\lambda|^2 \left(\sum_{k>m_B} \frac{|\alpha_k|^2}{|\mu_k - \lambda|^2 \ |\mu_k|^4} \right)^{1/2}$$

and therefore,

$$dist(u^B, X^B) \leq \max_{k>m_B} \frac{|\lambda|^2}{|\mu_k - \lambda| \ |\mu_k|^2} \|Eu^S\|_2. \tag{30}$$

Under the mild assumption that $\lambda < \mu_{m_B}$, we have the bound

$$dist(u^B, X^B) \leq \frac{|\lambda|^2}{|\mu_{m_B} - \lambda| \ |\mu_{m_B}|^2} \|Eu^S\|_2. \tag{31}$$

Concerning u^B, a comparison of the above bound with the bound (23) for the first order approximation, indicates that one may expect a good improvement when λ is small. Numerical experiments in section 8 will verify this prediction.

Concerning the source of errors, the situation is similar to the previous section. If we assume that instead of the exact bottom part u^S we use an approximation $u^S + \delta$, with $\|\delta\|$ being small, and we employ the expansion $E\delta = \sum_{k=1}^{N_B} \xi_k v_k$, then

$$u^B - \hat{u}^B - \lambda u^{(2)} = -\lambda^2 \ (B - \lambda I)^{-1} B^{-2} \ Eu^S = \sum_{k=1}^{N_B} \left(\frac{\lambda^2 \alpha_k}{(\mu_k - \lambda)\mu_k^2} - \frac{\xi_k}{\mu_k^2} \right) v_k,$$

which again leads to the conclusion that a small error $\|\delta\|$ will not severely affect the accuracy of the `AMLS` projection. In particular, based on the above relation, for the distance $dist(u^B, X^B)$ we can show that

$$dist(u^B, X^B) \leq \left(\sum_{k>m_B} \frac{|\lambda|^4 |\alpha_k|^2 + |\xi_k|^2 |\mu_k - \lambda|^2 - 2|\lambda^2 \alpha_k \xi_k (\mu_k - \lambda)|}{|\mu_k - \lambda|^2 \ |\mu_k|^4} \right)^{1/2}, \tag{32}$$

which reduces to (31) for small enough $\|\delta\|$.

Computational considerations. In order to compute the Schur complement S we need to compute the Cholesky factorization of $B = R^* R$, and then solve upper and lower triangular systems with p right-hand sides in order to compute $B^{-1}E$ (see the definition for matrix (1)). The previous computation is the same that is required in the original `AMLS` approach. Note that B is sparse and block diagonal. In fact, in the multilevel version of `AMLS`, factoring B with a sparse direct solver can be realized with a form of the nested dissection algorithm [9]. The second order method requires in addition the computation of $B^{-1}(B^{-1}E)$. Assuming that the result is computed as a dense matrix of size $n \times p$ and that the factors of B have $nnz(B_{fill})$ nonzero entries then this computation will cost an additional $p \times nnz(B_{fill})$ operations (p additional backward-forward solves with B). The solution of the generalized eigenvalue problem (26) is of dimension $2p$ with a cost of at most $O(p^3)$. However, note that in practice p is much smaller than the size n of the original problem. In the multilevel version of the method, we would need to recursively repeat the above calculations for each of the diagonal blocks of B which very quickly become much smaller than the original matrix A.

6. Expanding the projection space with Krylov subspaces. Section 4 described AMLS as a projection technique which uses expanded subspaces of the form (20) to extract Ritz values and vectors. We denote by $\mathcal{V}_S$ the subspace

$$\mathcal{V}_S = \mathtt{span}\left\{ \begin{pmatrix} -B^{-1}Eu_j^S \\ u_j^S \end{pmatrix} \right\}, \tag{33}$$

where u_j^S are eigenvectors of the generalized eigenvalue problem (10). In this section we propose a different approach that is based, however, on similar reasoning. Observe that, ideally, the top part of $\mathcal{V}_S$ should be $-(B-\lambda I)^{-1}Eu_j^S$. Therefore, instead of attempting to compensate for the difference $(B-\lambda I)^{-1} - B^{-1}$, we can approximate the resolvent $(B-\lambda I)^{-1}$ itself. The key is to expand the subspace $\mathcal{V}_S$ with the vectors $[V_{\mathcal{K}}; 0]$, where $V_{\mathcal{K}}$ is an orthonormal basis for the Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$ and U_S is a matrix of approximate eigenvectors of (10). Algorithm 6.1 is a description of a method based on this approach, which will be subsequently called Krylov AMLS. Observe that as in standard AMLS the block Gaussian eliminator matrix U (line 4) is never formed. Instead, a forward-backward solve with the factors of B is applied where needed, since we assume that B has been factored at the outset.

For simplicity, we examine the case where U_S has just one column. The error in the eigenvector of (2) will be of the order of the distance

$$\|(B-\lambda I)^{-1}Eu_j^S - u\|, \quad \text{where} \quad u \in \mathcal{K}_m(B^{-1}, Eu_j^S).$$

Note that $u = P_m(B^{-1})Eu_j^S$ for a certain polynomial P_m of degree $m-1$. If X^B denotes again the restriction to the B-part of the space of approximants, which in the present case is equal to $\mathcal{K}_m(B^{-1}, Eu_j^S)$, then expanding the resolvent $(B-\lambda I)^{-1}$ yields,

$$dist(u^B, X^B) = \min_{\gamma_k} \left\| \left(\sum_{k=0}^{\infty} \lambda^k B^{-k-1} - \sum_{k=0}^{m-1} \gamma_k B^{-k} \right) Eu_j^S \right\|.$$

Let $\mu_{\min}$ be the smallest eigenvalue of B. Then, notice that we can select the coefficients γ_k in order to exactly match the first $m-1$ terms of the Neumann series of the resolvent. Thus we have

$$dist(u^B, X^B) \leq \frac{1}{\mu_{\min} - \lambda} \left(\frac{\lambda}{\mu_{\min}} \right)^{m-1} \|Eu_j^S\|. \tag{34}$$

Therefore, if the ratio $\lambda/\mu_{\min}$ is small, then we can expect a very small distance $\|(B-\lambda I)^{-1}Eu^S - \mathcal{K}_m(B^{-1}, Eu_j^S)\|$. However, we must note that in practice we only have approximations to the eigenvalues and eigenvectors of the generalized eigenvalue problem (10). Similar to the previous sections, let us now assume that instead of an exact bottom part we use an approximate $u_j^S + \delta$, where u_j^S is exact and $\|\delta\|$ is small. The new distance $dist(u^B, X^B)$ now satisfies

$$dist(u^B, X^B) = \min_{\gamma_k} \left\| \left(\sum_{k=0}^{\infty} \lambda^k B^{-k-1} - \sum_{k=0}^{m-1} \gamma_k B^{-k} \right) Eu_j^S - \sum_{m=0}^{m-1} \gamma_k B^{-k} E\delta \right\|$$

$$\leq \min_{\gamma_k} \left\| \left(\sum_{k=0}^{\infty} \lambda^k B^{-k-1} - \sum_{k=0}^{m-1} \gamma_k B^{-k} \right) Eu_j^S \right\| + \left\| \sum_{m=0}^{m-1} \gamma_k B^{-k} E\delta \right\|. \tag{35}$$

ALGORITHM 6.1. **Krylov AMLS**

[illegible] $A \in \mathbb{C}^{n\times n}$, [illegible] $n_B < n$, $m_S < n$, m_K,
[illegible] n_e [illegible]
[illegible] n_e [illegible] A.

1. [illegible] $\mathtt{B} = \mathtt{A}(1:\mathtt{n_B}, 1:\mathtt{n_B})$, $\mathtt{C} = \mathtt{A}(\mathtt{n_B}+1:\mathtt{end}, \mathtt{n_B}+1:\mathtt{end})$,
   $\mathtt{E} = \mathtt{A}(1:\mathtt{n_B}, \mathtt{n_B}+1:\mathtt{end})$.
2. [illegible] $R^*R = B$.
3. [illegible] $S = C - E^*B^{-1}E$ [illegible] R.
4. [illegible] $U = [I, -B^{-1}E; 0, I]$.
5. [illegible] $M_S = I + (E^*B^{-1})(B^{-1}E)$ [illegible] R.
6. [illegible] m_S [illegible] $Su_j^S = M_S u_j^S$ [illegible] $U_S = [u_1^S, \ldots, u_{m_S}^S]$.
7. [illegible] V_K [illegible] $\mathcal{K}_{m_K}(B^{-1}, EU_S)$.
8. [illegible] $Z_K = [V_K, 0; 0, U_S]$.
9. [illegible] n_e [illegible] $(Z_K^*U^*AUZ_K)x = (Z_K^*U^*UZ_K)x$.

Consider again the expansion of $E\delta = \sum_{i=1}^{N_B} \xi_i v_i$ in the eigenbasis of B. We have that

$$\left\| \sum_{m=0}^{m-1} \gamma_k B^{-k} E\delta \right\| = \left\| \sum_{k=0}^{m-1} \sum_{i=1}^{N_B} \gamma_k \xi_i B^{-k} v_i \right\| \tag{36}$$

$$= \left\| \sum_{i=1}^{N_B} \xi_i \left(\sum_{k=0}^{m-1} \gamma_k \mu_i^{-k} \right) v_i \right\|. \tag{37}$$

Setting again γ_k in order to exactly match the first $m-1$ terms of the Neumann series of the resolvent we see that apart from the small coefficient ξ_i, we have

$$\sum_{k=0}^{m-1} \gamma_k \mu_i^{-k} = \sum_{k=0}^{m-1} \lambda^k \mu_i^{-k} = \frac{1 - (\frac{\lambda}{\mu_i})^{m-1}}{1 - \frac{\lambda}{\mu_i}},$$

and thus the distance $dist(u^B, X^B)$, can be large for eigenvalues λ that are very close to eigenvalues μ_i of B.

The bound (34) justifies the use of the block Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$, where U_S is a matrix of eigenvectors u_j^S of (10). Numerical experiments in section 8 will confirm that using block Krylov subspaces will result in significantly improved approximations of the smallest eigenvalues relative to the approximations of standard AMLS.

Computational considerations. As in the original AMLS method, in order to compute the Schur complement S we need to compute the Cholesky factorization of $B = R^*R$, and then solve upper and lower triangular systems with p right-hand sides. The same comments as before apply for the cost of factoring B. Instead of eigenvectors of B associated with the smallest eigenvalues, the block Krylov method requires the Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$. This can be computed by means of the (block) Lanczos method [20]. Since we already have available the Cholesky decomposition of B, then each of the m steps of Lanczos will require the solution of m_S upper and lower triangular system solves that entail a total cost of $m \times m_S \times nnz(B_{fill})$ operations.

However, it is well-known that in the Lanczos algorithm, the vectors of the Krylov basis rapidly loose orthogonality and thus some form of reorthogonalization is often required. Since the size of B is $n-p$, where p is the size of the Schur complement, the cost of reorthogonalization can reach $O((n-p)(mm_S)^2)$. On the other hand, in the practical multilevel version of the method, the computation of the Krylov basis will be performed only at the very last level l of subdivisions on submatrices $B_{i,l}$ that are much smaller than the original matrix A. Remember that in the original `AMLS` method we need to compute the smallest eigenvalues of the same submatrices $B_{i,l}$, a computation that requires again the construction of Krylov subspaces with $B_{i,l}^{-1}$. Therefore, there is no reason to expect the overall cost of the Krylov `AMLS` method to be noticeably higher than that of the original `AMLS` method.

7. Combining AMLS with second order and Krylov AMLS. In the two previous sections we presented improvements of the standard method that are based on our two basic observations concerning the approximation scheme of `AMLS`: a) second order approximations to the nonlinear eigenvalue problem (11) and b) block Krylov basis in the projection step. It is therefore natural to consider combinations of these improvements. In particular, the projection matrix (28) can be substituted by

$$Z = \begin{pmatrix} V_{\mathcal{K}} & -B^{-1}EU_S & -B^{-2}EU_S \\ 0 & U_S & 0 \end{pmatrix}, \tag{38}$$

where instead of the matrix of eigenvectors V_B we use the Krylov basis $V_{\mathcal{K}}$ for the block Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$. The effects of this combination will be further explored in the numerical experiments section.

Observe that in the second order `AMLS` the projection basis for the approximation of the $B-$part of the eigenvectors of the original matrix A is augmented by directions in the vectors $-B^{-2}EU_S$, which clearly belong to the block Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$. Thus, the projection step of the second order `AMLS` can be viewed as a special case of the projection step of Krylov `AMLS`. However, observe that in the case when U_S has several columns the cost for the computation of the Krylov basis $V_{\mathcal{K}}$ can be uneconomical. Thus, a practical application of the above scheme may limit the number of initial vectors from the columns of the matrix U_S. Clearly in this case, the projection step of the block Krylov method will differ from the projection step of the second order method. Using the basis (38) (where $V_{\mathcal{K}}$ utilizes only a subset of initial vectors from EU_S) attempts to combine the advantages of the improvements suggested in the two previous sections.

An alternative type of combination occurs when in Krylov `AMLS` we augment the approximation subspace of the projection step with eigenvectors of B associated with its smallest eigenvalues. The reasoning for such a combination is based on the fact that in order for the block Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$ to adequately approximate the resolvent $(B-\lambda I)^{-1}$ for several different eigenvalues λ, without significantly increasing the Krylov steps m, we need to include many eigenvectors in the starting vectors matrix U_S. In an attempt to avoid this we can insert eigenvectors of B in the subspace as in standard `AMLS`. Thus, the projection matrix is

$$Z = \begin{pmatrix} V_B & V_{\mathcal{K}} & 0 \\ 0 & 0 & U_S \end{pmatrix}, \tag{39}$$

where V_B is a matrix of eigenvectors of B and $V_{\mathcal{K}}$ is an orthonormal basis for the block Krylov subspace. The projected eigenproblem is $(Z^*U^*AUZ)x = (Z^*U^*UZ)x$, where Z is as defined above and U is the block Gaussian eliminator matrix (4).

Finally, we can consider a combination of all of the above. The matrix of smallest eigenvectors U_S of (11) is approximated by solving for the smallest eigenvalues of (25). Define the projection basis

$$Z = \begin{pmatrix} V_B & V_{\mathcal{K}} & -B^{-1}EU_S & -B^{-2}EU_S \\ 0 & 0 & U_S & 0 \end{pmatrix}, \tag{40}$$

and approximate the smallest (in modulus) eigenvalues of A by the smallest (in modulus) eigenvalues of the projected problem $(Z^*AZ)x = \lambda(Z^*Z)x$. Matrices V_B and $V_{\mathcal{K}}$ correspond to the smallest eigenvectors of B and to the block Krylov subspace $\mathcal{K}_m(B^{-1}, E\hat{U}_S)$, respectively, where the columns of $\hat{U}_S$ form a subset of the columns of U_S.

At this point it is important to emphasize that the main goal of this paper has been to provide an algebraic framework for the analysis of `AMLS`. This viewpoint and the related analysis lead to a few practical improvements of the basis scheme. However, it must be understood that the full potential of the method, as it was demonstrated in [16], lies in its recursive nature. This attribute of the method is particularly appealing as it enables the computation of a large number of eigenvalues of very large matrices. The main difference between `AMLS` and its competitors, the Lanczos based methods, is that `AMLS` is capable of building large projection bases, while requiring a much lower reorthogonalization cost. On the other hand, it should also be clear that efficient implementations of `AMLS` are far more complex than those of competitive approaches [13, 23, 24].

8. Numerical experiments. The goal of the numerical tests described in this section is to verify the theoretical predictions of sections 4, 5, and 6. Furthermore, we investigate the application of the combinations suggested in the previous section.

8.1. AMLS and Krylov AMLS. Bounds (23) and (34) suggest that if the smallest eigenvalue $\mu_{\min}$ of B is much larger than the smallest eigenvalue λ of matrix A, then the Krylov subspace framework in `AMLS` will yield better results than standard `AMLS`. We tested this prediction in the following experiment. The test matrix is `BCSSTK11` from the Harwell–Boeing collection (available from the Matrix Market[1]; see [6]). This is an SPD matrix of size $n = 1473$, which has $\texttt{nnz} = 34241$ nonzero entries. The matrix was symmetrically reordered using the nested dissection graph partitioning algorithm so that the matrix B will have block diagonal structure with two blocks (corresponding to two subdomains).

Figure 2 illustrates the nonzero structure of the matrix before (left) and after the partitioning (right). The sizes of the resulting two blocks of B are $n_1 = 696$ and $n_2 = 683$. The smallest eigenvalue of the matrix is $\lambda \approx 2.9641$ and the smallest eigenvalue of B is $\mu_{\min} \approx 40.3973$, so that $\lambda/\mu_{\min} \approx 0.0734$. We computed the five smallest eigenvalues ($m_S = 5$) and respective eigenvectors $u_j^S, j = 1, \ldots, 5$ of the generalized Schur eigenvalue problem (8). We then computed the $m_B = 10, 20, 30$, and 40 smallest eigenvalues and corresponding eigenvectors of the block diagonal simple eigenvalue problem (7). For the Krylov subspace framework in `AMLS`, we computed a basis for the block Krylov subspace $\mathcal{K}_m(B^{-1}, EU_S)$, where the columns of U_S are the eigenvectors u_j^S, for $m_K = 2, 4, \ldots, 8$. Figure 3 illustrates the absolute relative error obtained for the 5 smallest eigenvalues of the matrix for the selected values of m_K and m_B. We selected these values so that the dimension of the projection space will be

[1] `http://math.nist.gov/MatrixMarket`

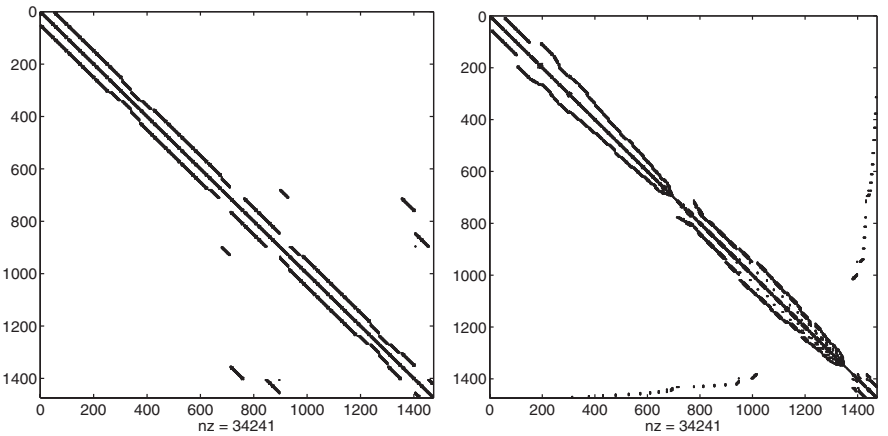


FIG. 2. *The nonzero structure of matrix* BCSSTK11 *before* (*left*) *and after* (*right*) *the nested dissection partitioning.*

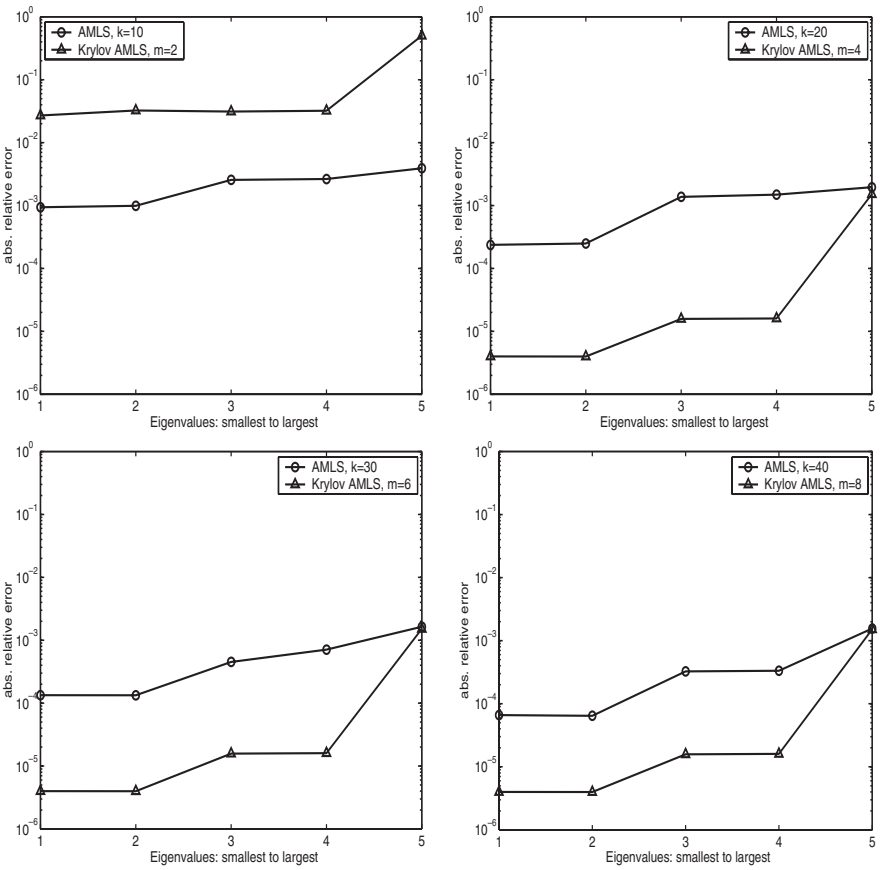


FIG. 3. *Relative errors for the five smallest eigenvalues of matrix* BCSSTK11 *computed, after a nested dissection reordering, by* AMLS *and Krylov* AMLS *for projection subspace dimensions equal to* $k = 10, 20, 30, 40$.

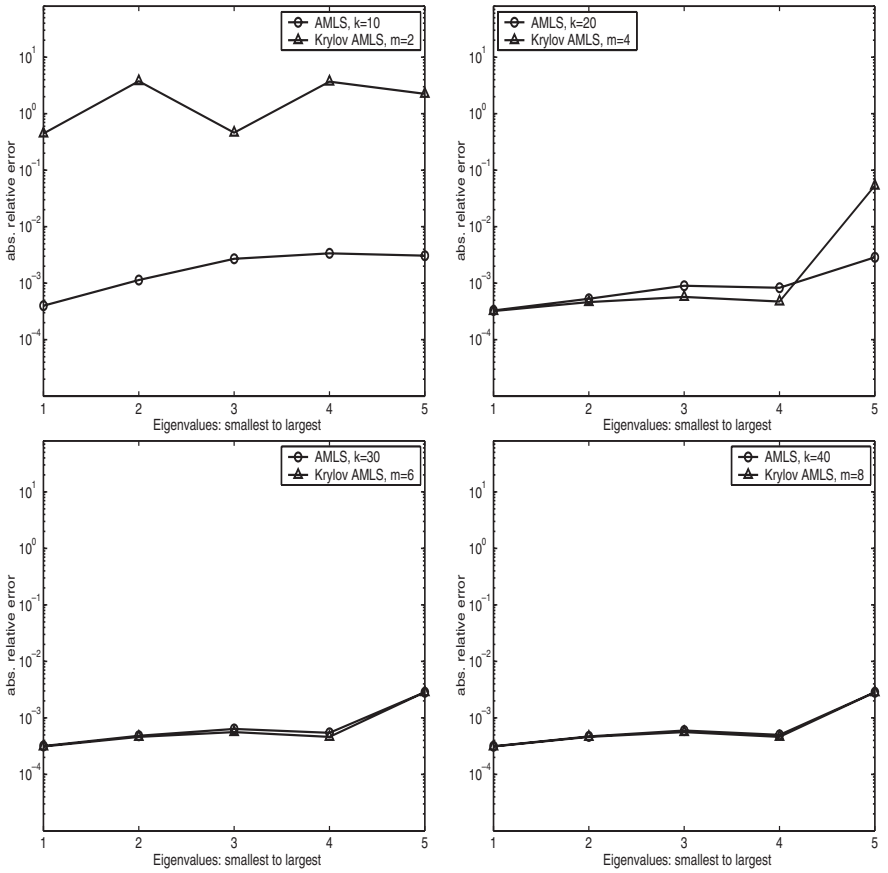


FIG. 4. *Relative errors for the five smallest eigenvalues of matrix* `BCSSTK11` *computed, without a nested dissection reordering, by* `AMLS` *and Krylov* `AMLS` *for projection subspace dimensions equal to* $k = 10, 20, 30, 40$.

equal in all cases. Observe that Krylov `AMLS` outperforms standard `AMLS` in all cases except when the dimension of the Krylov subspace is very small. It is interesting to note that Krylov `AMLS` for $m_K = 4$ (where the dimension of the projection subspace was 25) clearly outperforms standard `AMLS` for $m_B = 40$ (where the dimension of the projection subspace was 45).

We now consider the case where the ratio $\lambda/\mu_{\min}$ is close to 1. Observe that from interlacing properties of eigenvalues of Hermitian matrices, this ratio cannot be larger than 1. For this reason, we now fail to perform the nested dissection partitioning of the original matrix. As in the previous case we select the size of B to be $n_B = n_1 + n_2 = 1379$. In this case $\mu_{\min} \approx 3.3975$ and thus $\lambda/\mu_{\min} \approx 0.8724$. Figure 4 illustrates the absolute relative error obtained for the 5 smallest eigenvalues of the matrix for the same span of values of m_B and m_K as in the previous experiments. It is clear that in this case the gains of the Krylov `AMLS` framework compared to standard `AMLS` are not as compelling as when the ratio $\lambda/\mu_{\min}$ is much smaller than 1.

8.2. Experiments with the second order AMLS. In order to test the second order `AMLS` described in section 5, we used matrix `BCSSTK11` reordered as in the

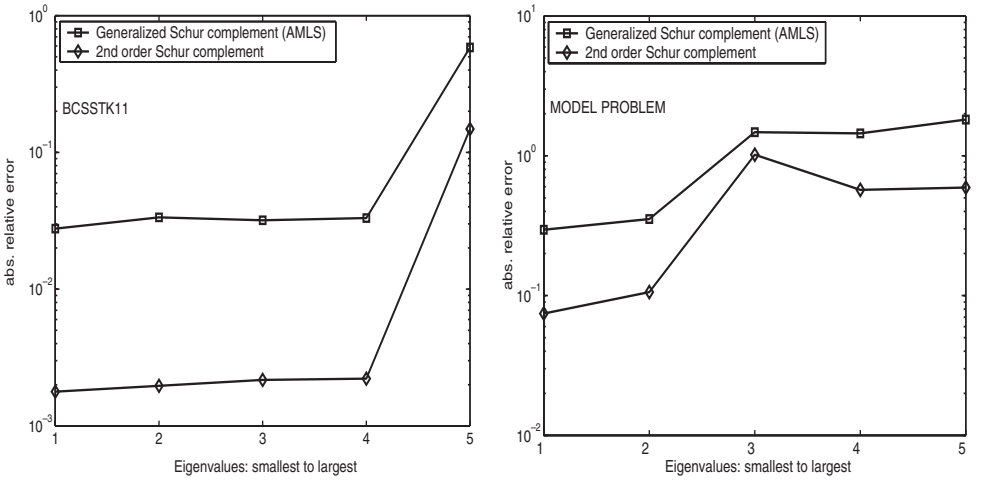


FIG. 5. *Absolute relative errors for the five smallest eigenvalues of matrix* `BCSSTK11` (*left*) *and of the model problem* (3)*, as they were approximated by the five smallest eigenvalues of* (16) *and by the five smallest, in modulus, eigenvalues of* (26).

previous experiment. A second test matrix results from a 5-point stencil discretization of the simple model problem (3) on the unit square, with $n_x = n_y = 30$ internal mesh nodes in each direction, so that the matrix size is $n = 900$. The matrix is reordered in order for the principal submatrix B to be block diagonal with two blocks and size $n_B = 840$.

The second order polynomial eigenvalue problem (25) is solved via the linearized problem (26). We first show that we are able to extract improved approximations to the smallest eigenvalues of the nonlinear Schur eigenvalue problem (13) by means of the second order method. Figure 5 illustrates the absolute relative error for the approximations of the five smallest eigenvalues of (13) (for both test matrices), which correspond to the five smallest eigenvalues of the original matrices. It is clear that the second order approximation produces improved approximations. We now use these approximations, and the corresponding eigenvectors, in the Rayleigh–Ritz projection with the basis (28).

For standard `AMLS` we used an increasing number m_B of added eigenvectors of B: $m_B = 20, 40, 60$, and 100. On the other hand, for the second order `AMLS` we used $m_B = 10$ eigenvectors of B and $m_S = 2 \times 10$ additional vectors: 10 for $B^{-1}EU_S$ and another 10 for $B^{-2}EU_S$. Thus, in all cases the matrix U_S had 10 columns which are eigenvectors corresponding to the smallest eigenvalues (in modulus) of the quadratic eigenvalue problem (25). Figure 6 illustrates the results for both test matrices (left: `BCSSTK11`, right: model problem). The accuracies achieved by the second order method are significantly better than those of the standard method. Notice in particular that for `BCSSTK11`, the error achieved by the second order `AMLS` method is of the order of the square of that obtained by standard `AMLS`, even when this method utilizes as many as $m_B = 100$ eigenvectors of B.

8.3. Combining the second order AMLS with Krylov subspaces. We now experiment with the combinations of the second order `AMLS` with Krylov subspaces. Note that this entails computing a block Krylov subspace with B^{-1} rather

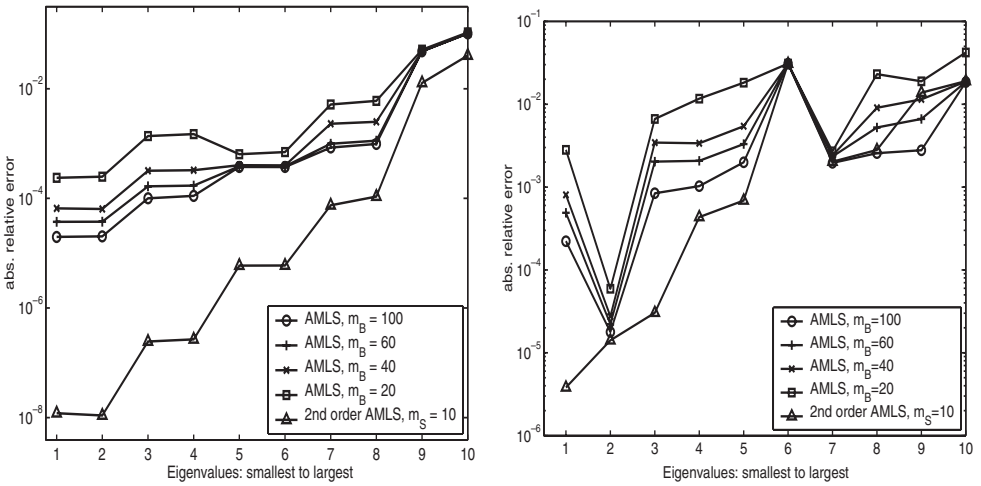


FIG. 6. *Absolute relative errors for the* 10 *smallest eigenvalues computed by standard* `AMLS` *and second order* `AMLS` $(-\triangle-)$ *for matrices* `BCSSTK11` (*left*) *and for the model problem* (*right*).

than a set of eigenvectors of B (cf the algorithmic description in Algorithm 5.1). We use matrix `BCSSTK11` and the model problem matrix as in the previous section.

We begin with the computation of the two smallest eigenvalues. We use five vectors in U_S, corresponding to the five smallest in modulus eigenvalues of (25). For the second order `AMLS` we use ten eigenvectors of B, corresponding to its ten smallest eigenvalues, while in the block Krylov version we use $E[u_1^S, u_2^S]$ as starting vectors, where $u_j^S, j = 1, 2$ correspond to the two smallest eigenvalues in modulus of (25). We then compute a basis of dimension ten; Figures 7 and 8 (left) illustrate the results. It is clear that introducing Krylov basis vectors, instead of eigenvectors of B, yields improved results which are especially pronounced for the smallest eigenvalues in the case of the model problem. For larger eigenvalues, however, the situation is reversed. This is due to the fact that the starting vectors of the block Krylov subspace are poor in directions towards $Eu_j^S, j = 1, \ldots$.

We then used five eigenvectors from (25) and computed a Krylov subspace of dimension twenty five. For comparison, we also augmented the partial eigenbasis of B that we use in the second order `AMLS` to a total of 25 eigenvectors. Figures 7 and 8 (right) illustrate the results. It is clear that in this case the additional starting vectors in the block Krylov subspace have a positive effect on improving the error for the corresponding largest eigenvalues, while the approximations for the two smallest eigenvalues remain practically unchanged. Compared to using eigenvectors of B, we observe that for the two smallest eigenvalues the block Krylov subspace approach remains in general superior, though the difference is not as pronounced as in the previous case. This is due to the increased dimension of the projection basis. On the other hand, for the larger eigenvalues we witness similar approximations for both approaches. It appears that in order to achieve an improved approximation for larger eigenvalues we need to either use more starting vectors for the block Krylov basis, or to use a significantly larger basis while keeping the number of starting vectors low. However, both of the above approaches will lead to increased computational cost.

8.4. Combining AMLS and Krylov AMLS. Finally, we experiment with combinations of the Krylov and second order `AMLS` with the original `AMLS` scheme. In

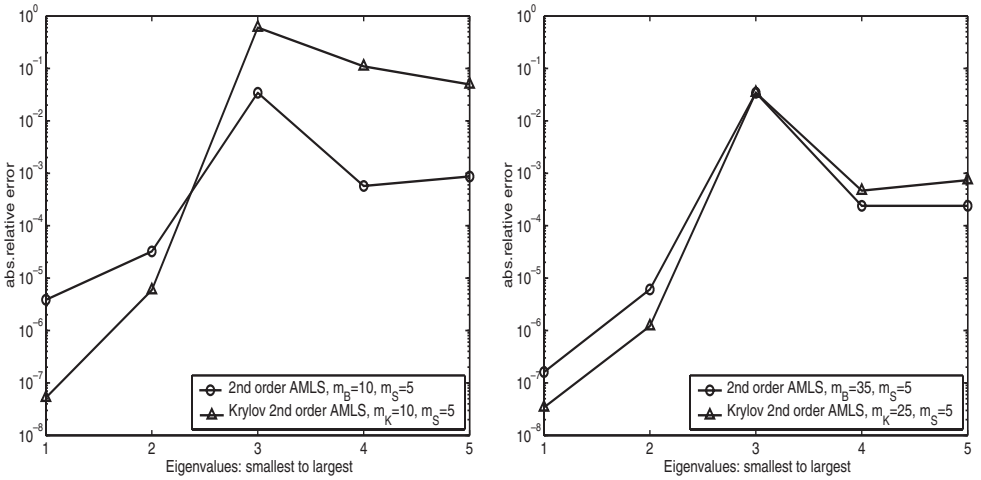


FIG. 7. *Model problem: absolute relative errors for the* 5 *smallest eigenvalues of* (3) *computed by the second order* AMLS $(-\mathrm{o}-)$ *and second order* AMLS *with block Krylov subspace* $(-\triangle-)$.

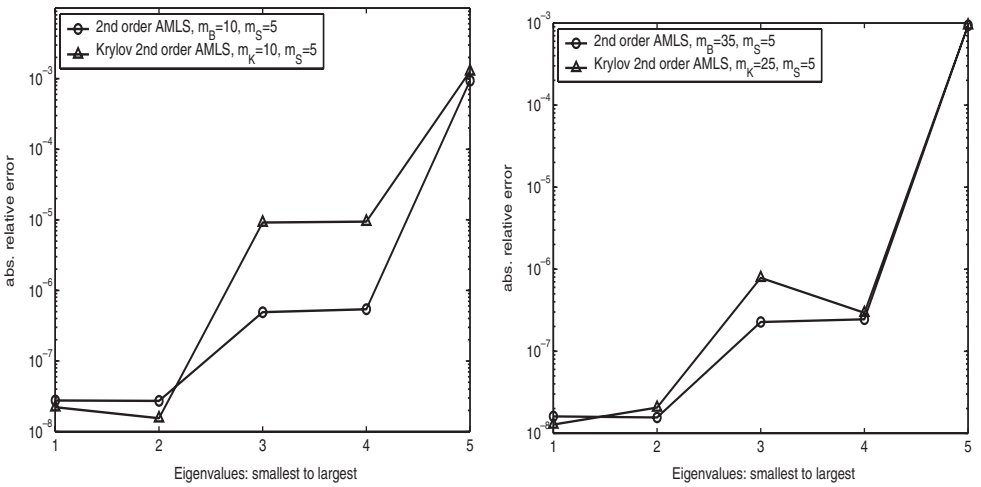


FIG. 8. BCSSTK11*: absolute relative errors for the* 5 *smallest eigenvalues of* (3) *computed by the second order* AMLS $(-\mathrm{o}-)$ *and second order* AMLS *with block Krylov subspace* $(-\triangle-)$.

Figure 4 (top-right) we observe that for larger eigenvalues of the matrix it is possible that Krylov AMLS may be inferior to standard AMLS. This is now explored further. We experimented with the same matrices (as they were reordered).

We first see how introducing eigenvectors of B, i.e., using the projection basis (38), can improve Krylov AMLS. Let U_S contain eigenvectors corresponding to the ten smallest eigenvalues of (16) and $V_{\mathcal{K}}$ be an orthonormal basis for $\mathcal{K}_{m_K}(B^{-1}, E[u_1^S, u_2^S])$ with $m_K = 5$. Thus, $V_{\mathcal{K}}$ has ten columns. Now let V_B contain eigenvectors that correspond to the ten smallest eigenvalues of B. Figure 9 (left column, top plot for the model problem and bottom plot for BCSSTK11) illustrates the absolute relative errors for the approximations of the ten smallest eigenvalues of the test matrices as they were computed by standard AMLS and Krylov AMLS. In both cases the dimension

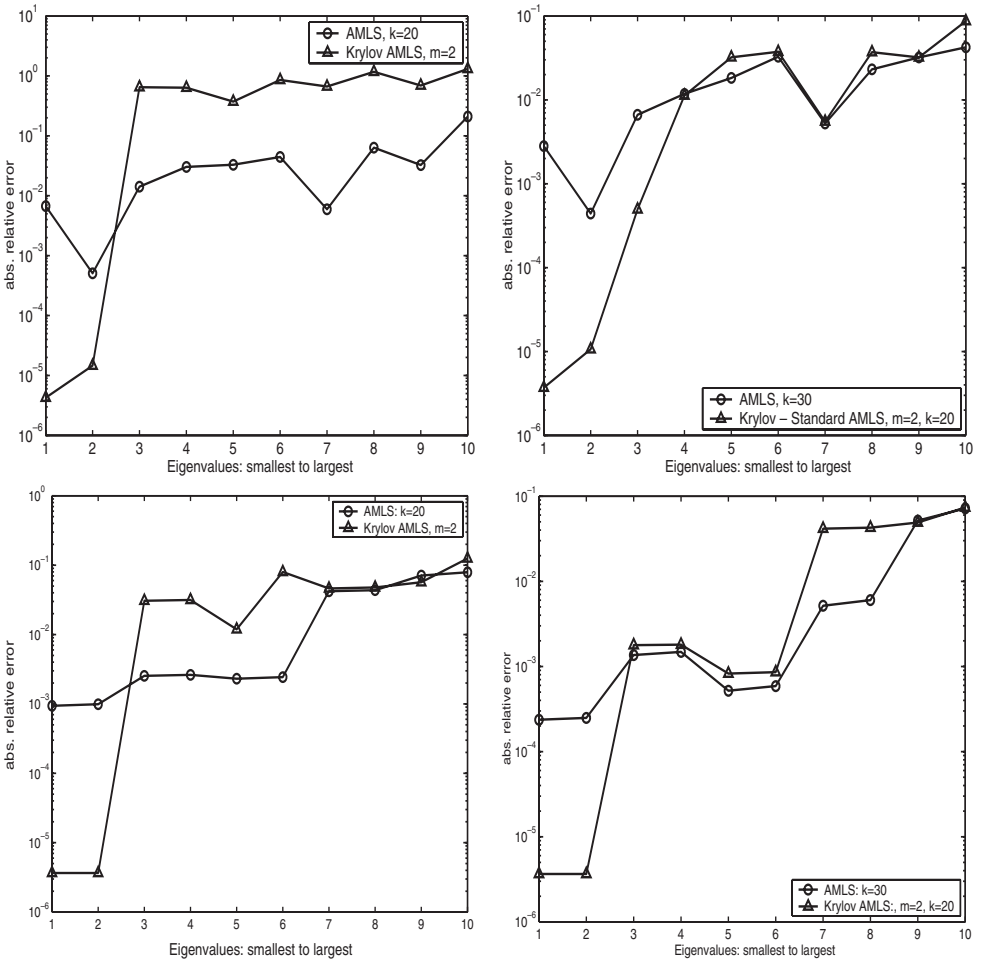


FIG. 9. *Left plots: Krylov* `AMLS` $(-\triangle-)$ *vs. standard* `AMLS` $(-\mathtt{o}-)$ *for the model problem* (3) (*up*) *and for matrix* `BCSSTK11` (*down*). *Right plots: Combination of Krylov* `AMLS` *and* `AMLS` $(-\triangle-)$ *vs. standard* `AMLS` $(-\mathtt{o}-)$ *for the model problem* (3) (*up*) *and for matrix* `BCSSTK11` (*down*).

of the projection space was equal to twenty. It is evident that for larger eigenvalues Krylov `AMLS` is inferior. This is due to the fact that the starting matrix $E[u_1^S, u_2^S]$ has a very poor representation of eigenvectors associated with the larger eigenvalues. We then added to Krylov `AMLS` ten eigenvectors of B that correspond to its smallest eigenvalues. For comparison, in standard `AMLS` we use twenty eigenvectors of B so that the dimension of the projection spaces would be equal to thirty in both cases. The right column plots of Figure 9 illustrate the results. The new combination is now capable of obtaining errors similar to those of standard `AMLS` for the larger eigenvalues while achieving much better results for the smallest eigenvalues.

Finally, Figure 10 illustrates the comparison of standard `AMLS` with the Krylov-second order `AMLS` hybrid (40). We have used ten eigenvectors in U_S, computed by the second order approximation, and a block Krylov basis $\mathcal{K}_{m_K}(B^{-1}, E[u_1^S, u_2^S])$ with starting vectors corresponding to the two smallest singular values and $m_K = 5$. Furthermore, we added the $m_B = 10$ smallest eigenvalues of B. Therefore, the

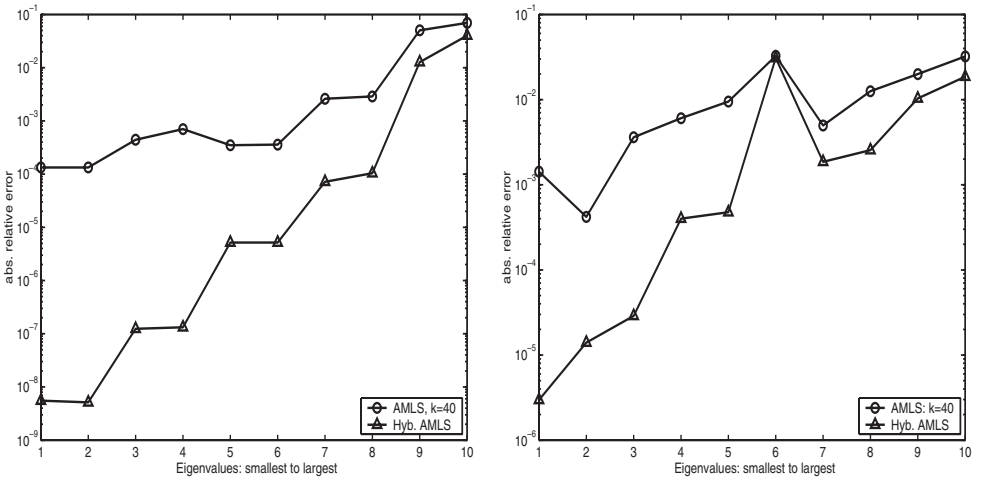


FIG. 10. `AMLS` *compared with the Krylov-second order* `AMLS` *hybrid* (40) *for* `BCSSTK11` (*left*) *and for the model problem* (*right*).

dimension of the projection basis (40) was equal to forty. For comparison, we employed standard `AMLS` with $m_S = 10$ smallest eigenvectors of (8) and $m_B = 30$ of the smallest eigenvalues of B. Clearly, the proposed hybrid outperforms the original method by combining all the improvements investigated in the previous sections.

The practical application of the above hybrids will depend on the particular situation. A large block of initial vectors in Krylov `AMLS` will give better accuracy, but this comes with increased computational costs. However, adding a few eigenvectors of B may help keep this added cost moderate. We believe that a careful selection of the dimensions of the various subspaces can lead to a successful method with controlled computational cost.

9. Conclusion. In recent years, `AMLS` has been successfully used in solving extremely large eigenvalue problems in structural dynamics which were intractable using the best alternative techniques such as shift-and-invert Lanczos. Therefore, the prospect of using `AMLS` in other application areas, such as computational materials science and nanotechnology appears particularly appealing. However, the levels of accuracy required in the latter applications are significantly higher than those required in the original application.

Thus, our primary goal in this paper was to investigate the approximation mechanism behind `AMLS` and in order to do so we have chosen to use a purely algebraic framework. `AMLS` can be viewed as a method which exploits a first order approximation to a nonlinear eigenvalue problem to extract a good subspace for a Rayleigh–Ritz projection process. This technique leads to approximations from a single Schur complement derived from a domain decomposition of the physical problem. Exploiting this observation, we have shown several possible enhancements to the original scheme. It is hoped that further improvements will be possible by exploiting similar approaches. For example, it would be interesting to explore the feasibility of an iterative scheme based on `AMLS`. Currently, `AMLS` is a one-shot algorithm in the sense that certain approximate eigenvectors are built from the last level up to the highest level and no further refinements are made. The current framework does not permit one to iteratively refine these approximations. This is an important shortcoming of the method

since eigenvalues obtained in this manner may not yield a satisfactory accuracy for certain applications.

A forthcoming paper will explore remedies which are based on new ways of exploiting the `AMLS` approximation mechanism [4]. In particular, `AMLS` can be viewed as an elaborate form of a shift-and-invert technique which uses only the zero shift. As was observed throughout our experiments, this leads to a diminishing accuracy for the larger eigenvalues. Improved approximations for the larger eigenvalues can be obtained from using other shifts, and this leads to developing strategies based on employing multiple shifts [4].

Acknowledgments. The authors would like to thank the referees for helpful comments and suggestions. We would also like to thank C. Yang, H. Voss, E. Gallopoulos, A. Stathopoulos, and P. Arbenz for insightful discussions concerning the practical application of the methods suggested in this paper.

While this paper was under review, the authors in [24] and [8] presented algebraic substructuring frameworks that, although related to the work presented here, are based on different settings.

REFERENCES

[1] A. Abramov, *On the separation of the principal part of some algebraic problems*, Zh. Vych. Mat. Fiz., 2 (1962), pp. 141–145.

[2] A. Abramov, *Remarks on finding the eigenvalues and eigenvectors of matrices which arise in the application of Ritz's method or in the difference method*, Zh. Vych. Mat. Fiz., 7 (1962), pp. 644–647.

[3] P. Arbenz and R. Lehoucq, *A Comparison of Algorithms for Modal Analysis in the Absence of a Sparse Direct Method*, Technical report 394, Institute of Scientific Computing, ETH Zurich, March 2003. Also available as Sandia National Laboratories, Technical report SAND2003-1028J.

[4] C. Bekas and Y. Saad, *Accurate computation of eigenvalues via multiple-shift AMLS*, in preparation.

[5] J. K. Bennighof and R. B. Lehoucq, *An automated multilevel substructuring method for eigenspace computation in linear elastodynamics*, SIAM. J. Sci. Comput., 25 (2004), pp. 2084–2106.

[6] R. F. Boisvert, R. Pozo, K. Remington, R. Barrett, and J. Dongarra, *The matrix market: A web repository for test matrix data*, in The Quality of Numerical Software, Assessment and Enhancement, R. Boisvert, ed., Chapman & Hall, London, 1997, pp. 125–137.

[7] V. S. Chichov, *A method for partitioning a high order matrix into blocks in order to compute its eigenvalues*, Zh. Vych. Mat., 1 (1961), pp. 169–173.

[8] K. Elssel and H. Voss, *An a priori bound for automated multi-level substructuring*, submitted, 2004.

[9] J. A. George and J. W-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[10] R. R. Craig, Jr. and M. C. C. Bampton, *Coupling of substructures for dynamic analysis*, AIAA Journal, 6 (1968), pp. 1313–1319.

[11] R. Grimes, J. G. Lewis, and H. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[12] W. C. Hurty, *Vibrations of structural systems by component-mode synthesis*, Journal of the Engineering Mechanics Division, ASCE, 86 (1960), pp. 51–69.

[13] M. Kaplan, *Implementation of Automated Multilevel Substructuring for Frequency Response Analysis of Structures*, Ph.D. thesis, The University of Texas at Austin, Austin, TX, 2001.

[14] T. Kato, *Perturbation Theory for Linear Operators*, 2nd ed., Springer-Verlag, Berlin, 1980.

[15] L. Komzsik and T. Rose, *Substructuring in MSC.NASTRAN for large scale parallel applications*, Computing Systems in Engineering, 2 (1991), pp. 167–173.

[16] A. Kropp and D. Heiserer, *Efficient Broadband Vibro-acoustic Analysis of Passenger Car Bodies using FE-based Component Mode Synthesis Approach*, in Fifth World Congress on Computational Mechanics (WCCM V), H. A. Mang, G. H. Rammerstorfer, and J. Eber-

hardsteiner, eds., Vienna University of Technology, Vienna, Austria, 2002.
[17] R. Lehoucq, D. C. Sorensen, and C. Yang, *Arpack User's Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998; also available online from http://www.caam.rice.edu/software/ARPACK.
[18] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.
[19] Y. Saad, *Shifts of origin for the LR and QR algorithms*, C.R.A.S. Proceedings of the French Academy of Sciences, Ser. A278, 1973, pp. 93–96. In French.
[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd edition, SIAM, Philadelphia, 2003.
[21] B. Smith, P. Bjørstad, and W. Gropp, *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, Cambridge, 1996.
[22] F. Tisseur and K. Meerbergen, *The quadratic eigenvalue problem*, SIAM Rev., 43 (2001), pp. 235–286.
[23] C. Yang, Private communication, October 2004.
[24] C. Yang, W. G. Gao, Z. Bai, X. Li, L. Lee, P. Husbands and E. G. Ng, *An Algebraic Sub-structuring Algorithm for Large-scale Eigenvalue Calculation*, LBNL Technical report, No. 55050, Lawrence Berkeley National Laboratory, Berkeley, CA, 2004.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 482–495
© 2005 Society for Industrial and Applied Mathematics

PSEUDOSPECTRAL DIFFERENCING METHODS FOR CHARACTERISTIC ROOTS OF DELAY DIFFERENTIAL EQUATIONS*

D. BREDA†, S. MASET‡, AND R. VERMIGLIO†

Abstract. In [D. BREDA, S. MASET, and R. VERMIGLIO, *IMA J. Numer. Anal.,* 24 (2004), pp. 1–19.] and [D. BREDA, *The Infinitesimal Generator Approach for the Computation of Characteristic Roots for Delay Differential Equations Using BDF Methods*, Research report UDMI RR17/2002, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Udine, Italy, 2002.] the authors proposed to compute the characteristic roots of delay differential equations (DDEs) with multiple discrete and distributed delays by approximating the derivative in the infinitesimal generator of the solution operator semigroup by Runge–Kutta (RK) and linear multistep (LMS) methods, respectively. In this work the same approach is proposed in a new version based on pseudospectral differencing techniques. We prove the "spectral accuracy" convergence behavior typical of pseudospectral schemes, as also illustrated by some numerical experiments.

Key words. pseudospectral methods, delay differential equations, characteristic roots

AMS subject classifications. 65J10, 65Q05, 34K06, 34K20

DOI. 10.1137/030601600

1. Introduction.

The interest for delay differential equations (DDEs) is still growing in many application fields, since the introduction of the delay in the model enriches its dynamics and allows a closer and more accurate description of the real-life phenomena. In this paper we consider a system of m-dimensional linear (or linearized) DDEs with multiple discrete and distributed delays

$$y'(t) = L_0 y(t) + \sum_{l=1}^{k} L_l y(t-\tau_l) + \int_{-\tau}^{0} M(\theta) y(t+\theta) d\theta, \quad t \geq 0, \tag{1.1}$$

where $L_0, L_1, \ldots, L_k \in \mathbb{C}^{m\times m}$, $0 = \tau_0 < \tau_1 < \cdots < \tau_k = \tau$, and $M : [-\tau, 0] \to \mathbb{C}^{m\times m}$ is a piecewise smooth function. Indeed in the implementation of the method we write the distributed term as

$$\int_{-\tau}^{0} M(\theta) y(t+\theta) d\theta = \sum_{l=1}^{k} \int_{-\tau_l}^{-\tau_{l-1}} M_l(\theta) y(t+\theta) d\theta, \quad t \geq 0, \tag{1.2}$$

where every function M_l, $l = 1, \ldots, k$, is smooth. Delay systems such as (1.1) are particularly important in control theory, where the stability effects of delays are a crucial problem (see the recent survey [Ric03]). In the exhaustive monograph [Nic00] many graphical and analytical tests to this aim are reported, but they work only for particular classes of system (1.1).

*Received by the editors November 3, 2003; accepted for publication (in revised form) December 6, 2004; published electronically September 23, 2005.
http://www.siam.org/journals/sisc/27-2/60160.html
†Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Via delle Scienze 208, I-33100 Udine, Italy (dbreda@dimi.uniud.it, vermiglio@dimi.uniud.it).
‡Dipartimento di Matematica e Informatica, Università degli Studi di Trieste, Via Valerio 12, I-34127 Trieste, Italy (maset@univ.trieste.it).

This motivates our interest in the numerical computation of the characteristic roots of (1.1), i.e., the (infinitely many) roots of

$$\det(\Delta(\lambda)) = 0, \tag{1.3}$$

where

$$\Delta(\lambda) = \lambda I - \sum_{l=0}^{k} L_l e^{-\lambda\tau_l} - \int_{-\tau}^{0} M(\theta)e^{\lambda\theta}d\theta, \quad \lambda \in \mathbb{C}, \tag{1.4}$$

since it is well known that the zero solution of (1.1) is asymptotically stable if and only if these roots have a strictly negative real part.

Recently, numerical approaches for characteristic roots computation have been proposed, which are based on the discretization of either the solution operator associated with (1.1) or the infinitesimal generator of the solution operator semigroup.

We briefly recall that the solution operator $T(t)$, $t \geq 0$, associated with (1.1) is defined by

$$T(t)\varphi = y_t, \quad \varphi \in X,$$

where $X = C([-\tau, 0], \mathbb{C}^m)$ endowed with the maximum norm, y_t is the function

$$y_t(\theta) = y(t + \theta), \quad \theta \in [-\tau, 0],$$

and y is the solution of (1.1) with initial data $\varphi \in X$. The family $\{T(t)\}_{t\geq 0}$ is a C_0-semigroup with infinitesimal generator $\mathcal{A} : D(\mathcal{A}) \subseteq X \to X$ given by

$$\mathcal{A}\varphi = \varphi', \quad \varphi \in D(\mathcal{A}), \tag{1.5}$$

with domain

$$D(\mathcal{A}) = \left\{ \varphi \in X \mid \varphi' \in X \text{ and } \varphi'(0) = \sum_{l=0}^{k} L_l\varphi(-\tau_l) + \int_{-\tau}^{0} M(\theta)\varphi(\theta)d\theta \right\}. \tag{1.6}$$

So (1.1) can be restated as the following abstract Cauchy problem ([DGVLW95]):

$$\begin{cases} \dfrac{dy_t}{dt} = \mathcal{A}y_t, & t > 0, \\ y_0 = \varphi. \end{cases}$$

The following two important results ([DGVLW95], [HVL93]):

1. $\det(\Delta(\lambda)) = 0 \Leftrightarrow \lambda = \frac{1}{t}\ln\mu$, $\mu \in \sigma(T(t)) \setminus \{0\}$;
2. $\det(\Delta(\lambda)) = 0 \Leftrightarrow \lambda \in \sigma(\mathcal{A})$;

where $\sigma(\cdot)$ denotes the spectrum, suggest the idea to turn the characteristic roots approximation problem into a corresponding eigenvalue problem for suitable matrix discretization of either $T(t)$ (i.e., [illegible] approach) or $\mathcal{A}$ (i.e., [illegible] approach).

Engelborghs and Roose propose in [ER02] and [ER99] the solution operator approach via LMS time integration for system (1.1) without the distributed delay term. Their method computes approximations to the roots from a large, standard, and sparse eigenvalue problem and is implemented in the MATLAB package DDE-BIFTOOL for DDEs bifurcation analysis ([ELR02] and [ELS01]). The distributed

delay case is considered in [LER03] by using LMS methods and in [Bre03] by using RK methods.

The complete development of the infinitesimal generator approach first appears in [BMV03] and [Bre02] where a matrix approximation to $\mathcal{A}$ is obtained by discretizing the derivative in (1.5) by the RK or LMS method, respectively, ending up with a large and sparse eigenvalue problem as in [ER99].

This work continues the investigation along this line and proposes the infinitesimal generator approach via pseudospectral differencing methods (see [Tre00] and the bibliography therein). This technique involves the exact differentiation of interpolants at selected sets of points. The resulting differentiation matrix is nonsparse, but we can take advantage of the well known "spectral accuracy" to obtain a very accurate approximation with small matrix dimension. This behavior represents in fact, for sufficiently small tolerance, the outstanding advantage of this method compared to the previously cited discretization schemes.

The paper is structured as follows. Section 2 describes the pseudospectral approach. Section 3 is about convergence analysis of the method. In section 4 we discuss implementation issues. In section 5 we perform numerical experiments on test delay systems of type (1.1) to illustrate the properties of the new method and to critically compare it with LMS and RK discretization techniques.

2. The pseudospectral approach. To simplify the notation, we rewrite the system of DDEs (1.1) as follows:

$$y'(t) = f(y_t),$$

where

$$f(\varphi) = \sum_{l=0}^{k} L_l \varphi(-\tau_l) + \int_{-\tau}^{0} M(\theta)\varphi(\theta)d\theta, \quad \varphi \in X. \tag{2.1}$$

Given a positive integer N, let us consider the mesh Ω_N of $N+1$ distinct points in $[-\tau, 0]$

$$\Omega_N = \{\theta_{N,i},\ i = 0, 1, \dots, N\}$$

with $0 = \theta_{N,0} > \theta_{N,1} > \cdots > \theta_{N,N} \geq -\tau$. We replace the continuous space X by the space X_N of the discrete functions defined on the mesh Ω_N, i.e., any $\varphi \in X$ is discretized into the block-vector $x \in X_N$ of components $x_i = \varphi(\theta_{N,i}) \in \mathbb{C}^m$, $i = 0, 1, \dots, N$. Let $\mathcal{L}_N x$, $x \in X_N$, be the unique $\mathbb{C}^m$-valued interpolating polynomial of degree $\leq N$ with $(\mathcal{L}_N x)(\theta_{N,i}) = x_i$, $i = 0, 1, \dots, N$. Thus we approximate the infinitesimal generator $\mathcal{A}$ by the matrix $\mathcal{A}_N : X_N \to X_N$, called the spectral differentiation matrix, defined as follows:

$$\begin{cases} (\mathcal{A}_N x)_0 = f_N(\mathcal{L}_N x), \\ (\mathcal{A}_N x)_i = (\mathcal{L}_N x)'(\theta_{N,i}), \quad i = 1, 2, \dots, N, \end{cases} \tag{2.2}$$

where f_N is an approximation of f in which the distributed delay integral term in (2.1) is substituted for by a suitable interpolatory quadrature rule or f_N is equal to f in the case of the simple matrix function M for which the integral can be exactly computed.

By using the Lagrange representation of $\mathcal{L}_N x$ we obtain

$$\mathcal{A}_N = \begin{pmatrix} a_0 & a_1 & \cdots & a_N \\ d_{10} & d_{11} & \cdots & d_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ d_{N0} & d_{N1} & \cdots & d_{NN} \end{pmatrix} \in \mathbb{C}^{m(N+1)\times m(N+1)}$$

with

$$\begin{cases} a_j = f_N(l_{N,j}(\cdot)I_m), & j = 0, 1, \dots, N, \\ d_{ij} = l'_{N,j}(\theta_{N,i})I_m, & i = 1, \dots, N,\ j = 0, 1, \dots, N, \end{cases} \tag{2.3}$$

where $l_{N,j}$, $j = 0, 1, \dots, N$, are the Lagrange coefficients relevant to the nodes in Ω_N and, by definition, $f_N(l_{N,j}(\cdot)I_m) = (f_N(l_{N,j}(\cdot)e^{(i)}))_{i=1}^m$ with $e^{(i)}$'s the canonical vectors in $\mathbb{R}^m$. Explicit expressions of the d_{ij}'s for particular choices of Ω_N (e.g., Chebyshev extremal points) can be found in [Tre00, section 6].

So, finally, the original problem of the numerical computation of the characteristic roots of (1.1) can be turned into the eigenvalue problem for the matrix $\mathcal{A}_N$, i.e., eigenvalues of $\mathcal{A}_N$ directly approximate the characteristic roots. How accurate are these approximations? In the next section we develop the convergence analysis.

3. Convergence analysis. The convergence analysis of the approximated roots to the exact ones is based on a comparison between the continuous (i.e., exact) characteristic equation (1.3) and its discrete (i.e., approximated) counterpart. Let us thus characterize the continuous and the discrete eigenvalue problems for $\mathcal{A}$ and $\mathcal{A}_N$, respectively. Let $\lambda \in \mathbb{C}$ and $\varphi \in D(\mathcal{A})$ be such that

$$\mathcal{A}\varphi = \lambda\varphi;$$

that means

$$\begin{cases} \varphi'(\theta) = \lambda\varphi(\theta), & \theta \in [-\tau, 0], \\ \varphi'(0) = f(\varphi). \end{cases}$$

Hence λ is an eigenvalue of $\mathcal{A}$ if and only if there exists $u \in \mathbb{C}^m$, $u \neq 0$, such that

$$\lambda u = f(e^{\lambda \cdot} u).$$

By introducing the linear operator $\hat{\mathcal{A}}(\lambda) : \mathbb{C}^m \to \mathbb{C}^m$ given by

$$\hat{\mathcal{A}}(\lambda)u = f(e^{\lambda \cdot} u), \quad u \in \mathbb{C}^m \tag{3.1}$$

we see that λ is an eigenvalue of $\mathcal{A}$ if and only if

$$\det(\lambda I_m - \hat{\mathcal{A}}(\lambda)) = 0 \tag{3.2}$$

which is the continuous characteristic equation (1.3).

In the same fashion, let $x \in X_N$ be such that

$$\mathcal{A}_N x = \lambda x;$$

thus by (2.2) we get

$$f_N(\mathcal{L}_N x) = \lambda x_0$$

and

$$(\mathcal{L}_N x)'(\theta_{N,i}) = \lambda x_i, \quad i = 1, 2, \dots, N. \tag{3.3}$$

By defining $p_N(\cdot; \lambda, u)$, $u \in \mathbb{C}^m$, as the collocation polynomial relevant to the nodes $\theta_{N,1}, \theta_{N,2}, \dots, \theta_{N,N}$ for the initial value problem

$$\begin{cases} y'(\theta) = \lambda y(\theta), & \theta \in [-\tau, 0], \\ y(0) = u, \end{cases} \tag{3.4}$$

whose solution is the exponential $y(\cdot;\lambda,u) = e^{\lambda\cdot}u$, one can easily derive that (3.3) is equivalent to $(\mathcal{L}_N x)(\cdot) = p_N(\cdot;\lambda,x_0)$. Consequently, λ is an eigenvalue of $\mathcal{A}_N$ if and only if there exists $u \in \mathbb{C}^m$, $u \neq 0$, such that

$$\lambda u = f_N(p_N(\cdot;\lambda,u)).$$

Finally, by introducing the linear operator $\hat{\mathcal{A}}_N(\lambda) : \mathbb{C}^m \to \mathbb{C}^m$ given by

$$\hat{\mathcal{A}}_N(\lambda)u = f_N(p_N(\cdot;\lambda,u)), \quad u \in \mathbb{C}^m, \tag{3.5}$$

we get that λ is eigenvalue of $\mathcal{A}_N$ if and only if

$$\det(\lambda I_m - \hat{\mathcal{A}}_N(\lambda)) = 0 \tag{3.6}$$

which is the discrete characteristic equation.

3.1. We assume to use the Chebyshev extremal nodes on $[-\tau,0]$, i.e.,

$$\theta_{N,i} = \frac{\tau}{2}\left(\cos\left(i\frac{\pi}{N}\right) - 1\right), \quad i = 0,\dots,N. \tag{3.7}$$

Let us denote by $B(\lambda,\rho)$ the closed ball in $\mathbb{C}$ of center λ and radius ρ. In the next lemma we give an upper bound of the error of the collocation polynomial for (3.4).

LEMMA 3.2. $\lambda^* \in \mathbb{C}$ $\rho_0 > 0$ 3.1 $N_0 \in \mathbb{N}$ $N \geq N_0$ $\lambda \in B(\lambda^*,\rho_0)$ $u \in \mathbb{C}^m$ $p_N(\cdot;\lambda,u)$ (3.4) (3.7)

$$\max_{\theta\in[-\tau,0]} |p_N(\theta;\lambda,u) - y(\theta;\lambda,u)| \leq \frac{C_0}{\sqrt{N}}\left(\frac{C_1}{N}\right)^N |u|,$$

$C_0 = C_0(\lambda^*,\rho_0)$ $C_1 = C_1(\lambda^*,\rho_0) = \tau(|\lambda^*|+\rho_0)e$ N

By introducing the (linear) integral Volterra operator $K_\lambda : X \to X$ as

$$(K_\lambda\varphi)(\theta) = \lambda\int_0^\theta \varphi(s)ds, \quad \varphi \in X,\ \lambda \in \mathbb{C},\ \theta \in [-\tau,0],$$

and the (linear and bounded) Lagrange interpolation operator $L_{N-1} : X \to X$ relevant to the nodes (3.7), we can rewrite (3.4) and its collocation equations, respectively, as the equations on the space X

$$y = u + K_\lambda y \tag{3.8}$$

and

$$p_N = u + K_\lambda L_{N-1} p_N, \tag{3.9}$$

where $y = y(\cdot;\lambda,u)$, $p_N = p_N(\cdot;\lambda,u)$, and u denotes the function of X of constant value u. By subtracting (3.8) from (3.9) and setting $e_N = p_N - y$ and $r_N = (L_{N-1} - I)y$, we get the following equation for e_N:

$$e_N = K_\lambda L_{N-1} e_N + K_\lambda r_N. \tag{3.10}$$

It can be directly shown that the solutions of (3.10) are the functions $K_\lambda \hat{e}_N$, where $\hat{e}_N$ is a solution of

$$\hat{e}_N = L_{N-1} K_\lambda \hat{e}_N + r_N.$$

Since $I - K_\lambda$ is invertible and, by the known estimate for the Lagrange interpolation on Chebyshev nodes, $\|L_{N-1}K_\lambda - K_\lambda\| \to 0$ as $N \to \infty$, it follows that, for sufficiently large N (and independently of λ), $I - L_{N-1}K_\lambda$ is invertible and

$$\|(I - L_{N-1}K_\lambda)^{-1}\| \le 2\|(I - K_\lambda)^{-1}\|.$$

Hence $e_N = K_\lambda (I - L_{N-1}K_\lambda)^{-1} r_N$ is the unique solution of (3.10) and as a consequence

$$\|e_N\| \le 2\|K_\lambda\|\|(I - K_\lambda)^{-1}\|\|r_N\|.$$

Since r_N is the error of the Lagrange interpolation of the exponential $y(\cdot;\lambda,u) = e^{\lambda \cdot} u$ relevant to the nodes (3.7), we conclude that

$$\|r_N\| \le \max\{1, e^{-\Re(\lambda)\tau}\} \frac{(\tau|\lambda|)^N}{N!} |u|.$$

The final assertion follows by application of the Stirling formula

$$N! \ge \sqrt{2\pi N}(N/e)^N. \qquad \square$$

Now let us set the following notation relevant to the continuous and discrete characteristic equations (3.2) and (3.6), respectively:

$$d(\lambda) = \det(\lambda I_m - \hat{\mathcal{A}}(\lambda)), \tag{3.11}$$

$$d_N(\lambda) = \det(\lambda I_m - \hat{\mathcal{A}}_N(\lambda)) \tag{3.12}$$

with $\hat{\mathcal{A}}(\lambda)$ and $\hat{\mathcal{A}}_N(\lambda)$ defined in (3.1) and (3.5), respectively. The following assumption and lemmas are the next necessary steps to state the final theorem of convergence.

3.3. We assume that the function f_N in (2.2) satisfies

$$\sup_{N\in\mathbb{N}} \|f_N\| < +\infty.$$

Lemma 3.4. $\lambda^* \in \mathbb{C}$ $\rho_0 > 0$ 3.1 3.3 $N_0 \in \mathbb{N}$ $N \ge N_0$ $\lambda \in B(\lambda^*, \rho_0)$

$$|d(\lambda) - d_N(\lambda)| \le C_2 \left(\varepsilon_N + \frac{1}{\sqrt{N}} \left(\frac{C_1}{N} \right)^N \right),$$

$$\varepsilon_N = \sup_{\lambda \in B(\lambda^*,\rho_0)} \frac{|f(y(\cdot;\lambda,u)) - f_N(y(\cdot;\lambda,u))|}{|u|},$$

$C_1 = C_1(\lambda^*, \rho_0)$ 3.2 $C_2 = C_2(\lambda^*, \rho_0)$ N

. By Lemma 3.2,

$$\begin{aligned}|\hat{\mathcal{A}}(\lambda)u - \hat{\mathcal{A}}_N(\lambda)u| &\le |f(y(\cdot;\lambda,u)) - f_N(y(\cdot;\lambda,u))| \\ &\quad + |f_N(y(\cdot;\lambda,u)) - f_N(p_N(\cdot;\lambda,u))| \\ &\le \left(\varepsilon_N + \sup_{N\in\mathbb{N}} \|f_N\| \frac{C_0}{\sqrt{N}} \left(\frac{C_1}{N}\right)^N\right) |u|\end{aligned}$$

for $N \ge N_0$ and $\lambda \in B(\lambda^*, \rho_0)$.

Now, since the derivative of $\det(A)$ with respect to $A \in \mathbb{C}^{m\times m}$ is continuous, the assertion easily follows. □

LEMMA 3.5. $d(\lambda)$ $\lambda^* \in \mathbb{C}$ ν $\rho_1 = \rho_1(\lambda^*) > 0$

$$|d(\lambda)| > C_3|\lambda - \lambda^*|^\nu$$

$\lambda \in B(\lambda^*, \rho_1) \setminus \{\lambda^*\}$, $C_3 = C_3(\lambda^*)$

. Expanding $d(\lambda)$ around λ^* and taking into account the multiplicity ν we obtain

$$d(\lambda) = \frac{d^{(\nu)}(\lambda^*)}{\nu!}(\lambda - \lambda^*)^\nu + O(|\lambda - \lambda^*|^{\nu+1})$$

with $d^{(\nu)}(\lambda^*) \neq 0$. The result follows. □

We are now able to state the convergence result.

THEOREM 3.6 (convergence). $\lambda^* \in \mathbb{C}$ $d(\lambda)$ ν ρ_1 3.5

$$\rho_N = \left(\frac{C_2}{C_3}\right)^{1/\nu} \left(\varepsilon_N + \frac{1}{\sqrt{N}} \left(\frac{C_1}{N}\right)^N\right)^{1/\nu} \tag{3.13}$$

C_1 C_2 3.4 $\rho_0 = \rho_1$ C_3 3.5 3.1 3.3 $\rho_N \le \rho_1$, $d_N(\lambda)$ ν λ_i $i = 1, \dots, \nu$

$$\max_{1\le i\le \nu} |\lambda^* - \lambda_i| \le \rho_N. \tag{3.14}$$

. Since $\rho_N \le \rho_1$, Lemmas 3.4 and 3.5 yield

$$|d(\lambda)| > |d_N(\lambda) - d(\lambda)|$$

for $|\lambda - \overline{\lambda}| = \rho_N$. Hence, by Rouché's theorem [Con78, section 5, Theorem 3.8], d and d_N have exactly the same number of zeros counted with their multiplicities in $B(\lambda^*, \rho_N)$. □

It is important to point out that the spectral accuracy of the collocation polynomial stated in Lemma 3.2 is preserved in the behavior of the final error (3.14) in Theorem 3.6, if the distributed term in (2.2) can be exactly computed, i.e., $\varepsilon_N = 0$. On the other hand, we also maintain the spectral accuracy in the error (3.14) if we make the further assumptions that

1. the function M in (1.1) is piecewise C^∞ and there exists $C > 0$ such that

$$\max_{\theta\in[-\tau,0]} \|M^{(q)}(\theta)\| \le C^q, \quad q = 0, 1, 2, \dots;$$

2. a piecewise interpolatory quadrature rule based on $N+1$ distinct nodes is used for the distributed term.

Finally, we observe that if the quadrature error is independent of N, i.e., $\varepsilon_N \leq$ TOL (e.g., machine precision), the error bound in Theorem 3.6 is still valid with ε_N replaced by TOL and thus the convergence follows spectral accuracy down to the tolerance TOL.

In the next proposition we also show the nonexistence of "ghost roots," i.e., if there exists a sequence of approximated characteristic roots that converges, then it converges to an exact characteristic root.

PROPOSITION 3.7. [illegible] $\{\Omega_{N^{(i)}}\}_{i\geq 1}$ [illegible] $[-\tau, 0]$ [illegible] $N^{(i)} \to \infty$ [illegible] $\varepsilon_{N^{(i)}} \to 0$ [illegible] $i \to \infty$ [illegible] $\lambda^{(i)} \to \lambda^*$ [illegible] $i \to \infty$ [illegible] $\lambda^{(i)}$ [illegible] $d_{N^{(i)}}$ [illegible] λ^* [illegible] d [illegible]. Since

$$|d(\lambda^*)| = |d_{N^{(i)}}(\lambda^{(i)}) - d(\lambda^*)| \leq |d_{N^{(i)}}(\lambda^{(i)}) - d(\lambda^{(i)})| + |d(\lambda^{(i)}) - d(\lambda^*)|,$$

the assertion follows by Lemma 3.4 and by continuity of d. ∎

4. Implementation issues. The application of the pseudospectral approach presented in this work allows two different alternatives. The first one uses a unique interpolating polynomial for the whole delay interval $[-\tau, 0]$. The second one can be interpreted as a piecewise pseudospectral approach since it makes use of a different N-degree interpolating polynomial for each delay interval $[-\tau_l, -\tau_{l-1}]$, $l = 1, \ldots, k$. The discretized infinitesimal generator results in the $m(1+kN) \times m(1+kN)$ matrix

$$\mathcal{A}_N = \begin{pmatrix} a_0^{(1)} & a_1^{(1)} & \cdots & a_N^{(1)} & \cdots & a_N^{(k-1)} & a_1^{(k)} & \cdots & a_N^{(k)} \\ d_{10}^{(1)} & d_{11}^{(1)} & \cdots & d_{1N}^{(1)} & & & & & \\ \vdots & \vdots & \ddots & \vdots & & & & & \\ d_{N0}^{(1)} & d_{N1}^{(1)} & \cdots & d_{NN}^{(1)} & & & & & \\ & & & & \ddots & & & & \\ & & & & & \ddots & & & \\ & & & & & d_{10}^{(k)} & d_{11}^{(k)} & \cdots & d_{1N}^{(k)} \\ & & & & & \vdots & \vdots & \ddots & \vdots \\ & & & & & d_{N0}^{(k)} & d_{N1}^{(k)} & \cdots & d_{NN}^{(k)} \end{pmatrix},$$

where, for $l = 1, \ldots, k$, $a_j^{(l)}$ with $j = 0, \ldots, N$, and $d_{ij}^{(l)}$ with $i = 1, \ldots, N$ and $j = 0, \ldots, N$, are the coefficients given by (2.3) and relevant to the interval $[-\tau_l, -\tau_{l-1}]$. Note that $a_0^{(l)} = a_N^{(l-1)}$ holds for $l = 2, \ldots, k$. Despite the increased approximant matrix dimension, this second solution avoids the evaluation of Lagrange coefficients at the discrete delay points when computing the first row since these are always mesh points. Moreover, if we are in the case of applying a quadrature rule to the distributed delay terms (1.2), using the interpolation points as quadrature nodes, the piecewise alternative permits avoidance of all Lagrange coefficients evaluation. This advantage clearly increases as the number of delays increases and, moreover, sparsity can be exploited when k is large.

The two alternatives in the pseudospectral approach have been implemented in two different [illegible] routines for the numerical computation of the characteristic

roots of systems of DDEs using Chebyshev extremal grids for both versions of the general case. The algorithms apply Clenshaw–Curtis quadrature (see [Tre00, section 12]) for the distributed delay terms, which is based on Chebyshev extremal nodes.

For stability analysis, one can be interested in computing either the first r-rightmost roots or the roots λ in the right half-plane $\Re(\lambda) \geq \alpha$ for a given real α. In the first case, since the roots with smaller modulus are approximated faster and, in general, they are not the rightmost ones, it is advisable to try with $N > r$ (so it is easier that the first r-rightmost roots are contained in the first r-roots ordered by modulus). In the second case, by using the following bound,

$$|\lambda| \leq \sum_{l=0}^{k} \|L_l\| e^{-\alpha\tau_l} + \tau B \max\{1, e^{-\alpha\tau}\} := K$$

for roots λ with $\Re(\lambda) \geq \alpha$, where B is a uniform upper bound of $M(\theta)$ in $[-\tau, 0]$, by (3.13) and (3.14) with $\varepsilon_N = 0$ one obtains for simple roots

$$|\lambda^* - \lambda| \leq \frac{C_2}{C_3} \left(\frac{1}{\sqrt{N}} \left(\frac{\tau(K+\rho_1)e}{N} \right)^N \right),$$

and by assuming $\frac{C_2}{C_3} \simeq 1$ and $\tau(K+\rho_1)e \simeq \tau K e$ one can roughly estimate the required N for a given tolerance.

For a more detailed discussion on the above and other implementation issues we refer the interested reader to [Bre02] and [BMV03].

5. Numerical results. We present here a sequence of results on some systems of linear DDEs (1.1) in order to show the features of the pseudospectral approach and to compare it with RK and LMS methods proposed in [Bre02] and [BMV03], respectively.

All the results concern the following systems and equations:

' . ', 5.1 (see [ER02]).

$$y'(t) = 2y(t) - ey(t-1). \tag{5.1}$$

' . ', 5.2 (see [ER02]).

$$\begin{cases} y_1'(t) = -0.5y_1(t) - \tanh\left(y_1(t-1.57)\right) + \tanh\left(y_2(t-0.2)\right), \\ y_2'(t) = -0.5y_2(t) + 2.34\tanh\left(y_1(t-0.2)\right) - \tanh\left(y_2(t-1.57)\right). \end{cases} \tag{5.2}$$

' . ', 5.3 (see [FSD00]).

$$y'(t) = L_0 y(t) + L_1 y(t-1) + \int_{-1}^{0} M(\theta) y(t+\theta) d\theta \tag{5.3}$$

with coefficients matrices

$$L_0 = \begin{pmatrix} -3 & 1 \\ -24.646 & -35.430 \end{pmatrix}, \qquad L_1 = \begin{pmatrix} 1 & 0 \\ 2.35553 & 2.00365 \end{pmatrix},$$

$$M(\theta) = \begin{pmatrix} 1 & -\theta \\ 1 & -\theta \end{pmatrix}, \quad \theta \in [-\tau, 0].$$

' . ,' 5.4.

$$y'(t) = L_0 y(t) + L_1 y(t-1) + \int_{-0.3}^{-0.1} M_1 y(t+\theta)d\theta + \int_{-1}^{-0.5} M_2 y(t+\theta)d\theta \tag{5.4}$$

with coefficients matrices

$$L_0 = \begin{pmatrix} -3 & 1 \\ -24.646 & -35.430 \end{pmatrix}, \qquad L_1 = \begin{pmatrix} 1 & 0 \\ 2.35553 & 2.00365 \end{pmatrix},$$
$$M_1 = \begin{pmatrix} 2 & 2.5 \\ 0 & -0.5 \end{pmatrix}, \qquad M_2 = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}.$$

' . ,' 5.5 (see [ER02]).

$$y'(t) = L_0 y(t) + L_1 y(t-\tau) \tag{5.5}$$

with tridiagonal coefficients $m \times m$ matrices

$$L_0 = \frac{1}{h^2}\left(\frac{1-\alpha}{\beta} + \alpha\right) \begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} + RI_m$$

and

$$L_1 = \frac{1}{h^2}\frac{1-\alpha}{\beta} \begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix},$$

where α, β, and R are real parameters and $h = \pi/(m+1)$ with $m \in \mathbb{N}$.

Equation (5.1) presents an exact rightmost double root $\overline{\lambda} = 1$ while system (5.2), linearized around the steady state solution $(y_1^*, y_2^*) = (0, 0)$, exhibits a rightmost root $\overline{\lambda} \simeq 0.3475$. Systems (5.3) and (5.4) present, respectively, a rightmost root $\overline{\lambda} \simeq -0.6873$ and $\overline{\lambda} \simeq 1.2462$. The latter is a modified version of the former: the distributed delay is split in two separate terms. Moreover, the matrices M_1 and M_2 are independent of θ. This modification is introduced in order to exalt the advantages of the piecewise alternative approach with respect to the standard one. The last test equation (5.5) presents a complex-conjugate rightmost root $\overline{\lambda} \simeq -0.06362 \pm 1.007\mathrm{i}$ and it arises from the discretization in space by a classical second order central difference scheme of a PDE with time delay in the diffusion term.

Figure 5.1 shows the trend of the absolute error on the rightmost root with respect to the number of interpolation points N for the single delay equation (5.1): convergence is confirmed as predicted in Theorem 3.6. Observe that the method works also in the case of a root with multiplicity $\nu > 1$, but in this case the maximum tolerance reached is equal to the machine precision under the νth root. The convergence behavior confirms the typical "spectral accuracy" of spectral methods.

The different approaches, single and piecewise interpolation, are compared in Figure 5.2 for the computation of the rightmost root of system (5.4). Performances are

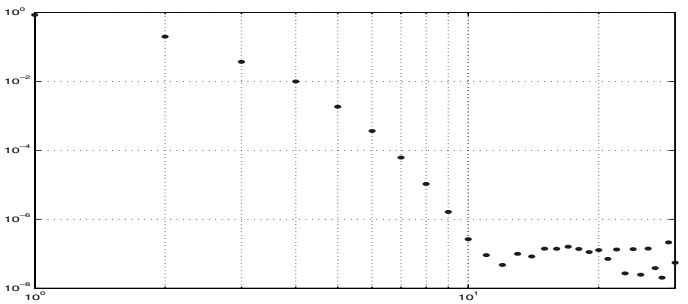


FIG. 5.1. *Rightmost root absolute error vs number of grid points N for system* (5.1).

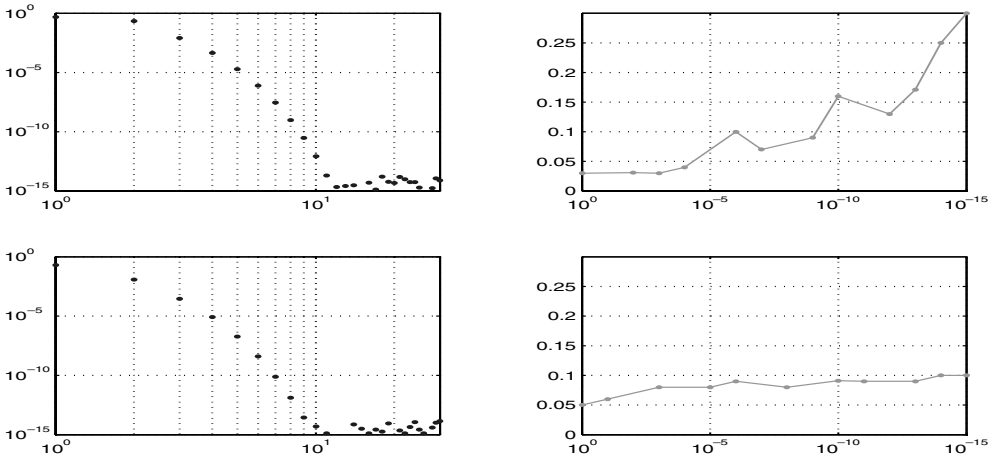


FIG. 5.2. *Rightmost root absolute error vs. number of grid points N (*1*st column) and computational time (seconds,* 2*nd column) vs. required tolerance TOL for system* (5.4) *with single (*1*st row) and piecewise (*2*nd row) pseudospectral approach.*

slightly different in favor of the piecewise method: this is a consequence of avoiding the computation of Lagrange coefficients as explained in section 4, which would be more evident in the case of systems with many delays. The approximant matrix dimension is four times larger for the piecewise approach: this disadvantage is recovered (in terms of computational time; see the 2nd column) by using sparse eigenvalue solvers (e.g., the function of).

The same analysis is carried out in Figure 5.3 for system (5.3) comparing the piecewise pseudospectral approach with LMS and RK methods of order 5 proposed in [Bre02] and [BMV03], respectively, both showing convergence of order 5 as expected. For what concerns the density of the delay interval discretization (i.e., the number N for the pseudospectral method and $N = \tau/h$, h being the steplength, for the other methods), one can see in the 1st column that the RK approach is more accurate than the pseudospectral one up to a tolerance of about 10^{-9}, an advantage that falls rapidly for smaller values due to the "spectral accuracy" of the pseudospectral technique. This behavior holds in general for "linearly" convergent approaches (LMS and RK) compared to "superlinearly" convergent ones (pseudospectral), the "crossing" tolerance value depending on the particular system of DDEs analyzed by the constants involved in the error (here, "linear" and "superlinear" convergence refer

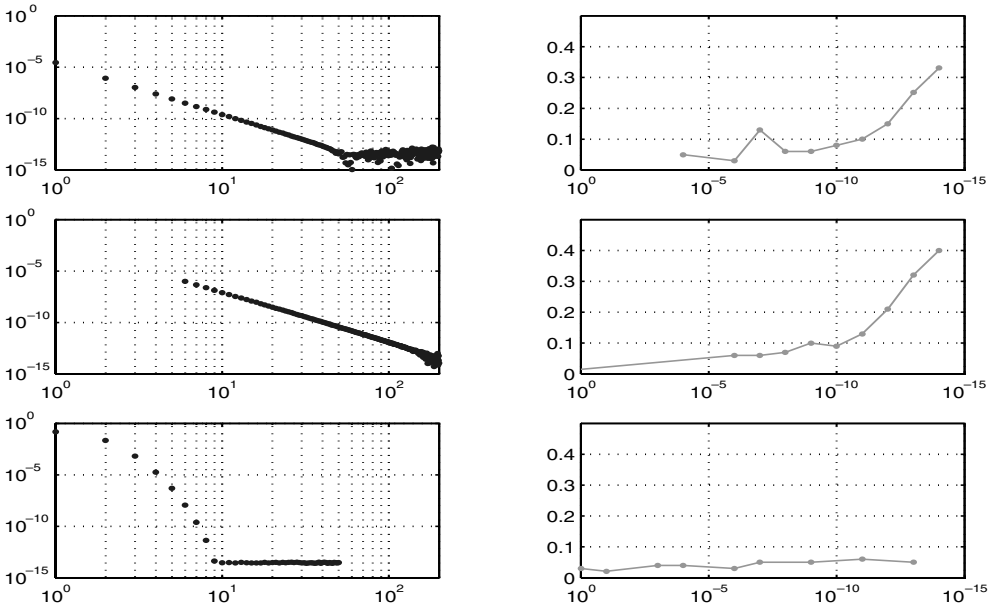


FIG. 5.3. *Rightmost root absolute error vs. number of grid points N (1st column) and computational time (seconds, 2nd column) vs. required tolerance TOL for system* (5.3) *with RK (1st row), LMS (2nd row), and pseudospectral (3rd row) approach.*

TABLE 5.1
First ten rightmost roots value and modulus for system (5.2).

characteristic root	modulus
0.34748172572643	0.34748172572643
−0.08116698520245	0.08116698520245
$-0.43412304132027 \pm \mathrm{i}1.62752128215701$	1.68442522507803
$-0.82061576063772 \pm \mathrm{i}5.11180446782392$	5.17725362947110
$-1.20975894908637 \pm \mathrm{i}4.82695048752634$	4.97624032015389
$-1.24622687378869 \pm \mathrm{i}8.90021501181034$	8.98704115253777

to a double-logarithmic plot context).

Moreover, we compare RK and pseudospectral approaches computing the first ten rightmost roots of system (5.2), listed in Table 5.1 and shown in Figure 5.4. The 1st column of Figure 5.5 shows the error trend for each root (simple-real or complex-conjugate) in ascending order with respect to their modulus. This last plays a fundamental role in the sense that approximated roots of smaller modulus converge sooner: for the pseudospectral approach this phenomenon clearly results from Theorem 3.6 and an analogous result can be deducted from the convergence analysis carried out in [BMV03]. The 2nd column evidences once more the superiority of the pseudospectral approach in terms of computational time with respect to the required tolerance for the maximum absolute error over all the computed roots.

Finally, to test the performance of our approach on time delay systems of higher order, we carry on the convergence analysis on the rightmost characteristic root of (5.5) with $m = 20$ and parameters $\alpha = 0.1$, $\beta = 2$, and $R = 0.08$. Results are given in Figure 5.6. Note that in general space discretization of PDEs with delay in time leads to DDE systems which already possess a quite large dimension m. This means

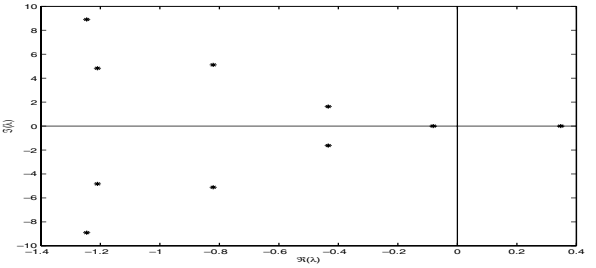


FIG. 5.4. *First ten rightmost roots of system* (5.2).

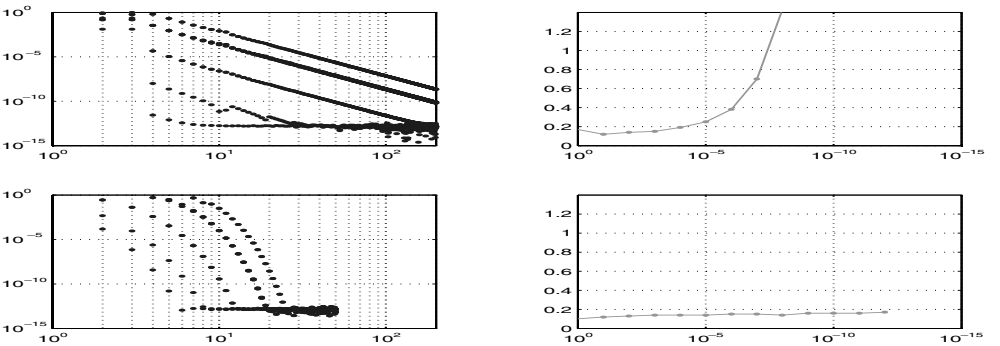


FIG. 5.5. *First ten rightmost roots absolute error vs. number of grid points N (*1*st column) and computational time (seconds,* 2*nd column) vs. required tolerance on the maximum absolute error TOL (seconds,* 2*nd column) for system* (5.2) *with RK (*1*st row) and pseudospectral (*2*nd row) approach.*

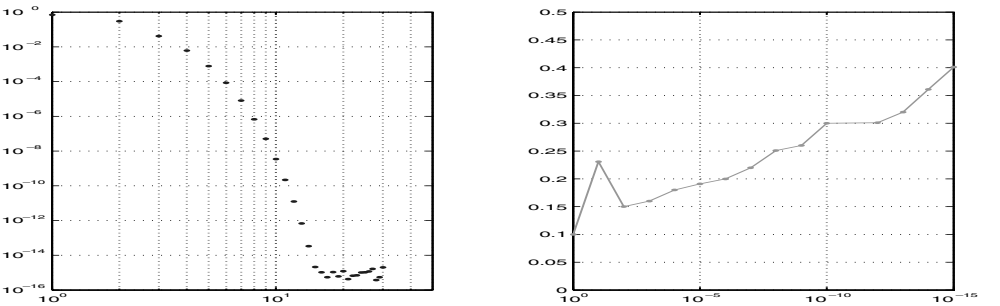


FIG. 5.6. *Rightmost root absolute error vs. number of grid points N (*1*st column) and computational time (seconds,* 2*nd column) vs. required tolerance TOL for system* (5.5) *with pseudospectral approach.*

that the choice of N for the approximation of the characteristic roots turns out to be crucial in saving computational time. Let us underline that in the second case we adopt a sparse eigenvalue solver to exploit the fact that the entries of $\mathcal{A}_N$ are (apart from the first block row) $m \times m$ diagonal blocks.

To complete this test we use $N = 100$ to allow the computation of $r = m(1+N) = 2020$ roots, which is the dimension of the discretized infinitesimal generator. The eigenvalues are shown in Figure 5.7 (left), while a zoom in Figure 5.7 (right) gives an idea of the particular conformation of the spectrum and of how clustered the roots are.

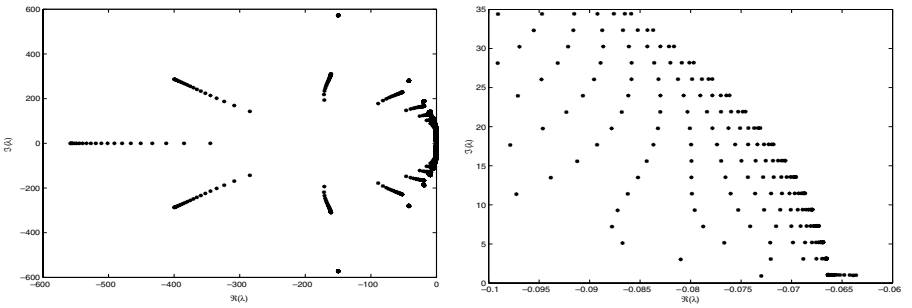


FIG. 5.7. *Characteristic roots of system* (5.5).

REFERENCES

[Bre03] D. BREDA, *Solution operator approximation for delay differential equation characteristic roots computation via Runge-Kutta methods*, special issue of Appl. Numer. Math. for the Third Conference on Volterra and Delay Equations dedicated to Professor Alan Feldstein's 70th Birthday, to appear.

[Bre02] D. BREDA, *The Infinitesimal Generator Approach for the Computation of Characteristic Roots for Delay Differential Equations Using BDF Methods*, Research report UDMI RR17/2002, Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Udine, Italy, 2002, submitted.

[BMV03] D. BREDA, S. MASET, AND R. VERMIGLIO, *Computing the characteristic roots for delay differential equations*, IMA J. Numer. Anal., 24 (2004), pp. 1–19.

[Con78] J.B. CONWAY, *Functions of One Complex Variable*, 2nd ed., Graduate Texts in Mathematics 11, Springer-Verlag, New York, 1978.

[DGVLW95] O. DIEKMANN, S.A. VAN GILS, S.M. VERDUYN LUNEL, AND H.O. WALTHER, *Delay Equations. Functional, Complex and Nonlinear Analysis*, American Mathematial Society 110, Springer-Verlag, New York, 1995.

[ELR02] K. ENGELBORGHS, T. LUZYANINA, AND D. ROOSE, *Numerical bifurcation analysis of delay differential equations using DDE-BIFTOOL*, ACM Trans. Math. Software, 28 (2002), pp. 1–21.

[ELS01] K. ENGELBORGHS, T. LUZYANINA, AND G. SAMAEY, *DDE-BIFTOOL V.* 2.00: *a Matlab Package for Bifurcation Analysis of Delay Differential Equations*, Technical report TW330, Department of Computer Science, K. U. Leuven, Leuven, Belgium, 2001.

[ER02] K. ENGELBORGHS AND D. ROOSE, *On stability of LMS methods and characteristic roots of delay differential equations*, SIAM J. Numer. Anal., 40 (2002), pp. 629–650.

[ER99] K. ENGELBORGHS AND D. ROOSE, *Numerical computation of stability and detection of Hopf bifurcations of steady state solutions of delay differential equations*, Adv. Comput. Math., 10 (1999), pp. 271–289.

[FSD00] A. FATTOUH, O. SENAME, AND J.M. DION, H_∞ *controller and observer design for linear systems with point and distributed delays*, in Proceedings of 2nd IFAC Workshop on Linear Time Delay Systems, Ancona, Italy, 2000.

[HVL93] J.K. HALE AND S.M. VERDUYN LUNEL, *Introduction to Functional-Differential Equations*, American Mathematical Society 99, Springer-Verlag, New York, 1993.

[LER03] T. LUZYANINA, K. ENGELBORGHS, AND D. ROOSE, *Computing stability of differential equations with bounded distributed delays*, Numer. Algorithms, 34 (2003), pp. 41–66.

[Nic00] S.I. NICULESCU, *Delay Effects on Stability: A Robust Control Approach*, Lecture Notes in Control and Information Science, 269, Springer-Verlag, New York, 2000.

[Ric03] J.P. RICHARD, *Time-delay systems: An overview of some recent advances and open problems*, Automatica, 39 (2003), pp. 1667–1694.

[Tre00] L.N. TREFETHEN, *Spectral Methods in MATLAB*, Software, Environment, and Tools, SIAM, Philadelphia, 2000.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 496–510
© 2005 Society for Industrial and Applied Mathematics

GLUED MATRICES AND THE MRRR ALGORITHM*

INDERJIT S. DHILLON†, BERESFORD N. PARLETT‡, AND CHRISTOF VÖMEL§

Abstract. During the last ten years, Dhillon and Parlett devised a new algorithm (multiple relatively robust representations (MRRR)) for computing numerically orthogonal eigenvectors of a symmetric tridiagonal matrix T with $\mathcal{O}(n^2)$ cost. It has been incorporated into LAPACK version 3.0 as routine STEGR.

We have discovered that the MRRR algorithm can fail in extreme cases. Sometimes eigenvalues agree to working accuracy and MRRR cannot compute orthogonal eigenvectors for them. In this paper, we describe and analyze these failures and various remedies.

Key words. multiple relatively robust representations, numerically orthogonal eigenvectors, symmetric tridiagonal matrix, tight clusters of eigenvalues, glued matrices, Wilkinson matrices

AMS subject classifications. 15A18, 15A23

DOI. 10.1137/040620746

1. Introduction.

Starting in the mid-90s, Dhillon and Parlett developed the algorithm of multiple relatively robust representations (MRRR) that computes numerically orthogonal eigenvectors of a symmetric tridiagonal matrix T with $\mathcal{O}(n^2)$ cost [13, 15, 16, 17, 7]. This algorithm was tested on a large and challenging set of matrices and has been incorporated into LAPACK version 3.0 as routine STEGR. The algorithm is described, in the detail we need here, in section 2. For a more detailed description see also [8]. An example of MRRR in action is given in section 3.

In 2003, one of us (Vömel) came to Berkeley to assist in the modification of STEGR to compute a subset of k eigenpairs with $\mathcal{O}(kn)$ operations. When testing STEGR on more and more challenging matrices, he discovered cases of failure.

Investigation of these cases brought to light assumptions made in STEGR that hold in exact arithmetic and in the majority of cases in finite precision arithmetic, but that can fail. These assumptions, why they are reasonable, and how they can fail are the subject of section 4.

In section 5, we propose and analyze various remedies for the aforementioned shortcomings and show how to incorporate them into MRRR. We also look at the cost of these modifications. We select as most suitable an approach that is based on small random perturbations and introduces artificial roundoff effects. This approach preserves the complexity of the original algorithm.

2. The algorithm of multiple relatively robust representations.

The MRRR algorithm [13, 15, 16, 17, 7] is a sophisticated variant of inverse iteration [11].

*Received by the editors December 13, 2004; accepted for publication (in revised form) March 14, 2005; published electronically October 7, 2005. This work was supported by a grant from the National Science Foundation (Cooperative Agreement ACI-9619020). The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights. A former version of this paper appeared as [20].

http://www.siam.org/journals/sisc/27-2/62074.html

†Department of Computer Science, University of Texas, Austin, TX 78712 (inderjit@cs.utexas.edu).

‡Department of Mathematics and Computer Science Division, University of California, Berkeley, CA 94720 (parlett@math.berkeley.edu).

§Computer Science Division, University of California, Berkeley, CA 94720 (voemel@eecs.berkeley.edu).

It addressed the challenge of computing numerically orthogonal eigenvectors of a symmetric tridiagonal matrix $T \in R^{n\times n}$ for eigenvalues that are very close to each other. The difficulty of this task is that each computed eigenvector tends to be corrupted by components from its neighbors, leading to nonorthogonal vectors.

One way to overcome this obstacle is to achieve highly accurate eigenpairs satisfying

$$\|(T - \lambda_i I)v_i\| = \mathcal{O}(n\epsilon|\lambda_i|); \tag{2.1}$$

the eigenpairs have a small residual norm with respect to $|\lambda_i|$. For a small eigenvalue λ, this is a much stronger demand on accuracy than the common condition

$$\|(T - \lambda_i I)v_i\| = \mathcal{O}(n\epsilon\|T\|). \tag{2.2}$$

(Throughout this paper, ϵ denotes the unit roundoff.)

2.1. Relatively robust representations. The first contribution of MRRR is the observation that for most shifts σ, the standard triangular factorization $T - \sigma I = LDL^T$, L unit bidiagonal, D diagonal exists and has the property that small relative changes in the nontrivial entries of L and D cause small relative changes in each small eigenvalue of LDL^T (despite possible element growth in the factorization). This is in sharp contrast to the entries in $T - \sigma I$ which have this property only in special cases; see [1]. We call (L, D) a if it determines a subset Γ of the eigenvalues to high relative accuracy. Armed with an RRR, we can then compute an approximate eigenvalue to high relative accuracy. This can be done by bisection for a single eigenvalue λ with $\mathcal{O}(n)$ work, yielding $\hat{\lambda}$ satisfying

$$|\lambda - \hat{\lambda}| \le Kn\epsilon|\hat{\lambda}|, \tag{2.3}$$

where K is a modest constant independent of T and λ. Moreover, when the matrix is definite, then the dqds algorithm [18] may be used to compute all eigenvalues to this accuracy in $\mathcal{O}(n^2)$ operations.

2.2. The FP vector. The second innovative feature of MRRR is based on the fact that, for an accurate approximate eigenvalue $\hat{\lambda}$, it is possible to find the index r of the largest component of the true wanted eigenvector v using a double factorization,

$$(LDL^T - \hat{\lambda}I) = L_+D_+L_+^T = U_-D_-U_-^T, \tag{2.4}$$

with $\mathcal{O}(n)$ work. This was discovered in the mid-1990s independently by Godunov and by Fernando; see [15] and the references therein. From

$$\sin^2 \angle(e_r, v) = 1 - \cos^2 \angle(e_r, v) = 1 - v^2(r), \tag{2.5}$$

we see that e_r, the rth column of the identity, cannot be a bad approximation to the true eigenvector. The algorithm solves

$$(LDL^T - \hat{\lambda}I)z = e_r\gamma_r, \;\; z(r) = 1, \tag{2.6}$$

with a normalization factor γ_r and $\gamma_r^{-1} = [(LDL^T - \hat{\lambda}I)^{-1}]_{rr}$. An eigenvector expansion [15] shows that

$$\frac{\|(LDL^T - \hat{\lambda}I)z\|_2}{\|z\|_2} = \frac{|\gamma_r|}{\|z\|_2} \le \frac{|\lambda - \hat{\lambda}|}{|v(r)|}. \tag{2.7}$$

In particular, if r is chosen such that the true eigenvector component $|v(r)| \geq 1/\sqrt{n}$, then an upper bound on (2.7) is $\sqrt{n}\,|\lambda - \hat{\lambda}|$. Now (2.7) together with (2.3) yields (2.1), a relatively small residual norm. In [15], the resulting z from (2.6) is called the [illegible], FP for Fernando and Parlett. The reward for the relatively small residual norm of the FP vector is revealed by the classical gap theorem, often attributed to Davis and Kahan [4, 12]. It shows that

$$|\sin \angle(z, v)| \leq \frac{\|LDL^T z - z\hat{\lambda}\|_2}{\|z\|_2 \operatorname{gap}(\hat{\lambda})} \leq \frac{|\lambda - \hat{\lambda}|}{\|v\|_\infty \operatorname{gap}(\hat{\lambda})}, \tag{2.8}$$

where $\operatorname{gap}(\hat{\lambda}) = \min\{|\hat{\lambda} - \mu| : \lambda \neq \mu, \mu \in \text{spectrum}(LDL^T)\}$. By (2.3), we thus obtain

$$|\sin \angle(z, v)| \leq \frac{Kn\epsilon|\hat{\lambda}|}{\|v\|_\infty \operatorname{gap}(\hat{\lambda})} = \frac{Kn\epsilon}{\|v\|_\infty \operatorname{relgap}(\hat{\lambda})}, \tag{2.9}$$

defining the relative gap, $\operatorname{relgap}(\hat{\lambda})$. Thus, when $\hat{\lambda}$ and $\operatorname{gap}(\hat{\lambda})$ are of not too different size, the angle between the FP vector and the eigenvector is $\mathcal{O}(n\epsilon)$.

2.3. The representation tree and the differential qd algorithm. The third new idea of the MRRR algorithm is to use [illegible] in order to achieve large relative gaps. By large we mean larger than a threshold, usually $1.E-3$. If the relative gaps of a group of eigenvalues are too small, then they can be increased by shifting the origin close to one end of the group. Furthermore, the procedure can be repeated for clusters within clusters of close eigenvalues. The tool for computing the various RRRs is the stationary [illegible] [17]. It has the property that tiny relative changes to the input (L, D) and the output (L_+, D_+) with $LDL^T - \sigma I = L_+ D_+ L_+^T$ give an exact relation. The MRRR procedure for computing eigenpairs is best represented by a rooted tree. Each node of the graph is a pair consisting of a representation (L, D) and a subset Γ of the wanted eigenvalues for which it is an RRR. The root node of this [illegible] is the initial representation that is an RRR for all the wanted eigenvalues; each leaf corresponds to a "singleton," that is, a (shifted) eigenvalue whose relative separation from its neighbor exceeds a given threshold.

2.4. Twisted factorizations. Another ingredient of the MRRR algorithm is the way the FP vector is computed. One uses a [illegible] to solve (2.6) as follows. With the multipliers from (2.4), let

$$N_r = \begin{pmatrix} 1 & & & & & & & & & & \\ L_+(1) & 1 & & & & & & & & & \\ & \ddots & \ddots & & & & & & & & \\ & & \ddots & \ddots & & & & & & & \\ & & & L_+(r-2) & 1 & & & & & & \\ & & & & L_+(r-1) & 1 & U_-(r) & & & & \\ & & & & & & 1 & U_-(r+1) & & & \\ & & & & & & & 1 & \ddots & & \\ & & & & & & & & \ddots & \ddots & \\ & & & & & & & & & \ddots & U_-(n-1) \\ & & & & & & & & & & 1 \end{pmatrix},$$

and, with the pivots from (2.4) and γ_r from (2.6), define

$$\Delta_r = \operatorname{diag}(D_+(1), \ldots, D_+(r-1), \gamma_r, D_-(r+1), \ldots, D_-(n)).$$

Then the twisted factorization at $\hat{\lambda}$ (with twist index r) is given by

$$LDL^T - \hat{\lambda} I = N_r \Delta_r N_r^T. \tag{2.10}$$

Since $N_r e_r = e_r$ and $\Delta_r e_r = \gamma_r e_r$, the solution of (2.6) is equivalent to solving

$$N_r^T z = e_r, \;\; z(r) = 1. \tag{2.11}$$

The great attraction of using this equation is that z can be computed solely with multiplications; no additions or subtractions are necessary. By (2.11), we obtain

$$z(i) = \begin{cases} -L_+(i) z(i+1), & i = r-1, \ldots, 1, \\ -U_-(i-1) z(i-1), & i = r+1, \ldots, n. \end{cases} \tag{2.12}$$

This procedure, together with the use of the differential qd algorithms, permits the error analysis that shows that roundoff does not spoil the overall procedure [7].

2.5. Putting it all together: The MRRR algorithm. Combining all the ideas from the previous sections, we obtain the following algorithm.

ALGORITHM 1: THE MRRR ALGORITHM.

for the current level of the representation tree **do**
  for each eigenvalue with a large relative gap **do**
    Compute an approximation that is good to high relative accuracy.
    Compute its FP vector.
  end for
  for each of the remaining groups of eigenvalues **do**
    Choose shift σ close to the outside of the group.
    Compute new RRR $L_+ D_+ L_+^T = LDL^T - \sigma I$.
    Refine the eigenvalues.
  end for
  Proceed to the next level of the representation tree.
end for

3. The MRRR algorithm in practice. In [21], Wilkinson introduced two classes of symmetric tridiagonal matrices W_{2m+1}^+ and W_{2m+1}^- for positive integers m. W_{2m+1}^+ is more interesting for our purposes. It is defined as

$$W_{2m+1}^+ = \text{tridiag} \left(\begin{array}{ccccccccccc} 1 & & 1 & & \ldots & & 1 & & 1 & & \\ & m & & m-1 & & \ldots & & 0 & & \ldots & m-1 & & m \\ 1 & & 1 & & \ldots & & 1 & & 1 & & \end{array} \right).$$

We want to illustrate the action of MRRR on W_{21}^+. The algorithm first computes a shift σ so that $T - \sigma I$ becomes definite; then all eigenvalues of the root representation $LDL^T = T - \sigma I$ are computed by dqds. In Table 1 we show the eigenvalues of W_{21}^+ and of the root representation whose shift is $\sigma = 10.746194215443904$, a computed upper bound on the largest eigenvalue of T (hence LDL^T is negative definite). The relative gaps are with respect to the root representation.

MRRR uses a threshold to decide which relative gaps are large enough, which is currently $1.E-3$. (That is, eigenvalues that agree to fewer than three digits are declared relatively isolated.) The last column of Table 1 shows that the relative gaps of the first nine eigenvalues exceed the threshold and thus are singletons for the algorithm. The corresponding FP vectors are computed immediately. The remaining

TABLE 1
Eigenvalues, right gaps, and relative right gaps of the negative definite root representation of W_{21}^+.

Index	$\lambda_i(T)$	$\lambda_i(LDL^T)$	$\lvert\lambda_{i+1}-\lambda_i\rvert$	$\lvert\lambda_{i+1}-\lambda_i\rvert/\lvert\lambda_i\rvert$
1	-1.125441522119984	-11.87163573756389	1.379	0.116
2	0.253805817096679	-10.49238839834723	0.694	0.066
3	0.947534367529293	-9.79865984791461	0.842	0.086
4	1.789321352695081	-8.95687286274882	0.341	0.038
5	2.130209219362507	-8.61598499608140	0.831	0.096
6	2.961058884185726	-7.78513533125818	0.082	0.011
7	3.043099292578824	-7.70309492286509	0.953	0.124
8	3.996048201383624	-6.75014601406028	8.306E-3	1.230E-3
9	4.004354023440857	-6.74184019200305	0.995	0.148
10	4.999782477742902	-5.74641173770100	4.619E-4	8.039E-5
11	5.000244425001912	-5.74594979044199	1.000	0.174
12	6.000217522257097	-4.74597669318681	1.651E-5	3.479E-6
13	6.000234031584167	-4.74596018385974	1.004	0.211
14	7.003951798616375	-3.74224241682753	4.109E-7	1.100E-7
15	7.003952209528675	-3.74224200591523	1.035	0.277
16	8.038941115814273	-2.70725309962963	7.015E-9	2.591E-9
17	8.038941122829025	-2.70725309261488	1.172	0.433
18	9.210678647304919	-1.53551556813899	5.641E-11	3.673E-11
19	9.210678647361332	-1.53551556808257	1.535	0.100
20	10.746194182903322	-3.25405817949E-8	7.160E-14	2.200E-6
21	10.746194182903393	-3.25405101953E-8		

eigenvalues come in pairs which are well isolated from each other. For each of these pairs, the algorithm finds a close shift, computes a new RRR, $LDL^T - \tau I = L_+ D_+ L_+^T$, and refines the eigenvalues of that new RRR. In Table 2 we show these refined local eigenvalues with respect to their new RRRs. Note that the relative gaps inside the pairs have been significantly improved over those of the root representation (they are all of order 1) because $|\lambda_{2i+1} - \lambda_{2i}| \approx |\lambda_{2i}|,\ i = 5, \ldots, 10$. With respect to the appropriate representation, each eigenvalue is now a singleton. In Figure 1 we show the representation tree associated to these computations.

TABLE 2
Eigenvalues and relative right gaps of the respective children of the root representation of W_{21}^+. *The last column shows the (incremental) shift* τ *of the respective representation with respect to the shift of the root representation.*

Index	Local λ ($L_+D_+L_+^T$)	$\lvert\lambda_{i+1}-\lambda_i\rvert/\lvert\lambda_i\rvert$	τ (shift of new RRR)
10	-4.619472590129691E-4	1.0	-5.745949790441989
11	-1.821136708729214E-15		
12	-1.6509327074115723E-5	1.0	-4.745960183859731
13	-5.239123482043716E-15		
14	-4.109123007352570E-7	1.0	-3.742242005915228
15	-3.071368719442505E-17		
16	-7.014749992198161E-9	1.0	-2.7072530926148803
17	-6.432284272916864E-17		
18	-5.641544663900284E-11	1.0	-1.5355155680825696
19	-1.932676195102966E-15		
20	-7.159957809285440E-14	1.0	-3.25405101953065E-8
21	-9.193866613291496E-23		

4. Fallible assumptions. In this section, we show how MRRR can fail. Our illustrations use (large) Wilkinson and glued Wilkinson matrices. By a glued (sym-

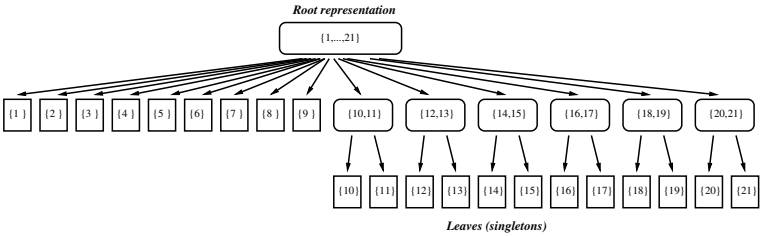


FIG. 1. *The representation tree for W_{21}^+. Square boxes correspond to singletons; boxes with round corners correspond to eigenvalue groups for which an individual representation is needed to improve relative gaps.*

metric tridiagonal) matrix, we denote a matrix

$$T(\gamma) = \begin{bmatrix} T_1 & & \\ & \ddots & \\ & & T_p \end{bmatrix} + \gamma \sum_{i=1}^{p-1} v_i v_i^T,$$

where v_i is a vector with only two nonzero entries of size 1 in the positions corresponding to the last row of T_i and the first row of T_{i+1} in T, where $T_1, \ldots, T_p$ are symmetric tridiagonal matrices.

4.1. Finding large relative gaps.

4.1. By suitable shifting, at least one eigenvalue in a tight cluster of eigenvalues of an unreduced symmetric tridiagonal matrix can be given a relative gap exceeding any given threshold.

In exact arithmetic, the eigenvalues of an unreduced symmetric tridiagonal matrix are simple. Thus, a shift to either end of a cluster of close eigenvalues yields a local eigenvalue that is (nearly) zero and a relative gap of (almost) ∞. Thus, at least one local eigenvalue becomes a singleton.

When working in finite precision with roundoff, we expect that a tight cluster, however close in exact arithmetic, will be computed with differences exceeding $\epsilon\lambda_i$. By shifting close to the end of a cluster, we can expect the relative gaps to become large enough. A typical example was shown in Table 2. For example, an incremental shift of -3.25E-8 from the root representation to the RRR for eigenvalues $20, 21$ increased the relative gap of eigenvalue 20 to 1 since $|\lambda_{21} - \lambda_{20}| \approx |\lambda_{20}|$. For matrices with tighter clusters, more steps of shifting and gradual refinement of the eigenvalues might be needed to increase the relative separation.

In the following experiment, we look at the second-smallest cluster of a matrix T obtained from five copies of W_{201}^+ glued together by $\sqrt{\epsilon}$. According to the strategy of finding large relative gaps, we refine the extremal eigenvalues of the cluster to high accuracy. Then we compute LDL^T factorizations at both ends of the cluster and choose the end with the smaller element growth for the representation. If the code performed as usual, by shifting closer and closer to the cluster, we would see for a certain shift the cluster splitting up into subgroups and singletons. Instead, we see a refinement of the eigenvalues of the cluster down to the underflow threshold without being able to find subclusters. In terms of the representation tree, this corresponds to a long chain and had not been observed in former tests. In Table 3 we show for some parts of the chain the extremal eigenvalues of the cluster together with the

corresponding depth of the representation tree. An explanation of this phenomenon is given in section 4.3.

TABLE 3

(Local) eigenvalues of the second-smallest cluster of the glued matrix T by representation tree level. (We define the root representation as Level 0 and increase the level number when descending in the tree from a parent to a child.)

Index	Level 1	Level 2	Level 10	Level 21
6	1.0524021997591E-13	-7.6189638190266E-30	3.83337214181111E-149	1.14585459648E-312
7	1.0524021997591E-13	-7.6189638190266E-30	3.83337214181111E-149	1.14585459649E-312
8	1.0524021997591E-13	-7.6189638190266E-30	3.83337214181111E-149	1.14585459651E-312
9	1.0524021997591E-13	-7.6189638190266E-30	3.83337214181111E-149	1.14585459653E-312
10	1.0524021997591E-13	-7.6189638190266E-30	3.83337214181111E-149	1.14585459654E-312

4.2. Finding an RRR.

4.2. It is always possible to find an RRR close enough to one (or both) ends of a tight cluster to yield a singleton eigenvalue.

This statement differs from Assumption 4.1 by bringing in the property that the representation should be relatively robust.

In exact arithmetic, the triangular factorization of a matrix $T - \sigma I = LDL^T$ exists for all but a finite set of shifts σ, the so-called Ritz values of T. Each Ritz value is the eigenvalue of a leading principal submatrix. The element growth, that is, $\|D\|$, is bounded except in a very small interval around these Ritz values.

It can be shown that the even-indexed eigenvalues of the Wilkinson matrices W^+_{2m+1} are also Ritz values. In fact, they are the eigenvalues of the leading (and trailing) $m \times m$ submatrix; see [21]. This implies that an LDL^T factorization with such a shift will break down or, in other words, an RRR with such a shift does not exist. Luckily, for the simple Wilkinson matrices it is usually not necessary to choose such a shift for a representation. For the well-isolated eigenvalues at the lower end of the spectrum, one need not shift close to any Ritz value in order to obtain large relative gaps between the eigenvalues. At the upper end of the spectrum, the eigenvalues come in pairs consisting of two eigenvalues, the smaller having an even and the larger an odd index. In this case, one need not shift close to the left (Ritz) eigenvalue; a shift to the right of the pair is excellent.

In order to find an RRR, the MRRR algorithm first factors at both ends of the cluster and monitors the element growth. If the element growth at one end is small, it is chosen as origin for the RRR. Otherwise, the shift backs off from the ends and tries again. The benefit from backing off is the possibility of smaller element growth; the danger is to obtain an RRR whose eigenvalues have small relative gaps.

In our experiment, we consider again the second-smallest cluster of five copies of W^+_{201} glued together by $\sqrt{\epsilon}$. As seen in Table 3, the glue does not change the finite precision approximation of the true eigenvalues. Thus, in limited precision, we face an eigenvalue/Ritz value of multiplicity five. A representation in a small neighborhood cannot exist. Normally, we would back off enough to ensure modest element growth. However, since the clustered eigenvalues have zero gaps, backing off is pointless. Even if we found an RRR, it would not be suitable for the algorithm. Note that the problem does not occur in the (unglued) matrix but only in the glued case for a reason that will be explained in section 4.3.

4.3. Why glued matrices are difficult for MRRR. Table 3 shows that the eigenvalues of the glued matrix are equal to working accuracy despite roundoff error.

The following simple model can provide a good intuition of the phenomenon: Suppose that the glue connecting the copies were zero but the algorithm "was not aware of it," that is, treated it as a nonsplitting matrix. Then [illegible] there would be five equal computed eigenvalues. So, in practice, if there are eigenvalues for which the glue parameter has a negligible effect, then the algorithm will fail. This is exactly what happens in this case. In fact, it can be shown that the magnitude of the effect of the glue on eigenvalues is tied to top and bottom entries of the corresponding eigenvectors. As an intuitive argument for the case of two glued matrices, we remind the reader of Theorem 4.3, which plays an important role in the divide-and-conquer algorithm [2, 3, 9]. A complete analysis of glued matrices is in preparation [19].

THEOREM 4.3. [illegible] $T_1 = Q_1\Lambda_1 Q_1^T, T_2 = Q_2\Lambda_2 Q_2^T$ [illegible]

$$D = \begin{bmatrix} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{bmatrix}, \quad u = \begin{bmatrix} Q_1^T & 0 \\ 0 & Q_2^T \end{bmatrix}, \quad v = \begin{bmatrix} \text{last column of } Q_1^T \\ \text{first column of } Q_2^T \end{bmatrix}.$$

[illegible] u [illegible] T [illegible]

$$\det(I + \gamma(D-\lambda)^{-1}uu^T) = 1 + \gamma u^T(D-\lambda)^{-1}u = 1 + \gamma\sum_{i=1}^{n}\frac{u_i^2}{d_i - \lambda} = 0. \tag{4.1}$$

It is well known that when u_i is exactly zero, then d_i is an eigenvalue of the glued matrix. Likewise, when u_i is tiny (as in the case of the top and bottom entry of the second eigenvector of W_{201}^+), the effect of the glue on the separation of the eigenvalues is negligible. Thus, for these glued matrices, the special structure can defeat the normal effects of rounding errors.

4.4. Computing the FP vector.

[illegible] 4.4. Let the eigenvalue approximation $\hat{\lambda}$ have high relative accuracy. By starting inverse iteration according to (2.6) with e_r, r being the largest component of the true eigenvector, we are guaranteed that the FP vector has a small angle to the true eigenvector according to (2.9) [illegible] (2.11) and can be computed by (2.12)

This is very satisfactory and correct in exact arithmetic. In finite precision arithmetic, we might not be able to accurately solve (2.6) (or (2.11)) by (2.12). Although we obtain a vector with small residual, we need not obtain the true FP vector which is guaranteed to have a small angle to the true eigenvector. The culprit is underflow, as we now explain.

The eigenvectors of the largest eigenvalue pair of a Wilkinson matrix W_{2m+1}^+, m sufficiently large, provide examples. Both eigenvectors have their "essential" nonzero part concentrated at the top and the bottom and decay rapidly towards the middle. This decay commonly causes MRRR (and other methods as well) to return the bisectors instead of the true eigenvectors; see the example of MATLAB in Figures 2 and 3.

These bisectors are perfectly satisfactory from the point of view of numerically small residual norms and numerical orthogonality. Moreover, the true eigenvectors can be obtained as the sum and the difference of the two bisectors.

We now look at the execution of (2.12) on the two largest eigenpairs of W_{201}^+. The largest entries of the true eigenvectors are at the first and last positions. When executing (2.12), underflow must occur for large enough m (and in particular occurs

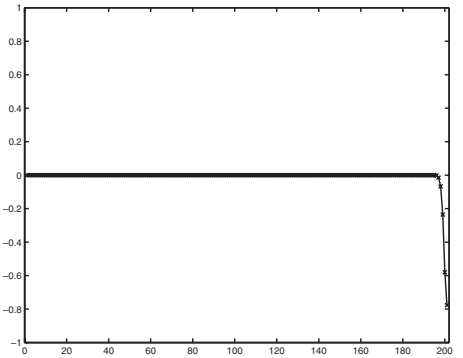


FIG. 2. *Eigenvector* 200 *of* W_{201}^{+} *as computed by MATLAB. The horizontal axis denotes the index. The vertical axis denotes the corresponding vector entry. Note the sharp decay at the right end.*

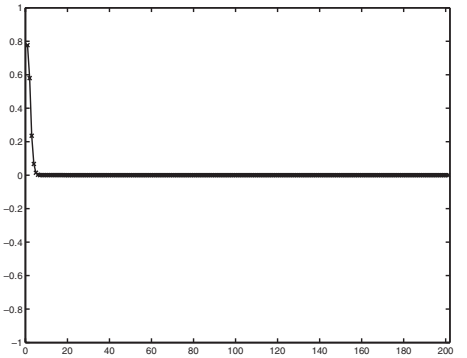


FIG. 3. *Eigenvector* 201 *of* W_{201}^{+} *as computed by MATLAB. The horizontal axis denotes the index. The vertical axis denotes the corresponding vector entry. Note the sharp decay at the left end.*

for $m = 201$), which sets the rest of the vector to zero, yielding the bisector instead of the FP vector.

The danger is that MRRR might choose the same twist index for both vectors, yielding the same bisector. In exact arithmetic, there are two maximal entries in the true eigenvectors, and all would be well if the code chose the twist indices at opposite ends. However, currently there is no mechanism in the MRRR algorithm that guarantees this.

In summary, the FP vector in exact arithmetic is always a good approximation to the true eigenvector. However, in the face of underflow, the computed vector will not satisfy (2.6) and its angle to the true eigenvector will not be small. Furthermore, MRRR is not guaranteed to produce orthogonal vectors.

5. Modifications to MRRR. This section sketches possible modifications to MRRR and discusses their principal features.

Section 5.1 essentially invokes the earlier LAPACK routine STEIN, based on inverse iteration with modified Gram–Schmidt orthogonalization. We apply it only to very tight clusters on which MRRR has failed (as seen in section 4).

Section 5.2 covers Parlett's submatrix method which is very efficient but is applicable only for clusters that are well isolated.

Both these methods can be used as fall-back for the MRRR algorithm on a difficult cluster. However, such a cluster might only be relatively isolated from neighboring clusters and not well separated by an absolute gap. Then the orthogonality among the two clusters cannot be guaranteed without additional work. For this purpose, we discuss the Rayleigh–Ritz projection in section 5.3.

In the worst case, all these modifications will lead to an algorithm that costs $\mathcal{O}(nk^2)$ operations, for a cluster of size k. In section 5.4, we describe Dhillon's elegant proposal to simply get rid of these difficult cases by small componentwise relative random perturbation of the root of the representation. This approach does not sacrifice the $\mathcal{O}(n^2)$ complexity of the original MRRR algorithm.

5.1. Selective inverse iteration with modified Gram–Schmidt orthogonalization. In this section, we take a look at inverse iteration with modified Gram–Schmidt orthogonalization [6, 10, 11], available in LAPACK's STEIN. Its efficiency

problems with matrices from the analysis of molecules prompted the research that led to the development of the MRRR algorithm. It is interesting that STEIN can cope with the very tight glued clusters and thus complement STEGR in this respect.

Given a set of eigenvalues ordered by increasing value, the procedure is as follows. First, if an eigenvalue is too close to its left neighbor, it is perturbed by a small relative amount. Then one step of inverse iteration with a random right-hand side is performed. The resulting vector is kept orthogonal to all previously computed eigenvectors that belong to eigenvalues within distance of the current eigenvalue. (The choice of the parameter $ORTOL = 10^{-3}||T||_1$ determines the level of numerical orthogonality.) The vector obtained is scaled and used as a new right-hand side until the iteration converges. However, it is the orthogonalization feature that makes STEIN inefficient.

Thus, we could use STEIN as backup for the case when MRRR cannot achieve relative separation for a cluster. However, this modification would destroy the claim that MRRR has $\mathcal{O}(n^2)$ complexity in the worst cases. While inverse iteration itself costs only $\mathcal{O}(nk)$ operations, the total cost of the algorithm is dominated by the orthogonalization of a computed vector to all previously found eigenvectors, yielding an $\mathcal{O}(nk^2)$ algorithm.

5.2. The submatrix method. In this section, we sketch briefly the ideas behind the submatrix method as a means of obtaining an orthogonal basis for the subspace defined by a tight isolated cluster. For more details, we refer the reader to [14].

The envelope vector of an invariant subspace of the (symmetric) matrix T is defined by the property that its jth entry is the maximal jth entry over all unit vectors in the subspace. For a tight isolated cluster of k eigenvalues, the envelope has k hills separated by $k-1$ valleys. The method chooses k submatrices of T with two properties:

- Each submatrix has a simple eigenvalue in the cluster interval whose normalized eigenvector has very small first and last entries (except for the first entry of the submatrix at the top and the last entry of the submatrix at the bottom).
- The indices of each submatrix overlap, at worst, the indices of its adjacent neighbors.

The respective k (small) eigenvectors of these k submatrices, when padded appropriately with zeros, form a distinguished basis for the subspace. When the cluster is well isolated, these basis vectors are numerically orthogonal and have a small residual norm (with respect to T). Otherwise, they might be not quite orthogonal (but, by construction, are guaranteed to be linearly independent). Suitable linear combinations are needed to form the eigenvectors of T. We show how to do that in the next section.

The main attraction of the submatrix method is that by construction, each submatrix has exactly one eigenvalue in the interval spanned by the cluster; thus, in terms of the submatrix, we have to compute the eigenvector of an isolated eigenvalue, which is an easy task.

Typically, the submatrix method is an $\mathcal{O}(nk)$ process. However, its limitation is that the tight cluster needs to be isolated. Some large glued Wilkinson matrices produce small tight clusters that are not well isolated; an example is shown in the next section. We could not find a modification of the submatrix method that did not spoil its complexity in that case.

5.3. Rayleigh–Ritz projections and how to couple MRRR with selective inverse iteration. In the previous sections, we have seen how to obtain numerically orthogonal vectors spanning the approximate invariant subspace of a tight cluster. These basis vectors will be (automatically) orthogonal to other eigenvectors belonging to eigenvectors outside the cluster, provided that the cluster is isolated. However, when the (absolute) gaps on either side of the tight cluster are not large enough, orthogonality cannot be guaranteed. For this reason, neither inverse iteration nor the submatrix method alone can provide a fall-back solution for tight clusters on which the MRRR algorithm fails.

A possible solution is the following postprocessing step whenever the MRRR algorithm fails to resolve a tight nonisolated cluster and resorts to a special method to compute a subspace basis. After all eigenvectors have been computed, we can group together those of the tight cluster and all those belonging to eigenvalues which are close to it so that the enlarged set has gaps greater than $ORTOL$ to the rest of the spectrum. (Here, we choose the parameter $ORTOL$ in the same way as in section 5.1.) Then, the Rayleigh–Ritz-values and -vector give us the best approximation from that subspace. The Ritz vectors are orthogonal among themselves and the orthogonality to the eigenvectors of the rest of the spectrum is governed in the same way as for STEIN; see section 5.1.

Specifically, let Z_k be the matrix whose k normalized linearly independent column vectors $z_i, 1 \leq i \leq k$, span the enlarged subspace. Furthermore, let

$$||Tz_i - \hat{\lambda}_i z_i|| = \mathcal{O}(n\epsilon||T||) \ \forall i \in \{1, \ldots, k\}\,.$$

Define the matrix $A = (a_{ij}) \in \mathcal{R}^{k\times k}$ by

$$a_{ij} = \frac{\lambda_i + \lambda_j}{2}(z_i, z_j)$$

and the matrix $B = (b_{ij}) \in \mathcal{R}^{k\times k}$ by $b_{ij} = (z_i, z_j)$.

If Λ holds the (generalized) eigenvalues and the columns of G hold the corresponding generalized eigenvectors of the pencil (A, B), $AG = BG\Lambda$, then Rayleigh–Ritz theory states that

$$(\Lambda, Z_k G) = \arg\min_{(\hat{\Lambda},\hat{Z})} ||T\hat{Z} - \hat{Z}\hat{\Lambda}||_F.$$

In practice, the need for the [illegible] step of computing Ritz approximations from a subspace does not occur very often. This is very fortunate since a cost analysis reveals that for a cluster of size k, it has $\mathcal{O}(nk^2)$ costs.

Next, we give an example where the postprocessing is needed. We consider a matrix T obtained from five copies of W_{201}^+ glued together by $\sqrt{\epsilon}$. Originally, the largest eigenvalue pair of the matrix W_{201}^+ is very close to 100.74619418290335. By gluing five copies of the matrix together, we obtain a cluster of ten eigenvalues that, at Level 1 of the representation tree (where we shift close to the right side of the cluster), splits up into two (tight) clusters of four eigenvalues each and a (not quite so tight) cluster of two in the middle. (The relative gaps between eigenvalues 999 and 1000 and eigenvalues 1001 and 1002 are both large enough to pass MRRR's threshold test.) This is shown in Table 4. In order to show that the eigenvalues belong to different clusters and thus to different representations, we separate them at the cluster boundaries by a horizontal line. Since the two eigenvalues in the middle cluster have large relative gaps between themselves, the corresponding eigenvectors

can be immediately computed by MRRR. However, the algorithm is not able to refine the two outer clusters any more, so it chooses representations closer and closer to the cluster ends without detecting large enough relative gaps in the eigenvalues which keep decreasing towards the roundoff level. Recall the behavior observed in section 4 and documented in Table 3. We can cure this situation in each cluster individually by applying inverse iteration with modified Gram–Schmidt orthogonalization on each group of eigenvalues as described in section 5.1.

TABLE 4
The largest (local) eigenvalues of the glued matrix T by representation tree level.

Index	Level 1	Level 2	Level 10
996	-3.841845502299982E-17	3.2287628082475236E-32	1.4072142703872178E-152
997	-3.841845502299985E-17	3.2287628082475225E-32	1.4072142703872178E-152
998	-3.841845502299985E-17	3.2287628082475225E-32	1.4072142703872178E-152
999	-3.841845502299985E-17	3.2287628082475225E-32	1.4072142703872178E-152
1000	-3.827326426940154E-17		
1001	-6.336625249605947E-22		
1002	-3.680255591970263E-21	-1.1612364990015818E-36	6.952868336216883E-157
1003	-3.680255591970262E-21	-1.1612364990015825E-36	6.952868336216888E-157
1004	-3.680255591970262E-21	-1.1612364990015825E-36	6.952868336216888E-157
1005	-3.680255591970262E-21	-1.1612364990015825E-36	6.952868336216888E-157

However, this procedure will only produce vectors that are orthogonal to the other three vectors within their cluster of four; mutual orthogonality within the cluster of ten eigenvalues of T is not guaranteed and indeed not given. However, the ten vectors are linearly independent and provide a good basis for the invariant subspace. By computing the orthogonal Ritz basis, we obtain orthogonal vectors with small residuals. In Table 5, we display the computed eigenvalues of T and the Ritz values from the postprocessing process.

TABLE 5
The largest eigenvalues of the glued matrix T and the corresponding Ritz values computed in the postprocessing step.

Index	Eigenvalues	Ritz values
996	100.74619418290335	100.74619418290335
997	100.74619418290335	100.74619418290337
998	100.74619418290335	100.74619418290337
999	100.74619418290335	100.74619418290341
1000	100.74619418290335	100.74619418290344
1001	100.74619418290335	100.74619418290345
1002	100.74619420089603	100.74619420089599
1003	100.74619420089603	100.74619420089603
1004	100.74619420089603	100.74619420089604
1005	100.74619420089603	100.74619420089614

5.4. Small relative perturbations of the root representation. In section 5.1, we presented inverse iteration with modified Gram–Schmidt orthogonalization as one remedy for tight clusters arising from gluing. Two key factors for that algorithm's success are

- the perturbation of an eigenvalue by a small (relative) amount to separate it from the rest of the spectrum;
- the choice of a random right-hand side for the inverse iteration.

As mentioned, the important drawback of the approach is its repeated use of reorthogonalization that increases its complexity.

Dhillon had the clever idea to perturb the elements of the root representation by a (relatively) small amount in order to achieve eigenvalue separation and break up the tight clusters. After having found an LDL^T factorization of T (with appropriate shift), he suggests using

$$\hat{d}_i = d_i * (1 + \mu_i k\epsilon),\ 1 \le i \le n, \tag{5.1}$$

$$\hat{l}_j = l_j * (1 + \nu_j k\epsilon),\ 1 \le j \le n, \tag{5.2}$$

where k is a small number, say 4, and $\mu_i, \nu_j \in [-1, 1]$ are chosen at random from a uniform distribution. The size of the perturbation is at the same level as the errors in the differential qd algorithms.

We have verified by experiments that this technique works very well and will show an example below. The rationale for our approach is as follows.

- Symmetric tridiagonal matrices do not necessarily define their eigenvalues to high relative accuracy. A result of Demmel and Kahan [5] states that bidiagonal matrices always do. As long as k from (5.1) and (5.2) is small enough, the eigenvalues of the root representation are perturbed only by a relatively small amount. For this reason, we apply the perturbation to the root representation and not the matrix T itself.
- So far, all our tests indicate that no other representations need to be perturbed. Thus, MRRR computes the eigenvalues of the perturbed root representation $\hat{L}\hat{D}\hat{L}^T$ instead of the computed LDL^T factorization of T.
- In order to reproduce numerical results, we advocate the use of a pseudorandom generator with fixed seed.

In brief, we introduce artificial roundoff into the root representation since machine roundoff might not come to the rescue.

In the following, we study the effect of random perturbations on the second-smallest cluster of a matrix T obtained from five copies of W_{201}^+ glued together by $\sqrt{\epsilon}$. As seen in section 4, this matrix is a counterexample to the basic assumptions of MRRR. Table 3 showed a refinement of the eigenvalues of the cluster up to the underflow threshold without being able to find subclusters. This behavior will be compared to the use of MRRR on a perturbed root representation with $k = 4$ in (5.1) and (5.2). We remark that the respective children at Level 1 of the unperturbed and the perturbed root representation have (slightly) different shifts. For this reason the local eigenvalues appear very different. However, for MRRR, we note that the child of the perturbed representation has (local) eigenvalues with large relative gaps; see the right-most column of Table 6. Hence we can compute the corresponding eigenvectors immediately with guaranteed numerical orthogonality. The relative changes in the eigenvalues of the root representation (Level 0) are of order $\mathcal{O}(n\epsilon)$, n = 1005 in this case.

In conclusion, random perturbations of the root representation work well in practice. The perturbations are made by default, not just in difficult cases which might be hard to recognize a priori. In all our tests, we have yet to find a case where the perturbed clusters are not broken up quickly.

6. Summary and conclusions. The MRRR algorithm is based on very reasonable assumptions that are theoretically sound and hold in the majority of cases in finite precision arithmetic. In this paper, we have shown very subtle and nonobvious

TABLE 6

Investigation of the second-smallest cluster of eigenvalues of the matrix T obtained from five copies of W_{201}^+ glued together by $\sqrt{\epsilon}$; see also Table 3. Shown are the unshifted eigenvalues of the (unperturbed) root representation LDL^T and its perturbed counterpart $\hat{L}\hat{D}\hat{L}^T$. Furthermore, we display the eigenvalues of the representation of the corresponding child clusters that belong to Level 1 of the respective representation trees.

	LDL^T (unperturbed root)		$\hat{L}\hat{D}\hat{L}^T$ (perturbed root)	
Index	Level 0	Level 1	Level 0	Level 1
6	0.2538058170966395	1.052402199759149E-13	0.2538058170965921	-6.243561316319056E-13
7	0.2538058170966395	1.052402199759149E-13	0.2538058170966135	-6.028454790143962E-13
8	0.2538058170966395	1.052402199759149E-13	0.2538058170966301	-5.862486570542836E-13
9	0.2538058170966395	1.052402199759149E-13	0.2538058170966449	-5.714964747742379E-13
10	0.2538058170966395	1.052402199759149E-13	0.2538058170966671	-5.492735741423265E-13

difficulties that can lead to failures. We have illustrated and explained these failures by examples. Furthermore, we have proposed and evaluated possible remedies. These will be adopted for an update of the current LAPACK 3.0 version of STEGR.

Acknowledgment. We thank the two anonymous referees for their fast feedback and constructive remarks.

REFERENCES

[1] J. BARLOW AND J. DEMMEL, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[2] J. BUNCH, P. NIELSEN, AND D. SORENSEN, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[3] J. J. M. CUPPEN, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[4] C. DAVIS AND W. M. KAHAN, *The rotation of eigenvectors by a perturbation.* III, SIAM J. Numer. Anal., 7 (1970), pp. 1–46.

[5] J. DEMMEL AND W. KAHAN, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.

[6] I. S. DHILLON, *Current inverse iteration software can fail*, BIT, 38 (1998), pp. 685–704.

[7] I. S. DHILLON AND B. N. PARLETT, *Orthogonal eigenvectors and relative gaps*, SIAM J. Matrix Anal. Appl., 25 (2004), pp. 858–899.

[8] I. S. DHILLON, B. N. PARLETT, AND C. VÖMEL, *LAPACK Working Note* 162: *The Design and Implementation of the MRRR Algorithm*, Technical report UCBCSD-04-1346, University of California, Berkeley, 2004.

[9] M. GU AND S. C. EISENSTAT, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

[10] I. C. F. IPSEN, *A history of inverse iteration*, in Helmut Wielandt, Mathematische Werke, Mathematical Works, Volume II: Matrix Theory and Analysis., B. Huppert and H. Schneider, eds., Walter de Gruyter, Berlin, 1996.

[11] I. C. F. IPSEN, *Computing an eigenvector with inverse iteration*, SIAM Rev., 39 (1997), pp. 254–291.

[12] B. N. PARLETT, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.

[13] B. N. PARLETT, *The new qd algorithms*, in Acta Numerica, A. Iserles, ed., Cambridge University Press, Cambridge, UK, 1995, pp. 459–491.

[14] B. N. PARLETT, *Invariant subspaces for tightly clustered eigenvalues of tridiagonals*, BIT, 36 (1996), pp. 542–562.

[15] B. N. PARLETT AND I. S. DHILLON, *Fernando's solution to Wilkinson's problem: An application of double factorization*, Linear Algebra Appl., 267 (1997), pp. 247–279.

[16] B. N. PARLETT AND I. S. DHILLON, *Relatively robust representations of symmetric tridiagonals*, Linear Algebra Appl., 309 (2000), pp. 121–151.

[17] B. N. Parlett and I. S. Dhillon, *Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices*, Linear Algebra Appl., 387 (2004), pp. 1–28.
[18] B. N. Parlett and O. Marques, *An implementation of the dqds algorithm (positive case)*, Linear Algebra Appl., 309 (2000), pp. 217–259.
[19] B. N. Parlett and C. Vömel, *Eigenvalues and Eigenvectors of Glued Matrices*, University of California, Berkeley, manuscript, 2005.
[20] B. N. Parlett and C. Vömel, *LAPACK Working Note* 163: *How the MRRR Algorithm Can Fail on Tight Eigenvalue Clusters*, Technical report UCBCSD-04-1367, University of California, Berkeley, 2004.
[21] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, UK, 1965.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 511–531
© 2005 Society for Industrial and Applied Mathematics

SOLVING LINEAR EQUATIONS BY MONTE CARLO SIMULATION*

GIRAY ÖKTEN†

Abstract. A new Monte Carlo estimator for solving the matrix equation $x = Hx+b$ is presented, and theoretical results comparing this estimator with the traditional terminal and collision estimators are given. We then make a detailed investigation of the average complexity of the Monte Carlo estimators when several popular random variable generation techniques are used, and we show that the average complexity can be as low as $C\|z\|_1 n + N$, where n is the number of random walks generated, N is the size of the matrix H, z is the solution of an associated matrix equation, and C is a small constant. As a consequence of the complexity results, we observe how scaling of matrices, a well-known technique in deterministic methods, can increase the efficiency of the Monte Carlo method.

One advantage of the Monte Carlo method is the efficiency at which it can be parallelized. The algorithms we discuss can provide fast and approximate solutions to systems of linear equations in massively parallel computing environments. Surprisingly, sequential (or adaptive) Monte Carlo methods can even be competitive in single-processor computing environments. We present numerical results and compare our Monte Carlo algorithms with Krylov subspace methods for some test matrices.

Key words. Monte Carlo, matrix equations, parallel computing

AMS subject classifications. 65F10, 65C05, 65C40

DOI. 10.1137/04060500X

1. Background and notation.

The first Monte Carlo algorithm for solving linear equations was discovered by von Neumann and Ulam, although in published form it appeared in a paper by Forsythe and Leibler [5]. The algorithm was further studied and extended by Wasow in [20]. The first study that compared the Monte Carlo approach with some deterministic matrix solvers was given by Curtiss [1]. Other important contributions were made by Curtiss [2] and Halton [6], [7].

Detailed descriptions of Monte Carlo algorithms for solving matrix equations can also be found in the books by Spanier and Gelbard [17] and Rubinstein [14]. Following Spanier, we will call the estimators of Forsythe and Leibler and of Wasow the terminal and collision estimators, respectively. We will introduce a new estimator that we call the expected value estimator. For a given matrix equation, there is an adjoint formulation of it, and solving the adjoint formulation is more efficient, especially when one is interested in only one component of the solution vector, as opposed to the full solution vector of the matrix equation. The view taken in this paper is that in very large systems of equations it might be desirable to obtain a fast estimate of the full solution vector using parallel Monte Carlo algorithms implemented over massively parallel computing environments. For this reason, we will consider only the "forward" formulation of the matrix equation, not its adjoint formulation.

Our main problem is estimating the solution of $x = Hx + b$, where H is an

*Received by the editors March 10, 2004; accepted for publication (in revised form) November 18, 2004; published electronically October 7, 2005. The investigation of parallel adaptive Monte Carlo algorithms for solving matrix equations was the subject of a research program funded by Los Alamos National Laboratory and was carried out by the author while he was visiting Claremont Research Institute of Applied Mathematical Sciences during the 2000/01 academic year. Additional support was provided by a Ball State research grant in the summer of 2003, where the author was an associate professor of mathematics.

http://www.siam.org/journals/sisc/27-2/60500.html

†Department of Mathematics, Florida State University, Tallahassee, FL 32306 (okten@math.fsu.edu).

$N \times N$ matrix and b is an N-dimensional vector satisfying certain conditions that will be mentioned later. The Monte Carlo estimators we will consider are based on the Neumann series representation

$$x_j = b_j + \sum_{i_1=1}^{N} h_{ji_1} b_{i_1} + \sum_{i_2=1}^{N} \sum_{i_1=1}^{N} h_{ji_2} h_{i_2 i_1} b_{i_1} + \cdots , \tag{1.1}$$

where x_j is the jth component of the solution vector x. The first step in estimating x_j is to introduce an absorbing (discrete) Markov chain with state space $\{1, \ldots, N+1\}$, a transition matrix $P = (p_{ij})$, where p_{ij} denotes the probability of transition from nonabsorbing states j to i, and a birth vector $a = (a_j)$. This being an absorbing chain, we introduce the N-dimensional vector $p = (p_j)$, where p_j is the probability that the Markovian random walk is absorbed in state j, when it visits state j. We will denote a typical random walk with k steps (collisions) as $\alpha = i_1, i_2, \ldots, i_k$, where $i_j \in \{1, \ldots, N\}$, i_1 is the birth state, and i_k is the final (terminal) state. The probability of occurrence of such a random walk is $a_{i_1} p_{i_2 i_1} \cdots p_{i_k i_{k-1}} p_{i_k}$. Clearly, being absorbed in state i_k is equivalent to making a transition from state i_k to the only recurrent state of the Markov chain (and staying there with probability 1).

In section 2, we will introduce the terminal and collision estimators, and the new expected value estimator, for the solution vector x. In section 3, we will consider the special case when $H = P$ and $b = a$ and compute the variances of the three estimators. As a corollary, we will prove a generalization of a classical result by Wasow. In section 4, we will analyze the complexity of the three Monte Carlo estimators when three popular methods for generating discrete distributions are used: the inverse transform method, acceptance-rejection method, and the alias method. Previous results on the complexity of the Monte Carlo method were restricted to certain estimators and methods of random variable generation. For example, Danilov, Ermakov, and Halton [3] compute the complexity of the collision estimator when $P = |H|$, and Knuth and Yao when the discrete distributions are generated using distribution generating trees [11]. One advantage of computing the complexity for a broader family of estimators and random variable generation methods was to assess the benefits of scaling (section 4.2), a well-known technique in deterministic linear solvers, to the Monte Carlo method. We also discuss implementational issues and the pros and cons of each random variable generation method when applied to certain matrix equations. Finally, in section 7, we present numerical results where the expected value estimator is used in conjunction with a sequential Monte Carlo method. In the literature, Monte Carlo methods are usually compared with classical iterative methods such as Gauss–Seidel, Jacobi, and conjugate gradient. Section 7 will provide numerical results when the Monte Carlo method is compared with two Krylov subspace methods.

2. The estimators. To solve the equation

$$x = Hx + b \tag{2.1}$$

by estimating the sum of the Neumann series mentioned earlier, we need to make sure the series is convergent. Hence we assume that the spectral radius of H is less than one. The Markovian random walks are generated using a transition matrix P and a birth probability vector a satisfying the following conditions:

1. $0 < a_j$, $p_{ij} < 1$ for all i, j,
2. $\sum_{i=1}^{N} p_{ij} < 1$ for all j,
3. $\sum_{j=1}^{N} a_j = 1$,
4. $h_{ij} \neq 0 \Rightarrow p_{ij} \neq 0$ and $b_i \neq 0 \Rightarrow a_i \neq 0$.

We note that the probability that the Markovian random walk is absorbed in state j, when it visits state j, is $p_j = 1 - \sum_{i=1}^{N} p_{ij}$. The first three conditions are simply the definitions of P and a, and the last condition enables us to go from the probability model, described by matrices P and a, to the Neumann series (1.1) that involves H and b. Indeed, let

$$\alpha = (i_1, \ldots, i_k)$$

be a typical random walk beginning at state i_1 and ending at state i_k. Define "weights"

$$w_{ij} = \begin{cases} \dfrac{h_{ij}}{p_{ij}} & \text{if } p_{ij} \neq 0, \\ 0 & \text{if } p_{ij} = 0 \end{cases}$$

and random variables W_m on the space of random walks α by

$$\begin{aligned} W_1(\alpha) &= \frac{b_{i_1}}{a_{i_1}}, \\ W_m(\alpha) &= W_{m-1}(\alpha) w_{i_m i_{m-1}} \\ &= \frac{b_{i_1}}{a_{i_1}} w_{i_2 i_1} \cdots w_{i_{m-1} i_{m-2}} w_{i_m i_{m-1}}, \quad m = 2, \ldots, k. \end{aligned} \tag{2.2}$$

We define the terminal estimator by

$$T_j(\alpha) = W_k(\alpha) \frac{\delta_{i_k j}}{p_{i_k}}, \tag{2.3}$$

the collision estimator by

$$C_j(\alpha) = \sum_{m=1}^{k} W_m(\alpha) \delta_{i_m j}, \tag{2.4}$$

and the expected value estimator by

$$e_j(\alpha) = b_j + \sum_{m=1}^{k} W_m(\alpha) h_{j i_m}, \tag{2.5}$$

where $j = 1, \ldots, N$ and $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.

The proofs that the terminal and collision estimators, as defined above, are unbiased estimators of x_j can be found in [17]. The unbiasedness of the expected value estimator can be shown easily from its definition. One can also observe the following relationship between e_j and C_j:

(2.6)

$$e_j(\alpha) = b_j + \sum_{m=1}^{k} W_m(\alpha) h_{j i_m} = b_j + \sum_{m=1}^{k} \sum_{i=1}^{N} W_m(\alpha) \delta_{i_m i} h_{ji} = b_j + \sum_{i=1}^{N} C_i(\alpha) h_{ji}.$$

Thus one can conclude that the unbiasedness of e_j follows from the unbiasedness of C_j:

$$E[e_j] = b_j + \sum_{i=1}^{N} E[C_i(\alpha)] h_{ji} = b_j + \sum_{i=1}^{N} x_i h_{ji} = x_j.$$

More important, (2.6) provides us with a relationship between the variances of these estimators.

THEOREM 2.1. [illegible] x_j. [illegible]

$$\operatorname{Var}(e_j) = \sum_{i=1}^{N} h_{ji}^2 \operatorname{Var}(C_i) + \sum_{i<k;\ i\neq k} 2h_{ji}h_{jk} \operatorname{Cov}(C_i, C_k).$$

It is possible to modify the probability space slightly and consider (nonabsorbing) random walks that make a fixed number, say K, of collisions. Then the estimators defined above can be used to estimate the Kth partial sum of the Neumann series (1.1). This approach was introduced by Curtiss [2] and further investigated by Sobol' [15]. It is also possible to accelerate the convergence of the Neumann series by applying appropriate transformations to the linear equation; see Dimov and Alexandrov [4] for details.

In the next section, we will investigate a special case of the problem (2.1) where $H = P$ and $b = a$. The weights are then all equal to one, and our estimators can be analyzed using results from the theory of absorbing Markov chains. In particular, we will prove a generalization of an earlier result by Wasow [20].

3. Special case. We consider the equation

$$x = Px + a, \tag{3.1}$$

where

$$0 < a_j, \quad p_{ij} < 1, \quad \sum_{i=1}^{N} p_{ij} < 1 \quad \text{for all } j, \quad \text{and} \quad \sum_{j=1}^{N} a_j = 1. \tag{3.2}$$

Put $Q = P^T$ and $\tilde{Q} = \left(\begin{smallmatrix} Q & p \\ 0 & 1 \end{smallmatrix}\right)$, where $p = (p_j)_{N\times 1}$ and $0 = (0)_{1\times N}$. Then $\tilde{Q}$ is the transition matrix of a discrete Markov chain $\{X_n, n = 0, 1, \dots\}$, or an (absorbing) random walk, with $N+1$ states. The matrix $Q = (q_{ij})$ gives the transition probabilities of the transient states of the chain. The birth probabilities are given by the vector $a = (a_j)$. The jth component of the vector $p = (p_j)_{N\times 1}$ gives the probability that the random walk is absorbed in state j, when it visits state j. We arrange the states so that the first N states are transient and the last state is an absorbing state, and hence recurrent. When convenient, we will let $\mathcal{T}$ denote the transient states and $\mathcal{R}$ denote the only recurrent (absorbing) state.

3.1. Terminal estimator. Let ε be the event that a random walk is absorbed in state j. We have

$$\begin{aligned}
\Pr(\varepsilon) &= \sum_{k=1}^{\infty} \Pr(\varepsilon \mid \text{random walk has } k \text{ steps}) \\
&= p_j a_j + \sum_{i_1=1}^{N} p_j p_{ji_1} a_{i_1} + \sum_{i_2=1}^{N}\sum_{i_1=1}^{N} p_j p_{ji_2} p_{i_2 i_1} a_{i_1} + \cdots \\
&= p_j x_j.
\end{aligned}$$

Now consider the random variable

$$Y_j = \begin{cases} 1 & \text{if random walk is absorbed in state } j, \\ 0 & \text{otherwise.} \end{cases}$$

The expected value of Y_j is $E[Y_j] = \Pr(\varepsilon) = p_j x_j$. The terminal estimator (2.3) takes the following form:

$$T_j = \frac{Y_j}{p_j}.$$

Clearly, $E[T_j] = x_j$, and the variance of the estimator is

$$\operatorname{Var}(T_j) = \frac{1}{p_j^2}\operatorname{Var}(Y_j) = \frac{x_j}{p_j} - x_j^2. \tag{3.3}$$

3.2. Collision estimator. Let C_{ij} denote the number of occurrences of state j when the random walk starts in state i. Here the states i and j are transient states. Put $E[C_{ij}] = s_{ij}$, $E[C_{ij}^2] = s_{ij}^{(2)}$, $S = (s_{ij})$, and $S^{(2)} = (s_{ij}^{(2)})$. For a given matrix $A = (a_{ij})$, let $\operatorname{diag}(A)$ be the diagonal matrix whose diagonal entries are a_{ii}. The following is a standard result from the theory of absorbing Markov chains (see page 103 of Iosifescu [9])

THEOREM 3.1. [illegible] $\tilde{Q} = \left(\begin{smallmatrix} Q & p \\ 0 & 1 \end{smallmatrix}\right)$. [illegible]

$$S = (I - Q)^{-1} \qquad S^{(2)} = S(2 \times \operatorname{diag} S - I).$$

Taking the transpose of $S = (I - Q)^{-1}$ and multiplying by a we obtain $S^T a = (I - P)^{-1} a = x$ from (3.1). Now let C_j be the number of times a random walk visits a transient state j. We have

$$\begin{aligned} E[C_j] &= \sum_k E[C_j \mid \text{random walk starts in state } k] P(\text{walk starts in } k) \\ &= \sum_k s_{kj} a_k = [S^T a]_j = x_j. \end{aligned} \tag{3.4}$$

C_j is the collision estimator, with its general form defined by (2.4).

Now we take the transpose of $S^{(2)} = S(2 \operatorname{diag} S - I)$ to get $S^{(2)^T} = (2 \operatorname{diag} S - I)^T S^T$. Then

$$S^{(2)^T} a = (2 \operatorname{diag} S - I)^T S^T a = (2 \operatorname{diag} S - I)^T x = (2 \operatorname{diag} S - I) x.$$

Since $2 \operatorname{diag} S - I$ is a diagonal matrix with (j, j)th entry equal to $2 s_{jj} - 1$, we get $[S^{(2)^T} a]_j = (2 s_{jj} - 1) x_j$. The second moment of the collision estimator is

$$\begin{aligned} E[C_j^2] &= \sum_k E[C_j^2 \mid \text{random walk starts in state } k] P(\text{walk starts in } k) \\ &= \sum_k s_{kj}^{(2)} a_k = [S^{(2)^T} a]_j = (2 s_{jj} - 1) x_j. \end{aligned}$$

We have obtained the following lemma.

LEMMA 3.2.

$$\mathrm{Var}(C_j) = (2s_{jj} - 1)x_j - x_j^2, \tag{3.5}$$

Now we can compare the variances of the two estimators.

THEOREM 3.3.

$$p_j < \frac{v_j}{2 - v_j},$$

v_j j j.

If j is a transient state, then starting in state j, the number of visits the random walk makes to state j is a geometric random variable with mean $1/v_j$. In other words $s_{jj} = 1/v_j$, and thus (3.5) can be rewritten as

$$\mathrm{Var}(C_j) = \left(\frac{2}{v_j} - 1\right) x_j - x_j^2. \tag{3.6}$$

Now from (3.3) and (3.6), we have

$$\mathrm{Var}(C_j) < \mathrm{Var}(T_j) \Leftrightarrow \left(\frac{2}{v_j} - 1\right) < \frac{1}{p_j} \Leftrightarrow p_j < \frac{1}{\frac{2}{v_j} - 1} = \frac{v_j}{2 - v_j}. \qquad \square$$

3.4. A special case of the above result was proven by Wasow when the birth vector a was given as $a_j = 1$ and $a_i = 0$ for $i \neq j$ (see Spanier and Gelbard [17, p. 54]).

3.3. Expected value estimator. The expected value estimator (2.5) takes the form

$$e_j(\alpha) = b_j + \sum_{i=1}^{N} C_i(\alpha) p_{ji}$$

and thus has variance

$$\mathrm{Var}(e_j) = \mathrm{Var}\left(\sum_{i=1}^{N} C_i(\alpha) p_{ji}\right).$$

Assume that the transition probabilities are chosen so that they are all equal to each other, and let this constant value be p_t. This is a very useful assumption, and it will be discussed further in section 4.2. In some numerical results of section 7, the transition probabilities are indeed chosen this way. The variance $\mathrm{Var}(e_j)$ then simplifies to

$$\mathrm{Var}(e_j) = p_t^2 \,\mathrm{Var}\left(\sum_{i=1}^{N} C_i(\alpha)\right).$$

The term $\sum_{i=1}^{N} C_i(\alpha)$ gives the total number of steps the random walk takes before absorption, and its higher moments, conditioned on the starting state of the random

walk, are known (see Theorem 3.2 of Iosifescu [9]). Using a simple conditional expectation argument, one can obtain the variance of $\sum_{i=1}^{N} C_i(\alpha)$ and reach the following result.

THEOREM 3.5.

$$\mathrm{Var}(e_j) = p_t^2 \left[((2S - I)Se) \cdot a - (Se \cdot a)^2 \right],$$

$S = (I-Q)^{-1}, e$ 1 p_t j.

4. Complexity of the Monte Carlo algorithm. Here we discuss the average complexity of the Monte Carlo algorithm that gives an approximate solution to the equation $x = Hx + b$, where the random walks are generated by a given transition matrix P and birth vector a. The main steps of the algorithm are as follows.

MONTE CARLO ALGORITHM.

1. n $\alpha_1, \ldots, \alpha_n$
2.

$$c_{\mathrm{sum}} = \begin{bmatrix} X_1(\alpha_1) + X_1(\alpha_2) + \cdots + X_1(\alpha_n) \\ \\ X_N(\alpha_1) + X_N(\alpha_2) + \cdots + X_N(\alpha_n) \end{bmatrix},$$

X_j

3. c_{sum} n

$$x = c_{\mathrm{sum}}/n.$$

We will study each step of the algorithm and compute its average complexity. To this end, let z denote the solution of $x = Px + a$, and let C_j be the number of times a random walk visits the transient (nonabsorbing) state j. Then the total number of steps a random walk makes before absorption is $\sum_{j=1}^{N} C_j$, with expected value

$$E\left[\sum_{j=1}^{N} C_j\right] = \sum_{j=1}^{N} z_j = \|z\|_1 .$$

In the following, we will investigate the complexity of random walk generation when the inverse transform, acceptance-rejection, and alias methods are used to sample from the discrete distributions specified by the probability transition matrix P. See Rubinstein [14] or Ross [13] for descriptions of the inverse transform and rejection methods, and Kronmal and Peterson [10] for a description of the alias method. We assume the cost of all arithmetical operations $(\times, \div, \pm)$ are equal and an IF statement is worth l arithmetic operations. By an IF statement we simply mean the logical operation of testing the inequality $\alpha < \beta$. We assume that the cost of generating one pseudorandom number is three arithmetic operations, which is indeed the case if a linear congruential generator is used as the random number generator. We will also assume that the matrix P and vector a have no zero entries. This will keep the exposition simpler: for matrices with zero entries our analysis can easily be modified to obtain tighter bounds for the complexity.

4.1. Complexity of generating the random walks.

4.1.1. Generating random walks using the inverse transform method. Generation of a random walk from birth until termination involves sampling from a number of discrete distributions. To start the random walk, we need to generate the discrete distribution described by the birth vector a. Once the random walk is in some state j, we need to generate the discrete distribution described by the jth column of matrix P with the corresponding probability of absorption adjoined, in order to determine the next state to visit or whether absorption takes place. We assume the implementation of the inverse transform method employs the simple linear search, where the search for the subinterval that contains the random number starts from the leftmost interval and continues to the right.

LEMMA 4.1. $\mathcal{C}^a_{\text{inv}}$ $\mathcal{C}^{\tilde{P}_j}_{\text{inv}}$ a $\tilde{P}_j = \{p_{1j}, p_{2j}, \ldots, p_{Nj}, p_j\}$ j P j $\mu_a = \sum_{i=1}^{N} i a_i$ $\mu_{\tilde{P}_j} = \sum_{i=1}^{N+1} i p_{ij}$ $p_{(N+1)j}$ p_j

$$E[\mathcal{C}^a_{\text{inv}}] = (l+1)(\mu_a - N a_N) + 2(1 - a_N)$$

$$E\left[\mathcal{C}^{\tilde{P}_j}_{\text{inv}}\right] = (l+1)(\mu_{\tilde{P}_j} - p_j N) - (l+3)p_j + 2.$$

We will prove only the second equation; the first one is proved similarly. Consider the discrete distribution $\tilde{P}_j = \{p_{1j}, p_{2j}, \ldots, p_{Nj}, p_j\}$ where $p_j = 1 - \sum_{i=1}^{N} p_{ij}$. Generate a random number ξ between 0 and 1. If

$$p_{1j} + \cdots + p_{i-1,j} \le \xi < p_{1j} + \cdots + p_{ij}, \quad i = 1, \ldots, N, \tag{4.1}$$

then state i is selected as the next state visited by the random walk. In the above inequality, the left-hand side is understood to be 0 when $i = 1$. If ξ does not satisfy (4.1) for any $i = 1, \ldots, N$, then absorption takes place. To determine that ξ satisfies (4.1) for some particular i, one has to perform $(i-1)$ additions and i IF statements, in addition to the cost of generating ξ. The total number of operations is $i - 1 + li + 3 = (l+1)i + 2$. Since the probability that ξ satisfies (4.1) for some i is p_{ij}, we have

$$P\left(\mathcal{C}^{\tilde{P}_j}_{\text{inv}} = 2 + (l+1)i\right) = p_{ij}, \quad i = 1, \ldots, N,$$

and the expected value of $\mathcal{C}^{\tilde{P}_j}_{\text{inv}}$ is

$$E\left[\mathcal{C}^{\tilde{P}_j}_{\text{inv}}\right] = \sum_{i=1}^{N} p_{ij}(2 + (l+1)i) = (l+1)\mu_{\tilde{P}_j} - (l+1)p_j N - (l+3)p_j + 2. \qquad \square \tag{4.2}$$

Now we investigate the number of times one needs to sample from these discrete distributions. Clearly, we sample from a only once during the course of one random walk. The number of times we need to sample from $\tilde{P}_j$ is the number of times the random walk visits state j, which is C_j. The expected number of times we generate the jth column is then $E[C_j] = z_j$. Putting all these results together, we obtain the following theorem.

THEOREM 4.2.

$$E[\mathcal{C}_{\text{inv}}] = E[\mathcal{C}^a_{\text{inv}}] + \sum_{j=1}^{N} z_j E\left[\mathcal{C}^{\tilde{P}_j}_{\text{inv}}\right] = (l+1)(\mu_a - Na_N) + 2(1-a_N)$$
$$+ \sum_{j=1}^{N} z_j((l+1)(\mu_{\tilde{P}_j} - p_j N) - (l+3)p_j + 2). \tag{4.3}$$

Since $\mu_a \le N$, $\mu_{\tilde{P}_j} < N+1$, $0 < a_N < 1$, and N will be usually a large number, we have

$$(l+1)(\mu_a - Na_N) + 2(1-a_N) < (l+1)N + 2 \approx (l+1)N,$$
$$(l+1)(\mu_{\tilde{P}_j} - p_j N) - (l+3)p_j + 2 < (l+1)(N+1) + 2 \approx (l+1)N,$$

so an upper bound for the average complexity is, approximately,

$$(l+1)\left(\|z\|_1 + 1\right) N. \tag{4.4}$$

There is essentially no additional storage required by our implementation of the inverse transform method. The calculations required in linear search are done as they are needed, and discarded immediately after they are used.

4.1.2. Generating random walks using the acceptance-rejection method. Let $q_i = \frac{1}{N+1}$, $i = 1, \dots, N+1$, describe the probability mass function for the discrete uniform distribution on $1, \dots, N+1$. Now we generate from the discrete distributions $\tilde{P}_j = \{p_{1j}, p_{2j}, \dots, p_{Nj}, p_j\}$ and a, via the rejection method. The rejection algorithm that samples from $\tilde{P}_j$ takes the following form. As before, $p_{(N+1)j}$ is understood to be p_j.

ALGORITHM 4.3.

1. k $1, \dots, N+1$. U_1 $k = [(N+1)U_1] + 1$ $(N+1)U_1$
2. U_2.
3. $U_2 < \frac{(N+1)p_{kj}}{c} = \frac{p_{kj}}{\|\tilde{P}_j\|_\infty}$ $\tilde{P}_j$ k 1.

Here $c = \max_{1 \le i \le N+1}(N+1)p_{ij} = (N+1)\|\tilde{P}_j\|_\infty$. The number of iterations of the algorithm needed to obtain a value from $\tilde{P}_j$ is a geometric random variable with mean c. Since each iteration requires the generation of 2 random numbers, 1 multiplication, 1 division, 1 addition, and 1 IF statement (we ignore the integer part operation), we have

$$E\left[\mathcal{C}^{\tilde{P}_j}_{\text{rej}}\right] = (l+9)c = (l+9)(N+1)\|\tilde{P}_j\|_\infty. \tag{4.5}$$

A slight modification of the above algorithm shows that the complexity of generating the birth vector a satisfies

$$E[\mathcal{C}^a_{\text{rej}}] = (l+9)N\left\|a\right\|_\infty. \tag{4.6}$$

Therefore, we have proved the following theorem.

THEOREM 4.4.

$$E[\mathcal{C}_{\text{rej}}] = (l+9)N\,\|a\|_\infty + (l+9)(N+1)\sum_{j=1}^{N} z_j \|\tilde{P}_j\|_\infty. \tag{4.7}$$

Since $\|a\|_\infty < 1$ and $\|\tilde{P}_j\|_\infty < 1$, an upper bound for the average complexity is

$$(l+9)N + (l+9)(N+1)\,\|z\|_1 \approx (l+9)(\|z\|_1 + 1)N. \tag{4.8}$$

For a given $N \times N$ matrix P, we precompute $\|\tilde{P}_j\|_\infty = \max\{p_{1j}, p_{2j}, \ldots, p_{Nj}, p_j\}$, $j = 1, \ldots, N$, and store these numbers in an array of size N. As we will see in section 5 the absorption probabilities p_j will usually be constants, and determined by the user, so no additional calculations or storage will be needed for them.

4.1.3. Generating random walks using the alias method. The alias method introduced by Walker [19] is an efficient method of generating discrete distributions; however, it requires significant storage. Here we consider the implementation of the method due to Kronmal and Peterson [10]. After the initial setup of their algorithm, which creates two tables for each distribution, the method only requires the generation of 2 random numbers,[1] 1 IF statement, 2 addition, and 1 multiplication operations, to sample once from the distribution. We ignore table lookups and one integer part operation. We have the following theorem.

THEOREM 4.5.

$$E[\mathcal{C}_{\text{alias}}] = (l+9) + \sum_{j=1}^{N}(l+9)z_j = (l+9)\left(\|z\|_1 + 1\right). \tag{4.9}$$

What is remarkable about this result is that unlike the inverse transform and rejection methods, the size of the matrix N does not appear in the average complexity explicitly. Of course, the trade-off is the significant storage required by the alias method.

To save storage space, it is best to combine two tables constructed for each distribution, as discussed by Kronmal and Peterson [10]. Then, a real array of size $N+1$ is stored to sample from the column $\tilde{P}_j$. Since we have N columns and one N-dimensional birth vector to sample from, a real square matrix of size $(N+1)$ is needed to be stored for the implementation of the alias method. In other words, we need to store a matrix about the size of the matrix of the linear equation we are trying to solve. This will be probably infeasible for very large matrices.

4.2. The effects of scaling on the complexity of the rejection method. The rejection method can be substantially more efficient than the inverse transform method if the entries of the column vectors of the matrix P are about the same size. We now make the following assumptions.

[1] It is possible to increase the efficiency by generating only one random number, as explained by Kronmal and Peterson. However, we do not pursue this approach, since it may cause correlations in the pseudorandom numbers generated.

1. The probability of absorption p_j is fixed for all states j and denoted by p_{abs}.
2. $p_{ij} \approx \frac{1-p_{\text{abs}}}{N}$, $1 \le i, j \le N$.
3. $a_i \approx \frac{1}{N}$, $1 \le i \le N$.

Assumption 1 implies $\|z\|_1 = \frac{1}{p_{\text{abs}}}$. Indeed, if p_j is fixed for all j, then the number of steps a random walk makes is a geometric random variable with expected value $1/p_{\text{abs}}$. Since the expected number of steps a random walk makes is $E[\sum_{j=1}^{N} C_j] = \|z\|_1$, we have the desired equality. Assumption 2 implies $\|\tilde{P}_j\|_\infty = \max(\frac{1-p_{\text{abs}}}{N}, p_{\text{abs}})$ and Assumption 3 implies $\|a\|_\infty \approx \frac{1}{N}$.

Let us first consider the case when $\|\tilde{P}_j\|_\infty = p_{\text{abs}}$. This situation is likely to arise when the matrix P is dense, and thus the (nonzero) entries are smaller than the probability of absorption. In such a case, one would generate the column vectors as follows: First generate a random number and decide whether the random walk is absorbed or not. If not, then sample from the jth column of P using the rejection method. Then the complexity of the rejection method (4.7) simplifies to $(13+l)\|z\|_1$. If, on the other hand, $\|\tilde{P}_j\|_\infty = \frac{1-p_{\text{abs}}}{N}$, then the complexity is, approximately, $(9+l)\|z\|_1$. In conclusion, the complexity of the rejection method when the matrices are scaled is bounded by

$$E[\mathcal{C}_{\text{rej}}] \le (13+l)\,\|z\|_1\,. \tag{4.10}$$

The bound is approximately equal to the complexity of the alias method (4.9) and does not include the size of the matrix, N. This implies substantial savings of computing time over the inverse transform method, whose complexity is proportional to N.

4.3. Complexity of evaluating the estimators. We now consider the second step of the Monte Carlo algorithm and discuss the complexity of evaluating the cumulative sums

$$X_j(\alpha_1) + X_j(\alpha_2) + \cdots + X_j(\alpha_n), \quad j = 1, \dots, N, \tag{4.11}$$

for each estimator. Let $\alpha = (i_1, \dots, i_k)$ be a typical random walk. We recall that the expected value of k, the number of steps a random walk takes before absorption, is $\|z\|_1$. Let r denote the expected number of [illegible] states a random walk visits before absorption. The exact value of r is known in terms of the matrix S (see [9, p. 105]); however, for simplicity we will use the estimate $r \le \|z\|_1$ in the computations below. We also assume in the following that the logical operation of testing the equality $i_k = j$ and getting ready for the next instruction is worth l arithmetic operations. As before, we measure complexity in terms of the number of arithmetic operations, which are all assumed to have equal complexity.

THEOREM 4.6. [illegible] (4.11) [illegible]

1. $(2\|z\|_1 + 1)n$ [illegible]
2. [illegible] $(l(r-1)+3)\|z\|_1 n$ [illegible]
3. $(2\|z\|_1 + 1)nN + (2\|z\|_1 - 1)n$ [illegible]

[illegible]. 1. Set $\hat{T}_j = 0$, $j = 1, \dots, N$. Consider the random walk $\alpha = (i_1, \dots, i_k)$. All terminal estimators T_j ($j = 1, \dots, N$) except T_{i_k} take the value zero when evaluated at α. Therefore we evaluate only $\hat{T}_{i_k} := \hat{T}_{i_k} + W_k(\alpha)/p_{i_k}$. The term $W_k(\alpha)$ is computed in $2k-1$ operations. So the total number of operations is $(2k-1)+1+1 = 2k+1$. The final result follows from the fact that k is a random variable with expected value $\|z\|_1$, and we generate n random walks.

2. Set $\hat{C}_j = 0$, $j = 1, \ldots, N$. Consider the random walk $\alpha = (i_1, \ldots, i_k)$. All collision estimators C_j $(j = 1, \ldots, N)$ except $C_{i_1}, \ldots, C_{i_k}$ take the value zero when evaluated at α. Therefore we only evaluate $\hat{C}_j := \hat{C}_j + C_j(\alpha)$, $j \in \{i'_1, \ldots, i'_r\}$, where $i'_1, \ldots, i'_r$ are the distinct states α visit. We need to compute

$$\begin{aligned} C_{i'_1}(\alpha) &= W_1(\alpha)\delta_{i_1 i'_1} + W_2(\alpha)\delta_{i_2 i'_1} + \cdots + W_k(\alpha)\delta_{i_k i'_1} \\ &\vdots \\ C_{i'_r}(\alpha) &= W_1(\alpha)\delta_{i_1 i'_r} + W_2(\alpha)\delta_{i_2 i'_r} + \cdots + W_k(\alpha)\delta_{i_k i'_r}. \end{aligned}$$

The Kronecker delta functions can be evaluated efficiently by noting that if, for example, $\delta_{i_1 i'_1} = 1$, then all other terms $\delta_{i_1 i'_2}, \ldots, \delta_{i_1 i'_r}$ are zero and thus can be discarded immediately. A rather loose upper bound on the number of logical operations performed in this manner is $\|z\|_1(r-1)$. The number of nonzero terms in the first equation is given by the random variable $C_{i'_1}$; the number of visits to state i'_1. Therefore there are $C_{i'_1} - 1$ additions to be performed in the first equation. This is true for the other equations as well, so the total number of additions is $\sum_j (C_{i'_j} - 1)$. Since the expected value of $C_{i'_1}$ is $z_{i'_1}$, the average number of additions is bounded by $\|z\|_1 - r$. The weights $W_1(\alpha), \ldots, W_k(\alpha)$ can be computed recursively in an average of $2\|z\|_1 - 1$ operations, as mentioned in the proof of 1. We also perform r additions in updating $\hat{C}_j := \hat{C}_j + C_j(\alpha)$. Overall we have $l(r-1)\|z\|_1 + (\|z\|_1 - r) + (2\|z\|_1 - 1) + r = (l(r-1)+3)\|z\|_1 - 1$ arithmetic operations.

3. Set $\hat{e}_j = 0$, $j = 1, \ldots, N$. Consider the random walk $\alpha = (i_1, \ldots, i_k)$. We need to evaluate $\hat{e}_j := \hat{e}_j + e_j(\alpha)$ for all $j \in \{1, \ldots, N\}$. We have

$$\begin{aligned} e_1(\alpha) &= b_1 + W_1(\alpha)h_{1i_1} + \cdots + W_k(\alpha)h_{1i_k} \\ &\vdots \\ e_N(\alpha) &= b_N + W_1(\alpha)h_{Ni_1} + \cdots + W_k(\alpha)h_{Ni_k}. \end{aligned}$$

The weights $W_1(\alpha), \ldots, W_k(\alpha)$ are computed in an average of $2\|z\|_1 - 1$ operations. The number of multiplications and additions in the above array is $2kN$; a random variable with expected value $2\|z\|_1 N$. So the total number of operations is $(2\|z\|_1 - 1) + 2\|z\|_1 N$. We also have N additions in updating $\hat{e}_j := \hat{e}_j + e_j(\alpha)$ for all j. Since we generate n random walks, the final count is $((2\|z\|_1 - 1) + 2\|z\|_1 N + N)n$ which simplifies to the expression given in the theorem. □

4.4. Total complexity. The final step of the algorithm is the division of the cumulative sums by the number of random walks n. This is done in N divisions. We summarize the results and report the total complexity for a variety of choices in Table 4.1. In the first column, IT refers to the inverse transform method, and AR refers to the rejection method. We use the bound (4.4) for the inverse transform complexity, (4.8) for the rejection method complexity, and (4.10) for the rejection

TABLE 4.1

Meth.	Scaled	Est.	Bounds for average complexity
IT	No	e_j	$\left(\|z\|_1 (l+3) + (l+2)\right) nN + \left(2\|z\|_1 - 1\right) n + N$
AR	No	e_j	$\left(\|z\|_1 (l+11) + (l+10)\right) nN + \left(2\|z\|_1 - 1\right) n + N$
AR	Yes	e_j	$\left(2\|z\|_1 + 1\right) nN + \left((15+l)\|z\|_1 - 1\right) n + N$
AR	Yes	T_j	$\left((15+l)\|z\|_1 + 1\right) n + N$

method complexity when the matrices are scaled. The second column of the table refers to whether the matrices P and a are scaled or not. The third column displays the estimator used.

5. Selection of P and a. The best way to select the transition matrix P and birth vector a from the given matrices H and b is not currently known. Here we will consider two approaches and mention their advantages and disadvantages.

5.1. Constant approach. In this approach we first specify the probability of absorption, p_{abs}, which is fixed for every state. In other words, the column sums of P are all equal to $1 - p_{\text{abs}}$. As mentioned earlier, when the probability of absorption is fixed for all states we have the relationship $p_{\text{abs}}\|z\|_1 = 1$. We define $P = (p_{ij})$ and $a = (a_i)$ as

$$p_{ij} = \frac{1 - p_{\text{abs}}}{N} \quad \text{and} \quad a_i = \frac{1}{N}.$$

An advantage of this approach is the speed at which the random walks can be generated. Sampling from a column of P essentially simplifies to sampling the discrete uniform distribution on integers $1, \ldots, N$. Indeed, to sample from a column of P, we generate a random number ξ and first determine whether $\xi > 1 - p_{\text{abs}}$. If the test gives a positive answer, the random walk terminates. If not, we need to decide on the next state, which is an integer between 1 and N with equal probabilities. So the next state is simply given by [illegible] $(N\xi/(1 - p_{\text{abs}})) + 1$. Hence the number of operations needed to sample from a column of P is $l + 3$ with probability p_{abs}, and $l + 6$ with probability $1 - p_{\text{abs}}$, yielding an expected value of $l + 6 - 3p_{\text{abs}}$. Sampling from the birth vector a is simply done by computing [illegible] $(N\xi) + 1$, which takes 5 operations. Since during the course of a random walk from birth until termination we sample from a only once and sample, on average, z_j times from the jth column of P, the total expected number of operations is $5 + \sum z_j(l + 6 - 3p_{\text{abs}})$. Using the relationship $p_{\text{abs}}\|z\|_1 = 1$, this expression simplifies to $(l + 6)\|z\|_1 + 2$. So we have proved the following lemma.

LEMMA 5.1. [illegible] P [illegible] a [illegible]

$$(l + 6)\,\|z\|_1 + 2.$$

We note that this result is close to the complexity of the alias method, and the rejection method when the matrices P and a are scaled but not necessarily constant. Using this bound, we compute the total complexity when terminal, collision, and expected value estimators are used in the Monte Carlo algorithm; see Table 5.1.

5.2. Proportional approach. Here we still fix the probability of absorption, p_{abs}, but define $P = (p_{ij})$ and $a = (a_i)$ as

$$p_{ij} = \frac{|h_{ij}|}{\sum_{k=1}^{N} |h_{kj}|}(1 - p_{\text{abs}}), \qquad a_i = \frac{|b_i|}{\sum_{j=1}^{N} |b_j|}.$$

TABLE 5.1

Est.	Bounds for average complexity
T_j	$\left(\|z\|_1 (l + 8) + 3\right) n + N = O(n) + N$
C_j	$\left(\|z\|_1 (lr + 9) + r\right) n + N = O(n) + N$
e_j	$\left(2\,\|z\|_1 + 1\right) nN + \left(\|z\|_1 (l + 8) + 1\right) n + N = O(nN) + O(n) + N$

This approach is significantly slower than the previous one. First, we need to compute p_{ij} from h_{ij} using the above formula. If storage is cheap, one can precompute all p_{ij} and store them. Otherwise these numbers need to be computed as they are needed during the generation of random walks.[2] We also no longer enjoy the constant complexity of Lemma 5.1 and must use either the rejection or inverse transform method to sample from the columns of P.

The proportional approach seems to make use of the specific information given by H. In particular, if H is a sparse matrix, then the random walks do not make transitions $j \to i$ for which $h_{ij} = 0$, since $h_{ij} = 0$ implies $p_{ij} = 0$. In the constant approach, however, transitions $j \to i$ occur with equal probabilities whether h_{ij} is zero or not.

Since the probability of absorption is a constant, and consequently the norm $\|z\|_1 = 1/p_{\text{abs}}$ is a constant (independent of n, N), the top three upper bounds presented in Table 4.1 are $O(nN) + O(n) + N$ and the last bound is $O(n) + N$.

6. Sequential correlated sampling. Halton introduced three sequential Monte Carlo methods for solving linear equations in [6]. Here we explain one of these methods, called sequential correlated sampling. In the following discussion, we adopt the notation in which x in an equation $x = Hx + r^{(i)}$ is simply the indeterminate (variable) of the equation, but not the solution of the equation.

1. Solve $x = Hx + b$ ($r^{(0)} = b$) and obtain an approximate solution $y^{(0)}$. Residual of the equation is $r^{(1)} = Hy^{(0)} + b - y^{(0)}$. Put $x^{(1)} = y^{(0)}$.

2. Solve $x = Hx + r^{(1)}$ and obtain an approximate solution $y^{(1)}$. Residual of the equation is $r^{(2)} = Hy^{(1)} + b - y^{(1)}$. Put $x^{(2)} = x^{(1)} + y^{(1)}$.

Continuing this way, we obtain the ith step.

i. Solve $x = Hx + r^{(i-1)}$ and obtain an approximate solution $y^{(i-1)}$. Residual is $r^{(i)} = Hy^{(i-1)} + b - y^{(i-1)}$. Put $x^{(i)} = x^{(i-1)} + y^{(i-1)}$.

This scheme produces a sequence of random variables that are all unbiased estimators of the solution vector for the original equation $x = Hx + b$. In addition, Halton [6, Lemma 6] shows that the variances of these estimators converge to zero geometrically if certain conditions are satisfied.

The idea behind sequential correlated sampling is in fact a well-known technique of deterministic linear algebra: the method of iterative improvement. Iterative improvement has been used since the 1940s, especially to improve the accuracy of estimates obtained by Gaussian elimination. A detailed deterministic treatment of this method is given by Higham [8].

7. Numerical results. The main thesis of this paper is that the Monte Carlo method can be quite useful in solving very large systems of equations over distributed and massively parallel computing environments due to the high parallelization efficiency of the method. There is a rich literature on parallel pseudorandom and quasirandom generators that can be used in conjunction with the Monte Carlo algorithms we have discussed (see, for example, Srinivasan, Mascagni, and Ceperley [18] and Ökten and Srinivasan [12] and the references in these papers). The Monte Carlo method provides a quick and approximate solution of a large system over a parallel environment; if a more accurate solution is needed, then the Monte Carlo estimate could be used to improve the effectiveness of a deterministic linear solver via preconditioning.

[2] This is what we implemented in the numerical results later presented in the paper.

TABLE 7.1

Iteration	Terminal	Collision	Expected value
1	1.5	5.0×10^{-1}	1.8×10^{-1}
2	2.7	2.0×10^{-1}	0.4×10^{-1}
3	2.6	0.7×10^{-1}	0.8×10^{-2}
4	7.0×10^{-1}	8.0×10^{-1}	0.2×10^{-2}
5	4.0×10^{-1}	3.0×10^{-1}	0.9×10^{-3}

TABLE 7.2

Iteration	Collision	Expected value
1	2.1×10^{-1}	1.0×10^{-3}
2	1.0×10^{-1}	1.0×10^{-5}
3	8.1×10^{-2}	7.4×10^{-8}
4	3.4×10^{-2}	2.5×10^{-10}
5	2.0×10^{-2}	1.6×10^{-11}
Time	70 sec	94 sec

Although our main objective is not to "beat" a deterministic solver by Monte Carlo in a traditional single-processor computing environment, we will next present some numerical results where the Monte Carlo method used with sequential correlated sampling can actually compete very well with modern deterministic solvers even in such environments. Earlier related numerical results were given by Halton [7], who compared classical iterative methods with the Monte Carlo method that used only the constant approach, when these methods were applied to some special matrices. In our work we will use Krylov subspace methods, the new expected value estimator, and the Monte Carlo method with the constant and proportional approach. We will see examples where the proportional approach gives faster convergence than the constant approach. Before we discuss these numerical results and see how the new expected value estimator compares with deterministic solvers, we will present a few examples that compare the terminal, collision, and expected value estimators when they are used in conjunction with the sequential correlated sampling technique.

EXAMPLE 7.1. Let $H = (h_{ij})$ be the 10-dimensional square matrix defined by

$$h_{ij} = \frac{10 - |i - j|}{100},$$

and $b = [1, 3, 4, 2, 1, 6, -2, 1, 4, 5]^T$. Table 7.1 displays the L^2 norm of the relative error for the first five iterations of the sequential correlated sampling technique. The proportional approach is used for generating the random walks, and the probability of absorption is taken as 0.1. The number of random walks generated at each iteration is 10,000. The expected value estimator provides faster convergence than the other two estimators.

We now consider a randomly generated matrix for another example, and report the norm of the absolute error as well as the computation time.

EXAMPLE 7.2. We generate a random square matrix (from the uniform distribution on (0,1)) of size 99. We then scale the entries of the matrix so that its column norm is 0.99 (and hence the spectral radius is less than one). Table 7.2 displays the L^2 norm of the absolute error at the end of each iteration of the sequential correlated sampling technique, as well as the time to complete the iterations (the terminal estimator fails to be competitive in accuracy as well as efficiency and thus is omitted in the table). The number of random walks generated at each iteration is 40,000. The convergence

TABLE 7.3

Time	$N = 500$	$N = 1000$	$N = 1500$	$N = 2000$	$N = 2500$
CGS	0.16	0.58	1.462	3.114	4.387
GMRES	0.21	0.511	1.272	2.734	3.886
MC(P)	0.11	0.48	0.792	1.312	1.923
MC(C)	0.1	0.33	0.551	0.911	1.352

rate of the expected value estimator is quite impressive compared to the collision estimator, and the additional time required by it is very minimal.

In the following we compare four methods for solving matrix equations: conjugate gradient squared (CGS), GMRES, the Monte Carlo algorithm that uses the constant approach for generating the matrices (MC(P)), and the Monte Carlo algorithm that uses the proportional approach for generating the matrices (MC(C)). MATLAB built-in functions **cgs** and **gmres** are used for the Krylov subspace methods CGS and GMRES. The Monte Carlo methods use the expected value estimator, the inverse transform approach, and the sequential correlated sampling technique. They compute the 2-norm of the relative residual at the end of each iteration of the sequential correlated sampling technique, and stop when this number is less than the user-specified tolerance. The deterministic methods in MATLAB output the 2-norm of the residual at the end of each iteration, and they also terminate when the 2-norm of the relative residual is less than the tolerance. In MATLAB we time from the moment **cgs** (**gmres**) is called to the moment the execution is over. In MC(P) and MC(C), we also time only the execution of the code.

We used MATLAB's built-in functions for the Krylov subspace methods, instead of C++ implementations, for convenience only. The Monte Carlo algorithms, on the other hand, were written in C++. It might be argued that this comparison is not fair, since MATLAB will probably run slower than C++ implementations of the Krylov subspace algorithms. At this point we emphasize again that our point here is not to give an accurate comparison of computing times of these algorithms in single-processor computing environments, but to show that Monte Carlo can be [illegible] even in such computing environments. In addition, the student-written C++ codes for the Monte Carlo algorithms can probably be optimized further, which will change the timing results. However, we can still get a fairly good understanding of various computing times that will be reported in this section: a C++ implementation of GMRES ran twice as fast as its MATLAB implementation when applied to the 1000-dimensional matrix introduced in the next example. If needed, the reader can use this as the rule of thumb in converting timings reported for MATLAB to C++. In particular in the next example, the effect of multiplying the timing results for MATLAB by 1/2 will just move the matrix size after which Monte Carlo algorithms outperform the deterministic methods to $N = 2000$.

[illegible] 7.3. This example is from [7]: Let $H = (h_{ij})$ and $b = (b_i)$, where

$$h_{ij} = \frac{0.9}{N + i + j}, \qquad b_i = 1.37 + \frac{i}{N}.$$

The best parameters for MC(C) and MC(P) are n (number of random walks per iteration) $= 100$, $p_{\text{abs}} = 0.4$. We set the tolerance to 10^{-6}. Table 7.3 and Figure 7.1 give the time to convergence for each method, as we increase the size of the matrix N from 500 to 2500.

The performance of MC(C) over the deterministic methods is very favorable. Table 7.4 gives the norm of the relative residual and iteration number at convergence

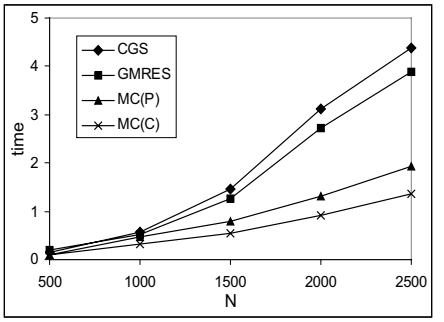


FIG. 7.1. *Time to convergence.*

TABLE 7.4

Rel. res. norm iter.	$N = 500$	$N = 1000$	$N = 1500$	$N = 2000$	$N = 2500$
CGS	5.9e-14 3	6.0e-14 3	6.0e-14 3	6.0e-14 3	6.0e-14 3
GMRES	2.7e-8 3	2.7e-8 3	2.7e-8 3	2.7e-8 3	2.7e-8 3
MC(P)	9.3e-7 3	3.0e-8 4	4.0e-7 3	5.7e-7 3	2.5e-7 3
MC(C)	8.5e-7 4	8.2e-8 4	2.5e-7 3	8.8e-7 3	2.4e-7 3

TABLE 7.5

MC(P) ($N = 1000, p_{\text{abs}} = 0.4$)	$n = 50$	$n = 100$	$n = 1000$	$n = 10{,}000$
Relative residual norm	1.1e-1 1.5e-2 1.2e-3 1.9e-4 1.1e-5 1.4e-6 1.6e-8	6.8e-2 7.8e-4 7.2e-6 3.0e-8	9.0e-3 8.6e-5 2.7e-7	3.7e-4 1.6e-6 1.2e-9
Time	0.54	0.48	1.853	16.284

TABLE 7.6

MC(C) ($N = 1000, p_{\text{abs}} = 0.4$)	$n = 50$	$n = 100$	$n = 1000$	$n = 10{,}000$
Relative residual norm	6.5e-2 3.4e-3 1.3e-4 5.6e-6 2.0e-7	4.1e-2 6.1e-5 1.8e-6 8.2e-8	1.5e-3 7.6e-6 6.8e-8	4.2e-3 2.3e-6 9.1e-9
Time	0.34	0.33	1.172	10.585

(termination). The Monte Carlo method is somewhat sensitive to the parameters p_{abs}, n, and we do not currently know how to choose them optimally. To illustrate this sensitivity, we vary the number of random walks while fixing the other parameters, and report the norm of the relative residual at termination as well as the time to termination. Table 7.5 is for MC(P) and Table 7.6 for MC(C). For both methods,

TABLE 7.7

MC(P) $(N = 1000, n = 1000)$	$p_{\text{abs}} = 0.1$	$p_{\text{abs}} = 0.4$	$p_{\text{abs}} = 0.7$
Relative residual norm	7.2e-3 5.7e-5 8.7e-8	3.5e-2 1.6e-3 4.0e-5 6.1e-7	2.3e-2 4.7e-4 1.2e-5 7.7e-7
Time	6.059	1.523	1.663

TABLE 7.8

MC(C) $(N = 1000, n = 1000)$	$p_{\text{abs}} = 0.1$	$p_{\text{abs}} = 0.4$	$p_{\text{abs}} = 0.7$
Relative residual norm	1.0e-3 1.2e-5 1.9e-8	7.2e-3 1.8e-5 2.1e-8	3.2e-2 1.7e-4 2.7e-6 1.8e-9
Time	2.925	1.151	1.172

TABLE 7.9

$N = 500$	Time	Iteration	Rel. res. norm
CGS	0.16	4	7.5×10^{-9}
GMRES(k)	0.2	(1,5)	5.5×10^{-7}
MC(P)	0.371	7	3.5×10^{-7}
MC(C)	0.431	11	6.4×10^{-7}

the optimal number of random walks is 100, given the selected values for the other parameters. Increasing the number of random walks per iteration slows down the convergence: although error reduction between iterations increases, the cost of generating more random walks eliminates any due advantages. Decreasing the number of random walks again slows down the process: although we generate less random walks, we now need more iterations to reach the tolerance which explains increasing time.

We now investigate the effects of varying the probability of absorption. Tables 7.7 and 7.8 are for MC(P) and MC(C). Here the optimal choice for p_{abs} seems to be 0.4. Decreasing p_{abs} results in producing random walks that make more transitions, which seems to increase the accuracy but at the expense of increased timing. Increasing p_{abs} gives less accurate results, and although generating shorter random walks saves time, the total time to termination actually increases since we need more iterations.

7.4. We choose the entries of H and b at random from a uniform distribution on $(0, 1)$. We then divide the matrix elements by appropriate constants to vary the column norm of the matrix. First we generate a 500×500 matrix H. It has spectral radius 0.84 and condition number 28787 (ratio of the magnitudes of the largest to smallest eigenvalue). The condition number of $I - H$ is 6.6.

After trying a few values for the parameters, we experimentally find the following optimal choices for MCSolver:

$n = 100$ and $p_{\text{abs}} = 0.16$ for MC(C),

$n = 100$ and $p_{\text{abs}} = 0.2$ for MC(P).

Table 7.9 gives the time to termination, the number of iterations, and the norm of the relative residual at termination. It is interesting to note that the constant approach requires eleven iterations to converge, whereas the proportional approach needs only seven iterations. The increased number of iterations explains why MC(C)

TABLE 7.10

Method	MC(P)	MC(C)
		2.6×10^{-1}
		5.1×10^{-2}
	1.3×10^{-1}	9.7×10^{-3}
	1.5×10^{-2}	1.5×10^{-3}
	1.8×10^{-3}	2.3×10^{-4}
Relative residual norm	2.2×10^{-4}	1.9×10^{-4}
	2.6×10^{-5}	4.0×10^{-5}
	3.5×10^{-6}	1.0×10^{-5}
	4.4×10^{-7}	4.5×10^{-6}
		1.8×10^{-6}
		6.4×10^{-7}

TABLE 7.11

$N = 1500$	Time	Iteration	Rel. res. norm
CGS	1.402	3	5.0×10^{-7}
GMRES(k)	1.532	(1,5)	7.5×10^{-8}
MC(P)	2.874	7	9.3×10^{-7}
MC(C)	3.244	12	2.1×10^{-7}

TABLE 7.12

Method	MC(P)	MC(C)
		3.9×10^{-1}
		9.5×10^{-2}
	1.4×10^{-1}	4.6×10^{-2}
	2.0×10^{-2}	9.4×10^{-3}
	3.2×10^{-3}	1.6×10^{-3}
Relative residual norm	4.1×10^{-4}	3.2×10^{-4}
	5.1×10^{-5}	4.4×10^{-5}
	6.8×10^{-6}	2.2×10^{-5}
	9.3×10^{-7}	6.5×10^{-6}
		2.0×10^{-6}
		1.0×10^{-6}
		2.1×10^{-7}

is not (significantly) faster than MC(P) like in the previous example. A closer look at the norm of the relative residuals at the end of each iteration reveals that MC(C) has significantly less error reduction between iterations than MC(P); see Table 7.10. We now increase the size of the matrix to 1500. The spectral radius is 0.86 and the condition number is 1.3×10^7. The condition number of $I - H$ is 7.2. The experimentally verified optimal choices for MCSolver are $n = 100$ and $p_{\text{abs}} = 0.14$ for both MC(P) and MC(C). We now report similar output to the previous case; see Tables 7.11 and 7.12.

At this point, we revisit the issue of different estimators and how they compare. A random matrix of size 80 was created to compare the collision and expected value estimators. The probability of absorption is 0.01. In Table 7.13 we display the L^2 norm of the absolute error at the end of each iteration, and observe that the expected value estimator has a significantly faster rate of convergence. The price paid for this gain was minimal: the computation time for the expected value estimator was only about 1.3 times the computation time for the collision estimator.

7.5. We first consider a 500×500 sparse matrix whose entries are all

TABLE 7.13

Iteration	Collision	Expected value
1	1.9×10^{-2}	5.9×10^{-4}
2	2.1×10^{-3}	5.5×10^{-6}
3	3.1×10^{-4}	1.5×10^{-7}
4	1.3×10^{-4}	5×10^{-10}

TABLE 7.14

$N = 500$	Time	Iteration	Rel. res. norm
CGS	0.22	4	1.1×10^{-8}
GMRES(k)	0.341	(2,1)	3.6×10^{-7}
MC(P)	1.3	8	3.3×10^{-7}

TABLE 7.15

$N = 1500$	Time	Iteration	Rel. res. norm
CGS	1.65	4	8.5×10^{-8}
GMRES(k)	2.25	(2,2)	1.2×10^{-7}
MC(P)	8.15	11	7.7×10^{-7}

zero except for -0.1 on the diagonal, 0.1 on the sub- and superdiagonals, and 0.1 on the 100th sub- and superdiagonals (i.e., for $|i - j| = 100$). The spectral radius of the matrix is 0.37. MC(P) has $p_{\text{abs}} = 0.4$ and $n = 1000$; see Table 7.14. We now increase the size of the matrix to 1500, put 0.05 on the sub- and superdiagonals, and -0.1 on the 100th sub- and superdiagonals. The diagonal elements are -0.1 as before. The spectral radius is 0.40. MC(P) has $p_{\text{abs}} = 0.6$ and $n = 1000$; see Table 7.15.

These results show, among other things, that the sequential Monte Carlo method can do pretty well when compared with modern deterministic methods. The sequential correlated sampling method, however, does not always converge. A theoretical analysis of convergence was given by Halton [6]. Numerically we were able to observe diverging behavior when the method was applied to `randsvd`-type matrices [8]. Since our main thesis in this paper is the efficiency of parallel Monte Carlo algorithms, with or without sequential improvements, over deterministic solvers in massively parallel computing environments, we do not pursue the pros and cons of sequential methods any further. A recent discussion of sequential methods in continuous problems is given by Spanier [16].

8. Conclusion. The amount of work for direct and iterative deterministic matrix solvers is typically $O(N^3)$ and $O(N^2)$. The fastest Monte Carlo algorithm we discussed has complexity $C\|z\|_1 n + N$, where $\|z\|_1 = 1/p_{\text{abs}}$. This is clearly a big advantage if the desired accuracy for the Monte Carlo estimate can be obtained by a moderate number of random walks n. The accuracy of the Monte Carlo estimate can be assessed using confidence intervals, since our estimators, as discussed in section 4, are approximately normal from the central limit theorem. Sequential Monte Carlo methods have the potential of lowering the necessary number of random walks to obtain a certain accuracy dramatically, and we have seen supporting numerical evidence in section 7. The parallel implementation of the Monte Carlo algorithms is straightforward, and essentially no communication is necessary between processors. The sequential Monte Carlo methods, however, will have some (probably very minimal) communications costs. One question that is left unanswered is, how to find the opti-

mal parameters in sequential correlated sampling in general. In numerical results that are not reported here, best choices for n (random walks per iteration) varied between 100 and 1000, and p_{abs} varied between 0.1 and 0.6, for several types of matrices with sizes between 500 and 2500.

Acknowledgments. I thank my graduate assistants Claudia Rangel and Jeho Park, both students at Claremont Graduate University during the 2000/01 academic year, for their work on this project. In particular, Jeho Park is the author of the C++ code that implements the Monte Carlo estimators discussed in this paper. The numerical results were obtained using this code. I also thank Dr. Irene Livshits, Ball State University, for her help in comparing the execution time of different implementations of GMRES.

REFERENCES

[1] J. H. Curtiss, *A theoretical comparison of the efficiencies of two classical methods and a Monte Carlo method for computing one component of the solution of a set of linear algebraic equations*, in Symposium on Monte Carlo Methods, University of Florida, 1954, John Wiley and Sons, New York; Chapman and Hall, London, 1956, pp. 191–233.

[2] J. H. Curtiss, *Monte Carlo methods for the iteration of linear operators*, J. Math. Phys., 32 (1954), pp. 209–232.

[3] D. L. Danilov, S. M. Ermakov, and J. H. Halton, *Asymptotic complexity of Monte Carlo methods for solving linear systems*, J. Statist. Plann. Inference, 85 (2000), pp. 5–18.

[4] I. T. Dimov and V. N. Alexandrov, *A new highly convergent Monte Carlo method for matrix computations*, Math. Comput. Simulation, 47 (1998), pp. 165–181.

[5] G. E. Forsythe and R. A. Leibler, *Matrix inversion by a Monte Carlo method*, J. Math. Tables Other Aids to Comput., 4 (1950), pp. 127–129.

[6] J. H. Halton, *Sequential Monte Carlo*, Proc. Cambridge Philos. Soc., 58 (1962), pp. 57–78.

[7] J. H. Halton, *Sequential Monte Carlo techniques for the solution of linear systems*, J. Sci. Comput., 9 (1994), pp. 213–257.

[8] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[9] M. Iosifescu, *Finite Markov Processes and Their Applications*, John Wiley and Sons, Chichester, UK, 1980.

[10] R. A. Kronmal and A. V. Peterson, Jr., *On the alias method for generating random variables from a discrete distribution*, Amer. Statist., 33 (1979), pp. 214–218.

[11] D. E. Knuth and A. C. Yao, *The complexity of nonuniform random number generation*, in Algorithms and Complexity, Academic Press, New York, 1976, pp. 357–428.

[12] G. Ökten and A. Srinivasan, *Parallel quasi-Monte Carlo methods on a heterogeneous cluster*, in Monte Carlo and Quasi-Monte Carlo Methods, 2000, K.-T. Fang, F. J. Hickernell, and H. Niederreiter, eds., Springer-Verlag, Berlin, 2002, pp. 406–421.

[13] S. M. Ross, *Simulation*, Academic Press, San Diego, CA, 1997.

[14] R. Y. Rubinstein, *Simulation and The Monte Carlo Method*, John Wiley and Sons, New York, 1981.

[15] I. M. Sobol', *Numerical Monte Carlo Methods*, Nauka, Moscow, 1973 (in Russian).

[16] J. Spanier, *Geometrically convergent learning algorithms for global solutions of transport problems*, in Monte Carlo and Quasi-Monte Carlo Methods, 1998, H. Niederreiter and J. Spanier, eds., Springer-Verlag, New York, 2000, pp. 98–113.

[17] J. Spanier and E. M. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*, Addison–Wesley, Reading, MA, 1969.

[18] A. Srinivasan, M. Mascagni, and D. Ceperley, *Testing parallel random number generators*, Parallel Comput., 29 (2003), pp. 69–94.

[19] A. J. Walker, *An efficient method for generating discrete random variables with general distributions*, ACM Trans. Math. Software, 3 (1977), pp. 253–256.

[20] W. Wasow, *A note on the inversion of matrices by random walks*, J. Math. Tables Other Aids to Comput., 6 (1952), pp. 78–81.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 532–552
© 2005 Society for Industrial and Applied Mathematics

BALANCED CENTRAL SCHEMES FOR THE SHALLOW WATER EQUATIONS ON UNSTRUCTURED GRIDS*

STEVE BRYSON† AND DORON LEVY‡

Abstract. We present a two-dimensional, well-balanced, central-upwind scheme for approximating solutions of the shallow water equations in the presence of a stationary bottom topography on triangular meshes.

Our starting point is the recent central scheme of Kurganov and Petrova (KP) for approximating solutions of conservation laws on triangular meshes. In order to extend this scheme from systems of conservation laws to systems of balance laws one has to find an appropriate discretization of the source terms. We first show that for general triangulations there is no discretization of the source terms that corresponds to a well-balanced form of the KP scheme. We then derive a new variant of a central scheme that can be balanced on triangular meshes.

We note in passing that it is straightforward to extend the KP scheme to general unstructured conformal meshes. This extension allows us to recover our previous well-balanced scheme on Cartesian grids. We conclude with several simulations, verifying the second-order accuracy of our scheme as well as its well-balanced properties.

Key words. shallow water equations, central schemes, balance laws, unstructured grids

AMS subject classifications. Primary, 65M06; Secondary, 76B15

DOI. 10.1137/040605539

1. Introduction.

We consider a flow in a two-dimensional channel with a bottom elevation given by $B(\vec{x})$, where $\vec{x} = (x, y)$. Let $H(\vec{x}, t)$ represent the fluid depth above the bottom, and let $\vec{u}(\vec{x}, t) = (u(\vec{x}, t), v(\vec{x}, t))$ be the fluid velocity. The top surface at any time t is denoted by $w(\vec{x}, t) = B(\vec{x}) + H(\vec{x}, t)$.

The shallow water equations, introduced by Saint-Venant in [24] (see also [10]), are commonly used to model flows in rivers or coastal areas. When written in terms of the top surface w and the momentum (Hu, Hv), these equations are of the form

$$\begin{cases} w_t + (Hu)_x + (Hv)_y = 0, \\ (Hu)_t + \left[\dfrac{(Hu)^2}{w-B} + \dfrac{1}{2}(w-B)^2\right]_x + \left[\dfrac{(Hu)(Hv)}{w-B}\right]_y = -g(w-B)B_x, \\ (Hv)_t + \left[\dfrac{(Hu)(Hv)}{w-B}\right]_x + \left[\dfrac{(Hv)^2}{w-B} + \dfrac{1}{2}(w-B)^2\right]_y = -g(w-B)B_y. \end{cases} \tag{1.1}$$

This choice of variables is particularly suitable for dealing with stationary steady-state solutions (see [14, 23] for details). For simplicity we fix the gravitational constant, g, from now on to be $g = 1$.

*Received by the editors March 23, 2004; accepted for publication (in revised form) November 18, 2004; published electronically October 7, 2005.

http://www.siam.org/journals/sisc/27-2/60553.html

†Program in Scientific Computing/Computational Mathematics, Stanford University, Stanford, CA 94305 and the NASA Advanced Supercomputing Division, NASA Ames Research Center, Moffett Field, CA 94035-1000 (bryson@nas.nasa.gov).

‡Department of Mathematics, Stanford University, Stanford, CA 94305-2125 (dlevy@math.stanford.edu). The work of this author was supported in part by the National Science Foundation Career Grant DMS-0133511.

In this work we are interested in approximating solutions of (1.1) on triangular meshes. Our goal is to investigate how to adapt the semidiscrete central schemes on triangular meshes that were recently introduced by Kurganov and Petrova in [16] to this problem. We are interested in deriving a discretization of the source terms in (1.1) that preserves stationary steady-state solutions, as such solutions play an important role in the dynamics of (1.1). While such a balance is not pursued for nonstationary shallow water problems, our simulations indicate that our method works well for such problems.

Central schemes for conservation laws have become popular in recent years as a tool for approximating solutions for multidimensional systems of hyperbolic conservation laws. Like other Godunov-type schemes, central schemes are based on a three-step procedure: , in which an interpolant is reconstructed from previously computed cell averages; , in which this interpolant is evolved exactly in time according to the equations; and , in which the solution is projected back to cell averages. When compared with other methods, central schemes are particularly appealing since they do not require any Riemann solvers and systems can be solved componentwise.

A first-order prototype of central schemes is the Lax–Friedrichs scheme [8]. A second-order extension is due to Nessyahu and Tadmor [21]. Extensions to two dimensions are due to Arminjon and Viallon [1] and Jiang and Tadmor [13]. By estimating bounds on the local speeds of propagation of information from discontinuities, it is possible to pass to the semidiscrete limit (see [15, 17] and the references therein). There are several extensions of central schemes to unstructured grids. A fully discrete method is due to Arminjon, Viallon, and Madrane [2]; a recent semidiscrete scheme was proposed by Kurganov and Petrova in [16]. Balanced central schemes for shallow water equations on Cartesian grids are due to Russo in the fully discrete framework [23] (see also [20]) and to Kurganov and Levy in the semidiscrete framework [14].

There are many approaches to approximating solutions of (1.1). We refer the reader to, e.g., [3, 4, 7, 9, 11, 18, 19, 22] and the references therein. An equilibrium technique for balancing scalar conservation laws is presented in [5, 6]. Our goal in this paper is to show that balancing is also possible with central schemes. We would like to emphasize that this is the first time in which the balancing issues are treated for central schemes on unstructured grids.

The paper is organized as follows. We start in section 2 with a brief overview of the Kurganov–Petrova (KP) central scheme on triangular meshes. We note that this scheme is not limited to triangular meshes and it can be equally well applied to general unstructured grids. We also make the necessary adjustments to incorporate source terms into the scheme. We proceed in section 3 with the discretization of the cell averages of the source terms for the shallow water equations. The goal is to find a discretization such that the scheme will preserve stationary steady states, i.e., zero velocities and a flat surface. We show that on general unstructured meshes, there is no discretization of the source terms in the shallow water equations that provide a well-balanced form of the KP scheme. We then proceed by showing how to modify the original scheme in such a way that it is possible to obtain a well-balanced discretization of the source terms. We conclude in section 4 with numerical examples that demonstrate the accuracy of our scheme as well as its well-balanced property.

2. Central schemes for balance laws on unstructured grids. We consider the two-dimensional balance law,

$$u_t + f(u)_x + g(u)_y = S(u, x, y, t), \tag{2.1}$$

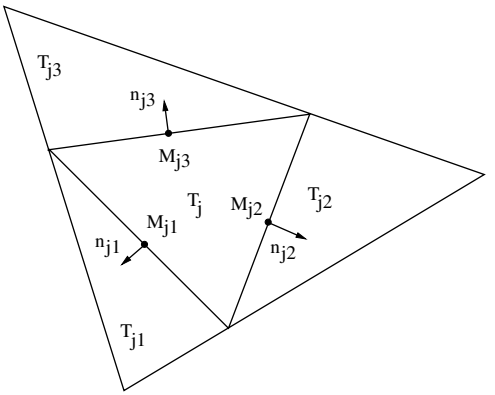


FIG. 2.1. *The triangular grid.*

subject to the initial data $u(x, y, 0) = u_0(x, y)$. We are interested in approximating solutions of (2.1) that are computed in terms of cell averages on a fixed unstructured conformal grid. To simplify this exposition we first consider the scalar case. The scheme that is described below can be easily extended componentwise to systems of balance laws. We will make this obvious extension later on when dealing with the specific problem of the system of shallow water equations (1.1).

We focus our attention on the central scheme on triangular meshes that was recently derived in [16]. We briefly overview the derivation of this scheme in the setup of conservation laws of the form

$$u_t + f(u)_x + g(u)_y = 0. \tag{2.2}$$

We assume a conformal triangulation of the domain consisting of cells T_j of area $|T_j|$. The neighboring cells to T_j are denoted by T_{jk}, $k = 1, 2, 3$, while the edge that is joint between T_j and T_{jk} is denoted by E_{jk} and is assumed to be of length h_{jk}. We also denote the outward unit normal to T_j on the kth edge as n_{jk}, and denote the midpoint of E_{jk} as M_{jk} (see Figure 2.1).

We assume that the cell averages on all the cells $\{T_j\}$ are known at time t^n,

$$\bar{u}_j^n \approx \frac{1}{|T_j|} \int_{T_j} u(\vec{x}, t^n) d\vec{x}, \tag{2.3}$$

and reconstruct a piecewise polynomial

$$\tilde{u}^n(x, y) = \sum_j u_j^n(x, y) \chi_j(x, y). \tag{2.4}$$

Here $u_j^n(x, y)$ is a two-dimensional polynomial that is yet to be determined and $\chi_j(x, y)$ is the characteristic function of the cell T_j. To simplify the notations we omit the time-dependence in $u_j(x, y)$. We also denote by $u_{jk}(x, y)$ the polynomial that is reconstructed in the cell T_{jk}.

Discontinuities in the interpolant u_j along the edges of T_j propagate with a maximal inward velocity a_{jk}^{in} and a maximal outward velocity a_{jk}^{out}. These velocities can be estimated (for convex fluxes in the scalar case) as

$$\begin{aligned} a_{jk}^{\text{in}}(M_{jk}) &= -\min\{\nabla F\left(u_j\left(M_{jk}\right)\right) \cdot n_{jk}, \nabla F\left(u_{jk}\left(M_{jk}\right)\right) \cdot n_{jk}, 0\}, \\ a_{jk}^{\text{out}}(M_{jk}) &= \max\{\nabla F\left(u_j\left(M_{jk}\right)\right) \cdot n_{jk}, \nabla F\left(u_{jk}\left(M_{jk}\right)\right) \cdot n_{jk}, 0\}, \end{aligned}$$

where $F = (f, g)$. These local speeds of propagation are then used to determine evolution points that are away from the propagating discontinuities. An exact evolution of the reconstruction at these evolution points is followed by an intermediate piecewise polynomial reconstruction and finally projected back onto the original cells, providing the cell averages at the next time step $\bar{u}_j^{n+1}$. Further details can be found in [16].

A semidiscrete scheme is then obtained at the limit

$$\frac{d}{dt}\bar{u}_j^n = \lim_{\Delta t \to 0} \frac{\bar{u}_j^{n+1} - \bar{u}_j^n}{\Delta t}. \tag{2.5}$$

Most of the terms on the right-hand side (RHS) of (2.5) vanish in the limit as $\Delta t \to 0$. The only quantity that has to be determined is a quadrature rule for the integrals of the flux functions over the edges of the cells. If we assume a Gaussian quadrature with m nodes

$$\int_0^1 \varphi(x)dx \approx \sum_{s=1}^{m} c_s \varphi(x_s),$$

which is scaled to h_{jk}, and denote the quadrature points on E_{jk} as G_{jk}^s, the KP scheme for triangular meshes is

$$\begin{aligned} \frac{d\bar{u}_j}{dt} = -\frac{1}{|T_j|} \sum_{k=1}^{3} \frac{h_{jk}}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \sum_{s=1}^{m} c_s \Big[\big(a_{jk}^{\text{in}} F(u_{jk}(G_{jk}^s)) + a_{jk}^{\text{out}} F(u_j(G_{jk}^s)) \big) \cdot n_{jk} \\ - a_{jk}^{\text{in}} a_{jk}^{\text{out}} \big(u_{jk}(G_{jk}^s) - u_j(G_{jk}^s) \big) \Big], \end{aligned} \tag{2.6}$$

where $F = (f, g)$. If the fluxes are integrated with a midpoint quadrature (as suggested in [16]) and we use the notation $u_{jk}^{\text{out}} := u_{jk}(M_{jk})$, $u_{jk}^{\text{in}} := u_j(M_{jk})$, $F_{jk}^{\text{in}} := F(u_{jk}^{\text{in}})$, and $F_{jk}^{\text{out}} := F(u_{jk}^{\text{out}})$, the semidiscrete scheme (2.6) becomes

$$\frac{d\bar{u}_j}{dt} = -\frac{1}{|T_j|} \sum_{k=1}^{3} \frac{h_{jk}}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \Big[\big(a_{jk}^{\text{in}} F_{jk}^{\text{out}} + a_{jk}^{\text{out}} F_{jk}^{\text{in}} \big) \cdot n_{jk} - a_{jk}^{\text{in}} a_{jk}^{\text{out}} \big(u_{jk}^{\text{out}} - u_{jk}^{\text{in}} \big) \Big]. \tag{2.7}$$

A basic observation that will be used below is that the semidiscrete scheme (2.7) is valid for any conformal grid, not necessarily triangular. All that one has to do is to make the suitable adjustments in the notations (e.g., $|T_j|$ being the area of the cell T_j regardless of the shape of that cell) and replace the sum over the three edges of the triangle by a sum over the N_j edges of each cell T_j. When approximating solutions to balance laws of the form (2.1) the scheme has one additional term due to the source term $S(u, x, y, t)$, i.e.,

$$\begin{aligned} \frac{d\bar{u}_j}{dt}(t) = -\frac{1}{|T_j|} \sum_{k=1}^{N_j} \frac{h_{jk}}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}} \Big[\big(a_{jk}^{\text{in}} F_{jk}^{\text{out}} + a_{jk}^{\text{out}} F_{jk}^{\text{in}} \big) \cdot n_{jk} - a_{jk}^{\text{in}} a_{jk}^{\text{out}} \big(u_{jk}^{\text{out}} - u_{jk}^{\text{in}} \big) \Big] \\ + \bar{S}_j(t). \end{aligned} \tag{2.8}$$

Here

$$\bar{S}_j \approx \frac{1}{|T_j|} \int_{T_j} S(u, \vec{x}, t) d\vec{x} \tag{2.9}$$

is a discretization of the cell average of the source term that should be obtained with an appropriate quadrature. It is the discretization of (2.9) that serves as the topic for the next section.

The above derivation carries on to systems of balance laws componentwise. In this case the maximal speeds of propagation can be estimated (for convex fluxes) as

$$a_{jk}^{\text{in}}(M_{jk}) = -\min\{\lambda_1[J(u_j(M_{jk}))\cdot n_{jk}], \lambda_1[J(u_{jk}(M_{jk}))\cdot n_{jk}], 0\},$$
$$a_{jk}^{\text{out}}(M_{jk}) = \max\{\lambda_N[J(u_j(M_{jk}))\cdot n_{jk}], \lambda_N[J(u_{jk}(M_{jk}))\cdot n_{jk}], 0\},$$

where $J(u_j(M_{jk}))$ is the Jacobian of the flux $F = (f, g)$ evaluated at M_{jk}, and $\lambda_1 < \cdots < \lambda_N$ are its N eigenvalues.

3. A well-balanced scheme. In this section we present a scheme for approximating the solution of (1.1) which is balanced via a discretized average of the source terms. In section 3.1 we show that the KP scheme (2.8) cannot be balanced using a straightforward discretization of the source terms on general conformal unstructured meshes. Using the insights gained from this analysis, in section 3.2 we present a new scheme based on (2.6), which does allow such balance on general conformal triangular meshes.

3.1. Balancing the KP scheme. Our goal now is to look for a discretization of the source term (2.9) such that the scheme (2.8) will preserve stationary steady-state solutions. Hence, we assume zero velocities, $u = v = 0$, and a constant surface, i.e., $w = \text{Const}$.

Since the first equation in (1.1) is homogeneous, we start by considering the second equation. Similar analysis applies to the third equation. We are therefore looking for a discretization of the average of the source term

$$-(w - B)B_x \tag{3.1}$$

over the cell T_j. The velocities are zero and therefore the only nonzero component of the flux in (1.1) is

$$f = \frac{1}{2}(w - B)^2. \tag{3.2}$$

This means that in order for the scheme (2.8) to preserve stationary steady-state solutions, the average of the source term (3.1) over the cell T_j has to be discretized such that, for f given in (3.2),

$$0 = -\frac{1}{|T_j|}\sum_{k=1}^{N_j}\frac{h_{jk}}{a_{jk}^{\text{in}} + a_{jk}^{\text{out}}}\left[\left(a_{jk}^{\text{in}} f_{jk}^{\text{out}} + a_{jk}^{\text{out}} f_{jk}^{\text{in}}\right) n_{jk,x}\right] + \bar{S}_j. \tag{3.3}$$

Here $n_{jk,x}$ is the component of the normal n_{jk} in the x-direction.

The eigenvalues of the Jacobian of the system (1.1) are $u \pm \sqrt{w - B}$ and u, which are $\pm\sqrt{w - B}$ and 0 in the case of zero velocities. If we assume that w is constant and that the point values of B are known, the one-sided velocities satisfy $a_{jk}^{\text{in}} = a_{jk}^{\text{out}}$. Under these assumptions the condition (3.3) can be rewritten as

$$\bar{S}_j = \frac{1}{|T_j|}\sum_{k=1}^{N_j} h_{jk}\left[\left(\frac{f_{jk}^{\text{out}} + f_{jk}^{\text{in}}}{2}\right) n_{jk,x}\right] = \frac{1}{|T_j|}\sum_{k=1}^{N_j}\frac{h_{jk}}{2}\left[(w - B_{jk})^2 n_{jk,x}\right], \tag{3.4}$$

where $B_{jk} = B(M_{jk})$.

We now assume a discretization of the cell average of the source (3.1) of the form

$$\bar{S}_j = -\sum_{k=1}^{N_j} g_{jk}(w - B_{jk})\mathcal{D}, \tag{3.5}$$

where $\mathcal{D} \approx B_x$, and g_{jk} are yet to be determined. To simplify the notations we denote $m_k = h_{jk} n_{jk,x}$. It is easy to check that $\sum_{k=1}^{N_j} m_k = 0$ and hence in order for the representation (3.5) to be consistent with (3.4) we must have

$$\mathcal{D} = -\frac{1}{2|T_j|} \frac{\sum_{k=1}^{N_j} m_k B_{jk}(B_{jk} - 2w)}{\sum_{k=1}^{N_j} g_{jk}(w - B_{jk})}. \tag{3.6}$$

Since $\mathcal{D}$ in (3.6) should not be a function of w we are seeking constants a_k such that

$$\sum_{k=1}^{N} m_k B_k(B_k - 2w) = \left[\sum_{k=1}^{N} g_k(w - B_k)\right]\left[\sum_{k=1}^{N} a_k B_k\right], \tag{3.7}$$

where for simplicity we omit the obvious j-dependence from all the notations. Equation (3.7) can be rewritten as

$$\begin{cases} -2\sum_{k=1}^{N} m_k B_k = \sum_{k=1}^{N} g_k \sum_{k=1}^{N} a_k B_k, \\ \sum_{k=1}^{N} m_k B_k^2 = -\sum_{k=1}^{N} g_k B_k \sum_{k=1}^{N} a_k B_k. \end{cases} \tag{3.8}$$

The coefficients of the powers of B_k in (3.8) produce the following system of equations:

$$\begin{cases} -2m_i = a_i \sum_{k=1}^{N} g_k, & i = 1, \dots, N, \\ m_i = -a_i g_i, & i = 1, \dots, N, \\ a_i g_j = -a_j g_i, & i \neq j,\ i, j = 1, \dots, N. \end{cases} \tag{3.9}$$

Finally, from (3.9) we have

$$g_i = \frac{1}{2}\sum_{k=1}^{N} g_k, \qquad i = 1, \dots, N. \tag{3.10}$$

Equation (3.10) can generally hold only for $N = 2$, which means that one can expect to be able to balance the scheme (2.8) for stationary steady-state solutions only in cells that have two edges that contribute to the flux in the x-direction. This obviously excludes most meshes. There are two cases of special interest.

1. When the mesh is composed of triangles with one side that is parallel to the x-axis (see Figure 3.1), each cell has only two edges that contribute to the flux in the x-direction. In this case the system (3.9) can be solved. This result will enable us to introduce in section 3.2 a modification of the scheme (2.8) that can be balanced on general meshes. Due to symmetry considerations, such a mesh will not satisfy the balance conditions in the y-direction (that come from the third equation in (1.1)) unless all triangles are right triangles that are aligned with both coordinate axes.

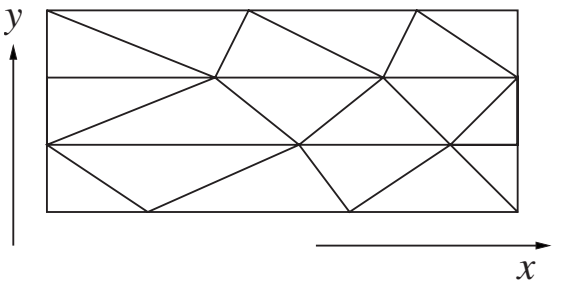


FIG. 3.1. *An admissible triangular mesh.*

2. The scheme can be balanced in both directions in the very special case of Cartesian grids. This corresponds to the case previously solved in [14]. The results of [14] when viewed from the point of view of the system (3.9) amount to the equality

$$B_1(B_1 - 2w) - B_2(B_2 - 2w) = (2w - B_1 - B_2)(B_2 - B_1).$$

If we assume a more general discretization of the cell average of (3.1) of the form

$$\bar{S}_j = -\sum_{k=1}^{N} g_k(w - B_k) \sum_{k'=1}^{N} a_{kk'} B_{k'}, \tag{3.11}$$

then the system (3.9) is replaced by (denoting $\tilde{m}_k = m_k/(2|T_j|)$)

$$\begin{cases} -2\tilde{m}_k = \sum_{k'=1}^{N} g_{k'} a_{k'k}, & k = 1, \dots, N, \\ \tilde{m}_k = -g_k a_{kk}, & k = 1, \dots, N, \\ g_k a_{kk'} = -g_{k'} a_{k'k}, & k \neq k',\ k, k' = 1, \dots, N. \end{cases} \tag{3.12}$$

The system (3.12) can be solved in the general case (unlike the system (3.9)). For example, in the case of a triangular grid ($N = 3$) one possible solution (assuming $\tilde{m}_1 \neq 0$ and $\tilde{m}_3 + 2\tilde{m}_2 \neq 0$) is

$$g = \begin{pmatrix} -\tilde{m}_1 \\ \tilde{m}_2 + \frac{\tilde{m}_3}{2} \\ \frac{\tilde{m}_3}{2} \end{pmatrix}, \qquad a = \begin{pmatrix} 1 & \frac{\tilde{m}_3 + 2\tilde{m}_2}{2\tilde{m}_1} & \frac{\tilde{m}_3}{2\tilde{m}_1} \\ 1 & -\frac{2\tilde{m}_2}{2\tilde{m}_2 + \tilde{m}_3} & -\frac{\tilde{m}_3}{2\tilde{m}_2 + \tilde{m}_3} \\ 1 & 1 & -2 \end{pmatrix}. \tag{3.13}$$

If $\tilde{m}_1 = 0$, the expression (3.13) can be replaced, e.g., by

$$g = \begin{pmatrix} 0 \\ \tilde{m}_2 + \frac{\tilde{m}_3}{2} \\ \frac{\tilde{m}_3}{2} \end{pmatrix}, \qquad a = \begin{pmatrix} 0 & \frac{\tilde{m}_3 + 2\tilde{m}_2}{\tilde{m}_3} & 1 \\ 1 & -\frac{2\tilde{m}_2}{2\tilde{m}_2 + \tilde{m}_3} & -\frac{\tilde{m}_3}{2\tilde{m}_2 + \tilde{m}_3} \\ 1 & 1 & -2 \end{pmatrix}. \tag{3.14}$$

If $\tilde{m}_3 + 2\tilde{m}_2 = 0$, a permutation of the numbering of the sides of the triangle will provide a solution. Other solutions of (3.12) exist.

While formally being able to balance the scheme with the expressions (3.13) and (3.14), it remains unclear in what sense $\sum_{k'} a_{kk'} B_{k'}$ approximates the derivative B_x. In other words, it is not obvious that the consistency of this discretization can be established. We therefore consider this approach unsatisfactory.

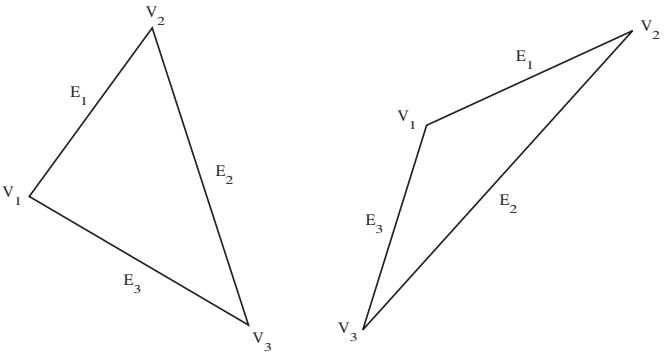


FIG. 3.2. *The two types of triangles. Left: type* 1. *Right: type* 2.

3.2. A new well-balanced scheme. The analysis in section 3.1 tells us that balancing the x-derivative flux term requires cells that have only 2 edges which have normals with nonzero x-components (see Figure 3.1). This can be accomplished by splitting our cells into subcells and solving the equation on these subcells. Similarly, the y-derivative flux term can be balanced by a different splitting of the cells. We now derive a new method based on this observation.

We focus on conformal triangular grids. Our ideas can be easily extended to other conformal unstructured meshes. In order to create a balanced scheme for (1.1), we propose to decompose every triangle T_j into several triangles as explained below. This requires a different decomposition for the two flux components. For the first component, f, we decompose T_j into two triangles, each of which has an edge parallel to the x-axis. For the second component of the flux we split T_j into two other triangles, each of which has an edge parallel to the y-axis.

We denote the vertices of T_j as V_{jk}, $k = 1, 2, 3$, and define the edges of T_j as $E_{j1} = V_{j2} - V_{j1}$, $E_{j2} = V_{j3} - V_{j2}$, and $E_{j3} = V_{j1} - V_{j3}$. The midpoint of the kth edge of T_j is M_{jk}. $I(a, b)$ denotes the closed interval with endpoints a and b. We also use $V_{jk,x}$ (or $V_{jk,y}$) to denote the x (or y) component of V_{jk}. To simplify the treatment, we classify the triangles into two categories as portrayed in Figure 3.2.

(i) Type 1: one vertex V_{j1} bisects the opposite edge in the y direction: the y-component $V_{j1,y} \in I(V_{j2,y}, V_{j3,y})$ and a different vertex V_{j2} bisects the opposite edge in the x-direction: $V_{j2,x} \in I(V_{j1,x}, V_{j3,x})$. (See Figure 3.2 (left).)

(ii) Type 2: the same vertex V_{j1} bisects the opposite edge in both the x and y directions: $V_{j1,y} \in I(V_{j2,y}, V_{j3,y})$ and $V_{j1,x} \in I(V_{j2,x}, V_{j3,x})$. (See Figure 3.2 (right).)

Note that these definitions specify our vertex numbering convention. The decomposition for the g-flux will depend on the type of triangle.

3.2.1. The decomposition in the x-direction. For the flux in the x-direction, we decompose T_j into two triangles $T_j^{x,1}$ and $T_j^{x,2}$ (see Figure 3.3). Define $V_{j3}^{x,1}$ to be the intersection of edge E_{j2} with the line $y = V_{j1,y}$. The vertices of $T_j^{x,1}$ are $V_{j1}^{x,1} := V_{j1}$, $V_{j2}^{x,1} := V_{j2}$, and $V_{j3}^{x,1}$. The vertices of $T_j^{x,2}$ are $V_{j1}^{x,2} := V_{j3}^{x,1}$, $V_{j2}^{x,2} := V_{j3}$, and $V_{j3}^{x,2} := V_{j1}$. For both triangles $T_j^{x,i}$, the lengths of the sides are $h_{jk}^{x,i}$, the corresponding midpoints are $M_{jk}^{x,i}$, the interior and exterior speeds of propagation are $a_{jk}^{\text{in,out},x,i}$, and the normals are $n_{jk}^{x,i} = (n_{jk,x}^{x,i}, n_{jk,y}^{x,i})$. These definitions imply $M_{j1}^{x,1} = M_{j1}$, $M_{j2}^{x,2} = M_{j3}$, $h_{j1}^{x,1} = h_{j1}$, $h_{j2}^{x,2} = h_{j3}$, $h_{j2}^{x,1} + h_{j1}^{x,2} = h_{j2}$, $n_{j1}^{x,1} =$

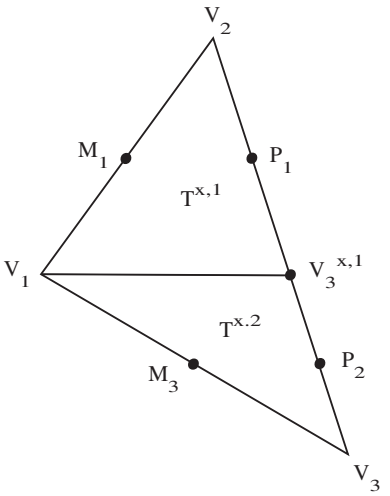


FIG. 3.3. *The triangle decomposition for the f-flux.*

$n_{j1,x}$, $n_{j2}^{x,1} = n_{j1}^{x,2} = n_{j2,x}$, $n_{j2}^{x,2} = n_{j3,x}$, and $n_{j3,x}^{x,1} = n_{j3,x}^{x,2} = 0$. For the velocities we have $a_{j1}^{\text{in,out},x,1} = a_{j1}^{\text{in,out}}$, $a_{j2}^{\text{in,out},x,1} = a_{j2}^{\text{in,out}}$, $a_{j1}^{\text{in,out},x,2} = a_{j2}^{\text{in,out}}$, and $a_{j2}^{\text{in,out},x,2} = a_{j3}^{\text{in,out}}$. The velocity $a_{j3}^{\text{in,out},x,i}$ does not play any role in the following computation since $n_{j3,x}^{x,1} = n_{j3,x}^{x,2} = 0$. Finally, the external reconstructions are denoted as $u_{j1}^{x,1} = u_{j1}$, $u_{j2}^{x,1} = u_{j1}^{x,2} = u_{j2}$, and $u_{j2}^{x,2} = u_{j3}$.

On each triangle $T_j^{x,i}$ we denote the cell average of the component of the flux in the x-direction by Φ_j^i. It is given by

$$\Phi_j^i = -\frac{1}{|T_j^{x,i}|}\sum_{k=1}^{2}\frac{h_{jk}^{x,i}n_{jk,x}^{x,i}}{a_{jk}^{\text{in},x,i}+a_{jk}^{\text{out},x,i}}\left[a_{jk}^{\text{in},x,i}f\left(u_{jk}^{x,i}\left(M_{jk}^{x,i}\right)\right)+a_{jk}^{\text{out},x,i}f\left(u_j\left(M_{jk}^{x,i}\right)\right)\right]$$

for $i = 1, 2$. The cell average of the x-component of the flux over the entire cell T_j is taken to be the weighted average

$$\Phi_j := \frac{\left|T_j^{x,1}\right|}{|T_j|}\Phi_j^1 + \frac{\left|T_j^{x,2}\right|}{|T_j|}\Phi_j^2.$$

Since $\sum_{k=1}^{3} h_{jk}n_{jk} = 0$ we have $h_{j1}^{x,1}n_{j1,x}^{x,1} = -h_{j2}^{x,1}n_{j2,x}^{x,1}$ and $h_{j2}^{x,2}n_{j2,x}^{x,2} = -h_{j1}^{x,2}n_{j1,x}^{x,2}$. Hence, Φ_j can be rewritten as

$$\begin{aligned}\Phi_j = &-\frac{h_{j1}n_{j1,x}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}f_{j1}^{\text{out}}+a_{j1}^{\text{out}}f_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}-\frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j1}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j1}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\ &-\frac{h_{j3}n_{j3,x}}{|T_j|}\left[\frac{a_{j3}^{\text{in}}f_{j3}^{\text{out}}+a_{j3}^{\text{out}}f_{j3}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}-\frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j2}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j2}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right],\end{aligned} \tag{3.15}$$

where $P_{j1} := M_{j2}^{x,1}$ and $P_{j2} := M_{j1}^{x,2}$.

We would like to note that the flux term (3.15) can be derived directly from (2.6) by changing the quadrature points on E_{j2} to P_1 and P_2.

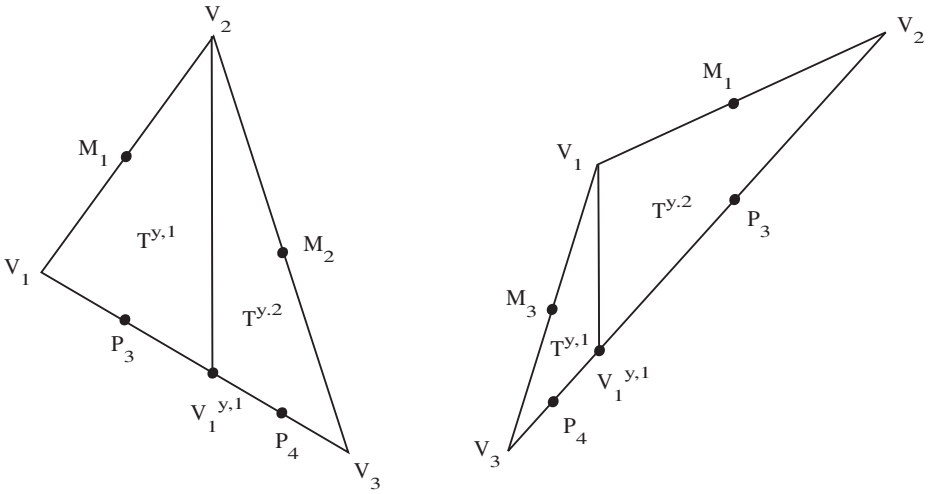


FIG. 3.4. *The triangle decomposition for the g-flux. Left: type* 1. *Right: type* 2.

3.2.2. The decomposition in the *y*-direction.

Type 1 triangles. We decompose T_j into two triangles $T_j^{y,1}$ and $T_j^{y,2}$ (see Figure 3.4 (left)). The intersection of the edge E_{j3} with the line $x = V_{j2,x}$ is denoted by $V_{j1}^{y,1}$. The vertices of $T_j^{y,1}$ are $V_{j1}^{y,1}$, $V_{j2}^{y,1} := V_{j1}$, and $V_{j3}^{y,1} := V_{j2}$. The vertices of $T_j^{y,2}$ are $V_{j1}^{y,2} := V_{j2}$, $V_{j2}^{y,2} := V_{j3}$, and $V_{j3}^{y,2} := V_{j1}^{y,1}$. As before, we have $M_{j2}^{y,1} = M_{j1}$, $M_{j1}^{y,2} = M_{j2}$, $h_{j2}^{y,1} = h_{j1}$, $h_{j1}^{y,2} = h_{j2}$, $h_{j1}^{y,1} + h_{j2}^{y,2} = h_{j3}$. The normals are $n_{j2}^{y,1} = n_{j1}$, $n_{j1}^{y,1} = n_{j2}^{y,2} = n_{j3}$, $n_{j1}^{y,2} = n_{j2}$, $n_{j3,y}^{y,1} = n_{j3,y}^{y,2} = 0$ and the velocities are $a_{j1}^{\text{in,out},y,1} = a_{j3}^{\text{in,out}}$, $a_{j2}^{\text{in,out},y,1} = a_{j1}^{\text{in,out}}$, $a_{j1}^{\text{in,out},y,2} = a_{j2}^{\text{in,out}}$, $a_{j2}^{\text{in,out},y,2} = a_{j3}^{\text{in,out}}$. The velocity $a_{j3}^{\text{in,out},y,i}$ does not appear in the result because $n_{j3,y}^{y,1} = n_{j3,y}^{y,2} = 0$. Finally, the external reconstructions are $u_{j1}^{y,1} = u_{j3}$, $u_{j2}^{y,1} = u_{j1}$, $u_{j1}^{y,2} = u_{j2}$, and $u_{j2}^{y,2} = u_{j3}$.

We denote the cell averages of the component of the flux in the y-direction on each triangle $T_j^{y,i}$ by Γ_j^i. It is given by

$$\Gamma_j^i := -\frac{1}{|T_j^{y,i}|}\sum_{k=1}^{2}\frac{h_{jk}^{y,i}n_{jk,y}^{y,i}}{a_{jk}^{\text{in},y,i}+a_{jk}^{\text{out},y,i}}\left[a_{jk}^{\text{in},y,i}g\left(u_{jk}^{y,i}\left(M_{jk}^{y,i}\right)\right)+a_{jk}^{\text{out},y,i}g\left(u_j\left(M_{jk}^{y,i}\right)\right)\right]$$

for $i = 1, 2$. The cell average of the y-component of the flux over the entire cell T_j is then given by the weighted average

$$\Gamma_j := \frac{\left|T_j^{y,1}\right|}{|T_j|}\Gamma_j^1 + \frac{\left|T_j^{y,2}\right|}{|T_j|}\Gamma_j^2.$$

Clearly $h_{j2}^{y,1}n_{j2,y}^{y,1} = -h_{j1}^{y,1}n_{j1,y}^{y,1}$ and $h_{j1}^{y,2}n_{j1,y}^{y,2} = -h_{j2}^{y,2}n_{j2,y}^{y,2}$. Therefore Γ_j can be rewritten as

$$\begin{aligned}\Gamma_j = &-\frac{h_{j1}n_{j1,y}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}g_{j1}^{\text{out}}+a_{j1}^{\text{out}}g_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}-\frac{a_{j3}^{\text{in}}g\left(u_{j3}\left(P_{j3}\right)\right)+a_{j3}^{\text{out}}g\left(u_j\left(P_{j3}\right)\right)}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}\right]\\ &-\frac{h_{j2}n_{j2,y}}{|T_j|}\left[\frac{a_{j2}^{\text{in}}g_{j2}^{\text{out}}+a_{j2}^{\text{out}}g_{j2}^{\text{in}}}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}-\frac{a_{j3}^{\text{in}}g\left(u_{j3}\left(P_{j4}\right)\right)+a_{j3}^{\text{out}}g\left(u_j\left(P_{j4}\right)\right)}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}\right],\end{aligned} \tag{3.16}$$

where $P_{j3} := M_{j1}^{y,1}$ and $P_{j4} := M_{j2}^{y,2}$.

Type 2 triangles. This case corresponds to Figure 3.4 (right). Analogous computations to those for type 1 triangles provide the cell average of the y-component of the flux over the cell T_j, which this time is given by

$$\Gamma_j = -\frac{h_{j1}n_{j1,y}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}g_{j1}^{\text{out}}+a_{j1}^{\text{out}}g_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}-\frac{a_{j2}^{\text{in}}g\left(u_{j2}\left(P_{j3}\right)\right)+a_{j2}^{\text{out}}g\left(u_j\left(P_{j3}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right] \tag{3.17}$$
$$-\frac{h_{j3}n_{j3,y}}{|T_j|}\left[\frac{a_{j3}^{\text{in}}g_{j3}^{\text{out}}+a_{j3}^{\text{out}}g_{j3}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}-\frac{a_{j2}^{\text{in}}g\left(u_{j2}\left(P_{j4}\right)\right)+a_{j2}^{\text{out}}g\left(u_j\left(P_{j4}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right],$$

where P_{j3} and P_{j4} are given in Figure 3.4 (right).

3.2.3. The new method (for conservation laws). We would now like to combine all the different ingredients that we developed in the previous section into one scheme. The scheme that we write here is still a scheme for approximating solutions of the conservation law (2.2) without the source term. Based on our preliminary analysis of section 3.1 we know that we will be able to find an admissible discretization of the source terms that will result with a well-balanced scheme. We will treat the source terms in the next section.

We write two versions of the scheme based on the type of triangle. For type 1 triangles, the discretization of the component of the flux in the x-direction, Φ_j is given by (3.15) while the discretization of the component of the flux in the y-direction, Γ_j, is given by (3.16). In this case the scheme takes the form

$$\begin{aligned}\frac{d\bar{u}_j}{dt} &= -\frac{h_{j1}n_{j1,x}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}f_{j1}^{\text{out}}+a_{j1}^{\text{out}}f_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}-\frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j1}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j1}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\ &\quad -\frac{h_{j3}n_{j3,x}}{|T_j|}\left[\frac{a_{j3}^{\text{in}}f_{j3}^{\text{out}}+a_{j3}^{\text{out}}f_{j3}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}-\frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j2}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j2}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\ &\quad -\frac{h_{j1}n_{j1,y}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}g_{j1}^{\text{out}}+a_{j1}^{\text{out}}g_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}-\frac{a_{j3}^{\text{in}}g\left(u_{j3}\left(P_{j3}\right)\right)+a_{j3}^{\text{out}}g\left(u_j\left(P_{j3}\right)\right)}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}\right]\\ &\quad -\frac{h_{j2}n_{j2,y}}{|T_j|}\left[\frac{a_{j2}^{\text{in}}g_{j2}^{\text{out}}+a_{j2}^{\text{out}}g_{j2}^{\text{in}}}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}-\frac{a_{j3}^{\text{in}}g\left(u_{j3}\left(P_{j4}\right)\right)+a_{j3}^{\text{in}}g\left(u_j\left(P_{j4}\right)\right)}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}\right]\\ &\quad +\frac{1}{|T_j|}\sum_{k=1}^{3}h_{jk}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\left[u_{jk}^{\text{out}}-u_{jk}^{\text{in}}\right].\end{aligned} \tag{3.18}$$

Here $P_1 = V_2 - \frac{1}{2}\frac{E_{1,y}}{E_{2,y}}E_2$, $P_2 = V_3 + \frac{1}{2}\frac{E_{3,y}}{E_{2,y}}E_2$, $P_3 = V_1 + \frac{1}{2}\frac{E_{1,x}}{E_{3,x}}E_3$, and $P_4 = V_3 - \frac{1}{2}\frac{E_{2,x}}{E_{3,x}}E_3$.

For type 2 triangles we replace the discretization of Γ_j by the one given in (3.17)

ending with

$$
\begin{aligned}
\frac{d\bar{u}_j}{dt} = &-\frac{h_{j1}n_{j1,x}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}f_{j1}^{\text{out}}+a_{j1}^{\text{out}}f_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}} - \frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j1}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j1}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\
&-\frac{h_{j3}n_{j3,x}}{|T_j|}\left[\frac{a_{j3}^{\text{in}}f_{j3}^{\text{out}}+a_{j3}^{\text{out}}f_{j3}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}} - \frac{a_{j2}^{\text{in}}f\left(u_{j2}\left(P_{j2}\right)\right)+a_{j2}^{\text{out}}f\left(u_j\left(P_{j2}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\
&-\frac{h_{j1}n_{j1,y}}{|T_j|}\left[\frac{a_{j1}^{\text{in}}g_{j1}^{\text{out}}+a_{j1}^{\text{out}}g_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}} - \frac{a_{j2}^{\text{in}}g\left(u_{j2}\left(P_{j3}\right)\right)+a_{j2}^{\text{out}}g\left(u_j\left(P_{j3}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\
&-\frac{h_{j3}n_{j3,y}}{|T_j|}\left[\frac{a_{j2}^{\text{in}}g_{j3}^{\text{out}}+a_{j2}^{\text{out}}g_{j3}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}} - \frac{a_{j2}^{\text{in}}g\left(u_{j2}\left(P_{j4}\right)\right)+a_{j2}^{\text{in}}g\left(u_j\left(P_{j4}\right)\right)}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}\right]\\
&+\frac{1}{|T_j|}\sum_{k=1}^{3}h_{jk}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\left[u_{jk}^{\text{out}}-u_{jk}^{\text{in}}\right].
\end{aligned}
\tag{3.19}
$$

In this case $P_1 = V_2 - \frac{1}{2}\frac{E_{1,y}}{E_{2,y}}E_2$, $P_2 = V_3 + \frac{1}{2}\frac{E_{3,y}}{E_{2,y}}E_2$, $P_3 = V_2 - \frac{1}{2}\frac{E_{1,x}}{E_{2,x}}E_2$, and $P_4 = V_3 + \frac{1}{2}\frac{E_{3,x}}{E_{2,x}}E_2$.

1. For a general triangulation our method will not be conservative because the fluxes from adjacent triangles will not in general be evaluated at the same point on the edge. Our tests with irregular triangulations in section 4 show, however, that this failure of conservation does not lead to serious problems.

2. A simple case of interest is that of coordinate-aligned right triangles. Such triangles can be considered to be of type 1 with $n_{j2,y} = n_{j3,x} = 0$ and $P_{j1} = M_{j2}$ and $P_{j3} = M_{j3}$, which means that the method becomes (taking into account that $\sum_{k=1}^{3} h_{jk}n_{jk} = 0$)

$$
\begin{aligned}
\frac{d\bar{u}_j}{dt} = &-\frac{1}{|T_j|}\left[\frac{a_{j1}^{\text{in}}f_{j1}^{\text{out}}+a_{j1}^{\text{out}}f_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}h_{j1}n_{j1,x} + \frac{a_{j2}^{\text{in}}f_{j2}^{\text{out}}+a_{j2}^{\text{out}}f_{j1}^{\text{in}}}{a_{j2}^{\text{in}}+a_{j2}^{\text{out}}}h_{j2}n_{j2,x}\right]\\
&-\frac{1}{|T_j|}\left[\frac{a_{j1}^{\text{in}}g_{j1}^{\text{out}}+a_{j1}^{\text{out}}g_{j1}^{\text{in}}}{a_{j1}^{\text{in}}+a_{j1}^{\text{out}}}h_{j1}n_{j1,y} + \frac{a_{j3}^{\text{in}}g_{j3}^{\text{out}}+a_{j3}^{\text{out}}g_{j1}^{\text{in}}}{a_{j3}^{\text{in}}+a_{j3}^{\text{out}}}h_{j3}n_{j3,y}\right]\\
&+\frac{1}{|T_j|}\sum_{k=1}^{3}h_{jk}\frac{a_{jk}^{\text{in}}a_{jk}^{\text{out}}}{a_{jk}^{\text{in}}+a_{jk}^{\text{out}}}\left[u_{jk}^{\text{out}}-u_{jk}^{\text{in}}\right].
\end{aligned}
\tag{3.20}
$$

As expected, in this case the method (3.20) coincides with the KP method (2.7).

3.2.4. Adding the source term. We return to the shallow water equations (1.1). Our goal is now to find an admissible discretization of the cell average of the source terms that will preserve stationary steady states. Such a term will be added to the RHS of (3.18) or (3.19) depending on the type of the triangle. In the following we assume type 1 triangles. Similar analysis holds for type 2 triangles. We also note that the first equation in the system (1.1) is homogeneous. This means that we only have to consider the remaining two equations when dealing with the source terms. We recall from section 3.1 that in stationary steady states all the velocities are equal, $a_{jk}^{\text{in}} = a_{jk}^{\text{out}} = a_{jk}$. Our new method (3.18) for the last two equations of (1.1) then

becomes

(3.21)
$$\begin{aligned}
0 = &-\frac{h_{j1}n_{j1,x}}{2\left|T_j\right|}\left[\begin{pmatrix}\left(w\left(M_{j1}\right)-B\left(M_{j1}\right)\right)^2\\0\end{pmatrix}-\begin{pmatrix}\left(w\left(P_{j1}\right)-B\left(P_{j1}\right)\right)^2\\0\end{pmatrix}\right]\\
&-\frac{h_{j3}n_{j3,x}}{2\left|T_j\right|}\left[\begin{pmatrix}\left(w\left(M_{j3}\right)-B\left(M_{j3}\right)\right)^2\\0\end{pmatrix}-\begin{pmatrix}\left(w\left(P_{j2}\right)-B\left(P_{j2}\right)\right)^2\\0\end{pmatrix}\right]\\
&-\frac{h_{j1}n_{j1,y}}{2\left|T_j\right|}\left[\begin{pmatrix}0\\\left(w\left(M_{j1}\right)-B\left(M_{j1}\right)\right)^2\end{pmatrix}-\begin{pmatrix}0\\\left(w\left(P_{j3}\right)-B\left(P_{j3}\right)\right)^2\end{pmatrix}\right]\\
&-\frac{h_{j2}n_{j2,y}}{2\left|T_j\right|}\left[\begin{pmatrix}0\\\left(w\left(M_{j2}\right)-B\left(M_{j2}\right)\right)^2\end{pmatrix}-\begin{pmatrix}0\\\left(w\left(P_{j4}\right)-B\left(P_{j4}\right)\right)^2\end{pmatrix}\right]+\begin{pmatrix}\bar{S}_j^1\\\bar{S}_j^2\end{pmatrix}.
\end{aligned}$$

The last term in (3.21) represents the average of the source, i.e., $\bar{S}_j^1 = \text{avg}(-(w - B)B_x)$ and $\bar{S}_j^2 = \text{avg}(-(w - B)B_y)$, where both averages are taken over T_j.

We use (3.21) to determine the admissible discretizations of the cell averages of the source terms. For constant w the source terms (given by (3.21)) can be rewritten as

$$\begin{aligned}
\bar{S}_j^1 = &-\frac{h_{j1}n_{j1,x}}{2|T_j|}[w_{M_{j1}} - B_{M_{j1}} + w_{P_{j1}} - B_{P_{j1}}][B_{M_{j1}} - B_{P_{j1}}]\\
&-\frac{h_{j3}n_{j3,x}}{2|T_j|}[w_{M_{j3}} - B_{M_{j3}} + w_{P_{j2}} - B_{P_{j2}}][B_{M_{j3}} - B_{P_{j2}}],\\
\bar{S}_j^2 = &-\frac{h_{j1}n_{j1,y}}{2|T_j|}[w_{M_{j1}} - B_{M_{j1}} + w_{P_{j3}} - B_{P_{j3}}][B_{M_{j1}} - B_{P_{j3}}]\\
&-\frac{h_{j2}n_{j2,y}}{2|T_j|}[w_{M_{j2}} - B_{M_{j2}} + w_{P_{j4}} - B_{P_{j4}}][B_{M_{j2}} - B_{P_{j4}}].
\end{aligned} \tag{3.22}$$

Here, we use the notation $w_{M_1} := w\left(M_{j1}\right)$, etc. We take the expressions in (3.22) as the discretization of the source even when w is not constant. Similar expressions can be easily written for type 2 triangles.

LEMMA 3.1. [illegible] $\bar{S}_j^i$ [illegible] (3.22) [illegible] (1.1) [illegible] (3.21).

Before proving Lemma 3.1 we consider the following geometrical lemma.

LEMMA 3.2. [illegible] M [illegible] P [illegible] 3.2.1 [illegible] j

$$M_{1,x} - P_{1,x} = -\frac{|T|}{h_2 n_{2,x}}, \quad M_{3,x} - P_{2,x} = -\frac{|T|}{h_2 n_{2,x}}.$$

[illegible] $M_1 = \frac{1}{2}\left(V_1 + V_2\right)$, and for some s, $P_1 = V_2 + s\left(V_3 - V_2\right)$. We require that $P_{1,y} = M_{1,y}$, which implies that

$$s = \frac{1}{2}\frac{V_{1,y} - V_{2,y}}{V_{3,y} - V_{2,y}}.$$

Therefore

$$M_{1,x} - P_{1,x} = \frac{1}{2}\left(V_{1,x} + V_{2,x}\right) - V_{2,x} - \frac{1}{2}\frac{V_{1,y} - V_{2,y}}{V_{3,y} - V_{2,y}}\left(V_{3,x} - V_{2,x}\right) = -\frac{1}{2}\frac{E_1 \times E_2}{E_{2,y}}.$$

When the orientation of the vertices of T_j is clockwise, $E_1 \times E_2 = -2|T|$ and $n_2 = \frac{1}{h_2}(-E_{2,y}, E_{2,x})$, while when the orientation is counterclockwise, $E_1 \times E_2 = 2|T|$ and $n_2 = \frac{1}{h_2}(E_{2,y}, -E_{2,x})$, so

$$M_{1,x} - P_{1,x} = -\frac{|T|}{h_2 n_{2,x}}.$$

Similar arguments hold for $M_{3,x} - P_{2,x}$. □

3.1. We show that $\bar{S}_j^1 \approx \text{avg}(-(w-B)B_x)$. To simplify the notations we suppress the index j. From Lemma 3.2 we know that $M_{1,x} - P_{1,x} = -\frac{|T|}{h_2 n_{2,x}}$. Since $M_{1,y} = P_{1,y}$ we have $\frac{B_{M_1} - B_{P_1}}{M_{1,x} - P_{1,x}} = B_x + O(|M_1 - P_1|^2)$ at the midpoint between M_1 and P_1. Hence the first part of $\bar{S}_j^1$ in (3.22) becomes

$$\begin{aligned}\frac{h_1 n_{1,x}}{2h_2 n_{2,x}}(w_{M_1} - B_{M_1} + w_{P_1} - B_{P_1})\,\frac{B_{M_1} - B_{P_1}}{M_{1,x} - P_{1,x}} \\ \approx \frac{h_1 n_{1,x}}{2h_2 n_{2,x}}\,(w_{M_1} - B_{M_1} + w_{P_1} - B_{P_1})\,B_x.\end{aligned}$$

Applying a similar argument to the second term of $\bar{S}_j^1$ in (3.22) gives

$$\begin{aligned}\bar{S}_j^1 \approx \frac{1}{2}\Bigg(&\frac{h_1 n_{1,x}}{h_2 n_{2,x}}\,(w_{M_1} - B_{M_1} + w_{P_1} - B_{P_1}) \\ &+ \frac{h_3 n_{3,x}}{h_2 n_{2,x}}\,(w_{M_3} - B_{M_3} + w_{P_2} - B_{P_2})\Bigg) B_x.\end{aligned} \tag{3.23}$$

Clearly, the coefficient of B_x in (3.23) is a discretization of a weighted average of $-(w-B)$. For example, when $w - B$ is constant we have

$$\bar{S}_j^1 \approx \frac{h_1 n_{1,x} + h_3 n_{3,x}}{h_2 n_{2,x}}\,(w-B)\,B_x = -\,(w-B)\,B_x.$$

Similar arguments hold for $\bar{S}_j^2$. □

If we consider a Cartesian mesh split into triangles along a diagonal, it is easy to see that our method reduces to the method in [14].

4. Numerical examples. The scheme developed in section 3.2 did not assume any particular reconstruction. There are several different second-order reconstructions on triangular meshes that are being used in the literature (see [16] and the references therein). We briefly describe the one we used in our simulations.

The starting point is the limited least-squares estimate of the gradients as done in [2]. The first step is to compute a least-squares estimate of the gradient of a field f on the triangle T_j, $\tilde{\nabla}_j f$. We then limit the gradient $\mathcal{D}_j f$ component by component as

$$\mathcal{D}_j f = \mathcal{MM}\left(\tilde{\nabla}_j f, \tilde{\nabla}_{j1} f, \tilde{\nabla}_{j2} f, \tilde{\nabla}_{j3} f\right),$$

where $\tilde{\nabla}_{jk} f$ is the least-squares gradient estimate on T_{jk} and $\mathcal{MM}$ stands for the usual MinMod limiter

$$\mathcal{MM}(x_1, x_2, \dots) := \begin{cases} \min_j \{x_j\} & \text{if } x_j > 0 \quad \forall j, \\ \max_j \{x_j\} & \text{if } x_j < 0 \quad \forall j, \\ 0 & \text{otherwise.} \end{cases}$$

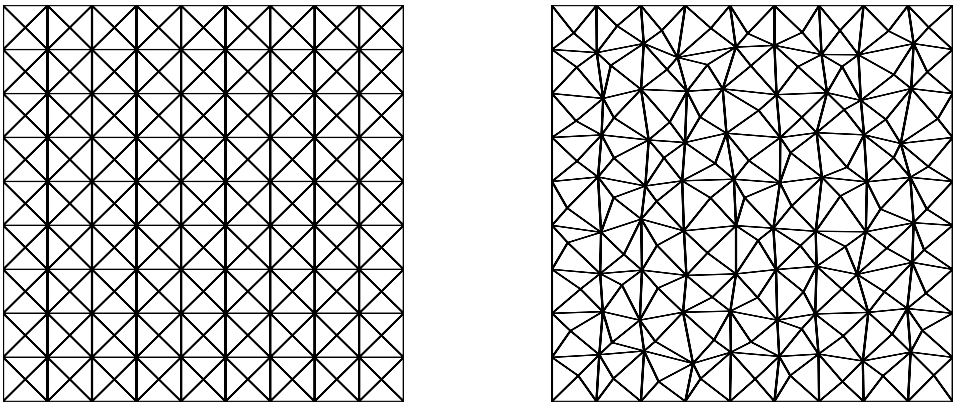


FIG. 4.1. *The two types of triangle meshes. Left: Uniform triangulation. Right: Perturbed (nonuniform) triangulation.*

We use the gradients $\mathcal{D}_j$ to construct a piecewise linear reconstruction for the point values of each triangle edge E_{jk} as

$$u_j(\vec{x}) = \bar{u}_j + \mathcal{MM}\left(\mathcal{D}_j u \cdot (\vec{x} - \vec{x}_j), \mathcal{D}_{jk} u \cdot (\vec{x} - \vec{x}_j)\right).$$

Here $\mathcal{D}_{jk}u$ is the limited gradient estimate on T_{jk}, $\vec{x}_j$ is the center of T_j, and $\vec{x} \in E_{jk}$. We find that this double use of the MinMod limiter minimizes spurious oscillations while preserving the second-order accuracy of the reconstruction.

We use an adaptive time step given by

$$\Delta t = 0.9 \min_j \frac{r_j}{|\lambda_N^j|},$$

where r_j is the radius of T_j and $|\lambda_N^j|$ is the norm of the largest eigenvalues of the Jacobian of the flux on T_j.

We use two types of meshes in most of our simulations. The first is a uniform triangulation of a Cartesian mesh with $N \times N$ nodes and constant spacings Δx and Δy, which divides each Cartesian cell into four triangles. To test our method on more general meshes, our second type of mesh is generated from a uniform triangular mesh with a perturbation of the coordinates of the interior vertices. Examples of both meshes are shown in Figure 4.1. The only exception is Example 6, which uses a uniform triangulation of a warped Cartesian mesh.

Example 1: Accuracy tests for a two-dimensional Burgers equation. We test the accuracy of our method (3.18)–(3.19) on nonlinear problems by applying it to the two-dimensional Burgers equation

$$\begin{cases} u_t + \frac{1}{2}\left(u^2\right)_x + \frac{1}{2}\left(u^2\right)_y = 0, & [-2\pi, 2\pi]^2, \\ u(x, y, t = 0) = \sin\left(\frac{x+y}{2}\right), \end{cases} \tag{4.1}$$

with periodic boundary conditions. The relative L^1-error (i.e., the L^1-error divided by the L^1-norm of the exact solution) of our method at $T = 0.5$ is shown in Table 4.1 for uniform and perturbed triangulations. The computed cell averages after singularity formation at $T = 1.5$ are shown in Figure 4.2. Note the sharp shocks that are captured by our method.

TABLE 4.1
L^1-error and convergence rates for Burgers equation (4.1) at $T = 0.5$ on a uniform and a perturbed triangulation of an $N \times N$ Cartesian mesh.

	Uniform triangulation		Perturbed triangulation	
N	Relative L^1-error	L^1-order	Relative L^1-error	L^1-order
10	0.064	—	0.064	—
20	0.014	2.20	0.014	2.20
40	4.68×10^{-3}	1.58	4.65×10^{-3}	1.59
80	1.37×10^{-3}	1.77	1.36×10^{-3}	1.77

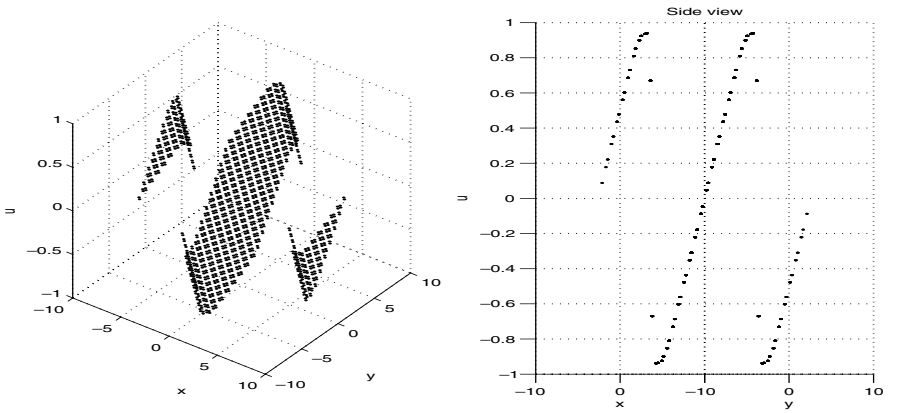


FIG. 4.2. *Burgers equation at $T = 1.5$ on a uniform triangulation of a 20×20 Cartesian mesh. Left: oblique view. Right: side view.*

Example 2: The Sod problem. To test our method (3.18)–(3.19) on a system of equations we apply it to the Sod problem for the Euler equations

$$\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ E \end{pmatrix}_t + \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u\,(E+p) \end{pmatrix}_x + \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v\,(E+p) \end{pmatrix}_y = 0. \tag{4.2}$$

Here ρ is the density, (u, v) is the velocity, and E is the energy. The equation of state for the pressure is $p = (\gamma - 1)\left[E - \frac{\rho}{2}\left(u^2 + v^2\right)\right]$. We set $\gamma = 1.4$ and take the initial conditions (constant in the y-direction)

$$(p, \rho, u, v) = \begin{cases} (1.0, 1.0, 0.0, 0.0)\,, & x < 0.5, \\ (0.1, 0.125, 0.0, 0.0)\,, & x > 0.5. \end{cases} \tag{4.3}$$

Figure 4.3 shows the computed cell averages at $T = 0.16$ using our method from section 3.2 projected onto the $y = 0$ plane. A reference one-dimensional solution is also shown. The two-dimensional problem is run on both a uniform triangulation based on a 300×30 Cartesian mesh on the domain $[0, 1] \times [0, 0.1]$ as well as a perturbation of that mesh. The one-dimensional reference solution is computed using the second-order central method of [15] with 3000 points in the domain $[0, 1]$. As expected, the quality of the solution slightly degrades on the nonuniform mesh.

Example 3: A balance test. In this example we demonstrate the well-balanced property of our scheme (3.18)–(3.19) with the source discretization (3.22).

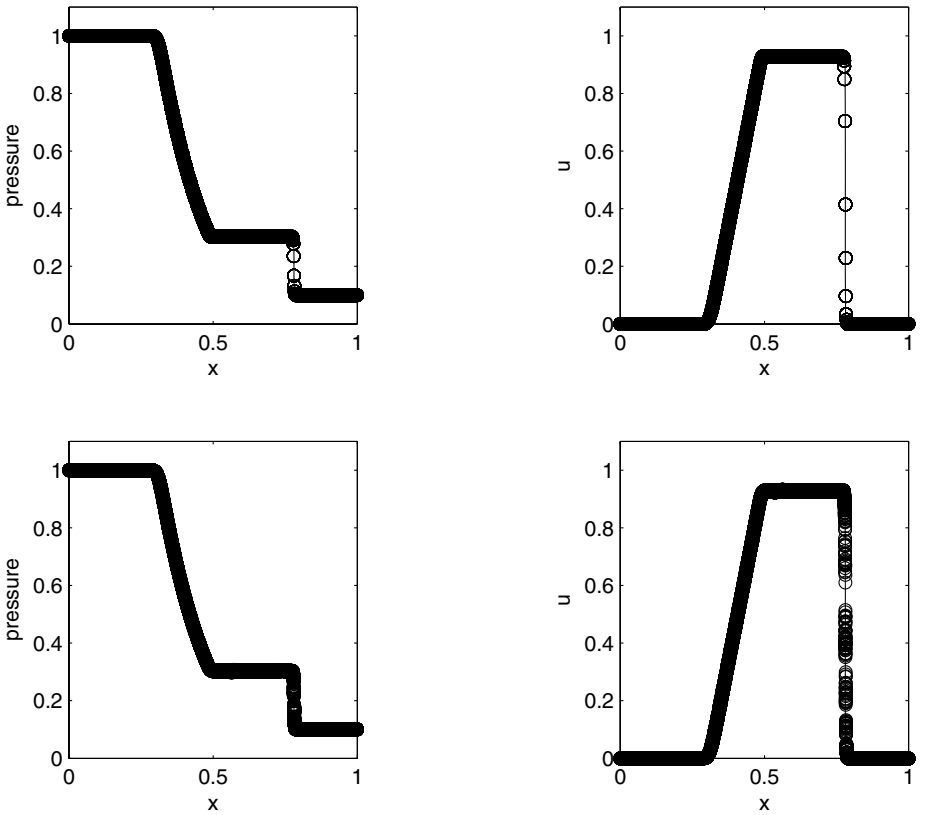


FIG. 4.3. *The pressure (left) and u-velocity (right) fields of the Sod problem at* $T = 0.16$ *on a uniform triangulation (top) and a perturbed triangulation (bottom). The circles show an orthogonal projection onto the* $y = 0$ *plane of the solution of the two-dimensional problem on a triangular mesh. The line shows a one-dimensional reference solution computed with the second-order central-upwind method of* [15].

As a simple test of balance, we consider the shallow water problem with initial conditions that represent a stationary steady state. We choose $w(x, y, t = 0) = 2$, $u(x, y, t = 0) = v(x, y, t = 0) = 0$, and a bottom topography given by $B(x, y) = \sin(2\pi x) + \cos(2\pi y)$ on the domain $[0, 1]^2$. We assume periodic boundary conditions. This initial value problem has the trivial stationary steady-state solution $w(x, y, t) = 2$, $u(x, y, t) = v(x, y, t) = 0$ for all t.

The relative L^1-errors at $T = 1$ on both uniform and perturbed triangulations based on a 10×10 Cartesian mesh are given in Table 4.2. This table also shows the result found using the KP method [16] with the same source discretization (3.22) which is not balanced for this scheme. We see that our method balances to machine accuracy.

Example 4: Propagating waves with a bottom topography. We apply our method to a test problem from [18] of a small perturbation of a steady-state problem on the domain $[0, 2] \times [0, 1]$ with periodic boundary conditions in the y-direction. The bottom topography is the elliptical Gaussian mound given by

$$B(x, y) = 0.8 \exp\left(-5(x - 0.9)^2 - 50(y - 0.5)^2\right),$$

TABLE 4.2
The relative L^1-error in the balance for (3.18), (3.19), (3.22) *and for the KP scheme with the source discretization* (3.22).

Method	Uniform triangulation	Perturbed triangulation
(3.18)–(3.19)–(3.22)	3.5×10^{-17}	6.9×10^{-19}
[16] + (3.22)	3.0×10^{-3}	3.5×10^{-3}

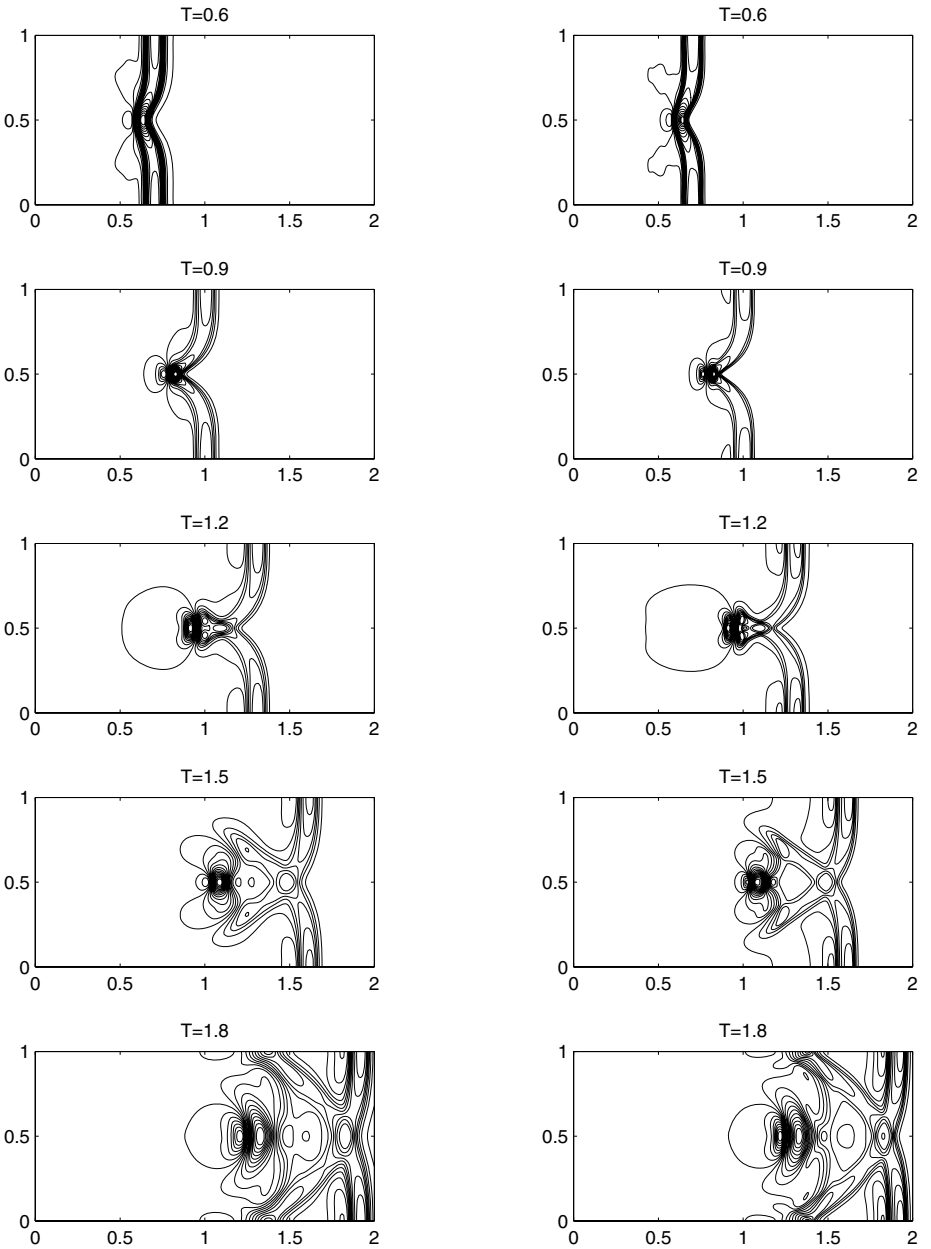


FIG. 4.4. *Wave propagation over an elliptical hump at various times. Left: uniform triangulation based on a* 200×100 *mesh. Right: uniform triangulation based on a* 400×200 *mesh.*

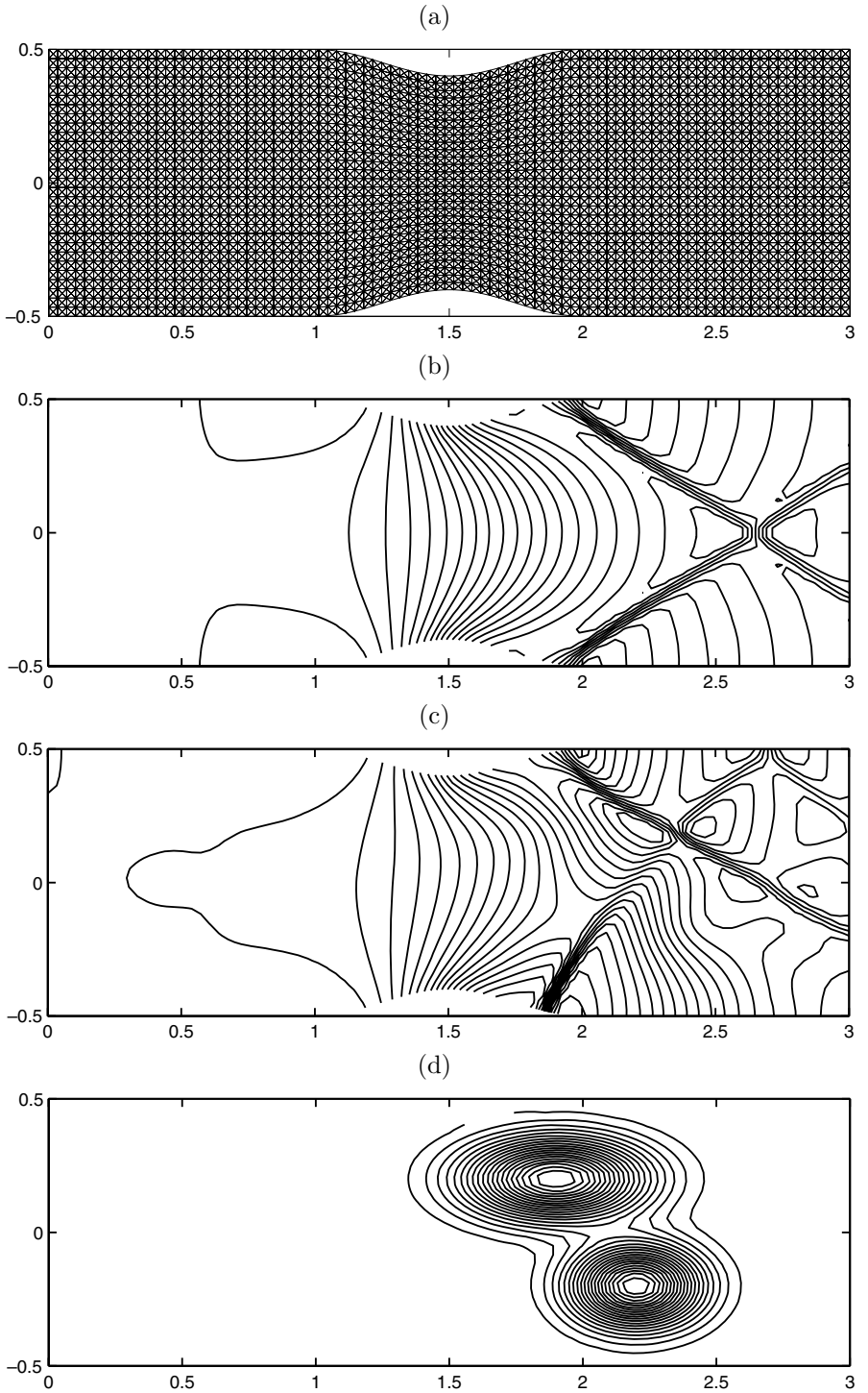


FIG. 4.5. *Example* 6: (a) *The mesh.* (b) *Contours of* w *for a critical flow through a converging-diverging channel with flat bottom at* $T = 7$. (c) *Contours of* w *for a critical flow through a converging-diverging channel with the topography shown below at* $T = 7$. (d) *The bottom topography for figure* (c).

and the initial conditions are

$$(w,u,v) = \begin{cases} (1.01, 0.0, 0.0) & \text{if } \ 0.05 < x < 0.15, \\ (1.0, 0.0, 0.0) & \text{otherwise.} \end{cases}$$

Figure 4.4 shows the result of our method at various times on a uniform triangulation of both a 200×100 and 400×200 Cartesian mesh. These results are in good agreement with other methods on Cartesian grids (see [14, 18]).

Example 5: Converging-diverging channel with bottom topography. Our final example is that of a converging-diverging channel with critical flow adapted from [12]. The channel is defined on the domain $[0,3] \times [-0.5, 0.5]$ with a half-cosine constriction centered at $x = 1.5$. The mesh for this example is shown in Figure 4.5(a). It is a nonuniform triangulation obtained by warping a uniform triangulation of a Cartesian mesh under the mapping $(x,y) \to \left(x, \left(1 - 0.2\cos^2\left(\pi\left(x - 1.5\right)\right)\right) y\right)$ when $|x - 1.5| < 0.5$. The initial data is $w = 1, u = v = 0$. The y-boundaries are reflecting. The left x-boundary is an inflow boundary with $u = 5.0$ and the right x-boundary is a zeroth-order outflow boundary. We run the simulations on a 90×30 mesh until $T = 7$ after the steady state is achieved. We would like to emphasize that this is not a stationary steady state.

We first present this example with a flat bottom, with contours of w shown in Figure 4.5(b). Figure 4.5(c) shows the same channel at the same time with bottom topography

$$B(x,y) = 0.8\left(\exp\left(-10\left(x - 1.9\right)^2 - 50\left(y - 0.2\right)^2\right) + \exp\left(-20\left(x - 2.2\right)^2 - 50\left(y + 0.2\right)^2\right)\right).$$

This topography is shown in Figure 4.5(d) and represents two elliptical Gaussian mounds just downflow from the constriction.

We choose the initial and boundary conditions such that stationary shocks are generated. The symmetry of the flat-bottom case (Figure 4.5(b)) is broken by the nonsymmetric bottom (Figures 4.5(c)–(d)). These results are in qualitative agreement with those that are shown in [12]. We note that the simulations in [12] correspond only to the flat bottom case.

5. Conclusion. In this paper we presented a well-balanced semidiscrete central scheme on triangular meshes for shallow water equations with bottom topography. This scheme is derived by changing the quadrature rules for the flux integrals that were proposed in [16]. A careful discretization of the source term then allows us to preserve stationary steady states.

The numerical flux is complemented by a new second-order reconstruction on triangular meshes. The accuracy and well-balanced properties of the scheme are demonstrated in a variety of numerical simulations.

Several generalizations of this scheme are left for future research. This includes dry beds and dissipative systems.

Acknowledgments. We would like to thank Jonathan Goodman and Benoit Perthame for helpful discussions.

REFERENCES

[1] P. Arminjon and M.-C. Viallon, *Généralisation du schéma de Nessyahu-Tadmor pour une équation hyperbolique à deux dimensions d'espace*, C. R. Acad. Sci. Paris Sér. I Math., 320 (1995), pp. 85–88.

[2] P. Arminjon, M.-C. Viallon, and A. Madrane, *A finite volume extension of the Lax-Friedrichs and Nessyahu-Tadmor schemes for conservation laws on unstructured grids*, Int. J. Comput. Fluid Dyn., 9 (1997), pp. 1–22.

[3] E. Audusse and M.-O. Bristeau, *Transport of pollutant in shallow water: A two time steps kinetic method*, M2AN Math. Model. Numer. Anal., 37 (2003), pp. 389–416.

[4] E. Audusse, M.-O. Bristeau, and B. Perthame, *Kinetic Schemes for Saint-Venant Equations with Source Terms on Unstructured Grids*, INRIA report RR-3989, 2000.

[5] R. Botchorishvili, B. Perthame, and A. Vasseur, *Equilibrium schemes for scalar conservation laws with stiff sources*, Math. Comp., 72 (2003), pp. 131–157.

[6] R. Botchorishvili and O. Pironneau, *Finite volume schemes with equilibrium-type discretization of source terms for scalar conservation laws*, J. Comput. Phys., 187 (2003), pp. 391–427.

[7] A. I. Delis and Th. Katsaounis, *Relaxation schemes for the shallow water equations*, Internat. J. Numer. Methods Fluids, 41 (2003), pp. 695–719.

[8] K. O. Friedrichs and P. D. Lax, *Systems of conservation equations with a convex extension*, Proc. Natl. Acad. Sci. USA, 68 (1971), pp. 1686–1688.

[9] T. Gallouët, J.-M. Hérard, and N. Seguin, *Some approximate Godunov schemes to compute shallow-water equations with topography*, Comput. & Fluids., 32 (2003), pp. 479–513.

[10] J. F. Gerbeau and B. Perthame, *Derivation of viscous Saint-Venant system for laminar shallow water; numerical validation*, Discrete Contin. Dyn. Syst. Ser. B, 1 (2001), pp. 89–102.

[11] L. Gosse, *A well-balanced scheme using non-conservative products designed for hyperbolic systems of conservation laws with source terms*, Math. Models Methods Appl. Sci., 11 (2001), pp. 339–365.

[12] M. E. Hubbard, *On the accuracy of one-dimensional models of steady converging/diverging open channel flows*, Internat. J. Numer. Methods Fluids, 35 (2001), pp. 785–808.

[13] G.-S. Jiang and E. Tadmor, *Nonoscillatory central schemes for multidimensional hyperbolic conservation laws*, SIAM J. Sci. Comput., 19 (1998), pp. 1892–1917.

[14] A. Kurganov and D. Levy, *Central-upwind schemes for the Saint-Venant system*, M2AN Math. Model. Numer. Anal., 36 (2002), pp. 397–425.

[15] A. Kurganov, S. Noelle, and G. Petrova, *Semidiscrete central-upwind schemes for hyperbolic conservation laws and Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 23 (2001), pp. 707–740.

[16] A. Kurganov and G. Petrova, *Central-upwind schemes on triangular grids for hyperbolic systems of conservation laws*, Numer. Methods Partial Differential Equations, 21 (2005), pp. 536–552.

[17] A. Kurganov and E. Tadmor, *New high-resolution central schemes for nonlinear conservation laws and convection-diffusion equations*, J. Comput. Phys., 160 (2000), pp. 214–282.

[18] R. J. LeVeque, *Balancing source terms and flux gradients in high-resolution Godunov methods: The quasi-steady wave-propagation algorithm*, J. Comput. Phys., 146 (1998), pp. 346–365.

[19] R. J. LeVeque and D. S. Bale, *Wave propagation methods for conservation laws with source terms*, in Hyperbolic Problems: Theory, Numerics, Applications, Vol. II (Zürich, 1998), Internat. Ser. Numer. Math. 130, Birkhäuser, Basel, 1999, pp. 609–618.

[20] S. F. Liotta, V. Romano, and G. Russo, *Central schemes for systems of balance laws*, in Hyperbolic Problems: Theory, Numerics, Applications, Vol. II (Zürich, 1998), Internat. Ser. Numer. Math. 130, Birkhäuser, Basel, 1999, pp. 651–660.

[21] H. Nessyahu and E. Tadmor, *Non-oscillatory central differencing for hyperbolic conservation laws*, J. Comput. Phys., 87 (1990), pp. 408–463.

[22] B. Perthame and C. Simeoni, *A kinetic scheme for the Saint-Venant system with a source term*, Calcolo, 38 (2001), pp. 201–231.

[23] G. Russo, *Central schemes for balance laws*, in Hyperbolic Problems: Theory, Numerics, Applications, Vol. I, II (Magdeburg, 2000), Internat. Ser. Numer. Math. 140, 141, Birkhäuser, Basel, 2001, pp. 821–829.

[24] A. J. C. de Saint-Venant, *Théorie du mouvement non-permanent des eaux, avec application aux crues des rivière at à l'introduction des marées dans leur lit*, C. R. Acad. Sci. Paris, 73 (1871), pp. 147–154.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 553–574
© 2005 Society for Industrial and Applied Mathematics

A FAST ALGORITHM FOR THE ELECTROMAGNETIC SCATTERING FROM A LARGE CAVITY*

GANG BAO† AND WEIWEI SUN‡

Abstract. A fast algorithm is presented for solving electromagnetic scattering from a rectangular open cavity embedded in an infinite ground plane. The medium inside the cavity is assumed to be (vertically) layered. By introducing a transparent (artificial) boundary condition, the problem in the open cavity is reduced to a bounded domain problem. A simple finite difference method is then applied to solve the model Helmholtz equation. The fast algorithm is designed for solving the resulting discrete system in terms of the discrete *Fourier transform* in the horizontal direction, a Gaussian elimination in the vertical direction, and a preconditioning conjugate gradient method with a complex diagonal preconditioner for the indefinite interface system. The existence and uniqueness of the finite difference solution are established for arbitrary wave numbers. Our numerical experiments for large numbers of mesh points, up to 16 million unknowns, and for large wave numbers, e.g., between 100 and 200 wavelengths, show that the algorithm is extremely efficient. The cost for calculating the radar cross section, which is of significant interest in practice, is $O(M^2)$ for an $M \times M$ mesh. The proposed algorithm may be extended easily to solve discrete systems from other discretization methods of the model problem.

Key words. electromagnetic scattering, FFT, preconditioning, indefinite systems

AMS subject classifications. 78M20, 65N22, 65N06

DOI. 10.1137/S1064827503428539

1. Introduction.

As a measure of the detectability of a target in radar systems, radar cross section (RCS) has been an important subject of study in electromagnetics. Accurate prediction of the RCS of complex objects such as aircraft presents a great challenge in computational electromagnetics. Of particular importance is the RCS prediction of cavities due to its dominance to the target's overall RCS. Examples of cavities include jet engine inlet ducts, exhaust nozzles, and cavity-backed antennas. The computation is especially challenging when the cavities are large compared to the wavelength of the fields because of the highly oscillatory nature of the fields. This paper is intended to develop a fast algorithm for solving the electromagnetic scattering problem from large cavities. More specifically, consider a time-harmonic electromagnetic plane wave incident on an open cavity embedded in an infinite ground plane. The geometry is shown in Figure 1. The ground plane and the walls of the open cavity are perfect electric conductors (PEC), and the interior of the open cavity is filled with nonmagnetic material which may be inhomogeneous. The half-space above the ground plane is filled with a homogeneous, linear, isotropic medium. In this setting, the electromagnetic scattering by the cavity is governed by the Helmholtz equation along with Sommerfeld's radiation conditions imposed at infinity.

*Received by the editors May 27, 2003; accepted for publication (in revised form) March 15, 2005; published electronically October 31, 2005.

http://www.siam.org/journals/sisc/27-2/42853.html

†Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027 (bao@math.msu.edu) and School of Mathematics, Jilin University, Changchun, China. The research of this author was partially supported by NSF grant DMS 01-04001, ONR grant N000140210365, and NSF of China grant 10428105.

‡Department of Mathematics, City University of Hong Kong, Hong Kong (maweiw@math.cityu.edu.hk). The research of this author was supported in part by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (project CityU 1141/01P).

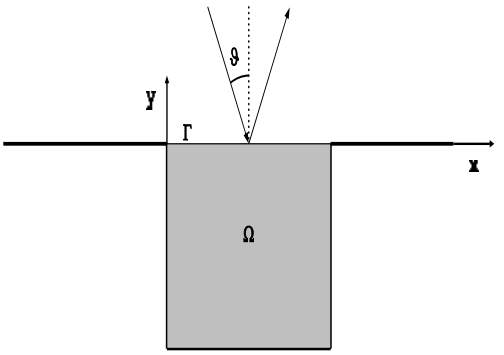


FIG. 1. *The cavity geometry.*

Because of its significant industrial and military applications, the cavity problem has attracted much attention recently. A variety of numerical methods including finite difference, finite element, boundary element methods, and hybrid methods have been developed by the engineering community for solving the open cavity problems [27], [34], [35], [42]. A detailed discussion and additional references may be found in Jin [23]. Mathematical analysis of open cavity models has been conducted by an integral equation method [2], and by a variational approach [3]. An important step for solving the open cavity problem is to introduce a transparent boundary condition on the top line of the cavity. The boundary condition reduces the infinite problem to a finite computational domain, which in turn can be solved by some classical numerical approximation. More recently, a time-domain finite element method has been presented in [41]. However, efficient solvers for the discrete linear systems have not been explored. Classical direct and iterative algorithms are not efficient, especially for large cavity problems due to their high computational costs.

The problem with large wave number, or more precisely with large "ka" number, is of significant interest. Here k is the wave number and a is the diameter of the computational domain. For electromagnetic cavity problems, microwaves are usually considered. Nonetheless, for large cavities, the large domain diameters give rise to high "ka" numbers. Our goal in this paper is to develop a fast algorithm that is capable of dealing with large cavities. For convenience, without loss of generalities, we focus primarily on high wave number problems in our discussion. A straightforward change of coordinates yields the equivalence of high wave number and large cavity problems in the current context. The main difficulty lies in the fact that the solution at a high wave number is highly oscillatory. Moreover, it is well known that error estimates strongly depend upon the wave number. Babuska and Sauter [5] showed that for a related model problem, the ratio of the error of the Galerkin solution and the error of the best approximation tends to infinity as the wave number increases. Furthermore, Aziz, Kellogg, and Stephen [4] pointed out that the condition "k^2h small" would be required to guarantee that the error of the Galerkin solution has the same magnitude as the error of the best approximation, where h is the mesh size. For a large cavity problem, say $1m \times 1m$, with a centimeter incident wave, this condition implies that $h \approx 10^{-4}m$, or 10^8 unknowns in the discrete linear systems of two-dimensional cases.

In this paper, we present a fast algorithm for solving the electromagnetic scattering from a rectangular cavity in which the medium is y-directional inhomogeneous. The algorithm is based on the use of discrete [illegible] in the horizontal

direction and a Gaussian elimination in the vertical direction. We reduce the global system with an $M \times M$ mesh to a (interface) linear system on the top line of the cavity by $O(M^2)$ operations. Since the interface system is complex and indefinite, i.e., its eigenvalues are located on both sides of the y-axis, many classical iterative methods may not be convergent. Here, we propose a complex diagonal preconditioner for the interface system. All eigenvalues of the preconditioned coefficient matrix lie on the right side of the y-axis and the condition number of the matrix increases slightly as the number of mesh points increase. The BiCG method with the diagonal preconditioner is applied to solve the interface system. The number of iterations used in the BiCG method is almost independent of the number of mesh points. Numerical experiments for large numbers of mesh points, up to 16 million unknowns, and for large wave number, e.g., k is between 100 and 200 wavelengths, show that the algorithm is efficient and the computational cost for RCS is $O(M^2)$ for an $M \times M$ mesh. The existence and uniqueness for discrete system with general approximations to the nonlocal boundary condition are established. Here a simple finite difference method is employed for the approximation to illustrate the idea for solving the scattering problem. It should be pointed out that the proposed algorithm and corresponding analysis may also be extended to linear systems resulting from other discrete approximations.

A related topic is the so-called [illegible], which were well studied in the last several decades; see the review papers [39], [40] for finite difference methods and [7], [36], [38] for other approximations. These methods can be applied mainly for solving Poisson's equation on a rectangular region with simple local boundary conditions, such as Dirichlet, Neumann, or periodic boundary conditions. The computational complexity is $O(N^2 logN)$ for an $N \times N$ mesh. Extensions to regions with rectangular substructures and to more general regions can be done in terms of domain decomposition or capacitance matrix techniques [8], [9], [33], [11].

Extensions of capacitance matrix methods to the Helmholtz equations were studied in [16], [17], [30]. For capacitance matrix methods, one needs to solve a capacitance system of small size by using either a direct method with an explicit formula of capacitance matrix or an iterative method in implicit form. At each iteration of the iterative method, the multiplication of a vector by the capacitance matrix can be performed by the classical fast Poisson solver. Usually, direct methods for solving capacitance systems are more expensive [33].

There are two related Helmholtz type models: exterior and interior Helmholtz problems. The former concerns the scattering from a bounded medium or obstacle; while the latter deals with the Helmholtz equation in a bounded domain with suitable boundary conditions, such as of Dirichlet, Neumann, or impedance (Robin) types. Numerous studies have been devoted to the development of iterative algorithms for solving large scale linear systems arising from these two Helmholtz models [12], [13], [14], [16], [17], [18], [21], [26], [31]. The methods are mostly based on domain decomposition and preconditioning techniques. Ernst and Golub [18] solved the interior Helmholtz model in a rectangular domain with impedance boundary conditions using preconditioning methods. In particular, they proposed several preconditioners including the discrete Helmholtz operators with Dirichlet or Neumann boundary conditions on two edges and a preconditioned Schur complement. Elman and O'Leary [12] further studied the first preconditioner proposed in [18] and presented eigenvalue analysis for the preconditioned systems. For the one-dimensional model, the identification of the eigenvalues of a preconditioned system entails the computation of the eigenvalues for a 2×2 matrix. They also extended the algorithm to three-dimensional models with large wavenumbers. More numerical experiments were given in [13].

Note that at each preconditioning iteration of their algorithm, one needs to solve the Helmholtz equation with Dirichlet or Neumann boundary conditions on two edges and impedance boundary conditions on the other two edges, which can be performed by classical FFT-type fast solver.

Exterior Helmholtz problems, in terms of some transformations or truncation techniques, can be reduced to finite domain problems with certain nonlocal boundary conditions: exact, approximate, periodic and Fourier-type, which have been done in the last 30 years; e.g., see [15], [19], [25]. Computational methods for exterior Helmholtz problems with some nonlocal boundary conditions were studied in [16], [17], [20], [28], [31]. In [16], [17], Ernst proposed a fast capacitance matrix algorithm for the exterior Helmholtz equation. A transparent (exact nonlocal) boundary condition introduced in [25] was applied on a circular artificial boundary. Due to the nature of axial symmetry, the nonlocal operator is circulant (periodic). A Krylov subspace method may be used for solving the resulting capacitance system. The multiplication of a vector by the capacitance matrix was performed using a fast Helmholtz solver in an annular domain, which should be viewed as a direct application of fast Poisson's solver since the nonlocal boundary condition is periodic. Otto and Larsson [31] presented a preconditioning algorithm for solving a Helmholtz problem in a duct with a curved bottom, in which a fast Helmholtz solver for a duct with straight lines was used as a preconditioner. The nonlocal boundary conditions at the near- and far-zone boundaries were given in the form of the coupled mode solution. The Fourier-sine transform was applied on both the Helmholtz equation and the nonlocal boundary condition. However, the coupled mode boundary condition proposed in [19] and [1] is only valid for waveguide models rather than the underlying cavity models. The algorithm proposed in the current paper is also based on FFT-type fast solvers and efficient preconditioning techniques. We focus on the cavity model which is different from both the exterior and interior Helmholtz problems. The nonlocal boundary operator arising from the cavity model is not circulant, and the coupled mode boundary condition is not suitable in this case.

Note that efficient numerical algorithms for solving linear systems arising from large cavity models have not been explored. Mathematical analysis on existence and uniqueness of solutions has been conducted only recently in [2], [3]. In our algorithm, the FFT-sine transform with a Gaussian elimination is used to reduce the global system to an aperture system. This step is equivalent to determining the discrete Dirichlet-to-Neumann map. The simple diagonal complex preconditioner applied to the resulting aperture system depends upon both the Helmholtz operator and the nonlocal boundary condition on the aperture. In comparison, the cost at each iteration of our algorithm is proportional to $NlogN$ for an $N \times N$ mesh while the cost at each iteration in [12], [13], [16], [17] is $O(N^2 logN)$.

The rest of the paper is organized as follows. In section 2, the model scattering problem is formulated and further reduced to a bounded domain problem. A detailed fast algorithm is introduced in section 3 for the TM (transverse magnetic) case. Issues on implementation and complexity of the algorithm are addressed. A complex diagonal preconditioner is proposed and the TE (transverse electric) case is also studied. The existence and uniqueness of the finite difference solution is proved in section 4. Section 5 is devoted to numerical experiments of the algorithm. The paper is concluded in section 5 by some general remarks on the usefulness of the algorithm and future directions.

2. The electromagnetic cavity problem. We focus on a two-dimensional geometry by assuming that the medium and material are invariant in the z-direction.

Assume also that the medium is nonmagnetic and a constant magnetic permeability $\mu = \mu_0$ exists everywhere. The electromagnetic property of the medium is characterized by the dielectric coefficient ε. Throughout, it is assumed that ε is layered inside the cavity.

For the TM polarization, in which the magnetic field is transverse to the invariant direction and $E = (0, 0, u(x, y))$, the time-harmonic Maxwell equations reduce to

$$\begin{aligned} &\Delta u + k^2 u = f(x,y) && (x,y) \in \Omega \cup R_2^+, \\ &u = 0 && \text{on } \partial(\Omega \cup R_2^+) \end{aligned} \tag{2.1}$$

together with the radiation boundary condition

$$\lim_{r\to\infty} r\left(\frac{\partial u}{\partial \eta} - ik_0 u\right) = 0, \tag{2.2}$$

where $k^2 = \omega^2 \varepsilon \mu_0$ and R_2^+ denotes the upper half-space. Inside Ω, assume that $\varepsilon = \varepsilon(y)$ is independent of the variable x. The fields are said to be source free if the source term $f = 0$.

For the TE polarization, in which the electric field is transverse to the invariant direction, $H = (0, 0, u(x, y))$, the Maxwell equations reduce to

$$\begin{aligned} &\nabla \cdot \left(\frac{1}{k^2}\nabla u\right) + u = f(x,y) && (x,y) \in \Omega \cup R_2^+, \\ &\frac{\partial u}{\partial n} = 0 && \text{on } \partial\Omega \setminus \Gamma \end{aligned} \tag{2.3}$$

with the same radiation boundary condition (2.2). Here, n denotes the unit outward normal.

Since the upper half-space is homogeneous, a so-called transparent boundary condition can be obtained by using either Green's function method (i.e., Hankel's function) [23] or the method of Fourier's transform [2], [3], [15]. In the TM case, the scattering problem may be further reduced to a bounded domain problem:

$$\begin{aligned} &\Delta u + k^2 u = f(x,y) && (x,y) \in \Omega, \\ &u = 0 && \text{on } \partial\Omega \setminus \Gamma, \\ &\frac{\partial u}{\partial n} = I(u) && \text{on } \Gamma, \end{aligned} \tag{2.4}$$

where $I(u)$ denotes a nonlocal integral operator on Γ.

Let $G_d(\boldsymbol{x}, \boldsymbol{x}')$ be the upper half-plane Dirichlet Green's function for the Helmholtz equation

$$G_d(\boldsymbol{x}, \boldsymbol{x}') = \frac{\mathrm{i}}{4}\left[H_0^{(1)}(k\boldsymbol{r}) - H_0^{(1)}(k\bar{\boldsymbol{r}})\right], \tag{2.5}$$

where $\boldsymbol{x} = (x, y)$, $\boldsymbol{x}' = (x', y')$, $\boldsymbol{r} = |\boldsymbol{x} - \boldsymbol{x}'|$, $\bar{\boldsymbol{r}} = |\boldsymbol{x} - \bar{\boldsymbol{x}}'|$ and $\bar{\boldsymbol{x}}' = (x', -y')$ is the image of $\boldsymbol{x}$ with respect to real axis; by Green's theorem and noting the homogeneous boundary conditions,

$$u(x,y) = \int_\Gamma \frac{\partial G_d(x, y; x', 0)}{\partial n'} u(x')dx' \quad x \in (0, a). \tag{2.6}$$

Then the nonlocal boundary condition is given by

$$\frac{\partial u}{\partial y} = I(u) := -\frac{\mathrm{i}k_0}{2} \Xint=_{\Gamma} \frac{1}{|x-x'|} H_1^{(1)}(k_0|x-x'|)u(x')dx' \quad x \in (0,a), \tag{2.7}$$

where $\Xint=_{\Gamma}$ denotes a Hadamard principle value (or finite part) integral. An alternative Fourier-type nonlocal boundary condition given in [1] is

$$I(u) = \frac{\mathrm{i}k_0}{2} \int_R \sqrt{k_0^2 - \xi^2}\mathcal{F}[\tilde{u}]e^{\mathrm{i}\xi x}d\xi,$$

where $\tilde{u}$ is an extension of u to $H^{1/2}(R)$ and $\mathcal{F}$ denotes the Fourier transformation.

Similar nonlocal boundary conditions for other geometries may be found in [15] and [19].

Assume that a plane wave $u_I = e^{\mathrm{i}(\alpha x - \beta y)}$ is incident on the cavity from the above, where $\alpha = k_0 \sin\theta$, $\beta = k_0 \cos\theta$, and $-\pi/2 < \theta < \pi/2$ is the angle of incidence with respect to the positive y-axis. The scattered field u^s can be expressed by

$$u^s = u^t - u_I + e^{\mathrm{i}(\alpha x + \beta y)},$$

where u^t is the total field of the scattering problem. Since u^s satisfies the Helmholtz equation (2.1) and the radiation condition (2.2), the total field u^t satisfies the following equation:

$$\begin{aligned} &\Delta u + k^2 u = f(x,y) && (x,y) \in \Omega, \\ &u = 0 && \text{on } \partial\Omega \setminus \Gamma, \\ &\frac{\partial u}{\partial n} = I(u) + g(x) && \text{on } \Gamma, \end{aligned} \tag{2.8}$$

where $g(x) = 2\mathrm{i}\beta e^{\mathrm{i}\alpha x}$.

Similarly, one can derive the formulation for TE polarization. In this case, the corresponding Helmholtz-type equation and the transparent boundary condition for the total field are

$$\begin{aligned} &\nabla \cdot \left(\frac{1}{k^2}\nabla u\right) + u = f(x,y) && (x,y) \in \Omega, \\ &\frac{\partial u}{\partial n} = 0 && \text{on } \partial\Omega \setminus \Gamma, \\ &\frac{\partial u}{\partial n} = \tilde{I}(u) + g && \text{on } \Gamma, \end{aligned} \tag{2.9}$$

where $\tilde{I}(u)$ is a nonlocal integral operator.

The existence and uniqueness of solutions of (2.8) and (2.9) for arbitrary wave numbers have been obtained in [3]. The proof is based on a variational approach by using the Fredholm alternative along with the fact that the imaginary part of the nonlocal integral operator is nonnegative.

When a numerical approximation is used for the discretization of (2.8), the global system can be written in general by

$$\begin{pmatrix} A & A_{12} \\ A_{21} & B \end{pmatrix} \begin{pmatrix} u \\ u_B \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.10}$$

where A denotes the discrete Helmholtz operator and B corresponds to the discrete boundary integral operator. The matrix A is sparse and B is full in general. Direct methods for such a system are generally less efficient. Most numerical algorithms are based on a Schur complement technique [23], [32]. A particular one is the outward-looking approach, i.e.,

$$\begin{aligned} &(A - A_{12}B^{-1}A_{21})u = f - A_{12}B^{-1}g, \\ &Bu_B = g - A_{21}u\,. \end{aligned} \tag{2.11}$$

Alternatively, there is also the inward-looking approach

$$\begin{aligned} &(B - A_{21}A^{-1}A_{12})u_B = g - A_{21}A^{-1}g, \\ &Au = f - A_{12}u_B. \end{aligned} \tag{2.12}$$

Some classical iterative algorithms, such as conjugate gradient and biconjugate gradient methods, have been applied for solving these subsystems or directly for solving the global system (2.10). The computational cost at each iteration in (2.11) or (2.12) is at least $O(M^2)$ or $O(M^2 log M)$, respectively. The main purpose of this paper is to present a fast algorithm for solving the Helmholtz equations (2.8) and (2.9) with the nonlocal boundary conditions. Due to the fixed position of aperture in this cavity model, approximate local boundary conditions are not suitable [15]. Also we point out that no convergence result for these iterative methods, particularly for problems with large wave numbers, is available at present.

3. Fast algorithm. For simplicity, we consider a simple five-point finite difference method for the discretization of the Helmholtz equation with uniform meshes and assume that the wave number k is real and positive. The approach may be extended easily to many other approximations. Also, for the layered medium problems in TE polarization, nonuniform meshes in y-direction and some special treatment on interface may be used to achieve better accuracy.

3.1. Algorithm in the TM case. Let $\{x_i, y_j\}_{i,j=0}^{M+1,N+1}$ define a uniform partition of $\Omega = [0, a] \times [-b, 0]$ with $x_{i+1} - x_i = h_x$ and $y_{j+1} - y_j = h_y$. Let v_{ij} be the finite difference solution at the point (x_i, y_j). The discrete finite difference system in the TM case can be given by

$$\frac{v_{i-1,j} - 2v_{ij} + v_{i+1,j}}{h_x^2} + \frac{v_{i,j-1} - 2v_{ij} + v_{i,j+1}}{h_y^2} + \omega\varepsilon(y_j)\mu_0 v_{ij} = f(x_i, y_j),$$
$$i = 1, 2, \ldots, M; j = 1, 2, \ldots, N$$

or in matrix form,

$$(A_x \otimes I_N + I_M \otimes A_y + I_M \otimes D)\, v + (I_M \otimes a_{N+1})\, v_{N+1} = f, \tag{3.1}$$

where $\otimes$ denotes the tensor product (Kronecker product), I_M is the $M \times M$ identity matrix,

$$A_x = \frac{1}{h_x^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \end{pmatrix} \qquad A_y = \frac{1}{h_y^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \end{pmatrix}$$

$$D = w^2\mu_0 \begin{pmatrix} \varepsilon(y_1) & & & \\ & \varepsilon(y_2) & & \\ & & \ddots & \\ & & & \varepsilon_(y_N) \end{pmatrix} \qquad a_{N+1} = \frac{1}{h_y^2} \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$$

and

$$v = (v_{11}, \ldots, v_{1N}, v_{21}, \ldots, v_{2N}, \ldots, v_{M1}, \ldots, v_{MN})^T,$$
$$v_j = (v_{1j}, v_{2j}, \ldots, v_{Mj})^T.$$

By using a backward finite difference scheme and a general approach for the nonlocal boundary condition, we obtain

$$\frac{v_{i,N+1} - v_{i,N}}{h_y} = \sum_{l=1}^{M} g_{il} v_{l,N+1} + g(x_i), \quad i = 1, 2, \ldots, M,$$

i.e.,

$$-v_N + (I_M - h_y G) v_{N+1} = h_y g,$$

where $G = (g_{ij})_{i,j=1}^{M}$ is an $M \times M$ matrix.

The global system is given by

$$\begin{pmatrix} A & A_{12} \\ A_{21} & I_M - h_y G \end{pmatrix} \begin{pmatrix} v \\ v_{N+1} \end{pmatrix} = \begin{pmatrix} f \\ h_y g \end{pmatrix}, \tag{3.2}$$

where

$$A = A_x \otimes I_N + I_M \otimes A_y + I_M \otimes D,$$
$$A_{12} = I_M \otimes a_{N+1},$$
$$A_{21} = -h_y^2 A_{12}^T.$$

For the tridiagonal Toeplitz matrix A_x, we have

$$S_M A_x S_M = \Lambda = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_M \end{pmatrix},$$

where S_M denotes the discrete Fourier-sine transformation,

$$S_M = \frac{1}{\sqrt{M}} \left(\sin \frac{lm\pi}{M+1} \right)_{l,m=1}^{M}, \qquad \lambda_l = -\frac{4(M+1)^2}{a^2} \sin^2 \frac{l\pi}{2(M+1)},$$

and $S_M^2 = I$.

By the discrete Fourier-sine transformation, we rewrite the discrete Helmholtz system (3.1) as

$$(\Lambda \otimes I_N + I_M \otimes A_y + I_M \otimes D)\, \bar{v} + (I_M \otimes a_{N+1})\, \bar{v}_{N+1} = \bar{f}, \tag{3.3}$$

where

$$\bar{v} = (S_M \otimes I_N)\, v = (\bar{v}_{11}, \ldots, \bar{v}_{1N}, \bar{v}_{21}, \ldots, \bar{v}_{2N}, \ldots, \bar{v}_{M1}, \ldots, \bar{v}_{MN})^T, \quad \bar{v}_{N+1} = S_M v_{N+1},$$
$$\bar{f} = (S_M \otimes I_N)\, f = (\bar{f}_{11}, \ldots, \bar{f}_{1N}, \bar{f}_{21}, \ldots, \bar{f}_{2N}, \ldots, \bar{f}_{M1}, \ldots, \bar{f}_{MN})^T.$$

Similarly, the discrete boundary equations become

$$-\bar{v}_N + S_M(I_M - h_y G)S_M\bar{v}_{N+1} = \bar{g},$$

where $\bar{g} = h_y S_M g$.

Reordering the unknowns and equations in (3.3), we obtain

$$\begin{aligned} &(A_y + \lambda_i I_N + D)\hat{v}_i + a_{N+1}\bar{v}_{i,N+1} = \hat{f}_i, \qquad i = 1, 2, \ldots, M, \\ &-\bar{v}_N + S_M(I_M - h_y G)S_M\bar{v}_{N+1} = \bar{g}, \end{aligned} \tag{3.4}$$

where

$$\begin{aligned} &\hat{v}_i = (\bar{v}_{i1}, \bar{v}_{i2}, \ldots, \bar{v}_{iN}), \\ &\hat{f}_i = (\bar{f}_{i1}, \bar{f}_{i2}, \ldots, \bar{f}_{iN}), \qquad i = 1, 2, \ldots, M. \end{aligned}$$

We use the Gaussian elimination method with a row partial pivoting to solve the first system in (3.4). Let

$$A_y + \lambda_i I_N + D = L_i U_i, \quad i = 1, 2, \ldots, M$$

be the LU-decomposition, where $A_y + \lambda_i I_N + D$ is a symmetric tridiagonal matrix.

Since L_i is nonsingular,

$$U_i\bar{v}_i + L_i^{-1}a_{N+1}\bar{v}_{i,N+1} = \hat{f}_i, \qquad i = 1, 2, \ldots, M, \tag{3.5}$$

where $U_i = (r_{pq}^i)$ and $\hat{f}_i = L_i^{-1}\bar{f}_l$.

Combining the last equations of the systems (3.5) and the boundary equations gives

$$\begin{aligned} &r_{NN}^i\bar{v}_{iN} + r_{N,N+1}^i\bar{v}_{i,N+1} = \hat{f}_{iN}, \quad i = 1, 2, \ldots, M, \\ &-\bar{v}_N + S_M(I_M - h_y G)S_M\bar{v}_{N+1} = \bar{g}, \end{aligned} \tag{3.6}$$

where $r_{N,N+1}^i$ is the last component of $L_i^{-1}a_{N+1}$.

Since

$$\begin{aligned} &\bar{v}_N = (\bar{v}_{1N}, \bar{v}_{2N}, \ldots, \bar{v}_{MN}), \\ &\bar{v}_{N+1} = (\bar{v}_{1,N+1}, \bar{v}_{2,N+1}, \ldots, \bar{v}_{M,N+1}), \end{aligned}$$

only $\bar{v}_N$ and $\bar{v}_{N+1}$ are involved in the system (3.6).

If k^2 is not an eigenvalue of the Helmholtz operator with Dirichlet boundary conditions, then for h small enough,

$$r_{NN}^i \neq 0, \qquad i = 1, 2, \ldots, M. \tag{3.7}$$

We can further eliminate the vector $\bar{v}_N$ from the second system in (3.6) to get

$$(S_M(I_M - h_y G)S_M + E)\bar{v}_{N+1} = \hat{g} \tag{3.8}$$

or equivalently

$$(I_M - h_y G + S_M E S_M)v_{N+1} = S_M\hat{g},$$

where

$$E = \begin{pmatrix} r_{1,N+1}^1/r_{NN}^1 & & \\ & \ddots & \\ & & r_{N,N+1}^M/r_{NN}^M \end{pmatrix}$$

and

$$\hat{g}_i = \bar{g}_i + \bar{f}_{iN}/r^i_{NN}, \quad i = 1, 2, \ldots, M.$$

Solving the linear system (3.8) gives the solution v_{N+1} on the interface Γ. The rest of the unknowns can be obtained by solving the following system:

$$(A_y + \lambda_i I_N + D)\bar{v}_i = -a_{N+1}\bar{v}_{N+1}, \quad i = 1, 2, \ldots, M, \tag{3.9}$$

in which $\bar{v}_N = -\bar{g} + \bar{B}\bar{v}_{N+1}$ may be used for those possible nearly singular systems.

If k^2 (approximately) is an eigenvalue of the Dirichlet–Helmholtz operator, then the matrix $A_y + \lambda_i I_N + D$ is singular (or nearly singular) for some $i \in J_s$, where J_s is a subset of $\{1, 2, \ldots, M\}$. Combining the equations in the first system of (3.6) with $i \in J_s$ and a system corresponding to (3.5), we have

$$\begin{bmatrix} D_s & W_{12} \\ W_{21} & I_M - h_y G + S_M E S_M \end{bmatrix} \begin{bmatrix} w \\ v_{N+1} \end{bmatrix} = \begin{bmatrix} f_s \\ g_s \end{bmatrix}, \tag{3.10}$$

where D_s is a diagonal matrix whose diagonal entries consist of r^i_{NN} with $i \in J_s$ and f_s and g_s can be defined analogously. It is obvious that for those i's satisfying

$$\frac{4(N+1)^2}{b^2} \sin^2 \frac{\pi}{2(N+1)} + \frac{4(M+1)^2}{a^2} \sin^2 \frac{i\pi}{2(M+1)} > k^2.$$

$A_y + \lambda_i I_N + D$ is symmetric negative definite. In all of our numerical tests, J_s is empty, i.e., it suffices to solve the system (3.8).

The algorithm is given below.

Algorithm I

(i) Generate the matrix G.
(ii) Calculate the LU decomposition to get U_i and E by using the forward Gaussian elimination with a row partial pivoting.
(iii) Calculate $\bar{f} = (S_M \otimes I)f$ and $S_M \hat{g}$.
(iv) Solve the system (3.8) for v_{N+1}.
(v) Solve the system (3.9) for the rest of the unknowns.

3.2. Implementation and preconditioning. It should be noted that the nonlocal integral operator is a symmetric Toeplitz-type operator. A suitable approximation to the nonlocal integral operator gives the symmetric Toeplitz matrix G and hence only M entries need be calculated. The nonlocal integral operator is hypersingular. Numerical calculation for such hypersingular integrals has been well studied; e.g., see [37]. Usually, the Hankel function can be evaluated in terms of its series representations or integral representations. One can use some high-order approximation to the Hankel function; hence the cost of Step (i) is $O(M^{1+\sigma})$, where $0 < \sigma < 1$. In Step (ii), one needs to calculate the LU-decomposition for $A_y + \lambda_i I_N + D$. By noting the tridiagonal structure of these matrices, only $5NM$ operations are needed if $f \neq 0$. Since S_M is the Fourier-sine transform, by using a standard FFT-sine code, the cost for calculating the product of the matrix S_M and a complex vector is $2MlogM$ (real operations). In Step (iv), one solves the system (3.8) for v_{N+1}. A direct method, such as a Gaussian elimination scheme, requires $M^3/3$ complex operations, which is not efficient. Since G is a complex Toeplitz matrix, the product of the coefficient matrix and a complex vector can be performed in terms of [illegible] [10], [24] and the cost for calculating the product is only $12MlogM$. If we use the BiCG

TABLE 1
The real operation counts of Algorithm I *for an* $M \times N$ *mesh (p is the number of iterations by BiCG).*

Step	Calculate v with $f \neq 0$	Calculate v_{N+1} with $f = 0$
(i)	$O(M^{1+\sigma})$	O(M)
(ii)	$5NM$	$3NM$
(iii)	$2MNlogM$	0
(iv)	$24pMlogM$	$24pMlogM$
(v)	$O(NMlogM)$	0
Total	$O(NMlogM) + 4pMlogM$	$O(NM) + 24pMlogM$

method or GMRES method for solving the linear systems, the cost of each iteration is $24MlogM$.

In most practical cases of electromagnetic scattering, particularly for radar systems, the fields are source free, i.e., $f(x, y) = 0$. Also, it often suffices to calculate the solution on the interface, i.e., v_{N+1}. In fact, many physical components of interest can be obtained by v_{N+1}. In that case, the solution in the whole computational domain may not be necessary. Hence, the computational procedure can be simplified significantly, and the first part of Steps (iii) and (v) are no longer needed.

The cost (real arithmetic operations) for each step is summarized in Table 1, where p is the number of iterations used for solving the system (3.8) and k is a real number. When $f \neq 0$, the cost of the algorithm is $O(NMlogM)$ if p is not large. More importantly, in the source free case, the cost decreases significantly to $O(M^2)$ for an $M \times M$ mesh if $p \leq M/logM$, which will be illustrated numerically in the next section.

Key to Algorithm I is the iterative solver for the interface system (3.8) with the coefficient matrix

$$\mathcal{A} = I - h_y S_M G S_M + E. \tag{3.11}$$

The efficiency of iterative algorithms for (3.8) depends upon many factors, such as the eigenvalue distribution, the condition numbers of the coefficient matrix, and the regularity of solution.

Consider a unit cavity ($a = b = 1$) and two different media, homogeneous medium $k = k_0$ and layered medium

$$k = \begin{cases} k_0 & -b/3 \leq y < 0, \\ 1.5k_0 & -2b/3 \leq y < -b/3, \\ 2k_0 & -b \leq y < -2b/3. \end{cases} \tag{3.12}$$

We present the eigenvalue distributions of the matrix $\mathcal{A}$ in Figure 2, with respect to some different wave numbers and numbers of mesh points. Observe from Figure 2 that there are some eigenvalues with negative real parts in all cases and the magnitude of the real parts becomes larger as the wave number increases, which may be responsible for the divergence of many classical iterative methods. As a comparison, note that the interface systems arising from Poisson-type equations in some simple domains are symmetric positive definite [8], [9], [11].

In order to solve such an indefinite system, we need an efficient preconditioner. Let $G = G^{re} + \mathrm{i}G^{im}$. Since G is a Toeplitz matrix, we denote by d_{im} the diagonal entries of G^{im}. Then we can rewrite the matrix $\mathcal{A}$ as

$$\mathcal{A} = (I - \mathrm{i}h_y d_{im} I + E) - h_y \left(S_M G S_M - \mathrm{i}d_{im} I\right).$$

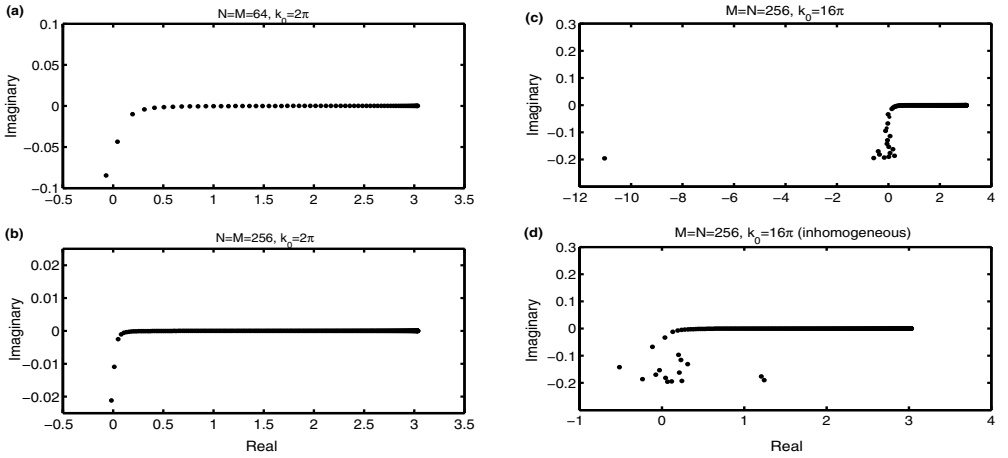


FIG. 2. *The eigenvalue distribution of* $\mathcal{A}$, (a)–(c) *for homogeneous media and* (d) *for the inhomogeneous medium in* (3.12).

TABLE 2
The extremal eigenvalues of the preconditioned matrix $P\mathcal{A}$.

k	N=M	$\min \lvert\lambda_i\rvert$	$\max \lvert\lambda_i\rvert$	$\max \lvert\lambda_i\rvert / \lvert\min \lambda_i\rvert$
$k = 2\pi$	64	1.505	3.718	2.470
	128	1.487	3.737	2.513
	256	1.478	3.745	2.534
	512	1.473	3.748	2.544
	1024	1.471	3.750	2.550
$k = 16\pi$	128	0.8449	7.658	9.064
	256	0.8751	1.453D1	1.661D1
	512	1.003	1.670D1	1.663D1
	1024	1.004	2.164D1	2.154D1
	2048	1.005	2.732D1	2.719D1
k given in (3.12) $k_0 = 16\pi$	128	0.9871	1.019D1	1.039D1
	256	1.011	4.811	4.758
	512	1.001	6.486	6.482
	1024	1.000	7.332	7.331
	2048	1.000	6.995	6.995

Let

$$P = (I - \mathrm{i}h_y d_{im} I + E)^{-1}$$

be a preconditioner. The interface system (3.8) becomes

$$P\mathcal{A}v_{N+1} = PS_M\hat{g}. \tag{3.13}$$

The extremal eigenvalues and the ratio of the maximal eigenvalue and the minimal eigenvalue of the preconditioned coefficient matrix $P\mathcal{A}$ are given in Table 2. The eigenvalues distributions are presented in Figure 3(a)–(c) for the homogeneous medium $k_0 = 2\pi$ with $N = M = 64$, 256, and 1024, in Figure 3(d) for the homogeneous medium $k = k_0 = 16\pi$, and in Figure 3(e)–(f) for the layered medium defined in (3.12) with $k_0 = 2\pi$ and $k_0 = 16\pi$, respectively.

Several observations are in order:

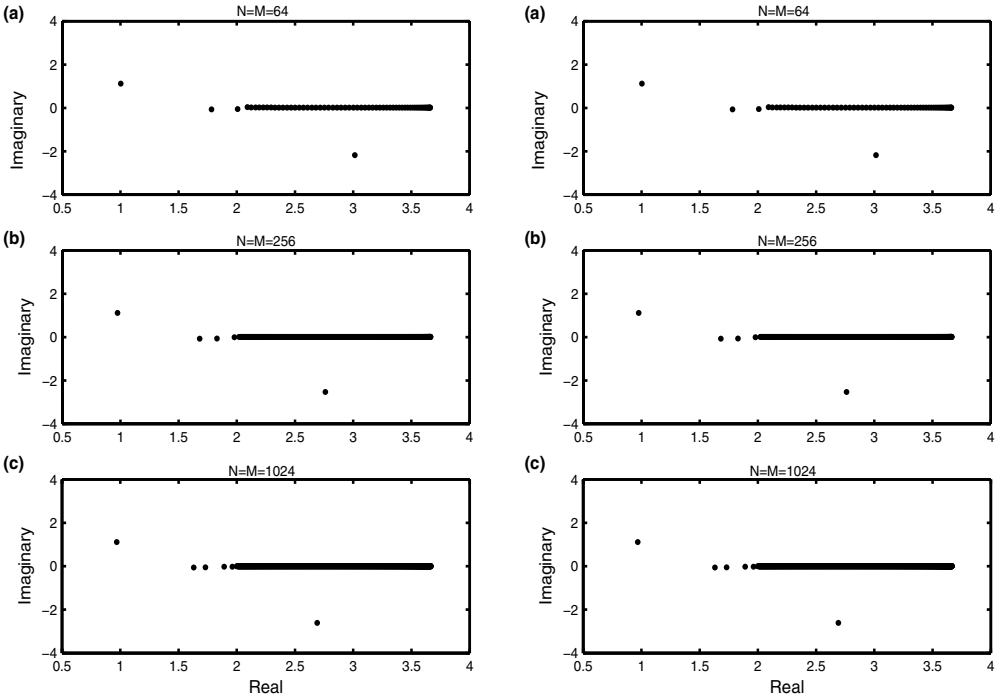


FIG. 3. *The eigenvalue distribution of preconditioned systems with* $\rho = 4.0$, (a)–(c) *homogeneous medium* $k_0 = 2\pi$ *and* (d)–(f) $N = M = 256$.

- In all cases, the real parts of eigenvalues are positive and larger than 0.854. Many iterative algorithms are convergent for such a linear system.
- For the small wave number $k_0 = 2\pi$, the ratio of the maximal eigenvalue and the minimal eigenvalue, which is equivalent to condition number, is almost bounded by a constant 2.55. The eigenvalues distributions are also independent of the number of mesh points. For large wave number $k_0 = 32\pi$, the ratio for the homogeneous medium increases approximately in the order of $M^{0.387}$ and the ratio for the inhomogeneous medium increases slightly as the number of mesh points increases. The minimal eigenvalues are approximately 1 for both homogeneous and inhomogeneous media.
- The ratio of the imaginary part and the real part of eigenvalues is also an important factor for most iterative algorithms. The eigenvalues with a large ratio are called ill-conditioning. For most eigenvalues shown in Table 2 and Figure 3, the imaginary part is smaller than the real part.
  In each case, all eigenvalues are located on a circle of radius about 1.2 to 13, depending on the wavenumber k. All these circles are in the right half-plane and the distance between each circle and the y-axis is larger than 0.85. Certain iterative algorithms are efficient for solving linear systems with this kind of coefficient matrices, such as Krylov subspace methods, and the convergent rate could be independent of the number of mesh points.

3.3. TE case. Rewrite (2.9) of TE case as

$$\begin{aligned} &\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}-\frac{k_y}{2k}\frac{\partial u}{\partial y}+k^2u=k^2f \qquad (x,y)\in\Omega,\\ &\frac{\partial u}{\partial n}=0 \qquad \text{on } \partial\Omega\backslash\Gamma,\\ &\frac{\partial u}{\partial n}=\tilde{I}(u)+g \qquad \text{on } \Gamma. \end{aligned} \tag{3.14}$$

Using the same approximation to the TE equation, we have the discrete system in matrix form

$$\left(A_x^N\otimes I_N+I_M\otimes(A_y^{ND}-D_kB_y^{ND})+I_M\otimes D\right)v+(I_M\otimes a_{N+1})\,v_{N+1}=f, \tag{3.15}$$

where D and a_{N+1} are defined as before, A_x^N denotes the finite difference discrete matrix corresponding to the second-order differential operator with Neumann boundary conditions, and A_y^{ND} and B_y^{ND} denote finite difference matrices corresponding to the second-order and the first-order differential operator with Neumann/Dirichlet boundary conditions, respectively. Thus

$$A_x^N=\frac{1}{h_x^2}\begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -1 \end{pmatrix} \qquad A_y^{ND}=\frac{1}{h_y^2}\begin{pmatrix} -1 & 1 & & \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 \end{pmatrix}$$

$$B_y^{ND}=\frac{1}{2h_y}\begin{pmatrix} -1 & 1 & & & \\ -1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 0 & 1 \\ & & & -1 & 1 \end{pmatrix}$$

and

$$D_k=\frac{1}{2}\begin{pmatrix} \varepsilon'(y_1)/\varepsilon(y_1) & & & \\ & \varepsilon'(y_2)/\varepsilon(y_2) & & \\ & & \ddots & \\ & & & \epsilon'(y_N)/\epsilon(y_N) \end{pmatrix}.$$

A fast algorithm for the TE case can be designed analogously by replacing A_x and A_y in Algorithm I with A_x^N and $A_y^{ND}-D_kB_y^{ND}$.

4. The existence and uniqueness of numerical solution. The existence and uniqueness of the scattering problem (2.1), (2.2) or the equivalent bounded domain problem in (2.8) were proved in [2] and [3]. Here we study the existence and uniqueness of the solution of the discrete system (3.2) with some simple discretization for the nonlocal boundary condition. Let $J_\nu(z)$ and $Y_\nu(z)$ be the Bessel functions. The Hankel function $H_1^1(z)$ is defined by

$$H_1^1(z)=J_1(z)+\mathrm{i}Y_1(z).$$

Since $Y_1(z)$ is hypersingular and $J_1(z)$ is continuous, the nonlocal boundary condition becomes

$$\frac{\partial u}{\partial n} = \frac{\mathrm{i}k_0}{2}\left(\int_0^a \frac{1}{|x-x'|}J_1(k_0|x-x'|)u(x',0)dx' + \mathrm{i}\int_0^a \frac{1}{|x-x'|}Y_1(k_0|x-x'|)u(x',0)dx'\right), \quad x \in (0,a).$$

Let $u(x,0) \approx \sum_{i=1}^{M} \phi_i(x)v_{i,N+1}$, where $\phi_i(x)$, $i = 1,2,\ldots,M$, denote the piecewise linear basis functions. Then the boundary discrete system can be written by

$$\frac{v_{l,N+1} - v_{l,N}}{h_y} = \sum_{l=1}^{M}(g_{lj}^{re} + \mathrm{i}g_{lj}^{im})v_{j,N+1} + g(x_l), \quad l = 1,2,\ldots,M,$$

where

$$g_{lj}^{im} = \frac{k_0}{2}\int_0^a \frac{1}{|x-x'|}J_1(k_0|x'-x_l|)\phi_j(x')dx'.$$

By the classical formulation

$$J_1(z) = \frac{z}{\pi}\int_{-1}^{1}(1-t^2)^{1/2}e^{\mathrm{i}zt}\,dt,$$

we have

$$g_{lj}^{im} = \frac{k_0^2}{2\pi}\int_{-1}^{1}(1-t^2)^{1/2}\left(\int_0^a \phi_j(x')e^{\mathrm{i}k_0|x'-x_l|t}\,dx'\right)dt. \tag{4.1}$$

The term g_{lj}^{re} can be calculated by some techniques in [29] since the Hadamard integral is involved.

THEOREM 4.1. [illegible] $g_{lj} = g_{lj}^{re} + \mathrm{i}g_{lj}^{im}$ [illegible] g_{lj}^{im} [illegible] (4.1) [illegible] k [illegible] (3.2) [illegible]. It suffices to prove that the system (3.2) has only the zero solution when $f = 0$ and $g = 0$. Let v^* denote the conjugate transpose of v. By (3.2)

$$v^*Av + v^*A_{12}v_{N+1} = 0,$$
$$v_{N+1}^*A_{21}v + v_{N+1}^*(I - h_y(G^{re} + \mathrm{i}G^{im}i))v_{N+1} = 0,$$

where $G = G^{re} + \mathrm{i}G^{im}$. Since $A_{21} = -h_y^2A_{12}^T$ and A is real and symmetric,

$$-h_y^2v^*Av + v_{N+1}^*(I - h_yG)v_{N+1} = 0,$$

which leads to

$$v_{N+1}^*G^{im}v_{N+1} = 0. \tag{4.2}$$

It is easy to see that

$$g_{lj}^{im} = \frac{k_0^2}{2\pi}\left[\int_0^1(1-t^2)^{1/2}\int_0^{x_l}\phi_j(x')e^{\mathrm{i}k_0(x'-x_l)t}\,dx'dt + \int_0^1(1-t^2)^{1/2}\int_{x_l}^a\phi_j(x')e^{-\mathrm{i}k_0(x'-x_l)t}\,dx'dt + \int_{-1}^0(1-t^2)^{1/2}\int_0^{x_l}\phi_j(x')e^{\mathrm{i}k_0(x'-x_l)t}\,dx'dt + \int_{-1}^0(1-t^2)^{1/2}\int_{x_l}^a\phi_j(x')e^{-\mathrm{i}k_0(x'-x_l)t}\,dx'dt\right].$$

Moreover, by using the change of variables in the last two integrals,

$$\begin{aligned} g_{lj}^{im} &= \frac{k_0^2}{2\pi}\left[\int_0^1 (1-t^2)^{1/2}\int_0^a \phi_j(x')e^{\mathrm{i}k_0(x'-x_l)t}\,dx'dt \right. \\ &\qquad \left. + \int_0^1 (1-t^2)^{1/2}\int_0^a \phi_j(x')e^{-\mathrm{i}k_0(x'-x_l)t}\,dx'dt\right] \\ &= \frac{k_0^2}{\pi}\int_0^1 (1-t^2)^{1/2}\int_0^a \phi_j(x')\cos(k_0(x'-x_l)t)\,dx'dt. \end{aligned}$$

A straightforward calculation gives

$$\int_0^a \phi_j(x')\cos(k_0(x'-x_l)t)\,dx' = \frac{2(1-\cos(k_0th_x))\cos((j-l)k_0th_x)}{t^2k_0^2h_x}.$$

Hence

$$g_{lj}^{im} = \frac{2}{\pi h_x}\int_0^1 \eta(t)\cos((j-l)k_0th_x)\,dt, \tag{4.3}$$

where $\eta(t) = (1-t^2)^{1/2}(1-\cos(k_0th_x))/t^2$. It follows that G^{im} is a symmetric Toeplitz matrix and furthermore

$$\begin{aligned} v_{N+1}^* G^{im} v_{N+1} &= \sum_{l=1}^M\sum_{j=1}^M g_{lj}^{im} v_{l,N+1}^* v_{j,N+1} \\ &= \frac{2}{\pi h_x}\sum_{l=1}^M\sum_{j=1}^M \int_0^1 \eta(t)\cos((j-l)k_0th_x)v_{l,N+1}^* v_{j,N+1}\,dt \\ &= \frac{1}{\pi h_x}\int_0^1 \eta(t)\sum_{l=1}^M\sum_{j=1}^M \left(e^{\mathrm{i}k_0(j-l)th_x} + e^{-\mathrm{i}k_0(j-l)th_x}\right) v_{l,N+1}^* v_{j,N+1}\,dt \\ &= \frac{1}{\pi h_x}\int_0^1 \eta(t)\left(\left|\sum_{j=1}^M e^{\mathrm{i}k_0jth_x}v_{j,N+1}\right|^2 + \left|\sum_{l=1}^M e^{-\mathrm{i}k_0lth_x}v_{l,N+1}\right|^2\right)dt. \end{aligned}$$

By (4.2), we have

$$\sum_{j=1}^M e^{\mathrm{i}k_0jth_x}v_{j,N+1} = 0\,, \quad \sum_{j=1}^M e^{-\mathrm{i}k_0jth_x}v_{j,N+1} = 0 \quad \text{for } 0<t<1 \tag{4.4}$$

which leads to $v_{j,N+1} = 0$, $j = 1, 2, \ldots, M$. From the second system in (3.2), $v_{j,N} = 0$ and we deduce further that $v_{ij} = 0$ from the first system in (3.2). The proof is complete. □

In practice, one may calculate g_{lj}^{im} numerically. A general quadrature form can be given by

$$g_{lj}^{im} \approx \frac{2}{\pi h_x}\sum_{m=1}^{M'} \omega_m\eta(t_m)\cos((j-l)k_0t_mh_x), \tag{4.5}$$

where w_m denotes the weight, and an alternative approximation is given by

$$g_{lj}^{im} \approx \frac{2}{\pi h_x}\int_0^1 \eta_h(t)\cos((j-l)k_0th_x)\,dt, \tag{4.6}$$

where $\eta_h(t)$ denotes a general approximation to $\eta(t)$ satisfying $\eta_h(t) > 0$ for $t \in (0, 1)$. The following corollary can be obtained immediately.

COROLLARY 4.2. $g_{lj} = g_{lj}^{re} + \mathrm{i} g_{lj}^{im}$ g_{lj}^{im} (4.5) $M' \geq M/2$ (4.6) k (3.2)

5. Numerical experiments. In this section, computational results are reported for two examples to test our algorithm. Our focus is on the efficiency of our algorithm to solve the discrete linear systems. Once again, error analysis of the discrete method and other high order methods remains open. Throughout, the computation is performed on a Blade 1000 Sun-workstation in double precision. The BiCG method given in [10] is applied to solve the system (3.13).

5.1. Consider a plane wave scattering from a rectangular groove with 1 meter wide and 0.25 meter deep at normal incidence ($a = 1.0$ and $b = 0.25$). Two different cases are considered: the groove filled with a homogeneous medium $k = k_0$ and the groove filled with some inhomogeneous medium defined in (3.12). The former has been a standard test problem [23], [41]. Numerical results are obtained by using our algorithm with 128 nodal points at each coordinate direction for the homogeneous case $k = k_0 = 2\pi$. The magnitude and phase of the field are given in Figure 4(a)–(b), compared with the experimental results in [11]. The physical parameter of interest is the RCS, which is defined by

$$\sigma = \frac{4}{k_0}|P(\phi)|^2,$$

where ϕ is the observation angle and P is the far-field coefficient given by

$$P(\phi) = \frac{k}{2}\sin\phi \int_\Gamma u e^{\mathrm{i}kx\cos\phi} dx.$$

The backscatter RCS(b), compared with experimental results in [23], is given in Figure 4(c).

Next, we consider problems with the high wave number $k_0 = 32\pi$ for both the homogeneous medium and the layered medium defined by (3.12). We present in Figure 4(d)–(f) the magnitudes of the fields on Γ, where $M = N = 1024$ and $\theta = 0$. Since no analytic solution is available, numerical solutions with $N = M = 2048$ are also calculated to confirm the convergence.

We use the BiCG method in Step (iv) to solve the linear system (3.13). As usual, the iteration stops with the absolute residual

$$e_R = \left(\frac{1}{M} \sum_{i=1}^{M} \left(\hat{g}_i - \sum_{j=1}^{M} a_{ij} v_{j,N+1}^p \right)^2 \right)^{1/2} \leq \delta,$$

where v^p is the numerical solution at the pth iteration and $P\mathcal{A} = (a_{ij})$. The CPU time for each step, the total CPU time, and the number of iterations in Step (iv) by the BiCG method are shown in Table 3 until $N = M = 4096$, where $\delta = h_x h_y/5$. The CPU time in Step (i) is proportional to M, the number of mesh points, since we have used an approximation independent of h_x, while the CPU time in Step (ii) is proportional to M^2.

Clearly, the number of iterations (p) in Step (iv) is almost independent of the number of mesh points and increases as the wavenumber k increases (approximately

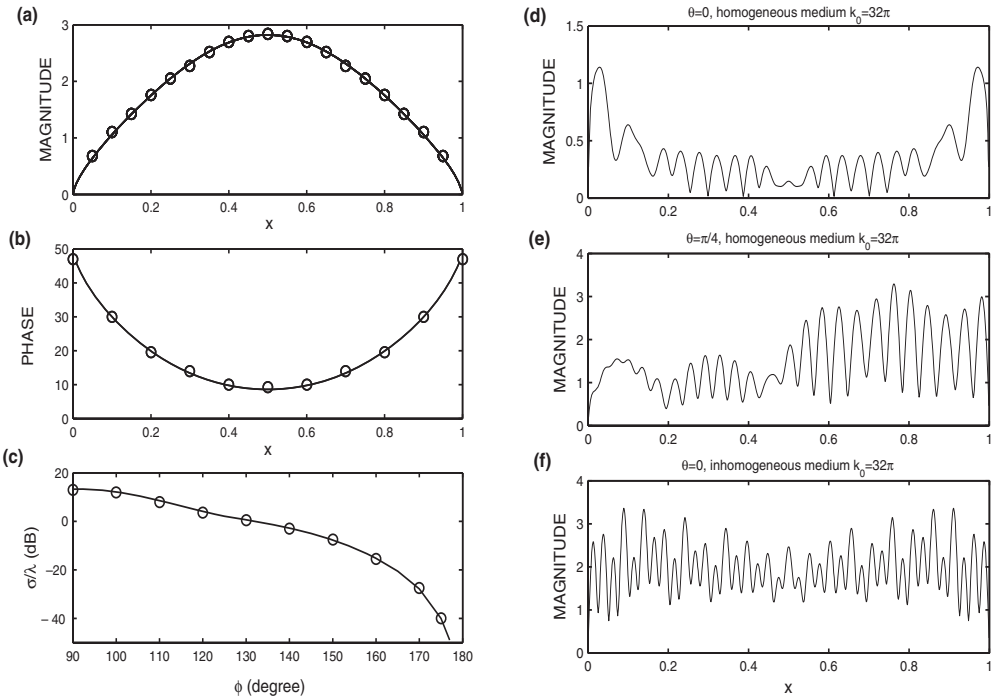


FIG. 4. *Aperture electric field and backscatter RCS.* (a)–(c) *normal incidence* ($\theta = 0$) *and homogeneous medium* $k_0 = 2\pi$ *and* (d)–(f) $k_0 = 32\pi$ *(Example* 5.1*).*

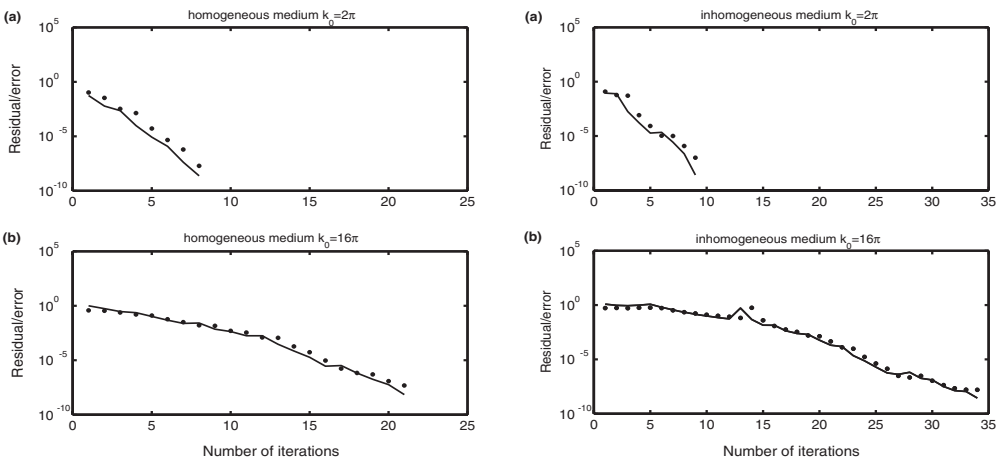


FIG. 5. *The hysteresis of convergence of the BiCG method for solving* (3.13) *with* $N = M = 512$*, where the solid line denotes the residual* e_R *and dashed line the error* e_G*.* (a) $k = 2\pi$*,* (b) $k = 16\pi$*, and* (c) k *is defined in* (3.12) *with* $k_0 = 16\pi$ *and* $\rho = 1$ *(Example* 5.1*).*

with $p = 4 + k/(2\pi)$). The results are in good agreement with our analysis. We present the convergence hysteresis of the BiCG method in Figure 5, where the solid

TABLE 3
CPU time (sec.) and the number of iterations for Example 5.1 *with* $\delta = h_x h_y/5$.

k	No. of unknowns (MN)	CPU (Step (i))	CPU (Step (ii))	CPU (No. of iteration) (Step (iv))	Total CPU
$k = 2\pi$	256^2	0.1181	0.0148	0.0405 (4 iterations)	0.1804
	512^2	0.2372	0.0594	0.1012 (4 iterations)	0.4013
	1024^2	0.4783	0.2375	0.2289 (4 iterations)	0.9467
	2048^2	0.9551	0.9873	0.4832 (4 iterations)	2.412
$k = 4\pi$	256^2	0.1502	0.0150	0.0502 (5 iterations)	0.1804
	512^2	0.3145	0.0598	0.1509 (6 iterations)	0.5268
	1024^2	0.6350	0.2376	0.3424 (6 iterations)	1.230
	2048^2	1.303	0.9876	0.7212 (6 iterations)	3.110
$k = 8\pi$	256^2	0.1906	0.0150	0.0799 (8 iterations)	0.2912
	512^2	0.3903	0.0597	0.2005 (8 iterations)	0.6587
	1024^2	0.8043	0.2379	0.5140 (9 iterations)	1.623
	2048^2	1.723	0.9882	1.073 (9 iterations)	3.867
$k = 16\pi$	256^2	0.2420	0.0152	0.1211 (12 iterations)	0.4122
	512^2	0.4853	0.0598	0.2776 (13 iterations)	0.8322
	1024^2	0.9770	0.2385	0.6164 (13 iterations)	1.867
	2048^2	1.972	0.9894	1.478 (14 iterations)	4.431
	4096^2	3.916	4.141	3.267 (15 iterations)	11.44
k given in (3.12) ($k_0 = 8\pi$)	256^2	0.2431	0.0153	0.1206 (12 iterations)	0.3827
	512^2	0.4889	0.0602	0.2636 (12 iterations)	0.8213
	1024^2	0.9856	0.2388	0.5812 (12 iterations)	1.832
	2048^2	1.974	0.9937	1.332 (12 iterations)	4.398
	4096^2	3.947	4.141	3.103 (13 iterations)	11.24

line represents the residual defined above and the dashed line denotes the L_2 error

$$e_G = \left(\frac{1}{M} \sum_{i=1}^{M} (v^p_{i,N+1} - v^G_{i,N+1})^2 \right)^{1/2}.$$

Here $v^G_{i,N+1}$ is the solution obtained by the Gaussian elimination.

Our numerical experiments illustrate that the computational complexity for the solution v_{N+1} is $O(M^2)$ for an $M \times M$ mesh.

5.2. We consider an artificial example defined by (2.8) with a large cavity $a = b = 1$ to test the accuracy of approximation. The $f(x, y)$ and $g(x)$ are chosen such that the exact solution is

$$u(x, y) = e^{xy} \left(\sin \frac{k_0 x}{2} \right) \sin \left(\frac{k_0}{2} + \frac{\pi}{4} \right) y.$$

The error measures in the L_2 norm and the L_∞ norm in the domain Ω are defined by

$$e_2(\Omega) = \left(\tfrac{1}{(N+1)M} \textstyle\sum_{i,j=1}^{M,N+1} (u^h_{ij} - u(x_i, y_j))^2 \right)^{1/2},$$
$$e_M(\Omega) = \max_{i,j} |u^h_{ij} - u(x_i, y_j)|,$$

where u^h_{ij} denotes the numerical solution at the point (x_i, y_j). Since the solution at the top line of cavity (interface) is more important, we also define the following error

TABLE 4
Error results for Example 5.2.

k	No. of unknowns (MN)	$e_M(\partial\Omega)$	$e_2(\partial\Omega)$	$e_M(\Omega)$	$e_2(\Omega)$
$k = k_0 = 2\pi$	64^2	8.384D-3	6.226D-2	1.170D-2	6.123D-3
	128^2	4.334D-3	3.197D-3	5.986D-3	3.149D-3
	256^2	2.203D-3	1.620D-3	3.032D-3	1.597D-3
	512^2	1.110D-3	8.152D-4	1.533D-3	8.040D-4
	1024^2	5.575D-4	4.089D-4	7.707D-4	4.024D-4
$k = k_0 = 32\pi$	64^2	6.601D-2	2.989D-2	2.873D-1	8.342D-2
	128^2	7.585D-2	4.327D-2	9.404D-2	3.039D-2
	256^2	3.387D-2	1.630D-2	7.249D-2	3.120D-2
	512^2	2.215D-2	1.165D-2	2.935D-2	1.250D-2
	1024^2	1.110D-2	5.945D-3	1.409D-2	5.979D-3
k given in (3.13) $k_0 = 32\pi$	64^2	1.544D-1	7.433D-2	1.544D-1	1.577D-2
	128^2	6.278D-2	3.684D-2	7.042D-2	2.669D-2
	256^2	3.453D-2	1.730D-2	5.490D-2	2.501D-2
	512^2	1.790D-2	9.243D-3	2.164D-2	7.358D-3
	1024^2	1.036D-2	6.008D-3	1.127D-2	4.796D-3

measures on Γ,

$$e_2(\Gamma) = \left(\frac{1}{M}\sum_i^M (u^h_{i,N+1} - u(x_i,0))^2\right)^{1/2},$$
$$e_M(\Gamma) = \max_{i,j} |u^h_{i,N+1} - u(x_i,0)|.$$

We analyze the errors in Table 4 with different wave numbers and different numbers of mesh points. It is evident that for the small wave number $k = k_0 = 2\pi$, the error is always in the order of $O(h)$. Since we use a first-order scheme for $\frac{\partial u}{\partial n}$, this is in good agreement with theoretical analysis. For the large wave number $k = k_0 = 32\pi$, however, the order pattern of errors is not clear for small M, $M = 64, 128$. By increasing M, we can still see the first-order accuracy in the numerical results. However, the errors for problems with large wave numbers are much larger than those for small wave numbers. This confirms previous theoretical analysis and the claim that the best error estimate is achieved only when the condition "$k^2h = constant$" is satisfied. Nevertheless, since the computational complexity of our algorithm is $O(M^2)$ for an $M \times M$ mesh and the memory requirement is $O(M)$ for calculating the RCS, our algorithm can be applied to solve model problems with very large numbers of mesh points.

6. Concluding remarks. A new fast algorithm is presented for solving electromagnetic scattering from a rectangular open (layered) cavity. The algorithm is designed to use the FFT in the layered direction and Gaussian elimination along the inhomogeneous direction to solve the discrete linear systems.

Based on our numerical experiments, the proposed complex preconditioner is efficient. However, theoretical analysis remains open. The problem is challenging mainly due to the complicated transparent boundary condition and the indefinite Dirichlet Helmholtz operator.

Our algorithm has the advantages of being simple in structures and easy to implement, in addition to its capability of handling large cavities or high wave numbers. It is also flexible enough to be incorporated with many types of discretization of the underlying model PDEs. The fast algorithm provides an alternative to the current

efforts of designing sophisticated basis functions for solving Helmholtz equations with high wave numbers.

We point out two important future directions along the line of our present work. The first is to solve the scattering from three-dimensional large cavities. Our algorithm may be extended to provide a fast solver for layered three-dimensional cavities and a preconditioner for two-dimensional problems with more general medium distribution. The algorithm for solving the resulting discrete systems may be designed again by a combination of the FFT and a Gaussian elimination scheme. Another important project is to apply our fast algorithm to the optimal design problems. An example is to design the shape and composition of a layered cavity to minimize the RCS. Computationally, the design problem can be challenging because of the need of solving the scattering problem repeatedly. The fast algorithm presented here certainly would provide the essential tool for the design problems. In addition, it is possible to extend existing preconditioning techniques for Toeplitz systems; e.g., see [10] and [24], to the interface system (3.8).

Acknowledgments. The authors would like to thank Professors Su, Chaowei, and Wu Ji-ming for providing a BiCG code and a code for the discretization of the nonlocal boundary conditions. We also thank Professor Howard Elman and the referee for useful suggestions and for informing us of several related references.

REFERENCES

[1] L. Abrahamsson and H. Kreiss, *Numerical solution of the coupled mode equations in duct acoustics*, J. Comput. Phys., 111 (1994), pp. 1–14.

[2] H. Ammari, G. Bao, and A. W. Wood, *An integral equation method for the electromagnetic scattering from cavities*, Math. Methods Appl. Sci., 23 (2000), pp. 1057–1072.

[3] H. Ammari, G. Bao, and A. W. Wood, *Analysis of the electromagnetic scattering from a cavity*, Japan J. Indust. Appl. Math., 19 (2002), pp. 301–310.

[4] A. K. Aziz, R. B. Kellogg, and A. B. Stephen, *A two point boundary value problem with a rapidly oscillating solution*, Numer. Math., 53 (1988), pp. 107–121.

[5] I. Babuska and S. A. Sauter, *Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?* SIAM J. Numer. Anal., 34 (1997), pp. 2392–2423.

[6] I. Babuska, F. Ihlenburg, E. T. Paik, and S. A. Sauter, *A generalized finite element method for solving the Helmholtz equation in two dimensions with minimal pollution*, Comput. Methods Appl. Mech. Engrg., 128 (1995), pp. 325–359.

[7] R. E. Bank, *Efficient algorithms for solving tensor product finite element equations*, Numer. Math., 31 (1978/79), pp. 49–61.

[8] P. E. Bjorstad and O. B. Widlund, *Iterative methods for the solution of elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal., 23 (1986), pp. 1097–1120.

[9] B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.

[10] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[11] T. F. Chan and T. Y. Hou, *Eigendecomposition of domain decomposition interface operators for constant coefficient elliptic problems*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1471–1479.

[12] H. Elman and D. P. O'Leary, *Efficient iterative solution of the three-dimensional Helmholtz equation*, J. Comput. Phys., 142 (1998), pp. 163–181.

[13] H. Elman and D. P. O'Leary, *Eigenanalysis of some preconditioned Helmholtz problems*, Numer. Math., 83 (1999), pp. 231–257.

[14] H. Elman, O. G. Ernst, and D. P. O'Leary, *A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations*, SIAM J. Sci. Comput., 23 (2001), pp. 1291–1315.

[15] B. Engquist and A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp., 31 (1977), pp. 629–651.

[16] O. G. Ernst, *Fast Numerical Solution of Exterior Helmholtz Problems with Radiation Boundary Condition by Imbedding*, Ph.D. thesis, Stanford University, Stanford, CA, 1994.
[17] O. G. Ernst, *A finite-element capacitance matrix method for exterior Helmholtz problems*, Numer. Math., 75 (1996), pp. 175–204.
[18] O. G. Ernst and G. H. Golub, *A domain decomposition approach to solving the Helmholtz equation with radiation boundary condition*, Contemp. Math, 157 (1994), pp. 177–192.
[19] G. Fix and S. P. Marin, *Variational methods for underwater acoustic problems*, J. Comput. Phys., 28 (1978), pp. 253–270.
[20] D. Givoli, *Numerical Methods for Problems in Infinite Domains*, Stud. Appl. Mech. 33, Elsevier, Amsterdam, 1992.
[21] E. Heikkola, T. Rossi, and J. Toivanen, *A parallel fictitious domain method for the three-dimensional Helmholtz equation*, SIAM J. Sci. Comput., 24 (2003), pp. 1567–1588.
[22] D. A. Jacobs, *A generalization of the conjugate-gradient method to solve complex systems*, IMA J. Numer. Anal., 6 (1986), pp. 447–452.
[23] J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley & Sons, New York, 1993.
[24] X. Q. Jin, *Developments and Applications of Block Toeplitz Iterative Solvers*, Science Press & Kluwer Academic Publishers, Beijing/Dordrecht, The Netherlands, 2002.
[25] J. B. Keller and D. Givoli, *Exact nonreflecting boundary conditions*, J. Comput. Phys., 82 (1989), pp. 172–192.
[26] S. Kim, *Domain decomposition iterative procedures for solving scalar waves in the frequency domain*, Numer. Math., 79 (1998), pp. 231–259.
[27] J. Liu and J. M. Jin, *A special higher-order finite-element method for scattering by deep cavity*, IEEE Trans. Antennas Propag., 48 (2000), pp. 694–703.
[28] R. C. MacCamy and S. P. Marin, *A finite element method for exterior interface problems*, Internat. J. Math. Math. Sci., 3 (1980), pp. 311–350.
[29] P. A. Martin, F. J. Rizzo, and T. A. Cruse, *Smoothness-relaxation strategies for singular and hypersingular integral equations*, Internat. J. Numer. Methods Engrg., 42 (1998), pp. 885–906.
[30] D. P. O'Leary and O. Widlund, *Capacitance matrix methods for the Helmholtz equation on general three-dimensional regions*, Math. Comp., 33 (1979), pp. 849–879.
[31] K. Otto and E. Larsson, *Iterative solution of the Helmholtz equation by a second-order method*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 209–229.
[32] A. F. Peterson, *Analysis of heterogeneous electromagnetic scatterers: Research progress of the past decades*, Proc. IEEE, 79 (1991), pp. 1431–1441.
[33] W. Proskurowski, *Capacitance matrix methods—a brief survey*, in Elliptic Problem Solvers, Academic Press, New York, 1981, pp. 391–398.
[34] D. C. Ross, J. L. Volakis, and H. T. Anastassiu, *Hybrid finite element-modal analysis of jet engine inlet scattering*, IEEE Trans. Antennas Propag., 43 (1995), pp. 277–285.
[35] P. R. Rousseau and R. J. Burkholder, *A hybrid approach for calculating the scattering from obstacles within large, open cavities*, IEEE Trans. Antennas Propag., 43 (1995), pp. 1068–1075.
[36] W. Sun, *Fast algorithms for high-order spline collocation systems*, Numer. Math., 81 (1998), pp. 143–160.
[37] W. Sun and J. Wu, *Newton-Cotes formulas for numerical evaluation of certain hypersingular integrals*, Computing, to appear.
[38] W. Sun and N. G. Zamani, *A fast algorithm for solving the tensor product collocation equations*, J. Franklin Inst., 326 (1989), pp. 295–307.
[39] P. N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis, and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle*, SIAM Rev., 19 (1977), pp. 490–501.
[40] P. N. Swarztrauber, *Fast Poisson solvers*, in Studies in Numerical Analysis, G. H. Golub, ed., MAA Stud. Math., 24, Math. Assoc. America, Washington, D.C., 1984, pp. 319–370.
[41] T. Van and A. W. Wood, *A time-domain finite element method for Helmholtz equations*, J. Comput Phys., 183 (2002), pp. 486–507.
[42] Z. Xiang and T. T. Chia, *A hybrid BEM/WTM approach for analysis of the EM scattering from large open-ended cavities*, IEEE Trans. Antennas Propag., 49 (2001), pp. 165–173.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 575–598
© 2005 Society for Industrial and Applied Mathematics

OPTIMAL SUPERCONVERGENT ONE STEP NODAL CUBIC SPLINE COLLOCATION METHODS*

BERNARD BIALECKI†, GRAEME FAIRWEATHER†, AND ANDREAS KARAGEORGHIS‡

Abstract. We formulate new optimal (fourth) order one step nodal cubic spline collocation methods for the solution of various elliptic boundary value problems in the unit square. These methods are constructed so that the respective collocation equations can be solved using matrix decomposition algorithms (MDAs). MDAs are fast, direct methods which employ fast Fourier transforms and require $O(N^2 \log N)$ operations on an $N \times N$ uniform partition of the unit square. The results of numerical experiments exhibit expected global optimal orders of convergence as well as desired superconvergence phenomena.

Key words. elliptic boundary value problems, nodal spline collocation, matrix decomposition, fast Fourier transforms, optimal convergence rates, superconvergence

AMS subject classifications. 65N35, 65N22

DOI. 10.1137/040609793

1. Introduction.

Assume that $\Omega = (0,1) \times (0,1)$, $D_x^i = \partial^i/\partial x^i$, $D_y^j = \partial^j/\partial y^j$, and consider the boundary value problem (BVP) comprising the differential equation

$$D_x^2 u + a(y) D_y^2 u + b(y) D_y u + c(y) u = f(x,y), \quad (x,y) \in \Omega, \tag{1.1}$$

and one of the following boundary conditions:
Dirichlet:

$$u(0,y) = u(1,y) = 0, \ \ 0 \le y \le 1, \quad u(x,0) = u(x,1) = 0, \ \ 0 < x < 1; \tag{1.2}$$

Neumann:

$$\begin{cases} D_x u(0,y) = D_x u(1,y) = 0, & 0 \le y \le 1, \\ D_y u(x,0) = D_y u(x,1) = 0, & 0 < x < 1; \end{cases} \tag{1.3}$$

mixed:

$$u(0,y) = D_x u(1,y) = 0, \ \ 0 \le y \le 1, \quad u(x,0) = D_y u(x,1) = 0, \ \ 0 < x < 1; \tag{1.4}$$

periodic:

$$\begin{cases} D_x^i u(0,y) = D_x^i u(1,y), & 0 \le y \le 1, \ \ i = 0,1, \\ D_y^j u(x,0) = D_y^j u(x,1), & 0 < x < 1, \ \ j = 0,1. \end{cases} \tag{1.5}$$

Over the past several decades, numerous fast direct methods have been developed for the efficient solution of the linear algebraic systems arising when various numerical techniques are applied to BVPs of this type. Since (1.1) includes Poisson's equation in Cartesian, polar, and cylindrical coordinates, these methods are frequently

*Received by the editors June 9, 2004; accepted for publication (in revised form) February 28, 2005; published electronically October 31, 2005.
http://www.siam.org/journals/sisc/27-2/60979.html

†Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401-1887 (bbialeck@mines.edu, gfairwea@mines.edu).

‡Department of Mathematics and Statistics, University of Cyprus/ Πανεπιστήμιο Κύπρου, P.O. Box 20537, 1678 Nicosia/ Λευκωσία, Cyprus/Κύπρος (andreask@ucy.ac.cy).

referred to as fast Poisson solvers. Such methods use matrix decomposition algorithms (MDAs) and often involve fast Fourier transforms (FFTs). In [4], an overview of MDAs for finite difference, finite element Galerkin, orthogonal spline collocation, and spectral methods applied to second order problems is presented. For recent work on fast direct methods, see [5] for references on orthogonal spline collocation methods, [11, 12] for quadratic spline collocation methods, [18, 20] for finite element Galerkin methods, [8, 24, 25, 26] for spectral methods, [15, 19, 21, 22] for finite differences, and [3, 9, 28] for Fourier-type methods, and references therein. In this paper, we develop new nodal cubic spline collocation methods for the solution of BVPs of the form (1.1) subject to each of the boundary conditions (1.2), (1.3), (1.4), and (1.5) in turn. These methods are constructed so that they have the desired convergence properties and their respective collocation equations can be solved using an MDA with FFTs.

In the basic nodal cubic spline collocation (NCSC) method, an approximate solution is sought in the space of bicubic splines and is determined by satisfying the differential equation and the boundary conditions at the partition nodes of $\overline{\Omega}$ and $\partial\Omega$, respectively. This method is well known to be suboptimal; in fact, it is no better than second order, whereas fourth order is expected since the approximate solution is piecewise bicubic. In [17], two optimal order NCSC methods, a two step method (TSM) and a one step method (OSM), were presented for the solution of the Dirichlet BVP. These methods are based on optimal NCSC methods for solving second-order two-point BVPs. The TSM is based on the method of Fyfe [14] in which the approximate solution is obtained using a deferred correction approach applied to the basic NCSC method. In the OSM, which is based on a method developed by Archer [1, 2], the approximation is determined by collocating the differential equation with a high order perturbation of the differential operator.

In [6], for the Helmholtz equation, MDAs with FFTs were formulated and implemented to solve the TSM collocation equations for each BVP, but, for the OSM, it was possible to formulate an MDA only for the Dirichlet BVP. In the numerical examples presented in [6], it is shown that, for smooth solutions u, the TSM exhibits superconvergence in the approximations to $D_x u$, $D_y u$ and $D_{xy} u$ ($D_{xy} = D_x D_y$). It is not surprising that the OSM is not superconvergent, as Archer [2] showed that the OSM for two-point BVPs lacks this property. For the Dirichlet BVP, the OSM is the more efficient method, requiring asymptotically one half of the cost of the TSM, but less accurate than the TSM with the same partition stepsize. It should be noted that, in [16], it is claimed that this method applied to the Neumann BVP is of optimal order, but no evidence, theoretical or numerical, is provided to support this claim. In fact, our numerical tests with this method for two-point BVPs with boundary conditions other than Dirichlet exhibit only third order accuracy from which one can only conclude that the optimality of the OSM for Neumann BVPs claimed in [16] is erroneous.

The purpose of this paper is to derive for each BVP an OSM with optimal global accuracy and the desired superconvergence properties and for which the collocation equations can be solved using an extension of the MDA with FFTs developed in [6]. The new OSMs differ from other OSMs in that they are constructed by perturbing judiciously both the differential operator and the right-hand side of the differential equation. It should be noted that, in previous discussions on OSMs, for example, in [16, 17], mixed and periodic boundary conditions were not considered.

An outline of this paper is as follows. In section 2, we introduce some notation and describe the MDA. Then, in section 3, we revisit the Archer and Daniel–Swartz

methods for solving two-point BVPs and derive new schemes which, when used in the discretization of two-dimensional problems, yield methods with the desired properties of optimal global accuracy and superconvergence, and are amenable to solution using MDAs with FFTs. Sections 4, 5, 6, and 7 are devoted to the Dirichlet, Neumann, mixed, and periodic BVPs, respectively. In section 8, numerical results are presented which demonstrate the expected orders of convergence. Concluding remarks are given in section 9. This work represents a significant extension of the work of [6, 16, 17].

2. Preliminaries. Let $\rho_x = \{x_i\}_{i=0}^{N+1}$ and $\rho_y = \{y_j\}_{j=0}^{N+1}$ be, respectively, uniform partitions of $[0,1]$ in the x- and y-directions such that $x_i = y_i = ih$, where the stepsize $h = 1/(N+1)$. The space of cubic splines

$$S_3 = \big\{v \in C^2[0,1] : v|_{I_i} \in \mathcal{P}_3, 1 \le i \le N+1\big\},$$

where $I_i = [x_{i-1}, x_i]$ and $\mathcal{P}_3$ is the set of polynomials of degree ≤ 3. To introduce basis functions for S_3, we extend the partition ρ_x using $x_i = ih$, $i = -3, -2, -1, N+2, N+3, N+4$. As a basis for S_3, we choose the functions $\{\mathcal{B}_m\}_{m=-1}^{N+2}$, where

$$\mathcal{B}_m(x) = \begin{cases} h^{-3} g_1(x - x_{m-2}), & x \in I_{m-1}, \\ g_2([x - x_{m-1}]/h), & x \in I_m, \\ g_2([x_{m+1} - x]/h), & x \in I_{m+1}, \\ h^{-3} g_1(x_{m+2} - x), & x \in I_{m+2}, \\ 0 & \text{otherwise}, \end{cases} \tag{2.1}$$

and

$$g_1(x) = x^3, \quad g_2(x) = 1 + 3x + 3x^2 - 3x^3.$$

These basis functions are such that, for $0 \le m \le N+1$,

$$\begin{cases} \mathcal{B}_{m-1}(x_m) = 1, & \mathcal{B}_m(x_m) = 4, & \mathcal{B}_{m+1}(x_m) = 1, \\ \mathcal{B}'_{m-1}(x_m) = -3/h, & \mathcal{B}'_m(x_m) = 0, & \mathcal{B}'_{m+1}(x_m) = 3/h, \\ \mathcal{B}''_{m-1}(x_m) = 6/h^2, & \mathcal{B}''_m(x_m) = -12/h^2, & \mathcal{B}''_{m+1}(x_m) = 6/h^2. \end{cases} \tag{2.2}$$

The space of bicubic splines is $S_3 \otimes S_3$, where, for two spaces V and W of functions on $[0,1]$, $V \otimes W$ denotes the space of functions on $\overline{\Omega}$ consisting of all finite linear combinations of products $v(x)w(y)$, where $v \in V$ and $w \in W$. In all our methods, the approximate solution lies in the subspace of $S_3 \otimes S_3$ comprising functions satisfying the boundary conditions under consideration. A basis for each subspace is obtained by modifying the functions $\{\mathcal{B}_m\}_{m=-1}^{N+2}$.

Each of the OSM methods for the solution of the BVPs of section 1 gives rise to a linear system of the form

$$\{A_1 \otimes B_2 + B_1 \otimes [D_a A_2 + D_b C + D_c B_2]\}\,\mathbf{u} = \mathbf{f}, \tag{2.3}$$

where the matrices A_1, B_1 are $M \times M$ with $M = N, N+1$, or $N+2$, the matrices A_2, C, and B_2 are $(N+2) \times (N+2)$, the symbol $\otimes$ denotes the matrix tensor product, and where, for a function g in $C[0,1]$,

$$D_g = \operatorname{diag}(g(y_0), \ldots, g(y_{N+1})).$$

If there exists a nonsingular matrix Z such that

$$Z^{-1}A_1Z = \Lambda_A, \quad Z^{-1}B_1Z = \Lambda_B, \tag{2.4}$$

where Λ_A and Λ_B are diagonal, then the system (2.3) is equivalent to

$$(Z^{-1}\otimes I)\{A_1\otimes B_2 + B_1\otimes[D_aA_2 + D_bC + D_cB_2]\}(Z\otimes I)(Z^{-1}\otimes I)\mathbf{u} = (Z^{-1}\otimes I)\mathbf{f},$$

which, on using (2.4), can be written as

$$\{\Lambda_A\otimes B_2 + \Lambda_B\otimes[D_aA_2 + D_bC + D_cB_2]\}\mathbf{v} = \mathbf{w}, \tag{2.5}$$

where

$$\mathbf{v} = (Z^{-1}\otimes I)\mathbf{u}, \quad \mathbf{w} = (Z^{-1}\otimes I)\mathbf{f}.$$

If $\Lambda_A = \mathrm{diag}(\lambda_i^A)_{i=1}^M$ and $\Lambda_B = \mathrm{diag}(\lambda_i^B)_{i=1}^M$, then (2.5) reduces to the collection of independent systems

$$\left\{\lambda_i^BD_aA_2 + \lambda_i^BD_bC + [\lambda_i^AI + \lambda_i^BD_c]B_2\right\}\mathbf{v}_i = \mathbf{w}_i, \quad 1\le i\le M, \tag{2.6}$$

where

$$\mathbf{v}_i = [v_{i,0}, v_{i,1}, \ldots, v_{i,N+1}]^T, \quad \mathbf{w}_i = [w_{i,0}, w_{i,1}, \ldots, w_{i,N+1}]^T.$$

We thus have the following MDA for solving (2.3):

Step 1. Compute $\mathbf{w} = (Z^{-1}\otimes I)\mathbf{f}$.
Step 2. Solve the M systems in (2.6).
Step 3. Compute $\mathbf{u} = (Z\otimes I)\mathbf{v}$.

As we shall see in the following, the elements of the matrix Z are sines or cosines, and Z^{-1} can be expressed in terms of Z or Z^T. As a consequence, in Steps 1 and 3, FFT routines can be employed to perform the matrix multiplications at a cost of $O(N^2\log N)$ operations. In Step 2, the linear systems are, or can be reduced to, pentadiagonal systems. Consequently, Step 2 requires $O(N^2)$ operations. Thus the total cost of the MDA is $O(N^2\log N)$ operations. As in [6], the key ingredient in this solution procedure is the determination of the matrices Λ_A, Λ_B, Z, and Z^{-1}.

Throughout the paper, we make use of the following. Let $\mathcal{I}$, $\mathcal{J}$, $\mathcal{M}$, and $\mathcal{N}$ be finite sets of increasing indices. Without loss of generality we assume

$$\mathcal{I} = \{1,\ldots,I'\}, \quad \mathcal{J} = \{1,\ldots,J'\}, \quad \mathcal{M} = \{1,\ldots,M'\}, \quad \mathcal{N} = \{1,\ldots,N'\}.$$

Then the matrix-vector form of

$$\phi_{i,j} = \sum_{m\in\mathcal{M}} c_{i,m}^{(1)} \sum_{n\in\mathcal{N}} c_{j,n}^{(2)}\psi_{m,n}, \quad i\in\mathcal{I}, \quad j\in\mathcal{J}, \tag{2.7}$$

is

$$\boldsymbol{\phi} = (C_1\otimes C_2)\boldsymbol{\psi}, \tag{2.8}$$

where

$$C_1 = (c_{i,m}^{(1)})_{i\in\mathcal{I},m\in\mathcal{M}}, \quad C_2 = (c_{j,n}^{(2)})_{j\in\mathcal{J},n\in\mathcal{N}},$$

and

$$\boldsymbol{\phi} = [\phi_{1,1},\ldots,\phi_{1,J'},\ldots,\phi_{I',1},\ldots,\phi_{I',J'}]^T,$$
$$\boldsymbol{\psi} = [\psi_{1,1},\ldots,\psi_{1,N'},\ldots,\psi_{M',1},\ldots,\psi_{M',N'}]^T.$$

3. Optimal superconvergent one step NCSC methods for two-point BVPs. In order to describe the new OSM approach for discretizing (1.1) in the x–direction with our choices of boundary conditions, we assume that κ is a constant and consider a two-point BVP comprising the differential equation

$$u'' + \kappa u = f(x), \quad x \in [0,1], \tag{3.1}$$

and either Dirichlet boundary conditions

$$u(0) = u(1) = 0, \tag{3.2}$$

or Neumann boundary conditions

$$u'(0) = u'(1) = 0, \tag{3.3}$$

or periodic boundary conditions

$$u(0) = u(1), \quad u'(0) = u'(1). \tag{3.4}$$

In Archer's method [1, 2] for (3.1) and (3.2), with

$$S^{\mathcal{D}} = \{v \in S_3 : v(0) = v(1) = 0\},$$

we seek $u_h \in S^{\mathcal{D}}$, such that

$$L_A u_h(x_i) = f(x_i), \quad 0 \le i \le N+1, \tag{3.5}$$

where

$$L_A v(x_i) = \begin{cases} v''(x_i) + \kappa v(x_i), & i = 0, N+1, \\ \dfrac{1}{12}[v''(x_{i-1}) + 10v''(x_i) + v''(x_{i+1})] + \kappa v(x_i), & 1 \le i \le N. \end{cases} \tag{3.6}$$

It is shown in [2] that, for $\kappa = 0$ and sufficiently smooth u,

$$\max_{0 \le x \le 1} |(u - u_h)^{(j)}(x)| = O(h^{4-j}), \quad j = 0, 1, 2, \tag{3.7}$$

and

$$\max_{0 \le i \le N+1} |(u - u_h)'(x_i)| = O(h^3). \tag{3.8}$$

Numerical experiments that we have conducted indicate that the exponent 3 in (3.8) is optimal.

For (3.1) and (3.3), in the context of the line cubic spline collocation method [16], one looks for $u_h \in S^{\mathcal{N}}$ satisfying (3.5). However, our numerical tests show that

$$\max_{0 \le i \le N+1} |(u - u_h)(x_i)| = O(h^3), \tag{3.9}$$

which indicates that the scheme of [16] is, at best, third (and not fourth) order accurate. In fact, our tests show that, for boundary conditions other than Dirichlet, the scheme (3.5) is only third order accurate.

In the new scheme for (3.1) and (3.2), which will be used for the discretization of (1.1) and (1.2) in the x-direction, we seek $u_h \in S^{\mathcal{D}}$ satisfying

$$L^{\mathcal{D}}_{MDA} u_h(x_i) = f^{\mathcal{D}}_h, \quad 0 \le i \le N+1, \tag{3.10}$$

where $L^{\mathcal{D}}_{MDA} = L_A$ of (3.6) and

$$f^{\mathcal{D}}_h(x_i) = \begin{cases} f(x_i) - \dfrac{h^2}{12}[f''(x_i) - \kappa f(x_i)], \ i = 0, N+1, \\ f(x_i), \ 1 \le i \le N. \end{cases}$$

In the new scheme for (3.1) and (3.3), which will be used for the discretization of (1.1) and (1.3) in the x-direction, with

$$S^{\mathcal{N}} = \{v \in S_3 : v'(0) = v'(1) = 0\},$$

we seek $u_h \in S^{\mathcal{N}}$, such that

$$L^{\mathcal{N}}_{MDA} u_h(x_i) = f^{\mathcal{N}}_h, \quad 0 \le i \le N+1, \tag{3.11}$$

where

$$L^{\mathcal{N}}_{MDA} u_h(x_i) = \begin{cases} \dfrac{1}{12}[10v''(x_i) + 2v''(x_{i+1})] + \kappa v(x_i), \ i = 0, \\ L_A v(x_i), \ 1 \le i \le N, \\ \dfrac{1}{12}[10v''(x_i) + 2v''(x_{i-1})] + \kappa v(x_i), \ i = N+1, \end{cases}$$

and

$$f^{\mathcal{N}}_h(x_i) = \begin{cases} f(x_i) + \dfrac{h}{6} f'(x_i), \ i = 0, \\ f(x_i), \ 1 \le i \le N, \\ f(x_i) - \dfrac{h}{6} f'(x_i), \ i = N+1. \end{cases}$$

The corresponding schemes for (3.1) with one of the conditions

$$u(0) = u'(1) = 0, \quad u'(0) = u(1) = 0,$$

are easily determined from (3.10) and (3.11).

In the new scheme for (3.1) and (3.4), which will be used for the discretization of (1.1) and (1.5) in the x-direction, with

$$S^{\mathcal{P}} = \{v \in S_3 : v(0) = v(1), v'(0) = v'(1)\},$$

we seek $u_h \in S^{\mathcal{P}}$ satisfying

$$L^{\mathcal{P}}_{MDA} u_h(x_i) = f^{\mathcal{P}}_h(x_i), \quad 0 \le i \le N+1, \tag{3.12}$$

where

$$L^{\mathcal{P}}_{MDA} u_h(x_i) = \begin{cases} \dfrac{1}{12}[10v''(x_i) + v''(x_{i+1}) + v''(x_N)] + \kappa v(x_i), \ i = 0, \\ L_A v(x_i), \ 1 \le i \le N-1, \\ \dfrac{1}{12}[v''(x_0) + v''(x_{i-1}) + 10v''(x_i)] + \kappa v(x_i), \ i = N, \\ \dfrac{1}{12}[v''(x_1) + v''(x_{i-1}) + 10v''(x_i)] + \kappa v(x_i), \ i = N+1, \end{cases}$$

$$f_h^{\mathcal{P}}(x_i) = \begin{cases} \frac{1}{12}[11f(x_0) + f(x_{N+1})] + \tilde{f} + 5\hat{f}, \ i = 0, \\ f(x_i), \ 1 \le i \le N-1, \\ f(x_N) + \frac{1}{12}[f(x_0) - f(x_{N+1})] + \hat{f}, \ i = N, \\ \frac{1}{12}[11f(x_{N+1}) + f(x_0)] + \tilde{f} - 5\hat{f}, \ i = N+1, \end{cases}$$

and

$$\tilde{f} = \frac{h}{12}[f'(x_0) - f'(x_{N+1})], \quad \hat{f} = \frac{h^2}{144}\{f''(x_{N+1}) - f''(x_0) - \kappa[f(x_{N+1}) - f(x_0)]\}.$$

Numerical experiments show that (3.7) holds for (3.10), (3.11), and (3.12), and that the approximation to the first derivative is superconvergent, that is,

$$\max_{0 \le i \le N+1} |(u - u_h)'(x_i)| = O(h^4). \tag{3.13}$$

To describe our schemes for the discretization in the y-direction, we consider a two-point BVP comprising the differential equation

$$a(y)u'' + b(y)u' + c(y)u = f(y), \quad y \in [0,1],$$

and one of the boundary conditions (3.2), (3.3), or (3.4). For any of the corresponding BVPs, in the method of Daniel and Swartz [13], one seeks $u_h \in S^{\mathcal{D}}$, $u_h \in S^{\mathcal{N}}$, or $u_h \in S^{\mathcal{P}}$, respectively, such that

$$L_{DS}u_h(y_j) = f(y_j), \quad 0 \le j \le N+1, \tag{3.14}$$

where

$$L_{DS}v(y_j) = \begin{cases} \frac{a(y_j)}{12}[14v''(y_j) - 5v''(y_{j+1}) + 4v''(y_{j+2}) - v''(y_{j+3})] \\ \qquad + b(y_j)u'(y_j) + c(y_j)u(y_j), \quad j = 0, \\ \frac{a(y_j)}{12}[v''(y_{j-1}) + 10v''(y_j) + v''(y_{j+1})] \\ \qquad + b(y_j)v'(y_j) + c(y_j)v(y_j), \quad 1 \le j \le N, \\ \frac{a(y_j)}{12}[14v''(y_j) - 5v''(y_{j-1}) + 4v''(y_{j-2}) - v(y_{j-3})] \\ \qquad + b(y_j)v'(y_j) + c(y_j)v(y_j), \quad j = N+1. \end{cases}$$

It is shown in [13] that (3.7) and (3.13) hold for the Dirichlet and Neumann schemes. In the new scheme of this paper, we seek $u_h \in S^{\mathcal{D}}$, $u_h \in S^{\mathcal{N}}$, or $u_h \in S^{\mathcal{P}}$, respectively, such that

$$L_{BFK}u_h(y_j) = f(y_j), \quad 0 \le j \le N+1, \tag{3.15}$$

where

$$L_{BFK}v(y_j) = \begin{cases} \frac{a(y_j)}{12}[13v''(y_j) - 2v''(y_{j+1}) + v''(y_{j+2})] \\ \qquad + b(y_j)v'(y_j) + c(y_j)v(y_j), \quad j = 0, \\ L_{DS}v(y_j), \quad 1 \le j \le N, \\ \frac{a(x_i)}{12}[13v''(y_j) - 2v''(y_{j-1}) + v(y_{j-2})] \\ \qquad + b(y_j)v'(y_j) + c(y_j)v(y_j), \quad j = N+1. \end{cases}$$

Numerical tests show that (3.7) and (3.13) hold for the new Dirichlet, Neumann, and periodic schemes.

We note that, using the comparison function w defined by (2.3) and (2.4) of [13], the endpoint collocation equations ($i = 0, N+1$) in Archer's scheme (3.5) have a local error which is second order, whereas in the Daniel–Swartz scheme (3.14), the local endpoint error is fourth order. In our new scheme (3.10), the right-hand side is perturbed so that the endpoint equations are fourth order accurate. In schemes (3.11) and (3.12), the left-hand side is perturbed so that (2.4) holds, and the right-hand side is perturbed so that the endpoint equations are third order accurate. Instead of using (3.14) involving a four point stencil, in (3.15) for $j = 0, N+1$, we use a three point stencil which guarantees third order accuracy of the endpoint equations. This is sufficient to yield the expected global accuracy of the scheme as well as its superconvergence.

4. The Dirichlet BVP. In this section, we consider the Dirichlet BVP comprising (1.1) and (1.2). In the OSM, we seek $u_h^{\mathcal{D}} \in S^{\mathcal{D}} \otimes S^{\mathcal{D}}$ such that

$$D_x^2 D_y^2 u_h^{\mathcal{D}}(x_i, y_j) = D_y^2 f(x_i, y_j), \quad i, j = 0, N+1, \tag{4.1}$$

$$\begin{aligned} D_x^2 u_h^{\mathcal{D}}(x_i, y_j) - \frac{h^2}{12}[1 + a(y_j)] D_x^2 D_y^2 u_h^{\mathcal{D}}(x_i, y_j) \\ = f_h(x_i, y_j),\ i = 0, N+1,\ 1 \le j \le N, \end{aligned} \tag{4.2}$$

$$(L_x + L_y) u_h^{\mathcal{D}}(x_i, y_j) = f(x_i, y_j), \quad 1 \le i \le N, \quad 0 \le j \le N+1, \tag{4.3}$$

where, for $0 \le j \le N+1$,

$$L_x v(x_i, y_j) = \frac{1}{12}\left[D_x^2 v(x_{i-1}, y_j) + 10 D_x^2 v(x_i, y_j) + D_x^2 v(x_{i+1}, y_j)\right],\ 1 \le i \le N,$$

for $1 \le i \le N$,

$$L_y v(x_i, y_j) = \begin{cases} \dfrac{a(y_j)}{12}\left[13 D_y^2 v(x_i, y_j) - 2 D_y^2 v(x_i, y_{j+1}) + D_y^2 v(x_i, y_{j+2})\right] \\ \qquad + b(y_j) D_y v(x_i, y_j) + c(y_j) v(x_i, y_j),\ j = 0, \\ \dfrac{a(y_j)}{12}\left[D_y^2 v(x_i, y_{j-1}) + 10 D_y^2 v(x_i, y_j) + D_y^2 v(x_i, y_{j+1})\right] \\ \qquad + b(y_j) D_y v(x_i, y_j) + c(y_j) v(x_i, y_j),\ 1 \le j \le N, \\ \dfrac{a(y_j)}{12}\left[D_y^2 v(x_i, y_{j-2}) - 2 D_y^2 v(x_i, y_{j-1}) + 13 D_y^2 v(x_i, y_j)\right] \\ \qquad + b(y_j) D_y v(x_i, y_j) + c(y_j) v(x_i, y_j),\ j = N+1, \end{cases} \tag{4.4}$$

and, for $i = 0, N+1$, $1 \le j \le N$,

$$f_h(x_i, y_j) = f(x_i, y_j) - \frac{h^2}{12}\left[\Delta f(x_i, y_j) - b(y_j) D_y f(x_i, y_j) - c(y_j) f(x_i, y_j)\right].$$

As a basis for $S^{\mathcal{D}}$, we choose the cubic splines $\{B_m^{\mathcal{D}}\}_{m=0}^{N+1}$ defined in terms of the splines (2.1) by

$$\begin{cases} \mathcal{B}_0^{\mathcal{D}} = \mathcal{B}_0 - 4\mathcal{B}_{-1}, \quad \mathcal{B}_1^{\mathcal{D}} = \mathcal{B}_1 - \mathcal{B}_{-1}, \\ \mathcal{B}_m^{\mathcal{D}} = \mathcal{B}_m, \quad 2 \le m \le N-1, \\ \mathcal{B}_N^{\mathcal{D}} = \mathcal{B}_N - \mathcal{B}_{N+2}, \quad \mathcal{B}_{N+1}^{\mathcal{D}} = \mathcal{B}_{N+1} - 4\mathcal{B}_{N+2}. \end{cases} \tag{4.5}$$

It follows from (4.5) and (2.2) that, for $i = 0, N+1$,

$$[\mathcal{B}_m^{\mathcal{D}}]''(x_i) = \begin{cases} -36h^{-2}, & m = i, \\ 0, & m \neq i, \end{cases} \qquad 0 \le m \le N+1. \tag{4.6}$$

Substituting

$$u_h^{\mathcal{D}}(x,y) = \sum_{m=0}^{N+1}\sum_{n=0}^{N+1} u_{m,n}\mathcal{B}_m^{\mathcal{D}}(x)\mathcal{B}_n^{\mathcal{D}}(y) \tag{4.7}$$

into (4.1) and using (4.6), we obtain

$$u_{i,j} = \frac{h^4}{1296} D_y^2 f(x_i, y_j), \quad i,j = 0, N+1.$$

Substituting (4.7) into (4.2) and using (4.6), we have

$$\sum_{n=1}^{N} u_{i,n}\left\{\frac{h^2}{12}[1+a(y_j)][\mathcal{B}_n^{\mathcal{D}}]''(y_j) - \mathcal{B}_n^{\mathcal{D}}(y_j)\right\} = \frac{h^2}{36} f_h(x_i, y_j)$$

$$- \sum_{n=0,N+1} u_{i,n}\left\{\frac{h^2}{12}[1+a(y_j)][\mathcal{B}_n^{\mathcal{D}}]''(y_j) - \mathcal{B}_n^{\mathcal{D}}(y_j)\right\}, \quad i = 0, N+1, \quad 1 \le j \le N.$$

These equations yield a linear system for $\{u_{i,n}\}_{n=1}^N$, $i = 0, N+1$, with the matrix

$$\frac{h^2}{12} D_{1+a} A - B_1, \tag{4.8}$$

where

$$A = [a_{j,n}]_{j,n=1}^N, \quad a_{j,n} = [\mathcal{B}_n^{\mathcal{D}}]''(y_j), \quad B_1 = [b_{j,n}^{(1)}]_{j,n=1}^N, \quad b_{j,n}^{(1)} = \mathcal{B}_n^{\mathcal{D}}(y_j). \tag{4.9}$$

Using (4.5) and (2.2), we have

$$A = 6h^{-2}T, \quad B_1 = T + 6I, \tag{4.10}$$

where the $N \times N$ matrix T is given by

$$T = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}.$$

Therefore, (4.8) and (4.10) yield the tridiagonal matrix $\frac{1}{2} D_{a-1} T - 6I$.

Substituting (4.7) into (4.3) and using (2.7)–(2.8), we obtain (2.3), where

$$\mathbf{u} = [u_{1,0}, \ldots, u_{1,N+1}, \ldots, u_{N,0}, \ldots, u_{N,N+1}]^T, \tag{4.11}$$

$$\mathbf{f} = [f_{1,0}, \ldots, f_{1,N+1}, \ldots, f_{N,0}, \ldots, f_{N,N+1}]^T, \tag{4.12}$$

$$f_{i,j} = f(x_i, y_j) - \sum_{m=0,N+1} \sum_{n=0}^{N+1} u_{m,n} \left[L_x \mathcal{B}_m^{\mathcal{D}}(x_i) \mathcal{B}_n^{\mathcal{D}}(y_j) + \mathcal{B}_m^{\mathcal{D}}(x_i) L_y \mathcal{B}_n^{\mathcal{D}}(y_j) \right],$$

$$A_1 = [a_{i,m}^{(1)}]_{i,m=1}^{N}, \quad A_2 = [a_{j,n}^{(2)}]_{j,n=0}^{N+1}, \tag{4.13}$$

$$a_{i,m}^{(1)} = \frac{1}{12} \left\{ [\mathcal{B}_m^{\mathcal{D}}]''(x_{i-1}) + 10[\mathcal{B}_m^{\mathcal{D}}]''(x_i) + [\mathcal{B}_m^{\mathcal{D}}]''(x_{i+1}) \right\}, \tag{4.14}$$

$$a_{j,n}^{(2)} = \frac{1}{12} \begin{cases} 13 \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_j) - 2 \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j+1}) + \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j+2}), & j = 0, \\ \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j-1}) + 10 \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_j) + \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j+1}), & 1 \le j \le N, \\ \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j-2}) - 2 \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_{j-1}) + 13 \left[\mathcal{B}_n^{\mathcal{D}}\right]''(y_j), & j = N+1, \end{cases} \tag{4.15}$$

B_1 is defined in (4.9) and (4.10),

$$B_2 = [b_{j,n}^{(2)}]_{j,n=0}^{N+1}, \quad b_{j,n}^{(2)} = \mathcal{B}_n^{\mathcal{D}}(y_j), \quad C = [c_{j,n}]_{j,n=0}^{N+1}, \quad c_{j,n} = [\mathcal{B}_n^{\mathcal{D}}]'(y_j). \tag{4.16}$$

It follows from (4.13)–(4.16), (4.10), (4.5), and (2.2) that

$$A_1 = \frac{1}{12} \begin{bmatrix} 1 & 10 & 1 & & & & \\ & 1 & 10 & 1 & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & & 1 & 10 & 1 & \\ & & & & & 1 & 10 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0}^T \\ A \\ \mathbf{0}^T \end{bmatrix} = \frac{1}{2h^2}(T + 12I)T,$$

$$A_2 = \frac{1}{2h^2} \begin{bmatrix} -80 & 5 & -4 & 1 & & & & \\ 4 & -19 & 8 & 1 & & & & \\ 1 & 8 & -18 & 8 & 1 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 8 & -18 & 8 & 1 \\ & & & & 1 & 8 & -19 & 4 \\ & & & & 1 & -4 & 5 & -80 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0 & \mathbf{0}^T & 0 \\ \mathbf{e}_1 & B_1 & \mathbf{e}_N \\ 0 & \mathbf{0}^T & 0 \end{bmatrix}, \quad \mathbf{e}_1 = [1, 0, \ldots, 0]^T, \quad \mathbf{e}_N = [0, \ldots, 0, 1]^T,$$

$$C = \frac{1}{h} \begin{bmatrix} 12 & 6 & & & \\ -3 & 0 & 3 & & \\ & \ddots & \ddots & \ddots & \\ & & -3 & 0 & 3 \\ & & & -6 & 12 \end{bmatrix}.$$

If we introduce $\Lambda = \operatorname{diag}(\lambda_i)_{i=1}^{N}$ and $Z = [z_{i,j}]_{i,j=1}^{N}$, where

$$\lambda_i = -4 \sin^2 \frac{i\pi}{2(N+1)}, \quad z_{i,j} = \left(\frac{2}{N+1} \right)^{1/2} \sin \frac{ij\pi}{N+1},$$

then (cf. section 4.5.3 in [29]) $Z^{-1}TZ = \Lambda$ and $Z^{-1} = Z$. Thus the system (2.3) can be solved using the MDA of section 2 with $\Lambda_A = (1/2)h^{-2}(\Lambda+12I)\Lambda$ and $\Lambda_B = \Lambda+6I$. The FFT routines SINTI and SINT from FFTPACK [27] are used in Step 1 and SINT in Step 3.

5. The Neumann BVP. We consider the Neumann BVP comprising (1.1) and (1.3). In the OSM, we seek $u_h^{\mathcal{N}} \in S^{\mathcal{N}} \otimes S^{\mathcal{N}}$ such that

$$(L_x + L_y)u_h^{\mathcal{N}}(x_i, y_j) = f_h(x_i, y_j), \quad 0 \le i, j \le N+1, \tag{5.1}$$

where, for $0 \le j \le N+1$,

$$L_x v(x_i, y_j) = \frac{1}{12}\begin{cases} 10D_x^2 v(x_i, y_j) + 2D_x^2 v(_{i+1}, y_j), \ i = 0, \\ D_x^2 v(x_{i-1}, y_j) + 10D_x^2 v(x_i, y_j) + D_x^2 v(x_{i+1}, y_j), \ 1 \le i \le N, \\ 2D_x^2 v(x_{i-1}, y_j) + 10D_x^2 v(x_i, y_j), \ i = N+1, \end{cases}$$

for $0 \le i \le N+1$, $L_y v(x_i, y_j)$ is defined by (4.4), and, for $0 \le j \le N+1$,

$$f_h(x_i, y_j) = \begin{cases} f(x_i, y_j) + \dfrac{h}{6} D_x f(x_i, y_j), & i = 0, \\ f(x_i, y_j), & 1 \le i \le N, \\ f(x_i, y_j) - \dfrac{h}{6} D_x f(x_i, y_j), & i = N+1. \end{cases}$$

As a basis for $S^{\mathcal{N}}$, we choose the cubic splines $\{B_m^{\mathcal{N}}\}_{m=0}^{N+1}$ defined in terms of the splines (2.1) by

$$\begin{cases} \mathcal{B}_0^{\mathcal{N}} = \mathcal{B}_0, \quad \mathcal{B}_1^{\mathcal{N}} = \mathcal{B}_1 + \mathcal{B}_{-1}, \\ \mathcal{B}_m^{\mathcal{N}} = \mathcal{B}_m, \quad 2 \le m \le N-1, \\ \mathcal{B}_N^{\mathcal{N}} = \mathcal{B}_N + \mathcal{B}_{N+2}, \quad \mathcal{B}_{N+1}^{\mathcal{N}} = \mathcal{B}_{N+1}. \end{cases} \tag{5.2}$$

Substituting

$$u_h^{\mathcal{N}}(x, y) = \sum_{m=0}^{N+1}\sum_{n=0}^{N+1} u_{m,n}\mathcal{B}_m^{\mathcal{N}}(x)\mathcal{B}_n^{\mathcal{N}}(y) \tag{5.3}$$

into (5.1) and using (2.7)–(2.8), we obtain (2.3), where

$$\mathbf{u} = [u_{0,0}, \ldots, u_{0,N+1}, \ldots, u_{N+1,0}, \ldots, u_{N+1,N+1}]^T,$$
$$\mathbf{f} = [f_{0,0}, \ldots, f_{0,N+1}, \ldots, f_{N+1,0}, \ldots, f_{N+1,N+1}]^T, \quad f_{i,j} = f_h(x_i, y_j),$$

$$A_1 = [a_{i,m}^{(1)}]_{i,m=0}^{N+1}, \quad A_2 = [a_{j,n}^{(2)}]_{j,n=0}^{N+1}, \tag{5.4}$$

$$a_{i,m}^{(1)} = \frac{1}{12}\begin{cases} 10\left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_i) + 2\left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_{i+1}), \ i = 0, \\ \left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_{i-1}) + 10\left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_i) + \left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_{i+1}), \ 1 \le i \le N, \\ 2\left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_{i-1}) + 10\left[\mathcal{B}_m^{\mathcal{N}}\right]''(x_i), \ i = N+1, \end{cases} \tag{5.5}$$

$$a^{(2)}_{j,n} = \frac{1}{12}\begin{cases} 13\left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_j) - 2\left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j+1}) + \left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j+2}), & j = 0, \\ \left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j-1}) + 10\left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_j) + \left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j+1}), & 1 \le j \le N, \\ \left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j-2}) - 2\left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_{j-1}) + 13\left[\mathcal{B}^{\mathcal{N}}_n\right]''(y_j), & j = N+1, \end{cases} \tag{5.6}$$

$$B_1 = B_2 = [b_{j,n}]_{j,n=0}^{N+1}, \quad b_{j,n} = \mathcal{B}^{\mathcal{N}}_n(y_j), \quad C = [c_{j,n}]_{j,n=0}^{N+1}, \quad c_{j,n} = [\mathcal{B}^{\mathcal{N}}_n]'(y_j). \tag{5.7}$$

Let $A = [a_{i,m}]_{i,m=0}^{N+1}$, where $a_{i,m} = [\mathcal{B}^{\mathcal{N}}_m]''(x_i)$. Then using (5.2) and (2.2), we have

$$A = 6h^{-2}T, \tag{5.8}$$

where the $(N+2)\times(N+2)$ matrix T is given by

$$T = \begin{bmatrix} -2 & 2 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & -2 & 1 & \\ & & & 2 & -2 & \end{bmatrix}.$$

It follows from (5.4)–(5.8), (5.2), and (2.2) that

$$A_1 = \frac{1}{12}(T + 12I)A = \frac{1}{2h^2}(T + 12I)T,$$

$$A_2 = \frac{1}{2h^2}\begin{bmatrix} -28 & 31 & -4 & 1 & & & \\ 8 & -17 & 8 & 1 & & & \\ 1 & 8 & -18 & 8 & 1 & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 8 & -18 & 8 & 1 \\ & & & 1 & 8 & -17 & 8 \\ & & & 1 & -4 & 31 & -28 \end{bmatrix},$$

$$B_1 = B_2 = T + 6I, \quad C = \frac{1}{h}\begin{bmatrix} 0 & 0 & & & \\ -3 & 0 & 3 & & \\ & \ddots & \ddots & \ddots & \\ & & -3 & 0 & 3 \\ & & & 0 & 0 \end{bmatrix}.$$

If we introduce $\Lambda = \operatorname{diag}(\lambda_i)_{i=1}^{N+2}$ and $Z = [z_{i,j}]_{i,j=1}^{N+2}$, where

$$\lambda_i = -4\sin^2\frac{(i-1)\pi}{2(N+1)}, \quad z_{i,j} = \left(\frac{2}{N+1}\right)^{1/2}\cos\frac{(i-1)(j-1)\pi}{N+1},$$

then (cf. section 4.5.5 in [29]) $Z^{-1}TZ = \Lambda$ and $Z^{-1} = DZD$, where the matrix $D = \operatorname{diag}(1/2, 1, \ldots, 1, 1/2)$. The system (2.3) can be solved using the MDA of section 2 with $\Lambda_A = (1/2)h^{-2}(\Lambda + 12I)\Lambda$ and $\Lambda_B = \Lambda + 6I$; the FFT routines COSTI and COST from FFTPACK [27] are used in Step 1 and COST in Step 3.

6. The mixed BVP. We consider the mixed BVP comprising (1.1) and (1.4). In this case, we introduce

$$S^{\mathcal{M}} = \{v \in S_3 : v(0) = v'(1) = 0\}.$$

In the OSM, we seek $u_h \in S^{\mathcal{M}} \otimes S^{\mathcal{M}}$ such that

$$D_x^2 D_y^2 u_h^{\mathcal{M}}(x_0, y_0) = D_y^2 f(x_0, y_0), \tag{6.1}$$

$$D_x^2 u_h^{\mathcal{M}}(x_0, y_j) - \frac{h^2}{12}[1 + a(y_j)] D_x^2 D_y^2 u_h^{\mathcal{M}}(x_0, y_j) = f_h(x_0, y_j),\ 1 \le j \le N+1, \tag{6.2}$$

$$(L_x + L_y) u_h^{\mathcal{M}}(x_i, y_j) = f_h(x_i, y_j),\ 1 \le i \le N+1,\ 0 \le j \le N+1, \tag{6.3}$$

where, for $0 \le j \le N+1$,

$$L_x v(x_i, y_j) = \frac{1}{12} \begin{cases} D_x^2 v(x_{i-1}, y_j) + 10 D_x^2 v(x_i, y_j) + D_x^2 v(x_{i+1}, y_j),\ 1 \le i \le N, \\ 2 D_x^2 v(x_{i-1}, y_j) + 10 D_x^2 v(x_i, y_j),\ i = N+1, \end{cases}$$

for $1 \le i \le N+1$, $L_y v(x_i, y_j)$ is defined by (4.4), and, for $0 \le j \le N+1$,

$$f_h(x_i, y_j) = \begin{cases} \left[\Delta f(x_i, y_j) - b(y_j) D_y f(x_i, y_j) - c(y_j) f(x_i, y_j)\right], i = 0, \\ f(x_i, y_j), \quad 1 \le i \le N, \\ f(x_i, y_j) - \dfrac{h}{6} D_x f(x_i, y_j), \quad i = N+1. \end{cases}$$

As a basis for $S^{\mathcal{M}}$, we choose the cubic splines $\{B_m^{\mathcal{M}}\}_{m=0}^{N+1}$ defined in terms of the splines (2.1) by

$$\begin{cases} \mathcal{B}_0^{\mathcal{M}} = \mathcal{B}_0 - 4\mathcal{B}_{-1}, \quad \mathcal{B}_1^{\mathcal{M}} = \mathcal{B}_1 - \mathcal{B}_{-1}, \\ \mathcal{B}_m^{\mathcal{M}} = \mathcal{B}_m, \quad 2 \le m \le N-1, \\ \mathcal{B}_N^{\mathcal{M}} = \mathcal{B}_N + \mathcal{B}_{N+2}, \quad \mathcal{B}_{N+1}^{\mathcal{M}} = \mathcal{B}_{N+1}. \end{cases} \tag{6.4}$$

It follows from (6.4) and (2.2) that

$$[\mathcal{B}_m^{\mathcal{M}}]''(x_0) = \begin{cases} -36h^{-2}, & m = 0, \\ 0, & 1 \le m \le N+1. \end{cases} \tag{6.5}$$

Substituting

$$u_h^{\mathcal{M}}(x, y) = \sum_{m=0}^{N+1} \sum_{n=0}^{N+1} u_{m,n} \mathcal{B}_m^{\mathcal{M}}(x) \mathcal{B}_n^{\mathcal{M}}(y) \tag{6.6}$$

into (6.1) and using (6.5), we obtain

$$u_{0,0} = \frac{h^4}{1296} D_y^2 f(x_0, y_0).$$

Substituting (6.6) into (6.2) and using (6.5), we have

$$\sum_{n=1}^{N+1} u_{0,n} \left\{ \frac{h^2}{12}[1 + a(y_j)][\mathcal{B}_n^{\mathcal{M}}]''(y_j) - \mathcal{B}_n^{\mathcal{M}}(y_j) \right\} = \frac{h^2}{36} f_h(x_0, y_j)$$
$$- u_{i,0} \left\{ \frac{h^2}{12}[1 + a(y_j)][\mathcal{B}_0^{\mathcal{M}}]''(y_j) - \mathcal{B}_0^{\mathcal{M}}(y_j) \right\}, \quad 1 \le j \le N+1.$$

These equations yield a linear system for $\{u_{0,n}\}_{n=1}^{N+1}$ with coefficient matrix

$$\frac{h^2}{12}D_{1+a}A - B_1, \tag{6.7}$$

where

$$A = [a_{j,n}]_{j,n=1}^{N+1}, \quad a_{j,n} = [\mathcal{B}_n^{\mathcal{M}}]''(y_j), \quad B_1 = [b_{j,n}^{(1)}]_{j,n=1}^{N+1}, \quad b_{j,n}^{(1)} = \mathcal{B}_n^{\mathcal{M}}(y_j). \tag{6.8}$$

Using (6.4) and (2.2), we have

$$A = 6h^{-2}T, \quad B_1 = T + 6I, \tag{6.9}$$

where the $(N+1)\times(N+1)$ matrix T is given by

$$T = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 2 & -2 \end{bmatrix}.$$

Therefore, (6.7) and (6.9) yield the tridiagonal matrix $\frac{1}{2}D_{a-1}T - 6I$.

Substituting (6.6) into (6.3) and using (2.7)–(2.8), we obtain (2.3), where

$$\mathbf{u} = [u_{1,0}, \ldots, u_{1,N+1}, \ldots, u_{N+1,0}, \ldots, u_{N+1,N+1}]^T,$$

$$\mathbf{f} = [f_{1,0}, \ldots, f_{1,N+1}, \ldots, f_{N+1,0}, \ldots, f_{N+1,N+1}]^T,$$

$$f_{i,j} = f_h(x_i, y_j) - \sum_{n=0}^{N+1} u_{0,n}\left[L_x\mathcal{B}_0^{\mathcal{M}}(x_i)\mathcal{B}_n^{\mathcal{M}}(y_j) + \mathcal{B}_0^{\mathcal{M}}(x_i)L_y\mathcal{B}_n^{\mathcal{M}}(y_j)\right],$$

$$A_1 = [a_{i,m}^{(1)}]_{i,m=1}^{N+1}, \quad A_2 = [a_{j,n}^{(2)}]_{j,n=0}^{N+1}, \tag{6.10}$$

$$a_{i,m}^{(1)} = \frac{1}{12}\begin{cases} \left[\mathcal{B}_m^{\mathcal{M}}\right]''(x_{i-1}) + 10\left[\mathcal{B}_m^{\mathcal{M}}\right]''(x_i) + \left[\mathcal{B}_m^{\mathcal{M}}\right]''(x_{i+1}), & 1 \le i \le N, \\ 2\left[\mathcal{B}_m^{\mathcal{M}}\right]''(x_{i-1}) + 10\left[\mathcal{B}_m^{\mathcal{M}}\right]''(x_i), & i = N+1, \end{cases} \tag{6.11}$$

$$a_{j,n}^{(2)} = \frac{1}{12}\begin{cases} 13\left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_j) - 2\left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j+1}) + \left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j+2}), & j = 0, \\ \left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j-1}) + 10\left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_j) + \left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j+1}), & 1 \le j \le N, \\ \left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j-2}) - 2\left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_{j-1}) + 13\left[\mathcal{B}_n^{\mathcal{M}}\right]''(y_j), & j = N+1, \end{cases} \tag{6.12}$$

B_1 is defined in (6.8) and (6.9),

$$B_2 = [b_{j,n}^{(2)}]_{j,n=0}^{N+1}, \quad b_{j,n}^{(2)} = \mathcal{B}_n^{\mathcal{M}}(y_j), \quad C = [c_{j,n}]_{j,n=0}^{N+1}, \quad c_{j,n} = [\mathcal{B}_n^{\mathcal{M}}]'(y_j). \tag{6.13}$$

It follows from (6.9)–(6.13), (6.4), and (2.2) that

$$A_1 = \frac{1}{12}\begin{bmatrix} 1 & 10 & 1 & & & \\ & 1 & 10 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 10 & 1 \\ & & & & 2 & 10 \end{bmatrix}\begin{bmatrix} \mathbf{0}^T \\ A \end{bmatrix} = \frac{1}{2h^2}(T + 12I)T,$$

$$A_2 = \frac{1}{2h^2}\begin{bmatrix} -80 & 5 & -4 & 1 & & & & \\ 4 & -19 & 8 & 1 & & & & \\ 1 & 8 & -18 & 8 & 1 & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 8 & -18 & 8 & 1 \\ & & & & 1 & 8 & -17 & 8 \\ & & & & 1 & -4 & 31 & -28 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} 0 & \mathbf{0}^T \\ \mathbf{e}_1 & B_1 \end{bmatrix}, \quad \mathbf{e}_1 = [1, 0, \ldots, 0]^T, \quad C = \frac{1}{h}\begin{bmatrix} 12 & 6 & & & \\ -3 & 0 & 3 & & \\ & \ddots & \ddots & \ddots & \\ & & -3 & 0 & 3 \\ & & & 0 & 0 \end{bmatrix}.$$

If we introduce $\Lambda = \operatorname{diag}(\lambda_i)_{i=1}^{N+1}$ and $Z = [z_{i,j}]_{i,j=1}^{N+1}$, where

$$\lambda_i = -4\sin^2\frac{(2i-1)\pi}{4(N+1)}, \quad z_{i,j} = \left(\frac{2}{N+1}\right)^{1/2}\sin\frac{i(2j-1)\pi}{2(N+1)},$$

then (cf. section 4.5.4 and (4.4.4) and (4.4.24) in [29]) $Z^{-1}TZ = \Lambda$ and $Z^{-1} = Z^T\operatorname{diag}(1, \ldots, 1, 1/2)$. The system (2.3) can be solved using the MDA of section 2 with $\Lambda_A = 6h^{-2}\Lambda$ $\Lambda_A = (1/2)h^{-2}(\Lambda + 12I)\Lambda$ and $\Lambda_B = \Lambda + 6I$. The FFT routines SINQI and SINQF are used in Step 1, and the FFT routine SINQB is used in Step 3, all from FFTPACK [27].

7. The periodic BVP. We consider the periodic BVP comprising (1.1) and (1.5). In the OSM, we seek $u_h^{\mathcal{P}} \in S^{\mathcal{P}} \otimes S^{\mathcal{P}}$ such that

$$(L_x + L_y)u_h^{\mathcal{P}}(x_i, y_j) = f_h(x_i, y_j), \qquad \begin{matrix} 0 \le i \le N, & 0 \le j \le N+1, \\ i = N+1, & 0 \le j \le N, \end{matrix} \tag{7.1}$$

and such that

$$\begin{aligned} &D_x^2 D_y^2 u_h^{\mathcal{P}}(x_0, y_{N+1}) - D_x^2 D_y^2 u_h^{\mathcal{P}}(x_{N+1}, y_{N+1}) \\ &\quad = D_y^2 f(x_0, y_{N+1}) - D_y^2 f(x_{N+1}, y_{N+1}), \end{aligned} \tag{7.2}$$

where, for $0 \le j \le N+1$,

$$L_x v(x_i, y_j) = \frac{1}{12}\begin{cases} 10D_x^2 v(x_i, y_j) + D_x^2 v(x_{i+1}, y_j) + D_x^2 v(x_N, y_j), & i = 0, \\ D_x^2 v(x_{i-1}, y_j) + 10D_x^2 v(x_i, y_j) + D_x^2 v(x_{i+1}, y_j), & 1 \le i \le N-1, \\ D_x^2 v(x_0, y_j) + D_x^2 v(x_{i-1}, y_j) + 10D_x^2 v(x_i, y_j), & i = N, \\ D_x^2 v(x_1, y_j) + D_x^2 v(x_{i-1}, y_j) + 10D_x^2 v(x_i, y_j), & i = N+1, \end{cases}$$

and for $0 \le i \le N+1$, $L_y v(x_i, y_j)$ is defined by (4.4). For $0 \le j \le N+1$,

$$f_h(x_i, y_j) = \begin{cases} \dfrac{1}{12}[11f(x_0, y_j) + f(x_{N+1}, y_j)] + \tilde{f}_j + 5\hat{f}_j, & i = 0, \\ f(x_i, y_j), & 1 \le i \le N-1, \\ f(x_N, y_j) + \dfrac{1}{12}[f(x_0, y_j) - f(x_{N+1}, y_j)] + \hat{f}_j, & i = N, \\ \dfrac{1}{12}[11f(x_{N+1}, y_j) + f(x_0, y_j)] + \tilde{f}_j - 5\hat{f}_j, & i = N+1, \end{cases}$$

where

$$\tilde{f}_j = \frac{h}{12}\left[D_x f(x_0, y_j) - D_x f(x_{N+1}, y_j)\right],$$

$$\hat{f}_j = \frac{h^2}{144}\{D_x^2 f(x_{N+1}, y_j) - D_x^2 f(x_0, y_j) + a(y_j)[D_y^2 f(x_0, y_j) - D_y^2 f(x_{N+1}, y_j)]$$
$$+ b(y_j)[D_y f(x_0, y_j) - D_y f(x_{N+1}, y_j)] + c(y_j)[f(x_0, y_j) - f(x_{N+1}, y_j)]\}.$$

As a basis for $S^{\mathcal{P}}$, we choose the cubic splines $\{B_m^{\mathcal{P}}\}_{m=0}^{N+1}$ defined in terms of the splines (2.1) by

$$\begin{cases} \mathcal{B}_0^{\mathcal{P}} = \mathcal{B}_0 + \mathcal{B}_{N+1}, \quad \mathcal{B}_1^{\mathcal{P}} = \mathcal{B}_1 + \mathcal{B}_{N+2}, \\ \mathcal{B}_m^{\mathcal{P}} = \mathcal{B}_m, \quad 2 \le m \le N-1, \\ \mathcal{B}_N^{\mathcal{P}} = \mathcal{B}_N + \mathcal{B}_{-1}, \quad \mathcal{B}_{N+1}^{\mathcal{P}} = 4(\mathcal{B}_{N+2} - \mathcal{B}_{-1}) - \mathcal{B}_{N+1} + \mathcal{B}_0. \end{cases} \tag{7.3}$$

It follows from (7.3) and (2.2) that

$$\begin{cases} [\mathcal{B}_m^{\mathcal{P}}]''(x_0) = [\mathcal{B}_m^{\mathcal{P}}]''(x_{N+1}), \quad 0 \le m \le N, \\ [\mathcal{B}_{N+1}^{\mathcal{P}}]''(x_0) = -[\mathcal{B}_{N+1}^{\mathcal{P}}]''(x_{N+1}) = -36h^{-2}. \end{cases} \tag{7.4}$$

For $0 \le j \le N$, subtracting (7.1) with $i = N+1$ from (7.1) with $i = 0$, using $u_h^{\mathcal{P}}(x_0, y) = u_h^{\mathcal{P}}(x_{N+1}, y)$, $0 \le y \le 1$, substituting

$$u_h^{\mathcal{P}}(x, y) = \sum_{m=0}^{N+1}\sum_{n=0}^{N+1} u_{m,n}\mathcal{B}_m^{\mathcal{P}}(x)\mathcal{B}_n^{\mathcal{P}}(y), \tag{7.5}$$

and using (7.4), we obtain

$$\sum_{n=0}^{N+1} u_{N+1,n}\mathcal{B}_n^{\mathcal{P}}(y_j) = -\frac{h^2}{60}\left[f_h(x_0, y_j) - f_h(x_{N+1}, y_j)\right], \quad 0 \le j \le N. \tag{7.6}$$

In a similar way, (7.2) gives

$$\sum_{n=0}^{N+1} u_{N+1,n}[\mathcal{B}_n^{\mathcal{P}}]''(y_{N+1}) = -\frac{h^2}{72}\left[D_y^2 f(x_0, y_{N+1}) - D_y^2 f(x_{N+1}, y_{N+1})\right]. \tag{7.7}$$

Equations (7.6) and (7.7) yield a linear system for $\{u_{N+1,n}\}_{n=0}^{N+1}$ with the coefficient matrix

$$B = [b_{j,n}]_{j,n=0}^{N+1}, \quad b_{j,n} = \begin{cases} \mathcal{B}_n^{\mathcal{P}}(y_j), & 0 \le j \le N, \\ \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_j), & j = N+1. \end{cases}$$

It follows from (7.4), (7.3), and (2.2) that

$$B = \begin{bmatrix} B_1 & \mathbf{a} \\ \mathbf{b}^T & 36h^{-2} \end{bmatrix}, \quad \mathbf{a} = [0, 1, 0, \ldots, 0, -1]^T, \quad \mathbf{b} = 6h^{-2}[-2, 1, 0, \ldots, 0, 1]^T,$$

and

$$B_1 = [b_{j,n}^{(1)}]_{j,n=0}^{N}, \quad b_{j,n}^{(1)} = \mathcal{B}_n^{\mathcal{P}}(y_j). \tag{7.8}$$

Using (7.3) and (2.2), we have

$$B_1 = T + 6I, \tag{7.9}$$

where the $(N+1)\times(N+1)$ matrix T is given by

$$T = \begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix}.$$

A system with coefficient matrix B can be easily solved using a Schur complement approach.

Substituting (7.5) into (7.1) with $0 \le i \le N$, $0 \le j \le N+1$, and using (2.7)–(2.8), we obtain (2.3) with

$$\mathbf{u} = [u_{0,0}, \dots, u_{0,N+1}, \dots, u_{N,0}, \dots, u_{N,N+1}]^T, \tag{7.10}$$

$$\mathbf{f} = [f_{0,0}, \dots, f_{0,N+1}, \dots, f_{N,0}, \dots, f_{N,N+1}]^T, \tag{7.11}$$

$$f_{i,j} = f_h(x_i, y_j) - \sum_{n=0}^{N+1} u_{N+1,n} \left[L_x \mathcal{B}_{N+1}^{\mathcal{P}}(x_i) \mathcal{B}_n^{\mathcal{P}}(y_j) + \mathcal{B}_{N+1}^{\mathcal{P}}(x_i) L_y \mathcal{B}_n^{\mathcal{P}}(y_j) \right],$$

$$A_1 = [a_{i,m}^{(1)}]_{i,m=0}^{N}, \quad A_2 = [a_{j,n}^{(2)}]_{j,n=0}^{N+1}, \tag{7.12}$$

$$a_{i,m}^{(1)} = \frac{1}{12} \begin{cases} 10 \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_i) + \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_{i+1}) + \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_N), & i = 0, \\ \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_{i-1}) + 10 \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_i) + \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_{i+1}), & 1 \le i \le N-1, \\ \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_0) + \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_{i-1}) + 10 \left[\mathcal{B}_m^{\mathcal{P}}\right]''(x_i), & i = N, \end{cases} \tag{7.13}$$

$$a_{j,n}^{(2)} = \frac{1}{12} \begin{cases} 13 \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_j) - 2 \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j+1}) + \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j+2}), & j = 0, \\ \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j-1}) + 10 \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_j) + \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j+1}), & 1 \le j \le N, \\ \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j-2}) - 2 \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_{j-1}) + 13 \left[\mathcal{B}_n^{\mathcal{P}}\right]''(y_j), & j = N+1, \end{cases} \tag{7.14}$$

B_1 is defined in (7.8) and (7.9),

$$B_2 = [b_{j,n}^{(2)}]_{j,n=0}^{N+1}, \quad b_{j,n}^{(2)} = \mathcal{B}_n^{\mathcal{P}}(y_j), \quad C = [c_{j,n}]_{j,n=0}^{N+1}, \quad c_{j,n} = [\mathcal{B}_n^{\mathcal{P}}]'(y_j). \tag{7.15}$$

Let $A = [a_{i,m}]_{i,m=0}^{N}$, where $a_{i,m} = [\mathcal{B}_m^{\mathcal{P}}]''(x_i)$. Then using (7.3) and (2.2), we have

$$A = 6h^{-2}T. \tag{7.16}$$

It follows from (7.12)–(7.16), (7.3), and (2.2) that

$$A_1 = \frac{1}{12}(T + 12I)A = \frac{1}{2h^2}(T + 12I)T,$$

$$A_2 = \frac{1}{2h^2}\begin{bmatrix} -28 & 18 & -4 & 1 & & & & 13 & -80 \\ 8 & -18 & 8 & 1 & & & & 1 & 4 \\ 1 & 8 & -18 & 8 & 1 & & & & 1 \\ & 1 & 8 & -18 & 8 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 8 & -18 & 8 & 1 & \\ 1 & & & & 1 & 8 & -18 & 8 & -1 \\ 8 & 1 & & & & 1 & 8 & -18 & -4 \\ -28 & 13 & & & & 1 & -4 & 18 & 80 \end{bmatrix},$$

$$B_2 = \begin{bmatrix} B_1 & \mathbf{c} \\ \mathbf{d}^T & 0 \end{bmatrix}, \quad \mathbf{c} = [0, 1, 0, \ldots, 0, -1]^T, \quad \mathbf{d} = [4, 1, 0, \ldots, 0, 1]^T,$$

$$C = \frac{1}{h}\begin{bmatrix} 0 & 3 & & -3 & 12 \\ -3 & 0 & 3 & & -3 \\ & \ddots & \ddots & \ddots & \\ 3 & & -3 & 0 & -3 \\ 0 & 3 & & -3 & 12 \end{bmatrix}.$$

If we introduce

$$\Lambda = \operatorname{diag}(\lambda_i)_{i=1}^{N+1}, \quad \lambda_i = -4\sin^2\frac{(i-1)\pi}{N+1},$$

and $Z = \left(\frac{2}{N+1}\right)^{1/2}[z_{i,j}]_{i,j=1}^{N+1}$, where, for $1 \le i \le N+1$, and N even,

$$z_{i,1} = \frac{1}{\sqrt{2}}, \quad z_{i,j+1} = \cos\frac{2(i-1)j\pi}{N+1}, \quad z_{i,N+2-j} = \sin\frac{2(i-1)j\pi}{N+1}, \quad 1 \le j \le \frac{N}{2},$$

and, for N odd,

$$z_{i,1} = \frac{1}{\sqrt{2}}, \quad z_{i,(N+3)/2} = \frac{1}{\sqrt{2}}(-1)^{i-1},$$

$$z_{i,j+1} = \cos\frac{2(i-1)j\pi}{N+1}, \quad z_{i,N+2-j} = \sin\frac{2(i-1)j\pi}{N+1}, \quad 1 \le j \le \frac{N-1}{2},$$

then (cf. section 1.5.4 in [23] and section 4.5.6 in [29]) $Z^{-1}TZ = \Lambda$ and $Z^{-1} = Z^T$. The system (2.3) can be solved using the MDA of section 2 with $\Lambda_A = (1/2)h^{-2}(\Lambda + 12I)\Lambda$ and $\Lambda_B = \Lambda + 6I$. The FFTPACK [27] routines RFFTI and RFFTF are used in Step 1, and RFFTB in Step 3.

8. Numerical experiments. Numerical results are presented in Tables 8.1–8.4 for the new OSM approach applied to the equation

$$D_x^2 u + \cos(y) D_y^2 u - \sin(y) D_y u + e^y u = f(x, y),$$

with exact solution

$$u(x, y) = e^{xy}(x^2 - x)(y^2 - y)$$

TABLE 8.1
Errors for the Dirichlet BVP.

N	$E_n(N)$	R_n	$E_g(N)$	R_g
u				
16	0.190(−7)		0.175(−6)	
32	0.662(−9)	5.064	0.117(−7)	4.077
64	0.510(−10)	3.782	0.761(−9)	4.030
128	0.346(−11)	3.925	0.489(−10)	4.004
256	0.220(−12)	3.999	0.308(−11)	4.010
D_xu				
16	0.165(−5)		0.690(−5)	
32	0.665(−7)	4.841	0.842(−6)	3.170
64	0.239(−8)	4.907	0.105(−6)	3.076
128	0.115(−9)	4.424	0.129(−7)	3.099
256	0.723(−11)	4.015	0.163(−8)	3.004
D_yu				
16	0.117(−5)		0.705(−5)	
32	0.832(−7)	3.989	0.876(−6)	3.144
64	0.566(−8)	3.964	0.108(−6)	3.089
128	0.371(−9)	3.976	0.129(−7)	3.099
256	0.238(−10)	3.986	0.163(−8)	3.004
D_x^2u				
16	0.937(−3)		0.942(−3)	
32	0.261(−3)	1.927	0.261(−3)	1.934
64	0.681(−4)	1.983	0.681(−4)	1.984
128	0.173(−4)	1.996	0.173(−4)	1.996
256	0.437(−5)	1.999	0.437(−5)	1.999
D_y^2u				
16	0.925(−3)		0.927(−3)	
32	0.256(−3)	1.939	0.256(−3)	1.942
64	0.671(−4)	1.973	0.671(−4)	1.973
128	0.172(−4)	1.987	0.172(−4)	1.987
256	0.435(−5)	1.994	0.435(−5)	1.994
$D_{xy}u$				
16	0.109(−3)		0.166(−3)	
32	0.806(−5)	3.921	0.209(−4)	3.129
64	0.551(−6)	3.956	0.264(−5)	3.052
128	0.362(−7)	3.975	0.315(−6)	3.100
256	0.232(−8)	3.986	0.398(−7)	3.002

for the Dirichlet BVP,

$$u(x,y) = e^{xy}(x^2 - x)^2(y^2 - y)^2$$

for Neumann and mixed BVPs, and

$$u(x,y) = u = [e^{2x} + (1 - e^2)x^2][e^{2y} + (1 - e^2)y^2]$$

for the periodic BVP. In each table, for several values of N, six quantities are given for the errors in u, D_xu, D_yu, D_x^2u, D_y^2u, and $D_{xy}u$:

- $E_n(N)$—the maximum absolute error at the nodes of the partition $\rho_x \times \rho_y$.
- $R_n(N)$—the experimental rate of convergence of the nodal error computed from

$$R_n(N) = \frac{\log(E_n(N/2)/E_n(N))}{\log((N+1)/((N/2)+1))}. \tag{8.1}$$

TABLE 8.2
Errors for the Neumann BVP.

N	$E_n(N)$	R_n	$E_g(N)$	R_g
u				
16	0.623(−6)		0.673(−6)	
32	0.449(−7)	3.966	0.499(−7)	3.922
64	0.302(−8)	3.983	0.342(−8)	3.956
128	0.196(−9)	3.991	0.224(−9)	3.974
256	0.125(−10)	3.996	0.144(−10)	3.986
D_xu				
16	0.139(−5)		0.964(−5)	
32	0.104(−6)	3.908	0.133(−5)	2.991
64	0.716(−8)	3.950	0.174(−6)	2.995
128	0.470(−9)	3.974	0.223(−7)	2.998
256	0.301(−10)	3.987	0.282(−8)	2.999
D_yu				
16	0.422(−5)		0.127(−4)	
32	0.334(−6)	3.826	0.157(−5)	3.148
64	0.237(−7)	3.903	0.192(−6)	3.104
128	0.158(−8)	3.946	0.235(−7)	3.063
256	0.102(−9)	3.972	0.290(−8)	3.035
D_x^2u				
16	0.168(−2)		0.169(−2)	
32	0.454(−3)	1.973	0.455(−3)	1.983
64	0.118(−3)	1.987	0.118(−3)	1.991
128	0.301(−4)	1.995	0.301(−4)	1.995
256	0.759(−5)	1.998	0.759(−5)	1.998
D_y^2u				
16	0.155(−2)		0.156(−2)	
32	0.435(−3)	1.916	0.436(−3)	1.926
64	0.115(−3)	1.958	0.115(−3)	1.960
128	0.297(−4)	1.979	0.297(−4)	1.979
256	0.754(−5)	1.989	0.754(−5)	1.989
$D_{xy}u$				
16	0.179(−4)		0.697(−4)	
32	0.150(−5)	3.745	0.874(−5)	3.132
64	0.111(−6)	3.842	0.110(−5)	3.052
128	0.761(−8)	3.906	0.139(−6)	3.019
256	0.450(−9)	3.951	0.175(−7)	3.010

- $E_g(N)$—the maximum absolute error computed on a 1000×1000 grid taken as an approximation to the L^∞ norm of the error.
- $R_g(N)$—the experimental rate of convergence of the global error computed as in (8.1) with E_g replacing E_n.

The numerical results were obtained in double precision on an IBM RISC F80-7025 (450 MHz).

The global errors in the approximations to u, D_xu, D_yu, D_x^2u, and D_y^2u behave as expected. In each problem, the global error in the approximation to $D_{xy}u$ is third order, as in the bicubic spline interpolant (see Theorem 4 in [10]). Moreover, the nodal errors in D_xu, D_yu, and $D_{xy}u$ exhibit superconvergent rates of convergence of order 4 as in the TSM of [6]. The proof of superconvergence results for the new OSM for the solution of one- and two-dimensional problems remains an open problem.

In Table 8.5, we give the CPU times for the new OSM for solving the Dirichlet problem, for the values of N for which FFTs are most efficient. The times obtained

TABLE 8.3
Errors for the mixed BVP.

N	$E_n(N)$	R_n	$E_g(N)$	R_g
u				
16	0.133(−5)		0.146(−5)	
32	0.975(−7)	3.939	0.108(−6)	3.928
64	0.661(−8)	3.970	0.733(−8)	3.963
128	0.430(−9)	3.985	0.479(−9)	3.980
256	0.275(−10)	3.992	0.306(−10)	3.990
D_xu				
16	0.235(−5)		0.962(−5)	
32	0.172(−6)	3.938	0.132(−5)	2.990
64	0.117(−7)	3.967	0.174(−6)	2.995
128	0.764(−9)	3.983	0.223(−7)	2.997
256	0.488(−10)	3.991	0.282(−8)	2.999
D_yu				
16	0.427(−5)		0.127(−4)	
32	0.336(−6)	3.833	0.157(−5)	3.151
64	0.238(−7)	3.907	0.192(−6)	3.105
128	0.159(−8)	3.949	0.235(−7)	3.064
256	0.103(−9)	3.974	0.290(−8)	3.035
D_x^2u				
16	0.168(−2)		0.169(−2)	
32	0.454(−3)	1.972	0.455(−3)	1.982
64	0.118(−3)	1.987	0.118(−3)	1.991
128	0.301(−4)	1.995	0.301(−4)	1.995
256	0.759(−5)	1.997	0.759(−5)	1.998
D_y^2u				
16	0.1 (−2)		0.156(−2)	
32	0.435(−3)	1.915	0.435(−3)	1.925
64	0.115(−3)	1.958	0.115(−3)	1.960
128	0.297(−4)	1.979	0.297(−4)	1.979
256	0.754(−5)	1.989	0.754(−5)	1.989
$D_{xy}u$				
16	0.179(−4)		0.697(−4)	
32	0.149(−5)	3.743	0.873(−5)	3.132
64	0.111(−6)	3.841	0.110(−5)	3.051
128	0.760(−8)	3.905	0.139(−6)	3.018
256	0.499(−9)	3.950	0.175(−7)	3.010

are comparable to the corresponding times reported in [6]. The results for the new OSM for the other BVPs are similar.

The results of extensive numerical experiments are given in [7], where it is shown that the results obtained using the new OSM are of comparable accuracy to those obtained using the TSM. However, the new method is asymptotically twice as fast as the TSM.

9. Concluding remarks. New one step nodal bicubic spline collocation methods for approximating certain separable equations on the unit square subject to various boundary conditions are derived. These methods involve not only a high-order perturbation of the differential operator at the interior collocation points but also perturbations of the differential operator and the right-hand side at the boundary collocation points. This ensures that the totality of collocation equations is amenable to solution by MDAs with FFTs. Numerical experimentation demonstrates optimal order accuracy of the new methods. Moreover, the numerical results exhibit global

TABLE 8.4
Errors for the periodic BVP.

N	$E_n(N)$	R_n	$E_g(N)$	R_g
u				
16	0.192(−4)		0.192(−4)	
32	0.142(−5)	3.927	0.142(−5)	3.928
64	0.964(−7)	3.963	0.964(−7)	3.963
128	0.630(−8)	3.981	0.629(−8)	3.982
256	0.402(−9)	3.993	0.402(−8)	3.992
D_xu				
16	0.216(−4)		0.232(−3)	
32	0.164(−5)	3.889	0.323(−4)	2.967
64	0.113(−6)	3.944	0.427(−5)	2.988
128	0.745(−8)	3.971	0.547(−6)	2.997
256	0.477(−9)	3.985	0.693(−7)	2.999
D_yu				
16	0.670(−4)		0.279(−3)	
32	0.522(−5)	3.848	0.360(−4)	3.087
64	0.366(−6)	3.920	0.452(−5)	3.060
128	0.243(−7)	3.959	0.564(−6)	3.037
256	0.156(−8)	3.980	0.703(−7)	3.020
D_x^2u				
16	0.421(−1)		0.422(−1)	
32	0.112(−1)	1.990	0.113(−1)	1.993
64	0.291(−2)	1.996	0.291(−2)	1.996
128	0.739(−3)	1.998	0.739(−3)	1.998
256	0.186(−3)	1.999	0.186(−3)	1.999
D_y^2u				
16	0.385(−1)		0.384(−1)	
32	0.107(−1)	1.926	0.107(−2)	1.925
64	0.284(−2)	1.962	0.284(−2)	1.961
128	0.730(−3)	1.980	0.730(−3)	1.980
256	0.185(−3)	1.990	0.185(−3)	1.990
$D_{xy}u$				
16	0.225(−3)		0.719(−3)	
32	0.165(−4)	3.943	0.101(−3)	2.957
64	0.112(−5)	3.966	0.135(−4)	2.972
128	0.733(−7)	3.980	0.168(−5)	3.042
256	0.469(−8)	3.988	0.217(−6)	2.966

TABLE 8.5
CPU times for the Dirichlet BVP.

N	63	127	255	511	1023
CPU time secs	0.04	0.17	0.68	2.88	12.03

third order accuracy of the approximations to $D_{xy}u$ and superconvergence at the nodes of the approximations to D_xu, D_yu, and $D_{xy}u$. A rigorous convergence analysis of the methods has yet to be derived. Future work includes the extension of the new one step nodal bicubic spline collocation approach to inhomogeneous boundary conditions, biharmonic problems, and problems in three dimensions. In the latter, the correct specification of the corner collocation equations and the construction of the collocation equations in the periodic case will require careful consideration.

Acknowledgment. The third author wishes to thank Y-S. Smyrlis and T. Tsangaris of the University of Cyprus for their assistance during the preparation of this paper.

REFERENCES

[1] D. Archer, *Some Collocation Methods for Differential Equations*, Ph.D. thesis, Rice University, Houston, TX, 1973.

[2] D. Archer, *An $O(h^4)$ cubic spline collocation method for quasilinear parabolic equations*, SIAM J. Numer. Anal., 14 (1977), pp. 620–637.

[3] A. Averbuch, M. Israeli, and L. Vozvoi, *A fast Poisson solver of arbitrary order accuracy in rectangular regions*, SIAM J. Sci. Comput., 19 (1998), pp. 933–952.

[4] B. Bialecki and G. Fairweather, *Matrix decomposition methods for separable elliptic boundary value problems in two space dimensions*, J. Comput. Appl. Math., 46 (1993), pp. 369–386.

[5] B. Bialecki and G. Fairweather, *Orthogonal spline collocation methods for partial differential equations*, J. Comput. Appl. Math., 128 (2001), pp. 55–82.

[6] B. Bialecki, G. Fairweather, and A. Karageorghis, *Matrix decomposition algorithms for modified spline collocation for Helmholtz problems*, SIAM J. Sci. Comput., 24 (2003), pp. 1733–1753.

[7] B. Bialecki, G. Fairweather, and A. Karageorghis, *Optimal Superconvergent One Step Nodal Cubic Spline Collocation Methods*, Technical Report TR/04/2004, Department of Mathematics and Statistics, University of Cyprus, Nicosia, Cyprus.

[8] B. Bialecki and A. Karageorghis, *Legendre Gauss spectral collocation for the Helmholtz equation on a rectangle*, Numer. Algorithms, 36 (2004), pp. 203–227.

[9] E. Braverman, M. Israeli, A. Averbuch, and L. Vozovoi, *A fast 3D Poisson solver of arbitrary order accuracy*, J. Comput. Phys., 144 (1998), pp. 109–136.

[10] R. E. Carlson and C. A. Hall, *Error bounds for bicubic spline interpolation*, J. Approx. Theory, 7 (1973), pp. 41–47.

[11] C. C. Christara and K. S. Ng, *Fast Fourier transform solvers and preconditioners for quadratic spline collocation*, BIT, 42 (2002), pp. 702–739.

[12] A. Constas, *Fast Fourier Transform Solvers for Quadratic Spline Collocation*, M.Sc. thesis, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada, 1996.

[13] J. W. Daniel and B. K. Swartz, *Extrapolated collocation for two-point boundary value problems using cubic splines*, J. Inst. Math. Appl., 16 (1975), pp. 161–174.

[14] D. J. Fyfe, *The use of cubic splines in the solution of two-point boundary value problems*, Comput. J., 12 (1969/1970), pp. 188–192.

[15] B. Gustafsson and L. Hemmingsson–Frändén, *A fast domain decomposition high order Poisson solver*, J. Sci. Comput., 14 (1999), pp. 223–243.

[16] A. Hadjidimos, E. N. Houstis, J. R. Rice, and E. Vavalis, *Analysis of iterative line spline collocation methods for elliptic partial differential equations*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 508–521.

[17] E. N. Houstis, E. A. Vavalis, and J. R. Rice, *Convergence of $O(h^4)$ cubic spline collocation methods for elliptic partial differential equations*, SIAM J. Numer. Anal., 25 (1988), pp. 54–74.

[18] D. B. Knudson, *A Piecewise Hermite Bicubic Finite Element Galerkin Method for the Biharmonic Dirichlet Problem*, Ph.D. thesis, Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO, 1997.

[19] M.-C. Lai and W.-C. Wang, *Fast direct solvers for Poisson equation on 2D polar and spherical geometries*, Numer. Methods Partial Differential Equations, 18 (2002), pp. 56–68.

[20] A. D. Lyashko and S. I. Solov'yev, *Fourier method of solution of FE systems with Hermite elements for Poisson equation*, Soviet J. Numer. Anal. Math. Modelling, 6 (1991), pp. 121–129.

[21] J. Martikainen, T. Rossi, and J. Toivanen, *A fast direct solver for elliptic problems with a divergence constraint*, Numer. Linear Algebra Appl., 9 (2002), pp. 629–652.

[22] W. J. Pickering and P. J. Harley, *On Robbins boundary conditions, elliptic equations, and FFT methods*, J. Comput. Phys., 122 (1995), pp. 380–383.

[23] A. A Samarskii and E. S. Nikolaev, *Numerical Methods for Grid Equations, Volume* I, *Direct Methods*, Birkhäuser Verlag, Basel, Switzerland, 1989.

[24] J. Shen, *Efficient spectral–Galerkin method.* I. *Direct solvers of second- and fourth-order equations using Legendre polynomials*, SIAM J. Sci. Comput., 15 (1994), pp. 1489–1505.

[25] J. Shen, *Efficient spectral–Galerkin method.* II. *Direct solvers for second- and fourth-order equations by using Chebyshev polynomials*, SIAM J. Sci. Comput., 16 (1995), pp. 74–87.

[26] J. Shen, *A new fast Chebyshev–Fourier algorithm for Poisson-type equations in polar geometries*, Appl. Numer. Math., 33 (2000), pp. 183–190.

[27] P. N. Swarztrauber, *FFTPACK*, NCAR, Boulder, CO, 1985, also available from netlib@ornl.gov.

[28] D. Trunec, J. Dřmál, and M. Vičar, *A subprogram for direct solution of Poisson equation in cylindrically symmetric geometry*, Comput. Phys. Comm., 98 (1996), pp. 339–345.

[29] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Frontiers Appl. Math. 10, SIAM, Philadelphia, 1992.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 599–621
© 2005 Society for Industrial and Applied Mathematics

ON THE NONNORMALITY OF SUBITERATION FOR A FLUID-STRUCTURE-INTERACTION PROBLEM*

E. H. VAN BRUMMELEN† AND R. DE BORST†

Abstract. Subiteration forms the basic iterative method for solving the aggregated equations in fluid-structure-interaction problems, in which the fluid and structure equations are solved alternatingly subject to complementary partitions of the interface conditions. In the present work we establish for a prototypical model problem that the subiteration method can be characterized by recursion of a nonnormal operator. This implies that the method typically converges nonmonotonously. Despite formal stability, divergence can occur before asymptotic convergence sets in. It is shown that the transient divergence can amplify the initial error by many orders of magnitude, thus inducing a severe degradation in the robustness and efficiency of the subiteration method. Auxiliary results concern the dependence of the stability and convergence of the subiteration method on the physical parameters in the problem and on the computational time step.

Key words. fluid-structure interaction, subiteration, nonnormality, monolithic methods, convergence and stability

AMS subject classifications. 65N12, 65N22

DOI. 10.1137/S1064827503431430

1. Introduction.

Numerical solution methods for fluid-structure interaction are of great importance in a multitude of physical and engineering disciplines, e.g., aerospace engineering [7, 8], offshore engineering [4], biomechanics [2, 13], and civil engineering [24]. Such solution methods are confronted with two fundamental problems. First, the disparity of the initial-boundary-value problems pertaining to the fluid and the structure generally yields severely ill-conditioned systems of equations. Second, the free-boundary character of the interface between the fluid and the structure yields an interdependence between the fluid and structure solutions and their domains of definition, and the computational expenses incurred by implicit treatment of this interdependence are prohibitive in actual applications. Furthermore, several practical impediments emanate from the inherent interconnection between the fluid and structure, such as the loss of software modularity; see [9].

The customary approach to sidestep the aforementioned problems is through , . , , , , . The fluid and structure equations are then solved alternately, subject to complementary partitions of the interface conditions. This iterative procedure is commonly referred to as subiteration, successive approximation, Picard iteration, or block Gauss–Seidel iteration. The conventional implementation of this method imposes the dynamic interface conditions (tractions) on the structure, and the kinematic interface conditions (displacements) on the fluid; see, e.g., [1, 4, 16, 17]. The principal advantage of partitioned solution methods is the affected separation of the fluid-structure-interaction problem into two common subproblems, viz., the independent solution of the initial-boundary-value problems corresponding to the fluid and the structure on prescribed computational domains, and the separate displacement of the sections of the boundaries of the computational domains associated with the interface. The

*Received by the editors July 16, 2003; accepted for publication (in revised form) December 16, 2004; published electronically October 31, 2005.
http://www.siam.org/journals/sisc/27-2/43143.html
†Delft University of Technology, Faculty of Aerospace Engineering, P.O. Box 5058, 2600 GB Delft, The Netherlands (E.H.vanBrummelen@lr.tudelft.nl, R.deBorst@lr.tudelft.nl).

essential disadvantage of partitioned solution methods pertains to their convergence behavior. Regularly, prohibitory small time steps are required to maintain stability, or convergence is excessively slow.

In the present work we investigate the properties of subiteration for a prototypical fluid-structure-interaction problem. An important new result is that the subiteration method can be characterized by recursion of a nonnormal operator. This implies that the appropriate norm of the operator and its spectral radius are disparate, which reflects in profoundly different transient and asymptotic convergence behavior. In particular, despite formal stability of the operator recursion, convergence can be nonmonotonous and transient divergence can precede the asymptotic convergence. This transient divergence can in fact amplify the initial error by many orders of magnitude, thus inducing a severe degradation in the robustness and efficiency of the method. Such transient behavior of one-parameter functions of nonnormal operators has recently also attracted much attention in the analysis of the convergence of numerical methods (see [12] for an overview), and in the field of hydrodynamic stability in [23, 26], where it has been propounded as a fundamental principle in the transition to turbulence.

The analysis in this paper moreover provides useful information on several other relevant properties of the subiteration method, e.g., the dependence of its stability and convergence on physical parameters and on the computational time step.

The contents of this paper are organized as follows: Section 2 states the piston problem. Section 3 describes the subiteration method for fluid-structure-interaction problems. Section 4 derives the characteristic operator associated with the subiteration method on the basis of a linear model problem. Section 5 establishes the nonnormality of the characteristic operator in the absence of reflections and investigates the properties of the operator by approximation on polynomials. Section 6 is concerned with the properties of the general characteristic operator. Finally, section 7 contains concluding remarks.

2. Problem statement. In this paper we restrict ourselves to the analysis of a prototypical fluid-structure model problem, viz., the piston problem. This one-dimensional problem possesses the prominent features of fluid-structure-interaction problems. Below, we present a concise specification of the problem. For further details, see, e.g., [5, 19].

2.1. Piston problem. The piston system comprises the Euler equations of gas dynamics in connection with a simple harmonic oscillator at the fluid/structure interface. We consider the Euler equations in conservative form:

$$\partial_t \mathbf{q} + \partial_x \mathbf{f}(\mathbf{q}) = 0, \qquad (x,t) \in \Omega_\alpha, \tag{2.1a}$$

with $\Omega_\alpha := \{(x,t) : 0 < t < \tau; 0 < x < \alpha(t)\}$ an open bounded space/time domain. The function $\mathbf{q}$ maps Ω_α into $\mathbb{R}^3$ and

(2.1b)

$$\mathbf{f}(\mathbf{q}) := \big(q_2, q_2^2/q_1 + p(\mathbf{q}), (p(\mathbf{q}) + q_3)\, q_2/q_1\big)^T, \quad p(\mathbf{q}) := (\gamma - 1)\left(q_3 - \frac{1}{2} q_2^2/q_1\right),$$

with, typically, $\gamma = 7/5$. The boundary $\partial\Omega_\alpha$ consists of the [illegible] between the fluid and the structure, $\Gamma_\alpha := \{(x,t) : x = \alpha(t); 0 \le t \le \tau\}$, and the fixed boundary $\partial\Omega_\alpha \setminus \Gamma_\alpha$. The function $\mathbf{q}$ in (2.1) is subject to the auxiliary conditions

$$q_2(0,t) = 0, \quad 0 < t < \tau, \quad \text{and} \quad \mathbf{q}(x,0) = \mathbf{q}^0(x), \quad 0 < x < \alpha(0), \tag{2.2}$$

with $\mathbf{q}^0(x)$ prescribed initial conditions. Moreover, $\mathbf{q}$ must comply with the kinematic interface conditions

$$q_2(\alpha(t),t) = q_1(\alpha(t),t)\,\dot{\alpha}(t), \qquad 0<t<\tau, \tag{2.3a}$$

$$\alpha(t) = \alpha^0 + z(t), \qquad 0<t<\tau, \tag{2.3b}$$

and the dynamic interface condition

$$p(\mathbf{q}(\alpha(t),t)) = \pi(t), \qquad 0<t<\tau, \tag{2.3c}$$

where $\pi(t)$ and $z(t)$ are interrelated by the harmonic-oscillator equation

$$M\ddot{z} + Kz = \pi(t) - p^0 \quad 0<t<\tau, \quad \text{subject to} \quad z(0)=z^0,\ \dot{z}(0)=\dot{z}^0, \tag{2.4}$$

for certain given constants $M, K \in \mathbb{R}_+$ and $z^0, \dot{z}^0 \in \mathbb{R}$.

To establish that the piston problem (2.1)–(2.3) constitutes a [illegible] problem, we notice first that the kinematic condition (2.3a) induces an interdependence between the state variables $\mathbf{q}$ and their domain of definition Ω_α. Second, the number of interface conditions is one more than the number of auxiliary conditions required by the initial-boundary-value problems for the fluid and the structure separately. For instance, (2.1)–(2.2) subject to (2.3a) with α given, and (2.4) subject to (2.3c) with p given, are valid problems independently. The above two properties are the distinguishing characteristics of a free-boundary problem.

2.2. Variational formulation. To facilitate the ensuing presentation, we condense the piston problem into the canonical variational form: Find $u \in \mathcal{U}$ such that

$$\mathsf{P}(\bar{u}, u) = \mathsf{p}(\bar{u}) \qquad \forall \bar{u} \in \bar{\mathcal{U}}. \tag{2.5a}$$

In (2.5), u represents the quadruple $u := (\mathbf{q}, \alpha, z, \pi)$ in the product space $\mathcal{U} := \mathcal{Q} \times \mathcal{A} \times \mathcal{Z} \times \mathcal{P}$, which consists of the product of the admissible functions for the fluid, the interface displacement, the structure displacement, and the interface traction. The test space $\bar{\mathcal{U}}$ has a similar structure. The functionals $\mathsf{P} : \bar{\mathcal{U}} \times \mathcal{U} \mapsto \mathbb{R}$ and $\mathsf{p} : \bar{\mathcal{U}} \mapsto \mathbb{R}$ can be expanded according to

$$\mathsf{P}((\bar{\mathbf{q}}, \bar{\alpha}, \bar{z}, \bar{\pi}), (\mathbf{q}, \alpha, z, \pi)) := \mathsf{E}(\bar{\mathbf{q}}, \mathbf{q}, \alpha) + \mathsf{S}(\bar{z}, z, \pi) + \mathsf{K}(\bar{\alpha}, \alpha, z) + \mathsf{D}(\bar{\pi}, \pi, \mathbf{q}, \alpha), \tag{2.5b}$$

$$\mathsf{p}((\bar{\mathbf{q}}, \bar{\alpha}, \bar{z}, \bar{\pi})) := \mathsf{e}(\bar{\mathbf{q}}) + \mathsf{s}(\bar{z}) + \mathsf{k}(\bar{\pi}). \tag{2.5c}$$

In (2.5), the functionals E, e are associated with the initial-boundary-value problem for the fluid, viz., (2.1) complemented with the kinematic condition (2.3a). The functionals S, s pertain to the initial-value problem for the harmonic oscillator (2.4), and K, k, and D are associated with the kinematic interface condition (2.3b) and the dynamic interface condition (2.3c), respectively. A specification of the above functionals for the piston problem is provided in [5]. In [6], van Brummelen, Michler, and Borst elaborate upon the formulation of a generic fluid-structure-interaction problem into the canonical variational form (2.5). On account of the canonical variational form, sufficient conditions for the existence of a unique solution to (2.5) can be derived from the generalized nonlinear Lax–Milgram theorem; cf. Appendix A.

3. Subiteration for fluid-structure-interaction problems. The interdependence of the state variables of the fluid and the structure and their domains of definition poses profound complications in the numerical treatment of fluid-structure-interaction problems. The subiteration method which we consider in this paper is the standard approach to bypass these complications. In this section, we first consider the problems pertaining to the application of Newton's method to the piston problem. We then proceed with a description of the subiteration method.

TABLE 1
Illustration of the connectivity in the fluid-structure-interaction problem.

	α	$\mathbf{q}$	π	z
K	$\times$	0	0	$\times$
E	$\times$	$\times$	0	0
D	$\times$	$\times$	$\times$	0
S	0	0	$\times$	$\times$

3.1. Newton's method. Provided with an initial approximation $u_0 \in \mathcal{U}$, Newton's method for solving the variational problem (2.5) is defined by the following iterative process: for $j = 1, 2, \ldots$, find u_j such that

$$\mathsf{P}'(\bar{u}, u_{j-1}, u_j - u_{j-1}) = \mathsf{p}(\bar{u}) - \mathsf{P}(\bar{u}, u_{j-1}) \quad \forall \bar{u} \in \bar{\mathcal{U}}, \tag{3.1}$$

with P' the (assumed existent) Fréchet derivative of the operator P with respect to its nonlinear argument. If the initial estimate u_0 is sufficiently close to the actual solution, then u_j converges to the solution of the variational problem as $j \to \infty$.

The above abstract formulation of Newton's method obscures two essential problems, related to the characteristic properties of fluid-structure-interaction problems. The first problem concerns the [illegible] E'_α, viz., the derivative of the functional pertaining to the initial-boundary-value problem for the fluid, E, with respect to the interface position, α, induced the free-boundary character. Discrete-approximation methods such as finite elements typically use boundary fitted meshes. As a perturbation of the interface generally yields a deformation of the mesh throughout the entire computational domain, E'_α behaves as a nonlocal operator. This renders the computational expenses incurred in its evaluation prohibitive in actual applications. The second problem is the inherent interconnection between the fluid and the structure, induced by the interface conditions. The functionals D and K depend on $\alpha, \mathbf{q}$, associated with the fluid, as well as on π, z, corresponding to the structure. This is illustrated in Table 1. Consequently, the matrix $\mathbf{P}$ associated with a discrete approximation of the operator P' in (3.1) is inseparable and the fluid and structure equations must be resolved simultaneously. This has severe practical disadvantages, such as the loss of modularity; see [9]. Moreover, due to the disparate properties of the initial-boundary-value problems associated with the fluid and the structure, the matrix $\mathbf{P}$ is generally severely ill-conditioned.

3.2. Subiteration. The aforementioned complications of Newton's method can be avoided by means of an iterative solution procedure often referred as [illegible]. Provided with an initial approximation of the structure position $z_0(t)$, the following iterative process defines the subiteration algorithm: for $j = 1, 2, \ldots$:

(S1) Solve the kinematic condition: find $\alpha_j \in \mathcal{A}$: $\mathsf{K}(\bar{\alpha}, \alpha_j, z_{j-1}) = \mathsf{k}(\bar{\alpha})\ \forall \bar{\alpha} \in \bar{\mathcal{A}}$.
(S2) Solve the fluid: find $\mathbf{q}_j \in \mathcal{Q}$: $\mathsf{E}(\bar{\mathbf{q}}, \mathbf{q}_j, \alpha_j) = 0\ \forall \bar{\mathbf{q}} \in \bar{\mathcal{Q}}$.
(S3) Solve the dynamic condition: find $\pi_j \in \mathcal{P}$: $\mathsf{D}(\bar{\pi}, \pi_j, \mathbf{q}_j, \alpha_j) = 0\ \forall \bar{\pi} \in \bar{\mathcal{P}}$.
(S4) Solve the structure: find $z_j \in \mathcal{Z}$: $\mathsf{S}(\bar{z}, z_j, \pi_j) = \mathsf{s}(\bar{z})\ \forall \bar{z} \in \bar{\mathcal{Z}}$.

Note that this procedure avoids the computation of the shape derivative as well as the simultaneous treatment of the fluid and the structure.

It is mentionable that the subiteration process forms an instance of the generic iterative method for free-boundary problems: (1) the boundary-value problem(s) is

(are) solved with a subset of the free-boundary conditions imposed, and (2) the free-boundary position is adjusted to relax the remaining free-boundary condition. Different names are used for this iterative procedure, e.g., subiteration, successive approximation, or Picard iteration. For applications to fluid-structure-interaction problems see, for example, [1, 16, 17]. Moreover, we remark that the customary [illegible] or [illegible] time-integration methods for fluid-structure-interaction problems (see [9, 20, 21, 22]) are essentially identical to the above subiteration method, with the provision that the process is not repeated.

The subiteration method can also be classified as a [illegible] method [3]. The defect correction approach for a variational problem of the form (2.5a) commences with the determination of an initial estimate $u_0 \in \mathcal{U}$ such that $\tilde{\mathsf{P}}(\bar{u}, u_0) = \mathsf{p}(\bar{u})$ for all $\bar{u} \in \bar{\mathcal{U}}$, where the functional $\tilde{\mathsf{P}} : \bar{\mathcal{U}} \times \mathcal{U} \mapsto \mathbb{R}$ represents a suitable approximation to P. It then proceeds with finding $u_j \in \mathcal{U}$ such that

$$\tilde{\mathsf{P}}(\bar{u}, u_j) = \mathsf{p}(\bar{u}) + \tilde{\mathsf{P}}(\bar{u}, u_{j-1}) - \mathsf{P}(\bar{u}, u_{j-1}) \qquad \forall \bar{u} \in \bar{\mathcal{U}}, \tag{3.2}$$

for $j = 1, 2, \ldots$. If $\tilde{\mathsf{P}}$ is sufficiently close to P, then u_j converges to the actual solution of the variational problem as $j \to \infty$. To derive the approximate functional for the subiteration method, we note that the approximations generated by (S1)–(S4) satisfy

$$\mathsf{E}(\bar{\mathbf{q}}, \mathbf{q}_j, \alpha_j) + \mathsf{S}(\bar{z}, z_j, \pi_j) + \mathsf{D}(\bar{\pi}, \pi_j, \mathbf{q}_j, \alpha_j) + \mathsf{K}(\bar{\alpha}, \alpha_j, z_{j-1}) = \mathsf{e}(\bar{\mathbf{q}}) + \mathsf{s}(\bar{z}) + \mathsf{k}(\bar{\alpha}), \tag{3.3}$$

for all admissible $(\bar{\mathbf{q}}, \bar{\alpha}, \bar{z}, \bar{\pi})$. In general, the functional K is associated with a variational projection, and is separable as $\mathsf{K}(\bar{\alpha}, \alpha, z) = \mathsf{K}_0(\bar{\alpha}, \alpha) + \mathsf{K}_1(\bar{\alpha}, z)$. Upon introducing this additive partition of K into (3.3) and adding suitable partitions of zero, we immediately recover (3.2) with the approximate functional

$$\tilde{\mathsf{P}}((\bar{\mathbf{q}}, \bar{\alpha}, \bar{z}, \bar{\pi}), (\mathbf{q}, \alpha, z, \pi)) := \mathsf{E}(\bar{\mathbf{q}}, \mathbf{q}, \alpha) + \mathsf{S}(\bar{z}, z, \pi) + \mathsf{K}(\bar{\alpha}, \alpha, \zeta) + \mathsf{D}(\bar{\pi}, \pi, \mathbf{q}, \alpha), \tag{3.4}$$

with ζ any fixed element of $\mathcal{Z}$. Note that the approximation only involves the functional K corresponding to the kinematic interface condition (2.3b). The connectivity table associated with the approximate operator is identical to Table 1, but with the right-upper entry nullified. Thus, the connectivity table is lower triangular, and the subproblems involving the inversion of $\tilde{\mathsf{P}}$ can be solved conveniently by the forward substitution process inherent in the subiteration method.

4. Model problem. In this section we develop a linear model problem, which serves as a basis for the ensuing analysis of the properties of the subiteration method in sections 5 and 6.

4.1. Generating solution. To set up the linearized model, we consider the piston problem and we note that if the initial conditions are specified as $\mathbf{q}^0(x) = \mathbf{q}^0$ with $\mathbf{q}^0$ a constant in $\{\mathbf{q} \in \mathbb{R}^3 : 0 < q_1; p(\mathbf{q}) = p^0; q_2 = 0\}$ and $z^0 = \dot{z}^0 = 0$, then the obvious solution to (2.5) is

$$u := (\mathbf{q}, \alpha, z, \pi) = (\mathbf{q}^0, \alpha^0, 0, p^0) =: u^0. \tag{4.1}$$

Hence, if we provide the subiteration method with an initial approximation $z_0(t) = 0$, then the first iteration immediately yields the actual solution. If, instead, the initial approximation is specified as $z_0(t) = \epsilon z_0'(t)$, with z_0' a suitable function independent of ϵ, then $u_1(\epsilon)$ approaches u^0 as $\epsilon \to 0$. In this context, u^0 is called a generating solution.

4.2. First-order conditions. The above motivates us to assume that the first approximation generated by the subiteration method can be expanded asymptotically:

(4.2)
$$\alpha_1(t) = \alpha^0 + \sum_{l=1}^{n} \eta_l(\epsilon)\,\alpha^{(l)}(x,t) + o(\eta_n), \quad q_{1,j}(x,t) = q_j^0 + \sum_{l=1}^{n} \psi_l^j(\epsilon)\,q_j^{(l)}(x,t) + o(\psi_n^j),$$
$$\pi_1(t) = p^0 + \sum_{l=1}^{n} \xi_l(\epsilon)\,\pi^{(l)}(x,t) + o(\xi_n), \qquad z_1(t) = \ 0 + \sum_{l=1}^{n} \chi_l(\epsilon)\,z^{(l)}(x,t) + o(\chi_n),$$

$(j = 1, 2, 3)$ for all $n = 1, 2, \dots$, with respect to certain asymptotic sequences $\{\psi_l^j(\epsilon)\}$, $\{\chi_l(\epsilon)\}$, $\{\eta_l(\epsilon)\}$, and $\{\xi_l(\epsilon)\}$, as $\epsilon \to 0$. For a definition of asymptotic sequences and the Landau symbols, o and O (used below), see, for instance, [15].

By the usual argument that terms of different orders in ϵ must vanish separately as $\epsilon \to 0$, it follows that terms with subscript l in (4.2) must be of $O(\epsilon^l)$, in order for (S1)–(S4) to provide meaningful relations between successive terms in the expansion. In particular, this implies that $u_1(\epsilon) = u^0 + \epsilon u^{(1)} + O(\epsilon^2)$, i.e., the error in the first-order approximation is $O(\epsilon^2)$.

For transparency of notation, in what follows the subscript 1, indicating the first iteration, is suppressed, and the superscript (1), referring to the infinitesimal perturbation, is replaced by a prime. 5the first iteration is suppressed and the superscript (1) referring Upon inserting (4.2) in (S1) and collecting terms of $O(\epsilon)$, we obtain the first-order condition $\alpha' = z_0'$. To simplify the analysis, we assume that the first-order perturbation in $\mathbf{q}$ is isentropic, i.e., the ratio p/q_1^γ is constant. Under this assumption, the energy equation becomes redundant and can be discarded. The first-order conditions corresponding to operation (S2) then translate into the initial-boundary-value problem

$$\frac{\partial}{\partial t}\begin{pmatrix} q_1' \\ q_2' \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ c^2 & 0 \end{pmatrix} \cdot \frac{\partial}{\partial x}\begin{pmatrix} q_1' \\ q_2' \end{pmatrix} = 0, \tag{4.3}$$

with $c := \sqrt{\gamma p^0/q_1^0}$ the speed of sound, subject to the initial and boundary conditions

$$q_1'(x,0) = 0, \quad x \in (0,\alpha^0), \qquad q_2'(x,0) = 0, \qquad x \in (0,\alpha^0), \tag{4.4a}$$
$$q_2'(0,t) = 0, \quad t \in (0,\tau), \qquad q_2'(\alpha^0,t) = \rho^0 \dot{z}_0'(t), \quad t \in (0,\tau). \tag{4.4b}$$

The generic solution to (4.3) is

$$\begin{pmatrix} q_1' \\ q_2' \end{pmatrix}(x,t) = \rho^0 \left(f_+(t + x/c)\begin{pmatrix} 1 \\ -c \end{pmatrix} + f_-(t - x/c)\begin{pmatrix} 1 \\ c \end{pmatrix}\right). \tag{4.5}$$

The initial and boundary conditions (4.4) imply the following conditions on f_+, f_-:

$$f_-(-s) + f_+(s) = 0, \qquad f_-(-s) - f_+(s) = 0, \qquad s \in (0,\alpha^0/c), \tag{4.6a}$$
$$f_-(s) - f_+(s) = 0, \quad f_-(s - \alpha^0/c) - f_+(s + \alpha^0/c) = \dot{z}_0'(s)/c, \quad s \in (0,\tau). \tag{4.6b}$$

From (4.5)–(4.6) it follows straightforwardly that

$$q_1'(\alpha^0,t) = -\frac{\rho^0}{c}\left(\dot{z}_0'(t) + 2\sum_{j=1}^{\lfloor ct/2\alpha^0 \rfloor} \dot{z}_0'(t - 2j\alpha^0/c)\right), \tag{4.7}$$

where $\lfloor\cdot\rfloor$ denotes the floor (truncation to integer) function. One can infer that the integer $\lfloor ct/2\alpha^0 \rfloor$ represents the number of reflections that have returned from the rigid boundary to the interface in the time interval $[0, t]$.

Under the assumption of isentropy, (S3) immediately yields the first-order perturbation in the aggregated force on the structure: $\pi' = c^2 q_1'(\alpha^0, t)$. From (4.7) it then follows that the structural displacement obtained in step (S4) of the subiteration algorithm complies with the first-order condition

(4.8)
$$M\ddot{z}'(t) + Kz'(t) = -c\rho^0 \left(\dot{z}_0'(t) + 2 \sum_{j=1}^{\lfloor ct/2\alpha^0 \rfloor} \dot{z}_0'\big(t - 2j\alpha^0/c\big) \right), \quad z'(0) = 0, \quad \dot{z}'(0) = 0.$$

Note that the initial-value problem (4.8) provides a unique relation between successive approximations of the structure displacement in the subiteration process.

To cast (4.8) into a convenient dimensionless form, we introduce characteristic time scales for the structure and the fluid, viz., $\tau_s := 2\pi\sqrt{M/K}$ and $\tau_f := 2\,\alpha^0/c$, respectively. Moreover, we define characteristic time-scale and mass ratios for the fluid-structure system, $\nu := 2\pi\,\tau_f/\tau_s$ and $\mu := \rho^0\alpha^0/M$, and a dimensionless time variable $s := 2\pi\,t/\tau_s$. With these definitions, we obtain

(4.9)
$$D^2 z'(s) + z'(s) = -\frac{2\,\mu}{\nu} \left(Dz_0'(s) + 2 \sum_{j=1}^{\lfloor s/\nu \rfloor} Dz_0'(s - j\nu) \right), \quad z'(0) = 0, \quad Dz'(0) = 0,$$

where D denotes differentiation with respect to the function argument.

4.3. Characteristic operators. The initial-value problem (4.9) associates a $z' \in \mathcal{Z}$ with each $z_0' \in \mathcal{Z}$. Accordingly, (4.9) generates an automorphic operator on $\mathcal{Z}$. We denote this operator by C. Because the subiteration process essentially reduces to recursion of the operator C, we refer to C as the [illegible] or [illegible].

An important simplification of the characteristic operator results from discarding the sum associated with the reflections. We designate the corresponding characteristic operator as the [illegible] C_0, in particular,

$$\mathsf{C}_0 : \{z_0' \in \mathcal{Z}\} \mapsto \big\{z' \in \mathcal{Z} : D^2 z' + z' = -2(\mu/\nu) Dz_0',\ z'(0) = 0,\ Dz'(0) = 0\big\}. \tag{4.10}$$

By virtue of the linearity of the differential equation and the homogeneity of the initial conditions in (4.9), the following relation between the null-reflection operator and the general characteristic operator holds:

$$\mathsf{C} = \left(\mathsf{I} + 2 \sum_{j}^{\lfloor \tau/\nu \rfloor} \mathsf{H}_\nu^j \right) \mathsf{C}_0, \tag{4.11}$$

where the shift operator $\mathsf{H}_\nu := \mathsf{H}(\nu)$ on $\mathcal{Z}$ is defined as

$$(\mathsf{H}_\nu z)(s) := \begin{cases} 0 & 0 < s \le \nu, \\ z(s - \nu) & \nu < s < \tau. \end{cases} \tag{4.12}$$

Note the group property $\mathsf{H}(\nu+\nu') = \mathsf{H}(\nu)\mathsf{H}(\nu')$ for any $\nu,\nu' \in \mathcal{T} =: (0,\tau)$ such that $\nu+\nu' \in \mathcal{T}$.

A second simplification pertains to the limit $\nu \to 0$. The right member of (4.9) can then be identified as a Riemann sum and it holds that

(4.13)
$$Du(s) + 2\sum_{j=1}^{\lfloor s/\nu \rfloor} Du(s-j\nu) = \frac{2}{\nu}\Big(\int_0^s Du(\xi)\,d\xi + O(\nu)\Big) \sim \frac{2}{\nu}\big(u(s)-u(0)\big) \text{ as } \nu \to 0.$$

Hence, as $\nu \to 0$ the characteristic operator approaches the operator C_∞,

(4.14)
$$\mathsf{C}_\infty : \{z_0' \in \mathcal{Z}\} \mapsto \big\{z' \in \mathcal{Z} : D^2 z' + z' = -4\mu/\nu^2\big(z_0'(s) - z_0'(0)\big), z'(0) = 0, Dz'(0) = 0\big\}.$$

We refer to C_∞ according to (4.14) as the [illegible]. Notice the different proportionalities $\mathsf{C}_0 \propto \mu/\nu$ and $\mathsf{C}_\infty \propto \mu/\nu^2$.

5. Nonnormality of the null-reflection operator. In this section we consider the nonnormality of the null-reflection operator. Such nonnormal behavior has important implications for the norm of powers of the operator and, hence, it is essential to the convergence properties of the subiteration method. The extension to the general subiteration operator including reflections is presented in section 6.

5.1. Theory of linear operators. This section reviews the basic theory of linear operators required for the ensuing exposition. For details we refer to the commendable [14].

Consider a Hilbert space $\mathcal{Z}$, equipped with an inner product $\langle\cdot,\cdot\rangle := \langle\cdot,\cdot\rangle_{\mathcal{Z}}$ and an associated norm $\|\cdot\| := \|\cdot\|_{\mathcal{Z}}$, and a bounded linear operator $\mathsf{L} : \mathcal{Z} \mapsto \mathcal{Z}$. The adjoint operator L^* is defined by the identity

$$\langle \mathsf{L}^* w, z\rangle = \langle w, \mathsf{L} z\rangle \qquad \forall w, z \in \mathcal{Z}. \tag{5.1}$$

In fact, it suffices that the identity holds for all w, z in a dense subspace of $\mathcal{Z}$; see [28, p. 263]. The operator L is said to be [illegible] if it commutes with its adjoint, i.e., if

$$\mathsf{L}^*\mathsf{L} = \mathsf{L}\mathsf{L}^*. \tag{5.2}$$

To present the specific properties pertaining to (5.2), we require some elementary concepts. The resolvent is defined as the operator-valued function

$$\mathsf{R}(\zeta) := \mathsf{R}(\mathsf{L},\zeta) := \big(\mathsf{L} - \zeta\,\mathsf{I}\big)^{-1} \tag{5.3}$$

on $\mathbb{C}$. Herein, I is the identity on $\mathcal{Z}$. The resolvent set $\mathcal{R}(\mathsf{L})$ is defined as the subset of $\mathbb{C}$ consisting of all ζ for which $\mathsf{R}(\zeta)$ is existent and bounded. The spectrum $\mathcal{S}(\mathsf{L})$ is defined as its complement in $\mathbb{C}$. If L is compact, which includes the case that $\dim \mathcal{Z} < \infty$, then the spectrum consists of at most a countable number of eigenvalues with finite multiplicities, possibly excepting zero if L is singular. The norm of the operator and its spectral radius are defined by

$$\|\mathsf{L}\| := \sup_{0\neq z\in\mathcal{Z}} \frac{\|\mathsf{L}z\|}{\|z\|} = \sup_{\substack{0\neq v\in\mathcal{Z}\\ 0\neq z\in\mathcal{Z}}} \frac{|\langle v, \mathsf{L}z\rangle|}{\|v\|\|z\|}, \quad \operatorname{spr}\mathsf{L} := \lim_{n\to\infty} \|\mathsf{L}^n\|^{1/n} = \inf_{n\in\mathbb{N}} \|\mathsf{L}^n\|^{1/n}, \tag{5.4}$$

respectively. It holds that

$$\operatorname{spr} \mathsf{L} = \sup_{\zeta \in \mathcal{S}(\mathsf{L})} |\zeta|. \tag{5.5}$$

A characteristic property of a normal operator is that $\|\mathsf{L}^n\| = \|\mathsf{L}\|^n$ and, accordingly, $\|\mathsf{L}\| = \operatorname{spr} \mathsf{L}$; see [14]. Hence, for a normal operator the spectral radius $\operatorname{spr} \mathsf{L}$ delineates the norm of monomials of the operator $\|\mathsf{L}^n\|$. In particular, if $\operatorname{spr} \mathsf{L} < 1$, then $\|\mathsf{L}^n\|$ converges monotonically to 0 as n increases. For nonnormal operators the spectral radius only represents the asymptotic convergence as $n \to \infty$. If $\|\mathsf{L}\| > 1 > \operatorname{spr} \mathsf{L}$, then the convergence of $\|\mathsf{L}^n\|$ is generally nonmonotonous and divergence can occur before asymptotic convergence sets in; cf. section 5.4 and, e.g., [12, p. 28]. Relevant in this respect are also the plots of $\|\exp(t\mathsf{L})\|$ (which has similar properties as $\|\mathsf{L}^n\|$) versus t for various nonnormal operators in [25].

Another important property of a normal operator is that it admits a spectral representation on a basis of orthonormal eigenvectors. If L is compact, we can index the nonzero eigenvalues as $\sigma_1, \sigma_2, \ldots$. To each eigenvalue σ_j corresponds an eigenprojection

$$\mathsf{P}_j := -\frac{1}{2\pi\iota} \oint_{\chi_j} \mathsf{R}(\zeta)\, d\zeta \tag{5.6}$$

with $\iota := \sqrt{-1}$ and χ_j any closed curve in $\mathbb{C}$ of which the intersection of its interior and the spectrum contains σ_j only; see[10, p. 5 ff.] or [14, p. 39 ff.]. For compact normal operators L we have the spectral representation

$$\mathsf{L} = \sum\nolimits_j \sigma_j \mathsf{P}_j \tag{5.7}$$

and $\mathsf{P}_j^* = \mathsf{P}_j$, $\dim \mathsf{P}_j < \infty$ and $\mathsf{P}_i \mathsf{P}_j = \delta_{ij} \mathsf{P}_i$. Furthermore, together with the orthonormal projection P_0 on the kernel of L the eigenprojections form a complete orthonormal family and, hence, $\sum \mathsf{P}_j = \mathsf{I}$. Alternatively, we can select orthonormal bases $\{\varphi_j^i\}$ $(i = 1, \ldots, \dim \mathsf{P}_j \mathcal{Z})$ for the subspaces $\mathsf{P}_j \mathcal{Z}$ and corresponding conjugate bases $\{\psi_j^i\}$ and rewrite (5.7) as

$$\mathsf{L}(\cdot) = \sum_j \sum_{i=1}^{\dim \mathsf{P}_i \mathcal{Z}} \sigma_j \varphi_j^i \langle \psi_j^i, \cdot \rangle. \tag{5.8}$$

The φ_i^j are the eigenvectors corresponding to the eigenvalue σ_i. The corresponding conjugate vectors $\{\psi_i^j\}$ are defined by $\langle \varphi_j^i, \psi_l^k \rangle = \delta_{jl}^{ik}$, i.e., $\langle \psi_j^i, \cdot \rangle$ represent the characteristic coordinates with respect to the basis $\{\varphi_j^i\}$. An operator which admits a representation conforming to (5.7) with $\mathsf{P}_j \mathsf{P}_k = \delta_{jk} \mathsf{P}_j$ and $\sum \mathsf{P}_j = \mathsf{I}$ is said to be [illegible] or [illegible]. If, in addition, it holds that the eigenprojections are orthogonal and, accordingly, $\mathsf{P}_j^* = \mathsf{P}_j$, then the normalized eigenvectors form an orthonormal basis of $\mathcal{Z}$ and the operator is said to be [illegible]. Hence, a compact operator is normal if and only if it is unitarily diagonalizable.

To express the degeneracy of the set of eigenvectors of a nonnormal operator, we define the coercivity coefficient and the continuity coefficient of the projection onto the basis of eigenvectors $\{\varphi_j\} := \{\varphi_j^i\}$, respectively, as the largest number $\gamma(\mathsf{L})$ and the smallest number $\gamma'(\mathsf{L})$ such that

$$\inf_{0 \neq \{\hat{z}_j\} \in \hat{\mathcal{Z}}} \sup_{0 \neq v \in \mathcal{Z}} \frac{\left|\Sigma_j \langle v, \varphi_j \rangle \hat{z}_j\right|}{\|v\|\, \|\hat{z}.\|} \geq \gamma(\mathsf{L}), \qquad \sup_{\substack{0 \neq \{\hat{z}_j\} \in \hat{\mathcal{Z}} \\ 0 \neq v \in \mathcal{Z}}} \frac{\left|\Sigma_j \langle v, \varphi_j \rangle \hat{z}_j\right|}{\|v\|\, \|\hat{z}.\|} \leq \gamma'(\mathsf{L}), \tag{5.9}$$

where $\|\hat{z}.\|^2 := \sum \hat{z}_i^2$ and $\hat{\mathcal{Z}}$ represents the space of coefficients $\{\hat{z}_i\}$ such that $\sum \hat{z}_i\varphi_i$ resides in $\mathcal{Z}$. From the spectral representation (5.8) it follows that

(5.10)
$$\langle v, \mathsf{L}^n z\rangle = \langle v,\varphi_{i_1}\rangle\sigma_{i_1}\langle\psi_{i_1},\varphi_{i_2}\rangle\sigma_{i_2}\cdots\langle\psi_{i_{n-1}},\varphi_{i_n}\rangle\sigma_{i_n}\langle\psi_{i_n},z\rangle = \sigma_i^n\langle v,\varphi_i\rangle\langle\psi_i,z\rangle,$$

where summation over repeated indices is implied. Furthermore, denoting by $\check{Z}$ the space of coefficients $\{\check{z}_i\}$ such that $\sum \check{z}_i\psi_i \in \mathcal{Z}$, it holds that

$$\sup_{0\neq z\in\mathcal{Z}} \frac{\|\langle z,\psi.\rangle\|}{\|z\|} = \sup_{\substack{0\neq\{\hat{z}_i\}\in\hat{\mathcal{Z}}\\ 0\neq\{\check{w}_j\}\in\check{\mathcal{Z}}}} \frac{\|\hat{z}_i\langle\varphi_i,\psi_j\rangle\check{w}_j\|}{\|\check{w}.\|\|\hat{z}_i\varphi_i\|} = \sup_{0\neq\{\hat{z}_i\}\in\hat{\mathcal{Z}}} \frac{\|\hat{z}.\|}{\|\hat{z}_i\varphi_i\|}$$

$$= \left(\inf_{0\neq\{\hat{z}_i\}\in\hat{\mathcal{Z}}} \frac{\|\hat{z}_i\varphi_i\|}{\|\hat{z}.\|}\right)^{-1} \leq \frac{1}{\gamma(\mathsf{L})}, \tag{5.11}$$

again with summation over repeated indices. Hence, we obtain

$$\|\mathsf{L}^n\| = \sup_{\substack{0\neq z\in\mathcal{Z}\\ 0\neq v\in\mathcal{Z}}} \frac{|\langle v,\mathsf{L}^n z\rangle|}{\|v\|\|z\|} \leq (\operatorname{spr}\mathsf{L})^n \sup_{\substack{0\neq v\in\mathcal{Z}\\ 0\neq z\in\mathcal{Z}}} \frac{\|\langle v,\varphi.\rangle\|}{\|v\|}\frac{\|\langle z,\psi.\rangle\|}{\|z\|} \leq (\operatorname{spr}\mathsf{L})^n\frac{\gamma'(\mathsf{L})}{\gamma(\mathsf{L})}. \tag{5.12}$$

The ratio $\gamma'(\mathsf{L})/\gamma(\mathsf{L}) =: \kappa(\mathsf{L})$ is called the condition number of the basis of eigenvectors. By (5.4) and (5.12), it holds that $(\operatorname{spr}\mathsf{L})^n \leq \|\mathsf{L}^n\| \leq \kappa(\mathsf{L})(\operatorname{spr}\mathsf{L})^n$. If L is normal and, accordingly, $\{\varphi_j\}$ forms an orthonormal basis of $\mathcal{Z}$, then $\gamma(\mathsf{L}) = \gamma'(\mathsf{L}) = 1$ by Parseval's identity and, hence, $\kappa(\mathsf{L}) = 1$. Contrastingly, if L is nonnormal, then $\kappa(\mathsf{L})$ is typically much larger than unity. Moreover, it is mentionable that (5.11) implies that the norm of the characteristic coordinates of any $z \in \mathcal{Z}$ are bounded by $\|\hat{z}.\| \leq \|z\|/\gamma(\mathsf{L})$. This implies that the characteristic coordinates of a typical element of $\mathcal{Z}$ can be enormous if the eigenfunctions are nearly linearly dependent and, accordingly, $\gamma(\mathsf{L})$ is small. Trefethen [25] notes that the properties of an analytic function of the operator, such as L^n or $\exp(t\mathsf{L})$, are then determined by the interactions of the eigenfunctions rather than their individual behavior.

Finally, we consider the relation between the normality of an operator and its pseudospectra, viz., the one-parameter family of subsets of $\mathbb{C}$ defined by

$$\mathcal{S}_\epsilon(\mathsf{L}) := \left\{\zeta\in\mathbb{C} : \|\mathsf{R}(\mathsf{L},\zeta)\| \geq \epsilon^{-1}\right\} \tag{5.13}$$

for all $\epsilon \in \mathbb{R}_+$. By convention, $\mathcal{S}_0(\mathsf{L})$ is identified with the spectrum $\mathcal{S}(\mathsf{L})$. More precisely, the spectrum consists of all complex numbers ζ for which the resolvent $\mathsf{R}(\mathsf{L},\zeta)$ is unbounded or its domain is only a proper subset of $\mathcal{Z}$; see also [14]. For any $\zeta \in \mathbb{C}$ it holds that $\mathcal{S}(\mathsf{L}-\zeta\,\mathsf{I}) = \mathcal{S}(\mathsf{L})-\zeta$. Hence, by the spectral mapping theorem, $\mathcal{S}\big(\mathsf{R}(\zeta_0)\big)$ is related to $\mathcal{S}(\mathsf{L})$ by the transformation $\zeta \mapsto (\zeta-\zeta_0)^{-1}$. On account of (5.5) it then follows that

(5.14)
$$\operatorname{spr}\mathsf{R}(\zeta_0) = \sup_{\zeta\in\mathcal{S}(\mathsf{R}(\zeta_0))}|\zeta| = \sup_{\zeta\in\mathcal{S}(\mathsf{L})}|\zeta-\zeta_0|^{-1} = \left(\inf_{\zeta\in\mathcal{S}(\mathsf{L})}|\zeta-\zeta_0|\right)^{-1} = \operatorname{dist}(\zeta_0,\mathcal{S}(\mathsf{L}))^{-1},$$

with $\operatorname{dist}(\zeta_0,\mathcal{S}(\mathsf{L})) := \inf_{\zeta\in\mathcal{S}(\mathsf{L})}|\zeta_0-\zeta|$ the distance of ζ_0 to the spectrum $\mathcal{S}(\mathsf{L})$. As the resolvent of a normal operator is itself a normal operator (see [14]), it holds that $\|\mathsf{R}(\mathsf{L},\zeta)\| = \operatorname{spr}\mathsf{R}(\mathsf{L},\zeta)$ if L is normal. This implies that the ϵ-pseudospectrum of a normal operator is the aggregate of all points in $\mathbb{C}$ at distance at most ϵ from the spectrum. For nonnormal operators, the pseudospectra can be much larger; cf. [25] for examples; see also section 5.4.

5.2. Functional setting. Considering an open interval $\mathcal{T} := (0, \tau)$, an appropriate setting for C_0 is the Sobolev space $H^1(\mathcal{T})$ defined as

$$H^1(\mathcal{T}) := \{z \in L^2(\mathcal{T}) : \|z\|_{H^1(\mathcal{T})} < \infty\}, \tag{5.15}$$

where the norm $\|\cdot\|_{H^1(\mathcal{T})}$ is induced by the inner product

$$\langle w, z\rangle_{H^1(\mathcal{T})} := \langle w, z\rangle_{L^2(\mathcal{T})} + \langle Dw, Dz\rangle_{L^2(\mathcal{T})}. \tag{5.16}$$

Whenever the interval is evident from the context, we will suppress it for brevity. The operator C_0 is bounded on H^1 as $\|\mathsf{C}_0\|_{H^1} \leq 2\tau$ and unbounded on L^2; see Appendix B. Equipped with the norm induced by (5.16), H^1 is a Hilbert space.

As a digression, we note a corollary of the upper bound $\|\mathsf{C}_0\|_{H^1} \leq 2\tau$ for the accuracy of the partitioned time-integration methods for fluid-structure-interaction problems (see, e.g., [9, 20, 21, 22]), and for the required number of subiterations for monolithic methods, i.e., methods that actually resolve the aggregated fluid and structure equations. Consider a time interval $\mathcal{T} := (0, T)$, partitioned into contiguous subintervals $\mathcal{T}_i$ of length τ ($i = 1, 2, \ldots, T/\tau$). Let z, z_i, and z_i^n denote the actual boundary displacement, its finite-element approximation of polynomial degree m on $\mathcal{T}_i$, and the approximation to z_i generated by n subiterations, respectively. Indicating by the binary relation $\lesssim$ that the left member is at most equal to the right member modulo a positive constant, basic interpolation theory conveys that for sufficiently regular z the interpolation error $d_i := z - z_i$ obeys $\|d_i\|_{H^1(\mathcal{T}_i)} \lesssim \tau^m |z|_{H^m(\mathcal{T}_i)} \lesssim \tau^{m+1/2}$, with $|\cdot|_{H^m}$ the seminorm; see, e.g., [18, Theorem 6.8]. Moreover, the approximation z_{i-1} can then be extended to $\mathcal{T}_i$ such that $\|z_{i-1} - z\|_{H^1(\mathcal{T}_i)} \lesssim \tau^{m+1/2}$. Therefore, if the initial approximation on the interval is generated by extrapolation, i.e., $z_i^0 = z_{i-1}$, then the corresponding evaluation error $e_i^n := z_i^n - z_i$ obeys $\|e_i^0\|_{H^1(\mathcal{T}_i)} \lesssim \tau^{m+1/2}$. The upper bound $\|\mathsf{C}_0\|_{H^1} \leq 2\tau$ indicates that $\|e_i^n\|_{H^1} \leq 2\tau \|e_i^{n-1}\|_{H^1}$. This implies that $\|e_i^1\|_{H^1} = O(\tau^{m+3/2}) \ll \|d_i\|_{H^1} = O(\tau^{m+1/2})$ as $\tau \to 0$. That is, for sufficiently small τ, a single subiteration renders the local evaluation error much smaller than the local interpolation error. However, to ensure that the global evaluation error is smaller than the global interpolation error, two iterations are required. To clarify this, we note that the pressure perturbation is related to the displacement perturbation by $\pi' = \text{const}\, Dz'$; cf. section 4.2. In the subiteration process the pressure approximation is based on the previous displacement approximation and, hence, the evaluation error in the pressure $\beta_i^n := \pi_i^n - \pi_i$ complies with $\|\beta_i^n\|_{L^2(\mathcal{T}_i)} \lesssim \|e_i^{n-1}\|_{H^1} \lesssim \tau^{m+n-1/2}$. Summation over the intervals imparts that the global evaluation error in the pressure satisfies $\|\beta^n\|_{L^2}^2 = \sum_i \|\beta_i^n\|_{L^2(\mathcal{T}_i)}^2 \lesssim \tau^{2(m+n-1)}$. The global evaluation error in the displacement complies with $D^2 e^n + e^n = \beta^n$ and the homogeneous initial conditions $e^n(0) = De^n(0) = 0$. A similar derivation as in Appendix B then conveys that $\|e^n\|_{H^1(\mathcal{T})} \leq 2T \|\beta^n\|_{L^2(\mathcal{T})} \lesssim \tau^{m+n-1}$. One subiteration ($n = 1$) thus renders the global evaluation error in the displacement only $O(\tau^m)$ and, hence, it is of the same order as the global interpolation error. The global evaluation error in the displacement after two iterations ($n = 2$) is $O(\tau^{m+1})$ and for sufficiently small τ it is thus negligible compared to the global interpolation error. In conclusion, partitioned methods and monolithic methods possess identical orders of accuracy. However, the proportionality constants can be very different. Furthermore, for appropriately small time steps a monolithic method requires only two subiteration per time step for convergence to sufficient accuracy.

5.3. Nonnormality of C_0. To establish the nonnormality of the null-reflection operator, we derive its adjoint and demonstrate that there exist admissible functions that violate the identity (5.2).

Consider an open interval $\mathcal{T} := (0,\tau)$. Let $\mathcal{Z} := H^1(\mathcal{T})$, equipped with the standard L^2 inner product. Being a closed linear subspace of the Hilbert space L^2, the closure $\overline{\mathcal{Z}}$ of $\mathcal{Z}$ in L^2 is a Hilbert space. To avoid the conveyance of constants, we set $2\mu/\nu = 1$ in (4.10) without loss of generality. A useful device in the derivation of the adjoint operator is the translation operator $\mathsf{T} : z \in \mathcal{Z} \mapsto z - z(0) \in \mathcal{Z}$. As $\mathsf{C}_0 z$ depends on z through Dz only, C_0 is translation invariant and $\mathsf{C}_0\mathsf{T} = \mathsf{C}_0$. To determine the adjoint operator of C_0 on $\overline{\mathcal{Z}}$, we select an arbitrary $w \in \mathcal{Z}$ and associate a function ψ by

$$D^2\psi + \psi = w,\ \psi(\tau) = 0, \quad D\psi(\tau) = 0. \tag{5.17}$$

Moreover, we consider an arbitrary $z \in \mathcal{Z}$ and denote by v its image under $\mathsf{C}_0\mathsf{T}$. The adjoint operator follows from the following chains of identities:

$$\begin{aligned}\langle w, \mathsf{C}_0 z\rangle = \langle w, \mathsf{C}_0\mathsf{T}z\rangle = \langle D^2\psi + \psi, v\rangle = v\, D\psi\big|_0^\tau - Dv\, \psi\big|_0^\tau + \langle \psi, D^2 v + v\rangle \\ = \langle \psi, D^2 v + v\rangle = \langle -\psi, D\,\mathsf{T}z\rangle = -\mathsf{T}z\, \psi\big|_0^\tau + \langle D\psi, \mathsf{T}z\rangle = \langle D\psi, \mathsf{T}z\rangle,\end{aligned} \tag{5.18}$$

and

$$\langle \mathsf{C}_0^* w, z\rangle = \langle w, \mathsf{C}_0 z\rangle = \langle w, \mathsf{C}_0\mathsf{T}z\rangle = \langle \mathsf{C}_0^* w, \mathsf{T}z\rangle. \tag{5.19}$$

The third and sixth identity in (5.18) follow by integration by parts. The boundary terms in (5.18) vanish in virtue of the homogeneous initial and final conditions on v and ψ, respectively, and the property $(\mathsf{T}z)(0) = 0$ of the translation operator. From (5.18)–(5.19) the adjoint operator is readily identified as

$$\mathsf{C}_0^* : \{w \in \mathcal{Z}\} \mapsto \big\{D\psi \in \mathcal{Z} : D^2\psi + \psi = w,\ \psi(\tau) = 0,\ D\psi(\tau) = 0\big\}. \tag{5.20}$$

As w, z in (5.18)–(5.19) are arbitrary, we have the implication $\langle w, \mathsf{C}_0 z\rangle = \langle \mathsf{C}_0^* w, z\rangle$ for all $w, z \in \mathcal{Z}$, in accordance with (5.1). Noting that $\mathcal{Z}$ is dense in $\overline{\mathcal{Z}}$, we find that C_0^* according to (5.20) is the adjoint operator of C_0 on $\overline{\mathcal{Z}}$.

The normality of C_0 can now be disproved straightforwardly by establishing that there exists a $z \in \mathcal{Z}$ such that $\mathsf{C}_0^*\mathsf{C}_0 z \neq \mathsf{C}_0\mathsf{C}_0^* z$. For instance, $\sin \in \mathcal{Z}$ and

$$\begin{aligned}(\mathsf{C}_0^*\mathsf{C}_0 - \mathsf{C}_0\mathsf{C}_0^*)\sin(s) = [2(s-\tau)\cos(s-2\tau) - \sin(s+2\tau) - 2\sin(s-2\tau) \\ -(4\tau s - 3 - 2\tau^2)\sin(s) - 2s\cos(s)]/16 \neq 0,\end{aligned} \tag{5.21}$$

which completes the proof.

5.4. Approximation on polynomials. To elaborate further the properties of the subiteration procedure, we consider the null-reflection characteristic operator on polynomials.

Let $\mathcal{Z}_m$ be the space of polynomials of degree m on the open interval $\mathcal{T} := (0,\tau)$ with bounded H^1 norm. By the Weierstrass approximation theorem, one can conceive of $\mathcal{Z}_m$ as an approximation to the continuous functions in $H^1(\mathcal{T})$. Equipped with the inner product (5.16), $\mathcal{Z}_m$ is a Hilbert space. For given m, we associate to any $z \in \mathcal{Z}_m$ a function $v \in \mathcal{Z}_m$ by the variational statement

$$\mathsf{A}(w, v) = \mathsf{B}(w, z) \qquad \forall w \in \mathcal{Z}_m, \tag{5.22a}$$

where

$$\text{(5.22b)}\quad \mathsf{A}(w,v) := \langle w, D^2 v + v\rangle_{L^2} + w\,Dv\big|_0 - v\,Dw\big|_0, \qquad \mathsf{B}(w,z) := -\langle w, Dz\rangle_{L^2}.$$

Equation (5.22) represents a weak formulation of $v = \mathsf{C}_0 z$ with weakly imposed initial conditions. To make (5.22) amenable to numerical treatment, we introduce a basis $\{\theta_0, \dots, \theta_m\}$ of $\mathcal{Z}_m$, where θ_i is the normalized Legendre polynomial of degree i on the interval $\mathcal{T}$. Equation (5.22) then translates into the linear algebra problem $\mathbf{A}\cdot\bar{\mathbf{v}} = \mathbf{B}\cdot\bar{\mathbf{z}}$, where $A_{ij} = \mathsf{A}(\theta_i,\theta_j)$ and $B_{ij} = \mathsf{B}(\theta_i,\theta_j)$, and the vectors $\bar{\mathbf{v}}$ and $\bar{\mathbf{z}}$ collect the coefficients of v and z with respect to the basis of Legendre polynomials, respectively. Hence, with respect to this basis the mapping C_0 is represented by the matrix $\mathbf{C}_0 := \mathbf{A}^{-1}\cdot\mathbf{B}$.

To express the H^1 norm of C_0 in terms of $\mathbf{C}_0$, we note that for any two functions $v, z \in \mathcal{Z}_m$ the H^1 inner product can be expressed as

$$\text{(5.23)}\quad \langle v, z\rangle_{H^1} = \sum_{i,j} \bar{v}_i W_{ij} \bar{z}_j \quad \text{with} \quad W_{ij} = \langle \theta_i, \theta_j\rangle_{H^1} = \langle \theta_i, \theta_j\rangle_{L^2} + \langle D\theta_i, D\theta_j\rangle_{L^2}.$$

The Gram matrix $\mathbf{W}$ is symmetric positive definite and, hence, it admits a Cholesky factorization, i.e., there exists an upper-triangular matrix $\mathbf{Q}$ such that $\mathbf{W} = \mathbf{Q}^T\cdot\mathbf{Q}$. Accordingly, we can express the H^1 norm of any $z \in \mathcal{Z}_m$ as

$$\text{(5.24)}\qquad \|z\|_{H^1}^2 = \langle z, z\rangle_{H^1} = \sum_{i,j} \bar{z}_i \langle \theta_i, \theta_j\rangle_{H^1} \bar{z}_j = \bar{\mathbf{z}}^T\cdot\mathbf{Q}^T\cdot\mathbf{Q}\cdot\bar{\mathbf{z}} = \|\mathbf{Q}\cdot\bar{\mathbf{z}}\|^2.$$

In (5.24), the vector norm $\|\cdot\| : \mathbb{R}^m \mapsto \mathbb{R}$ connotes the standard L^2-norm. The H^1 norm of C_0 on $\mathcal{Z}_m$ can thus be determined as follows:

$$\begin{aligned}\|\mathsf{C}_0\|_{H^1} &= \sup_{\substack{0\neq v\in\mathcal{Z}_m\\ 0\neq z\in\mathcal{Z}_m}} \frac{|\langle v, \mathsf{C}_0 z\rangle_{H^1}|}{\|v\|_{H^1}\|z\|_{H^1}} = \sup_{\substack{0\neq \bar{\mathbf{v}}\in\mathbb{R}^m\\ 0\neq \bar{\mathbf{z}}\in\mathbb{R}^m}} \frac{|\bar{\mathbf{v}}^T\cdot\mathbf{W}\cdot\mathbf{C}_0\cdot\bar{\mathbf{z}}|}{\|\mathbf{Q}\cdot\bar{\mathbf{v}}\|\|\mathbf{Q}\cdot\bar{\mathbf{z}}\|}\\ &= \sup_{\substack{0\neq \hat{\mathbf{v}}\in\mathbb{R}^m\\ 0\neq \hat{\mathbf{z}}\in\mathbb{R}^m}} \frac{|\hat{\mathbf{v}}^T\cdot\mathbf{Q}\cdot\mathbf{C}_0\cdot\mathbf{Q}^{-1}\cdot\hat{\mathbf{z}}|}{\|\hat{\mathbf{v}}\|\|\hat{\mathbf{z}}\|} = \|\mathbf{Q}\cdot\mathbf{C}_0\cdot\mathbf{Q}^{-1}\|,\end{aligned} \qquad \text{(5.25)}$$

where $\hat{(\cdot)} := \mathbf{Q}\cdot\bar{(\cdot)}$. Note that the final form in (5.25) is computable. Table 2 lists the norm $\|\mathsf{C}_0\|_{H^1(\mathcal{T})}$ for representative values of the interval length τ and the polynomial degree m. A discussion of the results is postponed to the end of this section.

The question pertaining to the relation between the spectrum of C_0 on a function space such as H^1, and the spectrum of C_0 on a finite-dimensional subspace such as $\mathcal{Z}_m$, is a profoundly difficult one, and we shall not pursue it here. Let us just mention that the spectrum of any bounded operator is nonempty [14, p. 176] and, in particular, the spectrum of C_0 on H^1 is nonempty. Below, we consider the spectrum of C_0 on $\mathcal{Z}_m$ independently. To relate the pseudospectra of C_0 on $\mathcal{Z}_m$ to the representation matrix $\mathbf{C}_0$, we state the following identities:

$$\text{(5.26)}\qquad \big\|(\mathsf{C}_0 - \zeta\mathsf{I})^{-1}\big\|_{H^1} = \big\|\mathbf{Q}\cdot(\mathbf{C}_0 - \zeta\mathbf{I})^{-1}\cdot\mathbf{Q}^{-1}\big\| = \big\|(\mathbf{Q}\cdot\mathbf{C}_0\cdot\mathbf{Q}^{-1} - \zeta\mathbf{I})^{-1}\big\|.$$

The second expression implies that the spectrum of C_0 on $\mathcal{Z}_m$ coincides with the spectrum of $\mathbf{C}_0$. The final expression demonstrates that the H^1 pseudospectra of C_0 on $\mathcal{Z}_m$ coincide with the pseudospectra of $\mathbf{Q}\cdot\mathbf{C}_0\cdot\mathbf{Q}^{-1}$. This reflects that the spectrum does not depend on the selected norm (H^1 or L^2), but the pseudospectra do. Figure 1

TABLE 2
H^1 norm and spectral radius of the null-reflection operator C_0 with $2\mu/\nu = 1$ on $\mathcal{Z}_m$.

	$m\backslash\tau$	10^{-2}	10^{-1}	10^{0}	10^{1}	10^{2}
$\|\mathsf{C}_0\|_{H^1}$	2^6	6.366×10^{-3}	6.363×10^{-2}	6.029×10^{-1}	3.684×10^{0}	3.235×10^{1}
	2^7	6.366×10^{-3}	6.363×10^{-2}	6.029×10^{-1}	3.684×10^{0}	3.235×10^{1}
	2^8	6.366×10^{-3}	6.363×10^{-2}	6.029×10^{-1}	3.684×10^{0}	3.235×10^{1}
$\operatorname{spr} \mathsf{C}_0$	2^6	3.947×10^{-4}	3.947×10^{-3}	3.941×10^{-2}	3.461×10^{-1}	3.872×10^{0}
	2^7	4.161×10^{-4}	4.161×10^{-3}	4.154×10^{-2}	3.600×10^{-1}	2.037×10^{0}
	2^8	4.314×10^{-4}	4.314×10^{-3}	4.307×10^{-2}	3.697×10^{-1}	2.099×10^{0}

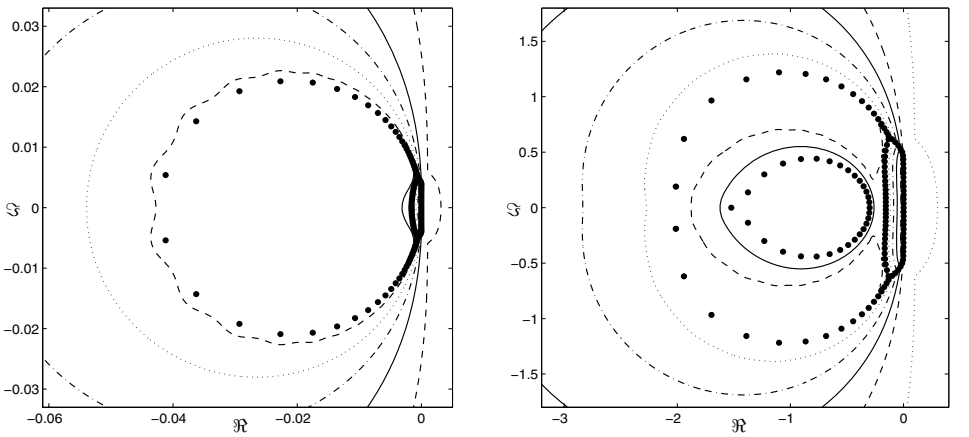


FIG. 1. *Spectra* (•) *and H^1 ϵ-pseudospectra of C_0 on $\mathcal{Z}_m$ for $m = 2^7$, $\tau = 1$* (left), *$\tau = 10^2$* (right), *$\epsilon = 10^{-13}$* (—), *10^{-11}* (−−), *10^{-9}* (···), *10^{-7}* (−·), *10^{-5}* (—), *10^{-3}* (−−), *10^{-1}* (···).

displays the H^1 pseudospectra[1] of C_0 for $m = 2^7$, and $\tau = 1$ and $\tau = 10^2$. Moreover, Table 2 lists the spectral radius of C_0 for representative values of τ and m.

The coercivity coefficient $\gamma(\mathsf{C}_0)$ can be determined as follows:

$$\inf_{0\neq\{\hat{z}_i\}\in\hat{\mathcal{Z}}_m} \sup_{0\neq v\in\mathcal{Z}_m} \frac{\left|\Sigma_i \langle v, \varphi_i\rangle_{H^1} \hat{z}_i\right|}{\|v\|_{H^1} \|\hat{z}.\|} = \inf_{0\neq\{\hat{z}_i\}\in\hat{\mathcal{Z}}_m} \sup_{0\neq\bar{\mathbf{v}}\in\mathbb{R}^m} \frac{\left|\Sigma_i \bar{\mathbf{v}}^T \cdot \mathbf{W} \cdot \bar{\boldsymbol{\varphi}}_j \|\mathbf{Q}\cdot\bar{\boldsymbol{\varphi}}_i\|^{-1} \hat{z}_i\right|}{\|\mathbf{Q}\cdot\bar{\mathbf{v}}\| \, \|\hat{z}.\|}$$

$$(5.27) = \inf_{0\neq\hat{\mathbf{z}}\in\mathbb{R}^m} \sup_{0\neq\mathbf{w}\in\mathbb{R}^m} \frac{\left|\mathbf{w}^T \cdot \mathbf{U} \cdot \hat{\mathbf{z}}\right|}{\|\mathbf{w}\| \, \|\hat{\mathbf{z}}\|} = \inf_{0\neq\mathbf{z}\in\mathbb{R}^m} \sup_{0\neq\mathbf{u}\in\mathbb{R}^m} \frac{\left|\mathbf{u}^T \cdot \mathbf{F}^T \cdot \mathbf{U} \cdot \mathbf{G} \cdot \mathbf{z}\right|}{\|\mathbf{u}\| \, \|\mathbf{z}\|} = \rho_{\min}.$$

with $\mathbf{w} := \mathbf{Q}\cdot\bar{\mathbf{v}}$, $\mathbf{U} := \left(\mathbf{Q}\cdot\hat{\boldsymbol{\varphi}}_1/\|\mathbf{Q}\cdot\hat{\boldsymbol{\varphi}}_1\|, \mathbf{Q}\cdot\hat{\boldsymbol{\varphi}}_2/\|\mathbf{Q}\cdot\hat{\boldsymbol{\varphi}}_2\|, \dots\right)$ and $\rho_{\min}$ the smallest singular value of $\mathbf{U}$. One simply infers that $\mathbf{U}$ contains the normalized eigenvectors of $\mathbf{Q}\cdot\mathbf{C}_0\cdot\mathbf{Q}^{-1}$. The matrices $\mathbf{F}$, $\mathbf{R}$, and $\mathbf{G}$ pertain to the singular-value decomposition of $\mathbf{U}$ according to $\mathbf{U} = \mathbf{F}\cdot\mathbf{R}\cdot\mathbf{G}^T$; see, e.g., [11, 12]. As the matrices $\mathbf{F}$ and $\mathbf{G}$ are unitary, the transformations $\mathbf{w} = \mathbf{F}\cdot\mathbf{u}$ and $\hat{\mathbf{z}} = \mathbf{G}\cdot\mathbf{z}$ are norm preserving. This leads to the third identity. Similarly, we obtain for the continuity coefficient $\gamma'(\mathsf{C}_0) = \rho_{\max}$. For the parameters in Table 2, the coercivity and continuity coefficients thus computed yield condition numbers $\kappa(\mathsf{C}_0) > 10^{10}$.

[1]We thankfully acknowledge the use of the *EigTool* package by T.G. Wright, M. Embree, and L.N. Trefethen for producing the pseudospectra plots in Figure 1; see [27] for further information.

The computed norms in Table 2 are virtually independent of the polynomial degree m. This is a result of the smoothness of the function that induces the norm and of its image, i.e., the functions $\bar{v}$ and $\bar{z}$ defined by

$$\{\bar{v}, \bar{z}\} := \arg \sup_{\substack{0 \neq v \in H^1 \\ 0 \neq z \in H^1}} \frac{\left|\langle v, \mathsf{C}_0 z \rangle_{H^1}\right|}{\|v\|_{H^1} \|z\|_{H^1}} \tag{5.28}$$

are sufficiently smooth to render the error in their approximation in $\mathcal{Z}_m$ with $m \in \{2^6, 2^7, 2^8\}$ negligible. The erratic behavior of the computed spectral radius conjecturally emanates from the nonnormality of C_0. Figure 1 illustrates that the ϵ-pseudospectra of C_0 are much larger than the union of ϵ-discs centered at the spectrum, in accordance with section 5.1. As $\mathcal{S}_\epsilon(\mathsf{C}_0)$ is the aggregate of the spectra of all perturbed operators $\mathsf{C}_0 + \mathsf{C}_0'$ with $\|\mathsf{C}_0'\| \leq \epsilon$ (see [14, p. 208] or [10, p. 4]), i.e.,

$$\mathcal{S}_\epsilon(\mathsf{C}_0) = \{\zeta \in \mathbb{C} : \zeta \in \mathcal{S}(\mathsf{C}_0 + \mathsf{C}_0'), \|\mathsf{C}_0'\| \leq \epsilon\} \tag{5.29}$$

(in the appropriate norm) this implies that the spectrum is sensitive to perturbations of the operator. An alternative measure for this sensitivity is the condition of the eigenvalues; see, e.g., [11, p. 323]. For the results in Table 2, the condition of the largest eigenvalue exceeds 10^{10}.

The results in Table 2 indicate that $\operatorname{spr} \mathsf{C}_0 \propto \|\mathsf{C}_0\|_{H^1} \propto \tau$ as $\tau \to 0$. To verify this hypothesis, consider C_0 on the interval $s \in (0, \tau)$ and define an auxiliary variable $\xi := s/\tau$. For any admissible z with image $v := \mathsf{C}_0 z$ it holds that

$$\frac{d^2 v}{d\xi^2} \frac{1}{\tau^2} + v = -\frac{dz}{d\xi} \frac{1}{\tau}, \qquad \xi \in (0, 1). \tag{5.30}$$

Because $D^2 v$ is the principal term in the operator, the term v can be ignored as $\tau \to 0$. Equation (5.30) is then equivalent to $D^2 v = -\tau\, Dz$. Noting that τ appears as a multiplicative factor, the result is immediate. In general, for sufficiently small τ, the spectrum and pseudospectra of C_0 scale with τ. As τ increases, the principal term $D^2 v$ becomes a singular perturbation to v and the spectrum deforms; see also Figure 1. For sufficiently small τ, the proportionality of $\operatorname{spr} \mathsf{C}_0$ and $\|\mathsf{C}_0\|$ to τ indicates that the convergence of the subiteration process deteriorates linearly with an increase of the time interval. Let us also note that the results in Table 2 comply with the upper bound $\|\mathsf{C}_0\|_{H^1} \leq 2\tau$ derived in Appendix B. Moreover, the results indicate that the bound is reasonably sharp.

To illustrate the effect of nonnormality on the convergence of the subiteration method, the left and right plots in Figure 2, respectively, plot $\|\mathsf{C}_0^n\|_{H^1}$ on the polynomials $\mathcal{Z}_m$ $(= \|(\mathbf{Q} \cdot \mathbf{C}_0 \cdot \mathbf{Q}^{-1})^n\|)$ with $m = 2^8$ versus the iteration counter n for $2\mu/\nu = 1$ and representative lengths of the time interval τ, and for $\tau = 1$ and representative values of the parameter $2\mu/\nu$. In all cases, the spectral radius is less than 1 and the subiteration method is formally stable. However, monotone convergence occurs only if $\|\mathsf{C}_0\|_{H^1} \leq 1$. The figures illustrate that if $\|\mathsf{C}_0\|$ is large, then the error can increase by many orders of magnitude before the asymptotic convergence sets in. Note that this behavior is in accordance with the indication provided by the condition numbers $\kappa(\mathsf{C}_0)$. Furthermore, it is to be noted that the difference between the results in the right plot in Figure 2 emanates exclusively from a rescaling of the spectrum, as $2\mu/\nu$ appears as a multiplicative factor to the operator; cf. (4.9). The nonnormality of the underlying operators (in any suitable measure, e.g., $\kappa(\mathsf{C}_0)$ or $\|\mathsf{C}_0\|/\operatorname{spr} \mathsf{C}_0$) is

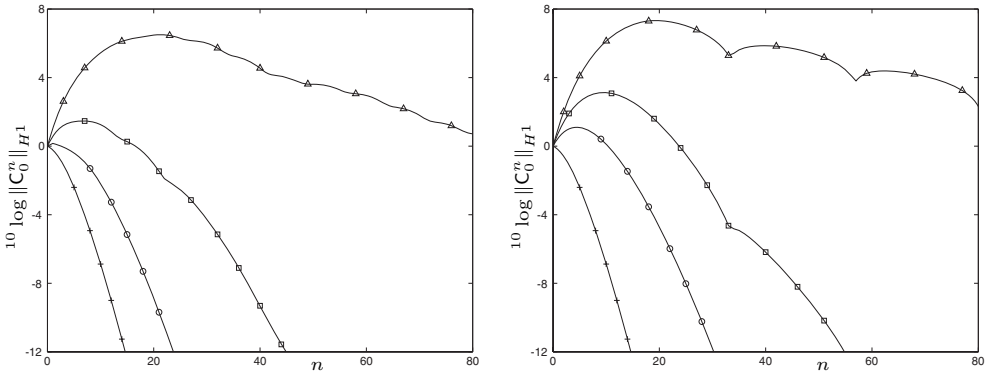


FIG. 2. *Convergence of subiteration:* $\|\mathsf{C}_0^n\|_{H^1}$ *on* $\mathcal{Z}_m$ $(m = 2^8)$ *with* (left) $2\mu/\nu = 1$ *and* $\tau = 10^0$ (+), $\tau = 10^{1/2}$ (○), $\tau = 10^1$ (□), $\tau = 10^{3/2}$ (△), *and* (right) $\tau = 1$ *and* $2\mu/\nu = 1$ (+), $2\mu/\nu = 5$ (○), $2\mu/\nu = 10$ (□), $2\mu/\nu = 20$ (△).

identical. This exemplifies that if the spectral radius is sufficiently small, then the effect of nonnormality on the convergence of the subiteration method is concealed.

By definition, the operator norm $\|\mathsf{C}_0^n\|_{H^1}$ is an upper bound on the convergence behavior pertaining to a particular element of H^1, i.e., $\|\mathsf{C}_0^n\|_{H^1} \geq \|\mathsf{C}_0^n z\|_{H^1}\|z\|_{H^1}^{-1}$ for all $z \in H^1$. The upper bound is sharp in the sense that there exists an element in H^1 for which equality holds. However, in practice, the initial error is typically confined to a proper subspace of H^1. For instance, if the solution is sufficiently regular and the initial approximation on a time interval is obtained by a k-order extrapolation of the solution on the previous time interval, then the initial error e is confined to a regular subspace of H^1 with, moreover, $[D^i e](0) = 0$ for all $i = 0, 1, \ldots, k-1$. The norm of C_0^n on this subspace can be much smaller, yielding $\|\mathsf{C}_0^n\|_{H^1}$ practically unsharp. To illustrate the effect of confinement of the initial error due to extrapolation, Figure 3 plots $\|\mathsf{C}_0^n z\|_{H^1}\|z\|_{H^1}^{-1}$ versus n for $\tau = 1$ and $2\mu/\nu = 10$, and $z(s) = s^k$, $k = 1, 2, 3, 4$. Note that s^k has $k-1$ vanishing derivatives at $s = 0$. Hence, we can conceive of s^k as a k-order-extrapolation error. Figure 3 indicates that the nonnormality-induced divergence is reduced with increasing k. Moreover, for large k, the discrepancy between $\|\mathsf{C}_0^n\|_{H^1}$ and $\|\mathsf{C}_0^n z\|_{H^1}\|z\|_{H^1}^{-1}$ can increase to several orders of magnitude.

6. Nonnormality of the subiteration operator. In this section we are concerned with the properties of the general characteristic operator , i.e., the operator including reflections.

6.1. Nonnormality of C. The normality of C can be disproved by deriving the associated adjoint operator and establishing that an admissible z exists that contradicts $\mathsf{C}^*\mathsf{C}z = \mathsf{C}\mathsf{C}^*z$. Let the space of admissible functions $\mathcal{Z}$ be defined as in section 5.3. Recalling the definition of the shift operator (4.12), from the sequence of identities

$$\langle \mathsf{H}_\nu^* v, z\rangle_{L^2} = \langle v, \mathsf{H}_\nu z\rangle_{L^2} = \int_\nu^\tau v(s)z(s-\nu)\,ds = \int_0^{\tau-\nu} v(s+\nu)z(s)\,ds \tag{6.1}$$

we readily identify the adjoint shift operator as

$$(\mathsf{H}_\nu^* z)(s) := \begin{cases} 0 & \nu \leq s < \tau, \\ z(s+\nu) & 0 < s < \tau - \nu. \end{cases} \tag{6.2}$$

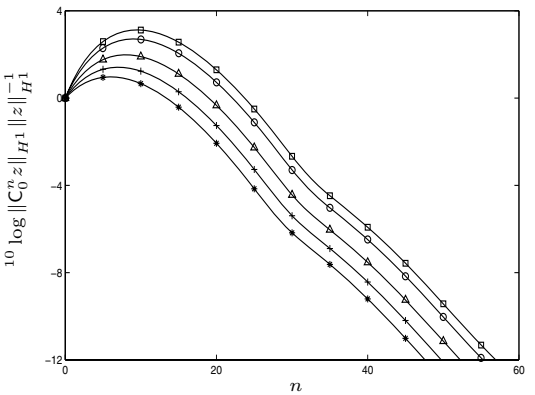


FIG. 3. *Convergence of subiteration:* $\|\mathsf{C}_0^n z\|_{H^1}\|z\|_{H^1}^{-1}$ *on* $\mathcal{Z}_m$ $(m = 2^8)$ *with* $\tau = 1$, $2\mu/\nu = 10$ *and* $z(s) = s^k$, $k = 1$ $(\circ)$, $k = 2$ $(\triangle)$, $k = 3$ $(+)$, *and* $k = 4$ $(*)$. *Norm* $\|\mathsf{C}_0^n\|_{H^1}$ *indicated by* $(\square)$.

Moreover, on account of (4.11) it holds that

$$\langle v, \mathsf{C}z\rangle_{L^2} = \big\langle v, \mathsf{C}_0 z + 2\textstyle\sum_j \mathsf{H}_{j\nu}\mathsf{C}_0 z\big\rangle_{L^2} = \big\langle \mathsf{C}_0^* v + 2\textstyle\sum_j \mathsf{C}_0^*\mathsf{H}_{j\nu}^* v, z\big\rangle_{L^2}. \tag{6.3}$$

One can infer that H_ν^* commutes with C_0^* and, hence,

$$\mathsf{C}^* = \left(\mathsf{I} + 2\sum_i^{\lfloor\tau/\nu\rfloor} \mathsf{H}_{i\nu}^*\right)\mathsf{C}_0^*. \tag{6.4}$$

The normality of C can now be disproved straightforwardly by establishing that there exists a $z \in \overline{\mathcal{Z}}$ such that $\mathsf{C}^*\mathsf{C}z \neq \mathsf{C}\mathsf{C}^* z$.

6.2. Approximation on polynomials. We consider the characteristic operator C on the polynomials $\mathcal{Z}_m$ on an interval $\mathcal{T} := (0, \tau)$ similarly as in section 5.4. The representation matrix $\mathbf{C}$ can be derived from the null-reflection matrix $\mathbf{C}_0$ and a shift matrix on the basis of (4.11). Recalling that θ_i designates the normalized Legendre polynomial of degree i, we define the representation matrix $\mathbf{H}_\nu$ of the shift operator H_ν according to (4.12) as

$$H_{ij}(\nu) := \langle \theta_i, \mathsf{H}_\nu \theta_j\rangle_{L^2} = \int_\nu^\tau \theta_i(s)\theta_j(s-\nu)\, ds. \tag{6.5}$$

Equation (4.11) leads to

$$\mathbf{C} := \left(\mathbf{I} + 2\sum_{j=1}^{\lfloor\tau/\nu\rfloor} \mathbf{H}^j(\nu)\right)\cdot \mathbf{C}_0. \tag{6.6}$$

By means of (6.6) we can establish the characteristics of C on $\mathcal{Z}_m$ through analogous expressions to (5.25)–(5.27).

The top and bottom plots in Figure 4, respectively, display the norm and the spectral radius of the subiteration operator C on $\mathcal{Z}_m$ with $m = 2^8$ versus ν^{-1} for $\tau = 10^{-2}, 10^{-1}, \ldots, 10^2$. Furthermore, the properly rescaled norm and spectral radius of the quasi-steady operator are displayed, viz., $(2/\nu)\,\mathrm{spr}\,\mathsf{C}_\infty$ and $(2/\nu)\,\|\mathsf{C}_\infty\|$ with $4\mu/\nu^2$ in (4.14) set to 1. For completeness, we mention that the representation matrix of

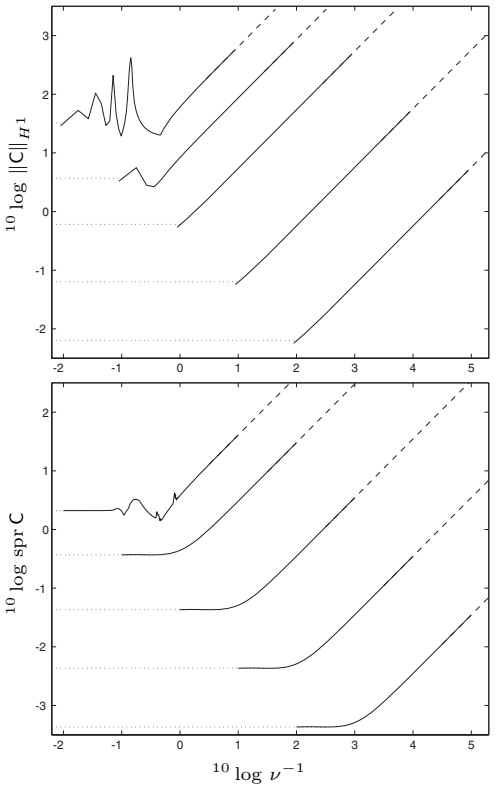


FIG. 4. *Plots* (log-log) *of the* H^1 *norm* (top) *and the spectral radius* (bottom) *of* C (—), C_0 $(\cdots)$ *and* C_∞ (– –, rescaled) *on* $\mathcal{Z}_m$ $(m = 2^8)$ *for* $\tau = 10^{-2}, 10^{-1}, \ldots, 10^2$ (bottom to top).

C_∞ with respect to $\{\theta_i\}$ is given by $\mathbf{C}_\infty := -\mathbf{A}^{-1} \cdot \mathbf{T}$ with $A_{ij} = \mathsf{A}(\theta_i, \theta_j)$ according to (5.22b) and $T_{ij} := \langle \theta_i, \mathsf{T}\theta_j \rangle_{L^2} = \delta_{ij} - \langle \theta_i, \theta_j(0) \rangle_{L^2}$. Note that if $\nu > \tau$, then $\mathsf{C} = \mathsf{C}_0$ and $\operatorname{spr}\mathsf{C}$ and $\|\mathsf{C}\|$ coincide with the corresponding entries in Table 2. The distance between the curves in Figure 4 indicates that $\operatorname{spr}\mathsf{C} \propto \|\mathsf{C}\|_{H^1} \propto \tau^2$ for appropriately small ν and τ ($\nu < \tau$). Noting that for any $z \in \mathcal{Z}$ with image $v := \mathsf{C}_\infty z$ it holds that

$$\frac{d^2 v}{d\xi^2} \frac{1}{\tau^2} + v = -(z - z(0)), \qquad \xi \in (0, 1), \tag{6.7}$$

by the same rationale as in section 5.4 we conclude that the norm and spectral radius of C_∞ are indeed proportional to τ^2 as $\tau \to 0$. In particular, this implies that for appropriately small ν and τ the convergence of the subiteration method degrades quadratically with increasing τ. Figure 4, moreover, indicates that for small τ the spectral radius of C is insensitive to a few reflections, but its norm is not: $\operatorname{spr}\mathsf{C}$ remains approximately constant upto $\tau/\nu = 10$, but $\|\mathsf{C}\|$ increases immediately if τ/ν exceeds 1. For large τ, the spectral radius and the norm of C behave erratically in the transition from C_0 to C_∞. We shall not further pursue the intricacies of this transition.

To demonstrate the effect of nonnormality on the convergence of the subiteration method in the presence of reflections, the top and bottom plots in Figure 5 plot $\|\mathsf{C}^n\|_{H^1}$ on $\mathcal{Z}_m$ $(= \|(\mathbf{Q} \cdot \mathbf{C} \cdot \mathbf{Q}^{-1})^n\|)$ with $m = 2^8$ versus the iteration counter n for $\tau/\nu = 2^2, 2^3, \ldots, 2^7$, and for $\tau = 1$ and $2\mu/\nu = 2$, and $\tau = 10^2$ and $2\mu/\nu = 0.2$, respectively. The values of $2\mu/\nu$ have been selected such that $\operatorname{spr}\mathsf{C} < 1$ in all

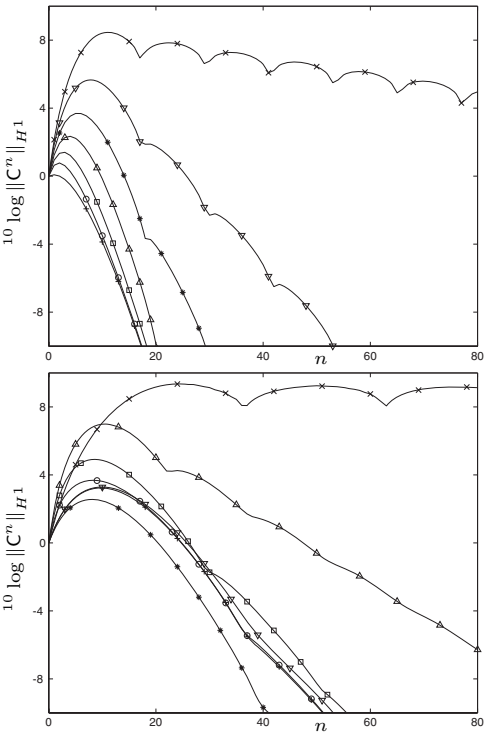


FIG. 5. *Convergence of subiteration:* $\|\mathsf{C}^n\|_{H^1}$ *on* $\mathcal{Z}_m$ $(m = 2^8)$ *with* $\tau = 1$ *and* $2\mu/\nu = 2$ (top), $\tau = 10^2$ *and* $2\mu/\nu = 0.2$ (bottom), *and* $\tau/\nu < 1$ (+), $\tau/\nu = 2^2$ ($\circ$), $\tau/\nu = 2^3$ ($\square$), $\tau/\nu = 2^4$ ($\triangle$), $\tau/\nu = 2^5$ ($*$) $\tau/\nu = 2^6$ ($\triangledown$), *and* $\tau/\nu = 2^7$ ($\times$).

cases. Figure 5 corroborates that the asymptotic convergence of the subiteration method is insensitive to a few reflections, in conformity with the aforementioned insensitivity of the spectral radius. However, as indicated by the sensitivity of the norm, the initial convergence behavior depends notably on the number of reflections. Furthermore, the figure shows that for $\tau = 1$ both the initial divergence and the asymptotic-convergence rate deteriorate gradually with decreasing ν. The gradual deterioration is in accordance with the smooth transition from $\operatorname{spr}\mathsf{C}_0$ to $\operatorname{spr}\mathsf{C}_\infty$ and from $\|\mathsf{C}_0\|$ to $\|\mathsf{C}_\infty\|$ as ν decreases in Figure 4. For $\tau = 10^2$ the convergence behavior behaves erratically with increasing τ/ν, in accordance with the erratic behavior of $\operatorname{spr}\mathsf{C}$ and $\|\mathsf{C}\|$ in the transition from C_0 to C_∞. On account of the nonnormality of the subiteration process, the error can again increase by many orders of magnitude before asymptotic convergence sets in.

7. Conclusion. We investigated the basic subiteration method for solving the aggregated equations in fluid-structure-interaction problems, on the basis of a prototypical model problem. An important new result is that the subiteration method is characterizable by recursion of a nonnormal operator. This implies that the norm and spectral radius of the operator are disparate, reflecting in profoundly dissimilar transient and asymptotic convergence behavior of the subiteration method. Convergence can be nonmonotonous and divergence can precede the asymptotic convergence, despite formal stability. By means of approximation on polynomials we established that the transient divergence can in fact amplify the initial error by many orders of

magnitude. Hence, the nonnormality induces a severe degradation in the robustness and efficiency of the subiteration method.

Auxiliary information provided by the analysis concerns the dependence of the stability and convergence of the subiteration method on the physical parameters in the problem and on the computational time step. We established that the convergence rate of the subiteration method is generally proportional to the ratio of the characteristic fluid mass over the structure mass. In the absence of reflections the convergence rate depends linearly on the ratio of the characteristic structure time to the characteristic fluid time. As the number of reflections increases, this dependence becomes quadratic. Moreover, in the absence of reflections the convergence rate is uniformly proportional to the time-step size. In particular, this proportionality implies that for appropriately small time steps, only two iterations are required for convergence to sufficient accuracy.

It is anticipated that the results generalize mutatis mutandis to other fluid-structure-interaction problems. In particular, we conjecture that the nonnormality of the subiteration method is independent of the underlying problem.

Appendix A. The generalized nonlinear Lax–Milgram theorem. Let $(\mathcal{U}, \langle\cdot,\cdot\rangle_{\mathcal{U}})$ and $(\mathcal{V}, \langle\cdot,\cdot\rangle_{\mathcal{V}})$ be two real Hilbert spaces with corresponding norms $\|\cdot\|_{\mathcal{U}}$ and $\|\cdot\|_{\mathcal{V}}$. Let the functional $\mathsf{P} : \mathcal{V} \times \mathcal{U} \mapsto \mathbb{R}$ be linear in its first argument and possibly nonlinear in its second argument, and let $\mathsf{p} : \mathcal{V} \mapsto \mathbb{R}$ be a linear functional. Sufficient conditions for the existence of a unique solution to the variational problem

$$\mathsf{P}(v,u) = \mathsf{p}(v) \qquad \forall v \in \mathcal{V} \tag{A.1}$$

are:

(i) p is a continuous linear functional on $\mathcal{V}$,
(ii) $\mathsf{P}(\cdot,u)$ is a continuous linear functional on $\mathcal{V}$ for all $u \in \mathcal{U}$,
(iii) $\inf_{\substack{u,u'\in\mathcal{U}\\ u\neq u'}} \sup_{0\neq v\in\mathcal{V}} \frac{|\mathsf{P}(v,u')-\mathsf{P}(v,u)|}{\|u'-u\|_{\mathcal{U}}\|v\|_{\mathcal{V}}} \geq C_1 > 0$,
(iv) $|\mathsf{P}(v,u') - \mathsf{P}(v,u)| \leq C_2\|u'-u\|_{\mathcal{U}}\|v\|_{\mathcal{V}} \qquad \forall u,u' \in \mathcal{U}$ and $\forall v \in \mathcal{V}$,
(v) $\sup_{u\in\mathcal{U}} |\mathsf{P}(v,u)| > 0 \quad \forall v \in \mathcal{V}\setminus\{0\}$

for certain positive constants C_1 and C_2.

The proof proceeds in a similar manner as for the linear case in [18, p. 316 ff.]. By (ii) and the Riesz representation theorem, for each fixed $u \in \mathcal{U}$ there exists a unique element $\Lambda_{\mathsf{P}}u$ of $\mathcal{V}$ such that $\mathsf{P}(v,u) = \langle v, \Lambda_{\mathsf{P}}u\rangle_{\mathcal{V}}$ for all $v \in \mathcal{V}$. This correspondence defines an operator $\Lambda_{\mathsf{P}} : \mathcal{U} \mapsto \mathcal{V}$. Similarly, by (i) and the Riesz theorem, there exists a $v_{\mathsf{p}} \in \mathcal{V}$ such that $\mathsf{p}(v) = \langle v, v_{\mathsf{p}}\rangle_{\mathcal{V}}$ for all $v \in \mathcal{V}$. Hence, the variational problem is equivalent to the operator equation $\Lambda_{\mathsf{P}}u = v_{\mathsf{p}}$. The existence of a unique solution to the operator equation is ensured if the operator $\Lambda_{\mathsf{P}} : \mathcal{U} \mapsto \mathcal{V}$ is bijective, i.e., if Λ_{P} is surjective and injective; see [28, pp. 16–17].

To prove the injectivity of Λ_{P}, we must establish that $\Lambda_{\mathsf{P}}u' = \Lambda_{\mathsf{P}}u$ implies $u' = u$ or, equivalently, that $\|\Lambda_{\mathsf{P}}u' - \Lambda_{\mathsf{P}}u\|_{\mathcal{V}} = 0$ implies $\|u'-u\|_{\mathcal{U}} = 0$. By property (iii) it holds that

$$\begin{aligned}\|\Lambda_{\mathsf{P}}u' - \Lambda_{\mathsf{P}}u\|_{\mathcal{V}} &= \sup_{0\neq v\in\mathcal{V}} \frac{|\langle v, \Lambda_{\mathsf{P}}u' - \Lambda_{\mathsf{P}}u\rangle_{\mathcal{V}}|}{\|v\|_{\mathcal{V}}} = \sup_{0\neq v\in\mathcal{V}} \frac{|\langle v, \Lambda_{\mathsf{P}}u'\rangle_{\mathcal{V}} - \langle v, \Lambda_{\mathsf{P}}u\rangle_{\mathcal{V}}|}{\|v\|_{\mathcal{V}}} \\ &= \sup_{0\neq v\in\mathcal{V}} \frac{|\mathsf{P}(v,u') - \mathsf{P}(v,u)|}{\|v\|_{\mathcal{V}}} \geq C_1\|u'-u\|_{\mathcal{U}}\end{aligned} \tag{A.2}$$

for all u' and u in $\mathcal{U}$. Hence, injectivity is ascertained.

To prove the surjectivity of Λ_{P}, we must verify that $\Lambda_{\mathsf{P}}\mathcal{U} = \mathcal{V}$, i.e., the range $\mathcal{R}(\Lambda_{\mathsf{P}})$ of the operator Λ_{P} with domain $\mathcal{U}$ coincides with $\mathcal{V}$. To this end, we show that $\mathcal{R}(\Lambda_{\mathsf{P}})$ is closed and, moreover, that its orthogonal complement in $\mathcal{V}$ is empty. To establish closedness of $\mathcal{R}(\Lambda_{\mathsf{P}})$, we note that for any pair u_n and u_m in $\mathcal{U}$ it holds that

$$\|\Lambda_{\mathsf{P}}u_n - \Lambda_{\mathsf{P}}u_m\|_{\mathcal{V}} = \sup_{0\neq v\in\mathcal{V}} \frac{|\langle v, \Lambda_{\mathsf{P}}u_n\rangle_{\mathcal{V}} - \langle v, \Lambda_{\mathsf{P}}u_m\rangle_{\mathcal{V}}|}{\|v\|_{\mathcal{V}}} = \sup_{0\neq v\in\mathcal{V}} \frac{|\mathsf{P}(v,u_n) - \mathsf{P}(v,u_m)|}{\|v\|_{\mathcal{V}}}$$

$$\leq C_2\|u_n - u_m\|_{\mathcal{U}}. \tag{A.3}$$

Hence, any Cauchy sequence $\{u_n\}$ in $\mathcal{U}$ generates a Cauchy sequence $\{\Lambda_{\mathsf{P}}u_n\}$ in $\mathcal{R}(\Lambda_{\mathsf{P}})$ and $u_n \to u \in \mathcal{U}$ implies $\Lambda_{\mathsf{P}}u_n \to \Lambda_{\mathsf{P}}u \in \mathcal{R}(\Lambda_{\mathsf{P}})$. Therefore, $\mathcal{R}(\Lambda_{\mathsf{P}})$ is closed. Now suppose that the orthogonal complement $\mathcal{R}(\Lambda_{\mathsf{P}})^{\perp}$ of $\mathcal{R}(\Lambda_{\mathsf{P}})$ in $\mathcal{V}$ is nonempty. Then there exists a $v' \in \mathcal{R}(\Lambda_{\mathsf{P}})^{\perp}$ such that $\langle v', \Lambda_{\mathsf{P}}u\rangle_{\mathcal{V}} = 0$ for all $u \in \mathcal{U}$. However, this is equivalent to $|\langle v', \Lambda_{\mathsf{P}}u\rangle_{\mathcal{V}}| = |\mathsf{P}(v', u)| = 0$ for all $u \in \mathcal{U}$, in contradiction to stipulation (v). Hence, such a v' is nonexistent. This implies that $\mathcal{V} = \mathcal{R}(\Lambda_{\mathsf{P}}) \oplus \mathcal{R}(\Lambda_{\mathsf{P}})^{\perp} = \mathcal{R}(\Lambda_{\mathsf{P}}) \oplus \emptyset = \mathcal{R}(\Lambda_{\mathsf{P}})$. Hence, the operator Λ_{P} is surjective.

Appendix B. Unboundedness of $\|\mathsf{C}_0\|_{L^2}$ and boundedness of $\|\mathsf{C}_0\|_{H^1}$. Considering a bounded open interval $\mathcal{T} := (0, \tau)$, the objective is to establish that (i) $\|\mathsf{C}_0\|_{L^2} = \infty$ and (ii) $\|\mathsf{C}_0\|_{H^1} \leq 2\tau < \infty$.

(i) To show that C_0 is unbounded on $L^2(\mathcal{T})$ we consider the sequence of continuous functions $\{z_n\}$ with elements

$$z_n(s) := \begin{cases} 1 - \sqrt{1 - (ns-1)^2} & \text{if } 0 \leq s \leq 1/n, \\ 0 & \text{if } s > 1/n. \end{cases} \tag{B.1}$$

The sequence $\{z_n\}$ is a Cauchy sequence in L^2 and, hence, $z_n \to z \in L^2$ as $n \to \infty$. It is important to note that $\|z_n\|_{L^2(\mathcal{T})} = O(1/\sqrt{n})$ as $n \to \infty$ for any τ independent of n. For the image of z_n under C_0 according to (4.10) we obtain

(B.2)

$$v_n := \mathsf{C}_0 z_n = c_1 \sin(s) + c_2 \cos(s) - \sin(s) \int \sin(s)\, z_n(s)\, ds - \cos(s) \int \cos(s)\, z_n(s)\, ds,$$

the constants c_1 and c_2 being determined by the homogeneous initial conditions on v_n. For $s > 1/n$, v_n can be expanded asymptotically as

$$v_n = \left(1 + \left(\frac{\pi}{2} - \frac{5}{6}\right)\frac{1}{n^2} + O\Big(\frac{1}{n^4}\Big)\right)\sin(s) \tag{B.3}$$

$$+ \left(\left(\frac{\pi}{2} - 1\right)\frac{1}{n} + \left(\frac{1}{2} - \frac{5\pi}{16}\right)\frac{1}{n^3} + O\Big(\frac{1}{n^5}\Big)\right)\cos(s), \tag{B.4}$$

as $n \to \infty$. The expression for v_n in the interval $0 \leq s \leq 1/n$ is not required in what follows. From (B.3) it follows that

$$\|\mathsf{C}_0\|_{L^2} := \sup_{0\neq z\in L^2} \frac{\|\mathsf{C}_0 z\|_{L^2}}{\|z\|_{L^2}} \geq \frac{\|v_n\|_{L^2}}{\|z_n\|_{L^2}} \to \infty \quad \text{as } n \to \infty, \tag{B.5}$$

which completes the proof.

(ii) Denoting by v the image of an arbitrary function $z \in H^1$ under C_0 and setting $2\mu/\nu := 1$ for convenience, (4.10) implies

$$D^2 v + v = -Dz,\ v(0) = 0,\ Dv(0) = 0. \tag{B.6}$$

Multiplying the differential equation by Dv and integrating from 0 to s, we obtain

$$\begin{aligned} -\int_0^s Dv(\xi)\, Dz(\xi)\, d\xi &= \int_0^s Dv(\xi)\left(D^2 v(\xi) + v(\xi)\right) d\xi \\ &= \int_0^s \frac{d}{d\xi}\Big(\frac{1}{2}[Dv(\xi)]^2 + \frac{1}{2}[v(\xi)]^2\Big) d\xi = \frac{1}{2}[Dv(s)]^2 + \frac{1}{2}[v(s)]^2. \end{aligned} \tag{B.7}$$

The final identity in (B.7) is obtained by invoking the initial conditions. Hence,

$$\begin{aligned} \frac{1}{2}\|v\|_{H^1}^2 = \frac{1}{2}\int_0^\tau [Dv(s)]^2 + [v(s)]^2\, ds &= \left| \int_0^\tau \int_0^s Dv(\xi)\, Dz(\xi)\, d\xi\, ds \right| \\ \leq \int_0^\tau \|Dv\|_{L^2(0,s)}\, \|Dz\|_{L^2(0,s)}\, ds &\leq \tau \|Dv\|_{L^2(\mathcal{T})}\, \|Dz\|_{L^2(\mathcal{T})} \leq \tau \|v\|_{H^1(\mathcal{T})}\, \|z\|_{H^1(\mathcal{T})}. \end{aligned} \tag{B.8}$$

The first upper bound is a consequence of Schwarz' inequality (see, e.g., [28]). Equation (B.8) implies $\|\mathsf{C}_0 z\|_{H^1} \leq 2\tau \|z\|_{H^1}$ and, since z is arbitrary, $\|\mathsf{C}_0\|_{H^1} \leq 2\tau$. This was to be proved.

REFERENCES

[1] J. Alonso and A. Jameson, *Fully-implicit time-marching aeroelastic solutions*, AIAA, 56 (1994), pp. 1–13.

[2] M. Bathe and R. Kamm, *A fluid-structure interaction finite element analysis of pulsatile blood flow through a compliant stenotic artery*, Journal of Biomechanical Engineering, 121 (1999), pp. 361–369.

[3] K. Bøhmer, P. Hemker, and H. Stetter, *The defect correction approach*, Comput. Suppl., 5 (1984), pp. 1–32.

[4] L. Broderick and J. Leonard, *Nonlinear response of membranes to ocean waves using boundary and finite elements*, Ocean Engrg., 22 (1995), pp. 731–745.

[5] E. van Brummelen, S. Hulshoff, and R. de Borst, *Energy conservation under incompatibility for fluid-structure interaction problems*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 2727–2748.

[6] E. van Brummelen, C. Michler, and R. de Borst, *Interface-GMRES(R) Acceleration of Subiteration for Fluid-Structure-Interaction Problems*, Tech. report DACS-05-001, Delft University of Technology, Delft, The Netherlands, 2005. Available at http://www.em.lr.tudelft.nl/downloads/DACS-05-001.pdf.

[7] C. Farhat, P. Geuzaine, and G. Brown, *Application of a three-field nonlinear fluidstructure formulation to the prediction of the aeroelastic parameters of an f-16 fighter*, Comput. & Fluids, 32 (2003), pp. 3–29.

[8] C. Farhat, P. Geuzaine, and C. Grandmont, *The discrete geometric conservation law and the nonlinear stability of ALE schemes for the solution of flow problems on moving grids*, J. Comput. Phys., 174 (2001), pp. 669–694.

[9] C. Felippa, K. Park, and C. Farhat, *Partitioned analysis of coupled mechanical systems*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 3247–3270.

[10] I. Gohberg and M. Kreĭn, *Introduction to the Theory of Linear Nonselfadjoint Operators*, Translations of Mathematical Monographs 18, AMS, Providence, RI, 1969.

[11] G. Golub and P. Loan, *Matrix Computations*, Johns Hopkins, 1996.

[12] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[13] M. Heil, *Stokes flow in an elastic tube—a large-displacement fluid-structure interaction problem*, Internat. J. Numer. Methods Fluids, 28 (1998), pp. 243–265.

[14] T. Kato, *Perturbation Theory for Linear Operators*, Grundlehren der Mathematischen Wissenschaft 132, Springer, Berlin, corrected print of the 2nd ed., 1984.
[15] J. Kevorkian and J. Cole, *Perturbation Methods in Applied Mathematics*, Appl. Math. Sci. 34, Springer-Verlag, Berlin, 1981.
[16] P. LeTallec and J. Mouro, *Fluid structure interaction with large structural displacements*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 3039–3067.
[17] R. B. Melville and S. A. Morton, *Fully-implicit aeroelasticity on overset grid systems*, 36th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV, 1998.
[18] J. Oden and J. Reddy, *An Introduction to the Mathematical Theory of Finite Elements*, Pure and Applied Mathematics, John Wiley & Sons, New York, 1974.
[19] S. Piperno, *Simulation Numérique de Phénomènes d'Interaction Fluide-Structure*, Ph.D. thesis, école Nationale des Ponts et Chaussées, France, 1995 (partly in French).
[20] S. Piperno, *Explicit/implicit fluid/structure staggered procedures with a structural predictor and fluid subcycling for* 2D *inviscid aeroelastic simulations*, Internat. J. Numer. Methods Fluids, 25 (1997), pp. 1207–1226.
[21] S. Piperno and C. Farhat, *Partitioned procedures for the transient solution of coupled aeroelastic problems.* II: *Energy transfer analysis and three-dimensional applications*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 3147–3170.
[22] S. Piperno, C. Farhat, and B. Larrouturou, *Partitioned procedures for the transient solution of coupled aeroelastic problems.* I: *Model problem, theory and two-dimensional application*, Comput. Methods Appl. Mech. Engrg., 124 (1995), pp. 79–112.
[23] S. C. Reddy, P. J. Schmid, and D. S. Henningson, *Pseudospectra of the Orr–Sommerfeld operator*, SIAM J. Appl. Math., 53 (1993), pp. 15–47.
[24] I. Robertson, S. Sherwin, and P. Bearman, *Prediction of flutter instability due to cross winds of the second forth road bridge*, in Computational Fluid Dynamics '01: Proceedings of the ECCOMAS CFD Conference 2001, Swansea, Wales, U.K., 2001, on CD-rom.
[25] L. Trefethen, *Pseudospectra of linear operators*, SIAM Rev., 39 (1997), pp. 383–406.
[26] L. Trefethen, A. Trefethen, S. Reddy, and T. Driscoll, *Hydrodynamic stability without eigenvalues*, Science, 261 (1993), pp. 578–584.
[27] T. Wright, *Matlab Pseudospectra GUI*, Oxford University Computing Laboratory. Available at http://www.comlab.ox.ac.uk/pseudospectra/eigtool/.
[28] E. Zeidler, *Applied Functional Analysis: Applications to Mathematical Physics*, Appl. Math. Sci. 108, Springer, Berlin, 1995.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 622–645
© 2005 Society for Industrial and Applied Mathematics

SECOND-ORDER CONE PROGRAMMING METHODS FOR TOTAL VARIATION-BASED IMAGE RESTORATION*

DONALD GOLDFARB† AND WOTAO YIN†

Abstract. In this paper we present optimization algorithms for image restoration based on the total variation (TV) minimization framework of Rudin, Osher, and Fatemi (ROF). Our approach formulates TV minimization as a second-order cone program which is then solved by interior-point algorithms that are efficient both in practice (using nested dissection and domain decomposition) and in theory (i.e., they obtain solutions in polynomial time). In addition to the original ROF minimization model, we show how to apply our approach to other TV models, including ones that are not solvable by PDE-based methods. Numerical results on a varied set of images are presented to illustrate the effectiveness of our approach.

Key words. image denoising, total variation, second-order cone programming, interior-point methods, nested dissection, domain decomposition

AMS subject classifications. 68U10, 65K10, 90C25, 90C51

DOI. 10.1137/040608982

1. Introduction.

Digital images, no matter what their source, usually contain noise. Consequently, a fundamental problem in image processing is the restoration or denoising of such images. Early methods for doing this were based on least squares and had the unfortunate property of either smoothing edges or creating spurious oscillations near edges, i.e., the well-known ringing phenomenon. In [19], Rudin, Osher, and Fatemi (ROF) proposed a method based on minimizing the total variation (TV) of the image. Specifically, they proposed solving the minimization problem

$$\begin{aligned} &\min && \int_\Omega \|\nabla u\| \, \mathrm{d}x \\ &\text{subject to (s.t.)} && u + v = f, \\ &&& \int_\Omega |v|^2 \, \mathrm{d}x \le \sigma^2, \end{aligned} \tag{1.1}$$

where $\|\nabla u\|$ denotes the Euclidean norm of ∇u at each x, Ω is an open (typically rectangular) domain in $\mathbb{R}^2$, $f : \Omega \to \mathbb{R}$ is a given noisy image in $L^2(\Omega)$, u is the computed estimate of the original image, and σ^2 is an estimate of the variance of the noise in the image f. Minimizing the TV of u allows u to have discontinuities; hence edges and important features in the original image are preserved by the ROF approach.

Rather than solving the constrained minimization problem (1.1), ROF and subsequent researchers also considered the unconstrained minimization problem

$$\min \quad \int_\Omega \|\nabla u\| \, \mathrm{d}x + \lambda \int_\Omega |f - u|^2 \, \mathrm{d}x, \tag{1.2}$$

*Received by the editors May 24, 2004; accepted for publication (in revised form) February 4, 2005; published electronically October 31, 2005. This research was supported by NSF grant DMS 01-04282, ONR grant N00014-03-1-0514, and DOE grant GE-FG01-92ER-25126.
http://www.siam.org/journals/sisc/27-2/60898.html
†Department of IEOR, Columbia University, New York, NY 10027 (goldfarb@columbia.edu, wy2002@columbia.edu).

which yields the same solution as (1.1) for a suitable choice of the Lagrange multiplier λ (see, e.g., [6]). To solve (1.2), ROF proposed using an artificial time stepping method to compute the steady state solution u of the parabolic system $\frac{\partial u}{\partial t} = g(u)$ with initial condition $u = f$ and homogeneous Neumann boundary conditions, where

$$g(u) \equiv \frac{1}{2\lambda} \nabla \cdot \left(\frac{\nabla u}{\|\nabla u\|} \right) + f - u. \tag{1.3}$$

Note that $g(u) = 0$ is the Euler–Lagrange equation for (1.2), which is a necessary and sufficient condition for u to be a solution of the convex minimization problem (1.2). Other algorithms that have been proposed for solving problem (1.2) can be found in [5, 7, 9, 12, 13, 17, 23].

We propose here a direct method for solving (1.1) based on the observation that a discretized version of this convex optimization problem can be reformulated as a second-order cone program (SOCP). One can also reformulate a discretized version of the convex unconstrained minimization problem (1.2) as an SOCP. Our approach is to solve the SOCP representation of (1.1) by an interior-point method taking advantage of the specific structure imparted to the SOCP by the image restoration problem and the TV-based denoising method. Of the many methods that have been proposed for solving (1.1) (or (1.2)) or related problems, it is most closely related to the one proposed by Chan, Golub, and Mulet in [9] based on the interior-point algorithm of Andersen et al. [3].

In the next section we provide a formal definition of an SOCP, present the discretized version of problem (1.1) that we will address, and introduce some notation. In section 3 we give an SOCP formulation of the ROF-based image restoration problem. We also present a dual formulation of problem (1.1) that is essentially equivalent to the dual formulation proposed by Chambolle [5] and show how it can be reformulated as an SOCP. In section 4 we discuss implementational aspects that enable interior-point methods to solve these problems efficiently in practice. In particular, we show how nested dissection can be used to reduce storage and computational effort. A detailed analysis is presented in the appendix. In theory the solution is obtained in polynomial time. Section 5 contains our numerical results. It also contains a discussion of how "overlapping domain decomposition" can be used to further reduce storage and run times. Finally, in section 6 we discuss how our approach can be applied to other image processing models. In particular, we show that it can be applied to the model introduced by Meyer in [15] that is not solvable by PDE-based methods.

2. Notation and preliminaries. The images we discuss in this paper are defined as two-dimensional $n \times n$ matrices in $\mathbb{R}^{n^2}$. We restrict our discussion to square domains only for the sake of simplicity. Let f be an observed image, which, in real applications, is a noisy version of an original image u. By introducing the noise term v, f and u have the following relation:

$$f_{i,j} = u_{i,j} + v_{i,j} \quad \text{for } i, j = 1, \ldots, n. \tag{2.1}$$

$f_{i,j}$ and $u_{i,j}$ are the values of the observed image and the original image, respectively, at pixel (i, j). Let ∂^+ be the discrete differential operator defined by

$$\partial^+ u_{i,j} \stackrel{\text{def}}{=} \left((\partial_x^+ u)_{i,j}, (\partial_y^+ u)_{i,j} \right), \tag{2.2}$$

where

$$\begin{aligned} (\partial_x^+ u)_{i,j} &\stackrel{\text{def}}{=} u_{i+1,j} - u_{i,j} \qquad \text{for } i = 1, \ldots, n-1, j = 1, \ldots, n, \\ (\partial_y^+ u)_{i,j} &\stackrel{\text{def}}{=} u_{i,j+1} - u_{i,j} \qquad \text{for } i = 1, \ldots, n, j = 1, \ldots, n-1. \end{aligned} \tag{2.3}$$

In addition, the differentials on the image domain boundary, $(\partial_x^+ u)_{n,j}$ for $j = 1, \ldots, n$ and $(\partial_y^+ u)_{i,n}$ for $i = 1, \ldots, n$, are defined to be zero. Consequently, they do not contribute to the total variation, which in the discrete case is defined by

$$TV(u) \stackrel{\text{def}}{=} \sum_{1 \le i,j \le n} \|\partial^+ u_{i,j}\|, \tag{2.4}$$

where $\|\cdot\|$ denotes the Euclidean norm, i.e., $\|\partial^+ u_{i,j}\| = (((\partial_x^+ u)_{i,j})^2 + ((\partial_y^+ u)_{i,j})^2)^{1/2}$.

Applying our definitions of f, u, v, and $\partial^+ u$ to the ROF restoration model (1.1), we obtain the finite-dimensional minimization problem

$$\begin{array}{ll} \min & \displaystyle\sum_{1 \le i,j \le n} \|\partial^+ u_{i,j}\| \\ \text{s.t.} & u + v = f, \\ & \|v\| \le \sigma. \end{array} \tag{2.5}$$

Problem (2.5) can be reformulated as an SOCP, the [illegible] of which we now define. The vector of variables $\mathbf{x} \in \mathbb{R}^n$ in a standard form SOCP is composed of subvectors $\mathbf{x_i} \in \mathbb{R}^{n_i}$—i.e., $\mathbf{x} \equiv (\mathbf{x_1}; \mathbf{x_2}; \ldots; \mathbf{x_r})$—where $n = n_1 + n_2 + \cdots + n_r$ and each subvector $\mathbf{x_i}$ must lie in an elementary [illegible] of dimension n_i

$$\mathcal{K}^{n_i} \equiv \{\mathbf{x_i} = (x_i^0; \bar{\mathbf{x}}_\mathbf{i}) \in \mathbb{R} \times \mathbb{R}^{n_i - 1} : \|\bar{\mathbf{x}}_\mathbf{i}\| \le x_i^0\}.$$

Definition 2.1. [illegible] [1]

$$\begin{array}{ll} \min & \mathbf{c_1}^\top \mathbf{x_1} + \cdots + \mathbf{c_r}^\top \mathbf{x_r} \\ \text{s.t.} & A_1 \mathbf{x_1} + \cdots + A_r \mathbf{x_r} = \mathbf{b}, \\ & \mathbf{x_i} \in \mathcal{K}^{n_i}, \quad \text{for } i = 1, \ldots, r, \end{array} \tag{2.6}$$

[illegible] $\mathbf{c_i} \in \mathbb{R}^{n_i}$ [illegible] $A_i \in \mathbb{R}^{m \times n_i}$ *for* $i = 1, \ldots, r$ [illegible] $\mathbf{b} \in \mathbb{R}^m$

Since a one-dimensional second-order cone corresponds to a semi-infinite ray, the above standard form SOCP can accommodate nonnegative variables. In fact if all cones $\mathcal{K}_i$ are one-dimensional, then the above SOCP is just a standard form linear program.

If we define $\mathcal{K}$ to be the Cartesian product $\mathcal{K} = \mathcal{K}^{n_1} \times \cdots \times \mathcal{K}^{n_r}$ and the vector $\mathbf{c}$ and the matrix A by $\mathbf{c} = (\mathbf{c_1}; \ldots; \mathbf{c_r})$, and $A = (A_1, \ldots, A_r)$, then the above SOCP can be written compactly as

$$\begin{array}{ll} \min & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} & A\mathbf{x} = \mathbf{b}, \\ & \mathbf{x} \in \mathcal{K} \end{array} \tag{2.7}$$

A crucial aspect of the above SOCP is that the cone $\mathcal{K}$ is pointed, closed, convex, and [illegible]—i.e., the [illegible] $\mathcal{K}^* \equiv \{\mathbf{z} : \mathbf{x}^\top \mathbf{z} \ge 0 \ \forall \mathbf{x} \in \mathcal{K}\} = \mathcal{K}$ as is the cone corresponding to the nonnegative orthant in $\mathbb{R}^n$. This similarity between SOCPs and linear programs is more than superficial; as is the case for linear programs, SOCPs can be solved in polynomial time by interior-point methods. This is the approach we propose in this paper.

3. Second-order cone programming formulation.

3.1. Primal formulation. By introducing $t_{i,j}$ and the relations $((\partial_x^+ u)_{i,j})^2 + ((\partial_y^+ u)_{i,j})^2 \le (t_{i,j})^2$ for each pixel $i,j = 1,\dots,n$, and including equations (2.3) that define $(\partial_x^+ u)_{i,j}$ and $(\partial_y^+ u)_{i,j}$, problem (2.5) can be reformulated as

$$
\begin{aligned}
\min \quad & \sum_{1\le i,j\le n} t_{i,j} \\
\text{s.t.} \quad & u_{i,j} + v_{i,j} = f_{i,j} && \text{for } i,j = 1,\dots,n, \\
& -(\partial_x^+ u)_{i,j} + (u_{i+1,j} - u_{i,j}) = 0 && \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n, \\
& -(\partial_y^+ u)_{i,j} + (u_{i,j+1} - u_{i,j}) = 0 && \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1, \\
& (\partial_x^+ u)_{n,j} = 0 && \text{for } j = 1,\dots,n, \\
& (\partial_y^+ u)_{i,n} = 0 && \text{for } i = 1,\dots,n, \\
& v_0 = \sigma, \\
& (t_{i,j}; (\partial_x^+ u)_{i,j}; (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3 && \text{for } i,j = 1,\dots,n, \\
& (v_0; v) \in \mathcal{K}^{n^2+1},
\end{aligned}
\tag{3.1}
$$

where, for $i,j = 1,\dots,n$, $u_{i,j}$, $(\partial_x^+ u)_{i,j}$, $(\partial_y^+ u)_{i,j}$, $v_{i,j}$, and $t_{i,j}$ are treated as variables and $f_{i,j}$ for $i,j = 1,\dots,n$ and σ as constants.

Although solving for u is our ultimate goal, in the above formulation u can be eliminated and $(\partial_x^+ u)_{n,j} = 0$ for $j = 1,\dots,n$ and $(\partial_y^+ u)_{i,n} = 0$ for $i = 1,\dots,n$ can be removed; hence, (3.1) simplifies to

$$
\begin{aligned}
\min \quad & \sum_{1\le i,j\le n,(i,j)\ne(n,n)} t_{i,j} \\
\text{s.t.} \quad & (\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j} && \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n, \\
& (\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j} && \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1, \\
& v_0 = \sigma, \\
& (t_{i,j}; (\partial_x^+ u)_{i,j}; (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3 && \text{for } i,j = 1,\dots,n-1, \\
& (t_{n,j}; (\partial_y^+ u)_{n,j}) \in \mathcal{K}^2 && \text{for } j = 1,\dots,n-1, \\
& (t_{i,n}; (\partial_x^+ u)_{i,n}) \in \mathcal{K}^2 && \text{for } i = 1,\dots,n-1, \\
& (v_0; v) \in \mathcal{K}^{n^2+1}.
\end{aligned}
\tag{3.2}
$$

After solving the above problem, u can be recovered via $u_{i,j} = f_{i,j} - v_{i,j}$ for $i,j = 1,\dots,n$.

Problem (3.2) is a standard form SOCP. It has $4n^2 - 2n$ variables, $2n(n-1)+1$ equality constraints, $(n-1)^2$ three-dimensional and $2(n-1)$ two-dimensional second-order cone constraints, and one large second-order cone constraint.

3.2. Alternative (dual) formulations. Recalling the definition (2.2) of $\partial^+ u_{i,j}$, let us define $\partial^+ u \equiv \{\partial^+ u_{i,j}\} \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2}$. If we further define $q_{i,j} \equiv (q_{i,j}^1, q_{i,j}^2)$ and

$(\mathrm{div}q) \equiv \{(\mathrm{div}q)_{i,j}\}$, where $(\mathrm{div}q)_{i,j} = (\partial_x^- q^1)_{i,j} + (\partial_y^- q^2)_{i,j}$ and

$$
\begin{aligned}
&(\partial_x^- q^1)_{i,j} \stackrel{\text{def}}{=} q^1_{i,j} - q^1_{i-1,j} && \text{for } i = 2,\dots,n-1, j = 1,\dots,n,\\
&(\partial_x^- q^1)_{1,j} \stackrel{\text{def}}{=} q^1_{1,j} && \text{for } j = 1,\dots,n,\\
&(\partial_x^- q^1)_{n,j} \stackrel{\text{def}}{=} -q^1_{n-1,j} && \text{for } j = 1,\dots,n,\\
&(\partial_y^- q^2)_{i,j} \stackrel{\text{def}}{=} q^2_{i,j} - q^2_{i,j-1} && \text{for } i = 1,\dots,n, j = 2,\dots,n-1,\\
&(\partial_y^- q^2)_{i,1} \stackrel{\text{def}}{=} q^2_{i,1} && \text{for } i = 1,\dots,n,\\
&(\partial_y^- q^2)_{i,n} \stackrel{\text{def}}{=} -q^2_{i,n-1} && \text{for } i = 1,\dots,n-1,
\end{aligned}
\tag{3.3}
$$

then $q \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2}$ and

$$
\langle q, \partial^+ u\rangle = \sum_{i,j} (q^1_{i,j}(\partial_x^+ u)_{i,j} + q^2_{i,j}(\partial_y^+ u)_{i,j}) = \sum_{i,j} -(\mathrm{div}q)_{i,j} u_{i,j} = \langle -\mathrm{div}q, u\rangle. \tag{3.4}
$$

Let z^* denote the optimal value of

$$
\begin{aligned}
\min \quad & \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| \\
\text{s.t.} \quad & \|u - f\|^2 \le \sigma^2.
\end{aligned}
\tag{3.5}
$$

Using the fact that $\|a\| = \max_{\|b\|\le 1} a^\top b$ (an immediate consequence of the Cauchy–Schwarz inequality), if we define the set

$$
S \equiv \{q \in \mathbb{R}^{n^2} \times \mathbb{R}^{n^2} : \|q_{i,j}\| \equiv \|(q^1_{i,j}, q^2_{i,j})\| \le 1 \ \forall\ i,j\},
$$

then

$$
z^* = \min_{\|u-f\|^2\le\sigma^2} \max_{q\in S} \langle q, \partial^+ u\rangle \tag{3.6}
$$

$$
= \min_{\|u-f\|^2\le\sigma^2} \max_{q\in S} \langle -\mathrm{div}q, u\rangle \tag{3.7}
$$

$$
= \max_{q\in S} \min_{\|u-f\|^2\le\sigma^2} \langle -\mathrm{div}q, u\rangle. \tag{3.8}
$$

From the inner minimization, u must satisfy $u = f + \frac{\mathrm{div}q}{\|\mathrm{div}q\|}\sigma$. Hence, substituting this expression for u and letting $p = -q$ yields the following maximization problem:

$$
z^* = \max_{p\in S} \{\langle \mathrm{div}p, f\rangle - \sigma\|\mathrm{div}p\|\}. \tag{3.9}
$$

In [5], Chambolle derived a dual formulation of a discrete formulation of problem (1.2), which is the Lagrangian version of (3.5), and proposed a method for solving this formulation using a PDE approach. A related dual formulation is presented in [21] and [4]. We now show how to derive Chambolle's formulation in a direct manner. Specifically, Chambolle considered

$$
\tilde{z}^* = \min_u \left\{ \frac{\|u-f\|^2}{2\lambda} + \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| \right\}, \tag{3.10}
$$

which, by arguments identical to those used to derive (3.8), is equivalent to

$$\tilde{z}^* = \frac{1}{2\lambda} \max_{q \in S} \left\{ \min_u \{ \|u - f\|^2 + \langle -2\lambda \mathrm{div} q, u \rangle \} \right\}. \tag{3.11}$$

Since $u = f + \lambda \mathrm{div} q$ solves the inner minimization problem, substituting this expression for u yields

$$\tilde{z}^* = \frac{1}{2\lambda} \max_{q \in S} \left\{ \|\lambda \mathrm{div} q\|^2 + \langle -2\lambda \mathrm{div} q, f + \lambda \mathrm{div} q \rangle \right\} \tag{3.12}$$

$$= \frac{1}{2\lambda} \max_{q \in S} \left\{ -\|\lambda \mathrm{div} q + f\|^2 + \|f\|^2 \right\}, \tag{3.13}$$

which, letting $p = -q$, is equivalent to

$$\tilde{z}^* = \frac{1}{2\lambda} \|f\|^2 - \frac{1}{2\lambda} \min_{p \in S} \left\{ \|\lambda \mathrm{div} p - f\|^2 \right\}. \tag{3.14}$$

This minimization problem is identical to problem (8) in [5].

In fact problem (3.9) is equivalent to problem (3.14) for an appropriate choice of σ in terms of λ. For this equivalence to hold, problem (3.5) and its Lagrangian version (3.10) must be equivalent. Hence, for a fixed q, the solutions to the inner minimization problem in (3.8) and (3.11) must be equal; i.e.,

$$u = f + \sigma \frac{\mathrm{div} q}{\|\mathrm{div} q\|} = f + \lambda \mathrm{div} q. \tag{3.15}$$

Since $p = -q$, this gives

$$\sigma = \lambda \|\mathrm{div} q\| = \lambda \|\mathrm{div} p\|. \tag{3.16}$$

Note that adding the term $\|u - f\|^2 / 2\lambda$ to the objective in (3.9) and then using (3.15) and (3.16) to substitute for $u - f$ and σ yields

$$\frac{1}{2\lambda} \max_{p \in S} \left\{ 2\lambda \langle \mathrm{div} p, f \rangle - \lambda^2 \|\mathrm{div} p\|^2 \right\},$$

which is easily seen to be equivalent to (3.14).

Next, we present the SOCP formulation of the problem $\min_{q \in S} \left\{ \|\lambda \mathrm{div} p - f\|^2 \right\}$. Following the definition of S, the discretization of the above problem gives the following SOCP:

$$\begin{array}{lll} \min & w_0 & \\ \text{s.t.} & \lambda(\mathrm{div} q)_{i,j} - w_{i,j} = f_{i,j} & \text{for } i,j = 1, \ldots, n, \\ & (w_0; (w_{i,j})) \in \mathcal{K}^{n^2+1}, & \\ & (q^0_{i,j}; q^1_{i,j}; q^2_{i,j}) \in \mathcal{K}^3 & \text{for } i,j = 1, \ldots, n-1, \\ & (q^0_{n,j}; q^2_{n,j}) \in \mathcal{K}^2 & \text{for } j = 1, \ldots, n-1, \\ & (q^0_{i,n}; q^1_{i,n}) \in \mathcal{K}^2 & \text{for } i = 1, \ldots, n-1, \\ & q^0_{i,j} = 1 & \text{for } i,j = 1, \ldots, n, \end{array} \tag{3.17}$$

in which $(\mathrm{div} q)_{i,j}$ is explicitly given by $(\mathrm{div} q)_{i,j} = (\partial_x^- q^1)_{i,j} + (\partial_y^- q^2)_{i,j}$ and (3.3).

Many other dual formulations are possible. For example, if rather than relaxing the constraint $\|f-u\|^2 \le \sigma^2$, we relax the equivalent one, $\|f-u\| \le \sigma$, we obtain the Lagrangian version of (2.5),

$$\min \quad \sum_{1\le i,j\le n} \|\partial^+ u_{i,j}\| + \lambda \|f-u\|, \tag{3.18}$$

which can be reformulated as the following SOCP:

$$\begin{aligned}
\min \quad & \sum_{1\le i,j\le n} t_{i,j} + \lambda\, v_0 \\
\text{s.t.} \quad & (\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j} && \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n, \\
& (\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j} && \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1, \\
& (t_{i,j}\,;\ (\partial_x^+ u)_{i,j}\,,\ (\partial_y^+ u)_{i,j}) \in \mathcal{K}^3 && \text{for } i,j = 1,\dots,n-1, \\
& (t_{n,j}\,;\ (\partial_y^+ u)_{n,j}) \in \mathcal{K}^2 && \text{for } j = 1,\dots,n, \\
& (t_{i,n}\,;\ (\partial_x^+ u)_{i,n}) \in \mathcal{K}^2 && \text{for } i = 1,\dots,n, \\
& (v_0; v) \in \mathcal{K}^{n^2+1}.
\end{aligned} \tag{3.19}$$

By assigning dual variables $r^1_{i,j}$ and $r^2_{i,j}$ to each of the first two constraints and associating $(s^0_{i,j}; s^1_{i,j}, s^2_{i,j})$ to $(t_{i,j}; (\partial_x^+ u)_{i,j}, (\partial_y^+ u)_{i,j})$ and $(w_0; w)$ to $(v_0; v)$, we obtain the SOCP dual of (3.19):

$$\begin{aligned}
\max \quad & \sum_{1\le i,j\le n, i\ne n} r^1_{i,j}(f_{i+1,j} - f_{i,j}) + \sum_{1\le i,j\le n, j\ne n} r^2_{i,j}(f_{i,j+1} - f_{i,j}) \\
\text{s.t.} \quad & -(\partial_x^- r^1)_{i,j} - (\partial_y^- r^2)_{i,j} + w_{i,j} = 0 && \text{for } i,j = 1,\dots,n, \\
& r^1_{i,j} + s^1_{i,j} = 0 && \text{for } i = 1,\dots,n-1,\ j = 1,\dots,n, \\
& r^2_{i,j} + s^2_{i,j} = 0 && \text{for } i = 1,\dots,n,\ j = 1,\dots,n-1, \\
& s^0_{i,j} = 1 && \text{for } i,j = 1,\dots,n, \\
& w_0 = \lambda, \\
& (s^0_{i,j}\,;\ s^1_{i,j}\,,\ s^2_{i,j}) \in \mathcal{K}^3 && \text{for } i,j = 1,\dots,n-1, \\
& (s^0_{n,j}\,;\ s^2_{n,j}) \in \mathcal{K}^2 && \text{for } j = 1,\dots,n, \\
& (s^0_{i,n}\,;\ s^1_{i,n}) \in \mathcal{K}^2 && \text{for } i = 1,\dots,n, \\
& (w_0\,;\ w) \in \mathcal{K}^{n^2+1}.
\end{aligned} \tag{3.20}$$

By eliminating variables $r^1_{i,j}$, $r^2_{i,j}$, $s^0_{i,j}$, and w_0, this maximization problem can be simplified to the following minimization problem:

$$\begin{aligned}
\min \quad & \sum_{1\le i,j\le n, i\ne n} s^1_{i,j}(f_{i+1,j} - f_{i,j}) + \sum_{1\le i,j\le n, j\ne n} s^2_{i,j}(f_{i,j+1} - f_{i,j}) \\
\text{s.t.} \quad & (\partial_x^- s^1)_{i,j} + (\partial_y^- s^2)_{i,j} + w_{i,j} = 0 && \text{for } i,j = 1,\dots,n, \\
& (1\,;\ s^1_{i,j}\,,\ s^2_{i,j}) \in \mathcal{K}^3 && \text{for } i,j = 1,\dots,n-1, \\
& (1\,;\ s^2_{n,j}) \in \mathcal{K}^2 && \text{for } j = 1,\dots,n, \\
& (1\,;\ s^1_{i,n}) \in \mathcal{K}^2 && \text{for } i = 1,\dots,n, \\
& (\lambda\,;\ w) \in \mathcal{K}^{n^2+1}.
\end{aligned} \tag{3.21}$$

4. Nested dissection. SOCPs can be solved very efficiently in practice and, in theory, in polynomial time by interior-point methods. For example, path-following interior-point methods generate a sequence of interior points (i.e., points in the interior of $\mathcal{K}$) that follow the so-called central path toward an optimal solution. At each iteration, a set of primal-dual equations, which depend on current variable values and a duality gap parameter μ, are solved to yield an improving search direction. Rather than solving this system of equations directly, most implementations of interior-point methods first reduce these equations to a smaller system of linear equations involving a symmetric positive definite matrix that has the form $AFA^\top$. This system of linear equations is solved by computing the Cholesky factorization $LDL^\top$ of $AFA^\top$, where L is a unit lower triangular matrix and D is a positive diagonal matrix. Although A remains fixed from one iteration to the next, F depends on the current primal-dual solution, so $AFA^\top$ must be factorized at every iteration. Even though interior-point methods terminate in a polynomial number of iterations, each iteration can be very time consuming due to the $O(m^3)$ computational complexity of solving the Cholesky factorization of the $m \times m$ matrix $AFA^\top$, or even fail because of the lack of enough memory to store a dense $m \times m$ lower triangular matrix L. In our applications, m is quite large (e.g., in the SOCP (3.2), $m = 2n(n-1)+1$, so that for $n = 1024$, m is roughly 2×10^6).

This difficulty can be overcome by exploiting the sparsity of the matrix A and the structure of F. In all interior-point methods, F is a block diagonal matrix $F \equiv \mathrm{Diag}\{F_1, \ldots, F_r\}$, where each of the diagonal submatrices $F_i \in \mathbb{R}^{n_i \times n_i}$ can be expressed as a sum of a scaled identity matrix and two rank-one terms (or for some methods, three rank-one terms), one with a positive coefficient and the other with a negative coefficient. Hence,

$$AFA^\top = \sum_{i=1}^{r} A_i F_i A_i^\top, \tag{4.1}$$

where $F_i = \gamma_i I + \mathbf{u_i}\mathbf{u_i}^\top - \mathbf{v_i}\mathbf{v_i}^\top$ ($\mathbf{u_i}$ and $\mathbf{v_i}$ are two columns vectors, not the signal and noise defined before).

It is easily verified that for the SOCP (3.2) the terms $A_i F_i A_i^\top$ have the same sparsity structure as $A_i A_i^\top$ for all i corresponding to two- and three-dimensional cones $\mathcal{K}^2$ and $\mathcal{K}^3$. However, for the one large cone $\mathcal{K}^{n^2+1}$, $A_i F_i A_i^\top$ has a totally dense $(n^2+1) \times (n^2+1)$ block, whereas $A_i A_i^\top$ is very sparse. To take advantage of the sparsity of the latter matrix, implementations of interior-point methods compute the Cholesky factorization of only the sparse part of $AFA^\top$—i.e., for large cones, they include only the sparse term $\gamma_i A_i A_i^\top$ in the sum (4.1) when computing this factorization and then update the factorization to account for the dense part using either a [illegible] approach (see [11]) or a version of the Sherman–Morrison–Woodbury updating identity (see [2]). The latter approach is simpler but unfortunately less numerically stable than the former. The interior-point codes that we used in our experiments treated large cones in this fashion; hence in the following discussion we consider only the sparsity structure of $A_i A_i^\top$.

It is well known that sparsity in a symmetric matrix M does not guarantee that the Cholesky factor L will be sparse. For example, Cholesky factorization of the

rather sparse matrix

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 1 & 0 & 0 & 7 & 0 \\ 1 & 0 & 0 & 0 & 43 \end{bmatrix} \tag{4.2}$$

yields $D = I$ and

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 \\ 1 & -1 & -2 & 1 & 0 \\ 1 & -1 & -2 & -6 & 1 \end{bmatrix}. \tag{4.3}$$

However, by reversing the order of the rows and columns of M, we obtain a matrix $\tilde{M}$ whose Cholesky factorization yields $\tilde{D} = \mathrm{Diag}(43, 7, 3, 2, 1)$ and

$$\tilde{L} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1/43 & 1/7 & 1/3 & 1/2 & 1 \end{bmatrix}. \tag{4.4}$$

[illegible] [10] is a scheme for symmetrically permuting the rows and columns of a symmetric matrix M so that the resulting Cholesky factorization takes optimal advantage of certain types of sparsity in M. To describe nested dissection, we let G_M be the [illegible] associated with M. If $M \in \mathbb{R}^{m\times m}$, then G_M has m nodes and has an edge (i, j) if the (i, j)th element of M is nonzero. Nested dissection recursively applies the following [illegible] strategy. Given a graph G_M, find a [illegible] set of nodes N_S, whose removal from G_M divides it into two unconnected components G_U and G_V with corresponding node sets N_U and N_V. If the nodes in N_S in G_M are numbered after those in N_U and N_V, then the matrix M corresponding to this row and column ordering has the form

$$M = \begin{bmatrix} M_{UU} & \mathbf{0} & M_{US} \\ \mathbf{0} & M_{VV} & M_{VS} \\ M_{US}^\top & M_{VS}^\top & M_{SS} \end{bmatrix}, \tag{4.5}$$

and most importantly, the Cholesky factor L of M inherits a similar structure, i.e.,

$$L = \begin{bmatrix} L_{UU} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & L_{VV} & \mathbf{0} \\ L_{SU} & L_{SV} & L_{SS} \end{bmatrix}. \tag{4.6}$$

This strategy is then applied recursively to the subgraphs G_U and G_V.

Let us now consider the application of nested dissection to the matrix $AA^\top$ arising from the SOCP (3.2), which as explained above corresponds to the sparse part of $AFA^\top$. $(AA^\top)_{l,k} \neq 0$ if and only if the lth and kth constraints share decision variables. The equality constraints in our problem are

$$(\partial_x^+ u)_{i,j} + (v_{i+1,j} - v_{i,j}) = f_{i+1,j} - f_{i,j} \quad \text{for } i = 1, \ldots, n-1,\ j = 1, \ldots, n, \tag{4.7}$$

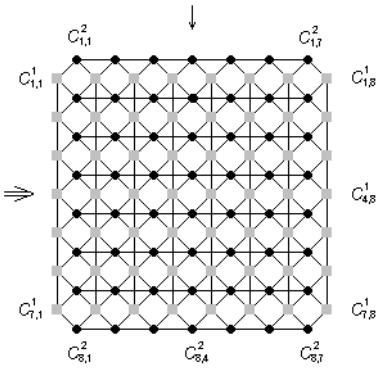


FIG. 1. *Constraint dependency of* (3.2).

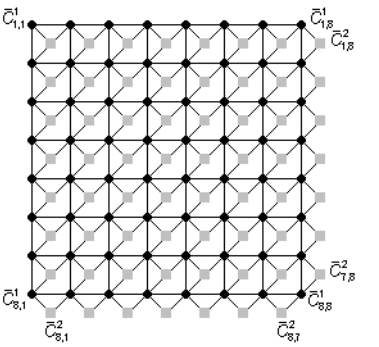


FIG. 2. *Constraint dependency of* (3.17).

$$(\partial_y^+ u)_{i,j} + (v_{i,j+1} - v_{i,j}) = f_{i,j+1} - f_{i,j} \quad \text{for } i = 1, \ldots, n,\ j = 1, \ldots, n-1, \tag{4.8}$$

$$v_0 = \sigma. \tag{4.9}$$

Clearly, the last constraint shares no variable with any other constraint; thus its position in the ordering of the constraints is irrelevant . Let $C^1_{i,j}$ and $C^2_{i,j}$ denote the (i,j)th constraints in (4.7) and (4.8), respectively. For a particular pair (i,j) (i.e., row of A), $C^1_{i,j}$ shares at least one decision variable with the following constraints, given their existence:

$$\begin{array}{llll} C^1_{i-1,j}, & C^2_{i,j-1}, & C^2_{i,j} & (\text{sharing } v_{i,j}), \\ C^1_{i+1,j}, & C^2_{i+1,j-1}, & C^2_{i+1,j} & (\text{sharing } v_{i+1,j}). \end{array}$$

$C^2_{i,j}$ shares at least one decision variable with the following constraints, given their existence:

$$\begin{array}{llll} C^1_{i-1,j}, & C^1_{i,j}, & C^2_{i,j-1} & (\text{sharing } v_{i,j}), \\ C^1_{i-1,j+1}, & C^1_{i,j+1}, & C^2_{i,j+1} & (\text{sharing } v_{i,j+1}). \end{array}$$

The constraint dependency relations (i.e., adjacency graph G_M of $AA^\top$ excluding the row and column corresponding to (4.9), which we shall refer to as M) are depicted in Figure 1 for the case of $n = 8$, in which squares represent $C^1_{i,j}$ and bullets $C^2_{i,j}$. Clearly, removal of the row of squares $C^1_{4,j}$, $j = 1, \ldots, 8$, separates the dependency graph in Figure 1 into two independent parts. If we let $S \equiv \{C^1_{4,j} : j = 1, \ldots, 8\}$ and let U and V denote the sets of constraints corresponding to the upper and the lower independent parts, and order the constraints in U before those in V and those in V before those in S, then M has the form of (4.5).

Furthermore, removal of the column of bullets $C^2_{i,4}$, $i = 1, \ldots, 4$, separates U into two independent parts—U_L and U_R—and removal of the column of bullets $C^2_{i,4}$, $i = 5, \ldots, 8$, separates V into two independent parts—V_L and V_R. As a result, M_{UU} and M_{VV} can be ordered so that each has the form (4.5). This process is then repeated recursively until every node has been assigned to some separator set (i.e., every row and column of M has been numbered). In the appendix, we analyze the storage cost and the number of multiplications for factorizing $M = AA^T$ corresponding to the

equality constraints (4.7) and (4.8) after ordering this matrix by nested dissection. These costs are $(31/4)n^2 \log_2 n + O(n^2)$ and $943n^3/84 + O(n^2 \log_2 n)$ for an $n \times n$ image, respectively.

Finally, we depict in Figure 2 the dependency graph of the dual SOCP formulation (3.17). In Figure 2 the (i,j)th bullet $\bar{C}^1_{i,j}$ represents the constraint $(\mathrm{div} q)_{i,j} - w_{i,j} = f_{i,j}$, and the (i,j)th square $\bar{C}^2_{i,j}$ represents the constraint $q^0_{i,j} = 1$. Since constraints $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$ contain either $q^1_{i,j}$ or $q^2_{i,j}$ or both of them, and the 3×3 fully dense block F_k corresponding to cone $(q^0_{i,j}; q^1_{i,j}; q^2_{i,j})$ causes the 3 corresponding columns in A to merge, constraint $\bar{C}^2_{i,j}$ in AFA^T essentially shares variables with constraints $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$; hence, in Figure 2, $\bar{C}^2_{i,j}$ is connected to $\bar{C}^1_{i,j}$, $\bar{C}^1_{i+1,j}$, and $\bar{C}^1_{i,j+1}$. The connections between bullets $\bar{C}^1_{i,j}$'s are due to the sharing of $q^1_{i,j}$ or $q^2_{i,j}$ for some i,j.

5. Numerical results. In this section, we present numerical results that illustrate the effectiveness of our SOCP approach and compare it to the PDE approach. We then discuss the use of domain decomposition to reduce the storage and computational effort needed by the SOCP approach.

5.1. SOCP/PDE comparison. We used the commercial optimization package Mosek as our SOCP solver. Mosek is designed to solve a variety of large-scale optimization problems, including SOCPs. Before solving a large-scale problem, Mosek uses a presolver to remove redundant constraints and variables and to reorder constraints. The Mosek presolver reorders the coefficient matrix A using a combination of a graph partitioning–based method and an approximate minimum local-fill-in ordering method to increase the sparsity of the Cholesky factors of $AFA^\top$. This general-purpose reordering produced results for our problem that were very good, but not as good as nested dissection, which further reduced the time per iteration and the total time by approximately 30 percent in our tests on a 512×512 image. Therefore, we turned off Mosek's reordering heuristic and replaced it by our own nested dissection reordering code in our tests.

Existing PDE approaches require the ROF model to be represented in the relaxed form (1.2) and the existence of the Euler–Lagrange equation $g(u) = 0$, where $g(u)$ is given by (1.3). To handle the case of $\|\nabla u\| = 0$, most PDE approaches add a small perturbation $\epsilon > 0$ to $\|\nabla u\|$ in (1.3). The PDE method that we used in our numerical tests is an artificial time stepping method which computes the steady state solution u of the parabolic system

$$\frac{\partial u}{\partial t} = \frac{1}{2\lambda} \nabla \cdot \left(\frac{\nabla u}{\|\nabla u\| + \epsilon} \right) + f - u. \tag{5.1}$$

To compare the two approaches for each noisy image, we first solved the SOCP formulation (3.2) of (1.1) and calculated λ from the optimal dual solution using the following theorem.

THEOREM 5.1. [illegible] (1.1) [illegible] (1.2) [illegible] u [illegible] $2\lambda\sigma = z_0$ [illegible] z_0 [illegible] $(z_0, z) \in \mathcal{K}^{n^2+1}$ (z_0, z) [illegible] (v_0, v) [illegible] (3.2) [illegible] (1.1) [illegible] $\int_\Omega |v|^2 \, \mathrm{d}x \le \sigma^2$ [illegible] $(v_0, v) \in \mathcal{K}^{n^2+1}$ [illegible] $v_0 = \sigma$

We then used this Lagrange multiplier λ in the PDE method. In practice, it is much easier to estimate the noise level of an image than it is to estimate an appropriate

TABLE 1
Numerical comparison between SOCP (MATLAB+Mosek) and PDE (C++) approaches.

Name	Size $n \times n$	Fig.	SOCP result u_1 vs. PDE result u_2	
			$\max_{i,j} \lvert u_1 - u_2 \rvert$	$\lVert u_1 - u_2 \rVert_{L^1}/n^2$
Books	64×64	3–5	0.58	0.04
Dotplus	100×100	6–8	0.97	0.10
Scale	252×252	9–11	0.74	0.02
Barbara	256×256	12–14	0.90	0.05
Barbara	350×350	15–17	0.95	0.06
Barbara	512×512	24–26	0.96	0.06

λ. Experiments show that a σ that is a little smaller than the true one (i.e., a slight amount of undersmoothing) yields the best results.

In our tests, we called Mosek from MATLAB, while we called the PDE-based code (an improved version of a C++ code provided to us by Stanley Osher and Jinjun Xu of UCLA) directly. The running times given in Table 2 for the problems whose sizes are given in Table 1 are the total times in seconds needed to solve the problem on a Sun E450 with four 300MHz SuperSparc II processors and 4GB memory. Since both codes are single-threaded, they run only on one processor and use 1GB memory at a time. The SOCP run times do not include the time taken by our nested dissection code to reorder constraints and the time taken by Mosek for presolving, since the process of constraint reordering and presolving remains the same for all images of the same size.

During their execution, both approaches record the objective values of (1.2):

$$z(t) = \int_\Omega \|\nabla u_t\| \, \mathrm{d}x + \lambda \int_\Omega |f - u_t|^2 \, \mathrm{d}x. \tag{5.2}$$

The objective values obtained by the PDE approach decrease with time t over the long run but fluctuate and never reach as low a value as that obtained by the SOCP approach. Given a nearly optimal objective value, the PDE approach may have to iterate further for a large number of steps to obtain a lower objective value. For these reasons, we used the following stopping rules for the PDE approach:

- initially, let $z_{\min} = \infty$ and $t_{\min} = 0$, where $z_{\min}$ and $t_{\min}$ denote the minimal objective value and its time;
- at time t, update $z_{\min} := z(t)$ and $t_{\min} := t$ if $z_{\min} < z(t) - \lambda$;
- to denoise an image of size $n \times n$, stop at time t once $t - t_{\min} \geq \max\{0.15 t_{\min}, n\}$.

These stopping criteria ensure that the PDE method spends a reasonable amount of time searching for a possibly lower objective value but does not waste too much time trying to obtain an insignificant improvement.

In all of our tests starting from the same noisy image, the two approaches produced images that were visually indistinguishable from one another. Table 1 shows that in all our tests the gray values (u_1 and u_2) produced by the two approaches differ by at most 0.97 over all pixels with an average difference of less than 0.10. Therefore, we present in Figures 3–17, in addition to the original and noisy images, only the SOCP images. Similar results were obtained on the other images.

From Table 2, one can see that the SOCP approach requires a smaller number of iterations than the PDE approach while every iteration takes more time. Note that as the size of the image grows, the number of iterations required by the SOCP approach remains relatively constant, while that number grows for the PDE approach. In spite

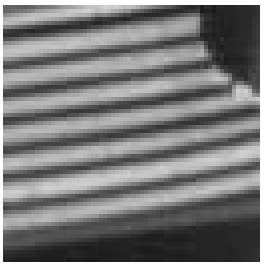


FIG. 3. *Original books* 64×64.

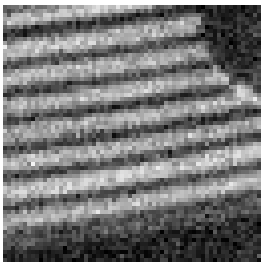


FIG. 4. *Noisy books* ($\sigma = 20$).

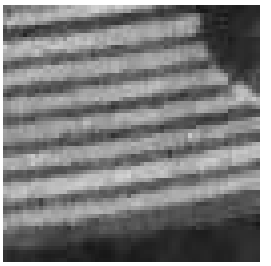


FIG. 5. *Denoised image by the SOCP method.*

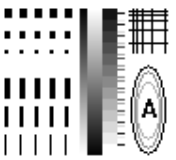


FIG. 6. *Original dotplus* 100×100.

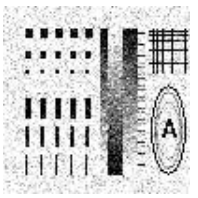


FIG. 7. *Noisy dotplus* ($\sigma = 40$).

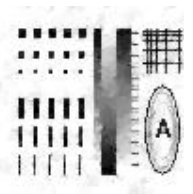


FIG. 8. *Denoised image by the SOCP method.*

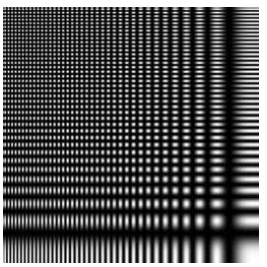


FIG. 9. *Original scale* 252×252.

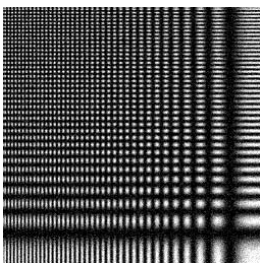


FIG. 10. *Noisy scale* ($\sigma = 20$).

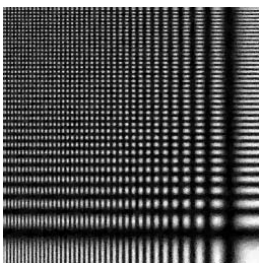


FIG. 11. *Denoised image by the SOCP method.*



FIG. 12. *Original Barbara* 256×256.



FIG. 13. *Noisy Barbara* ($\sigma = 20$).



FIG. 14. *Denoised image by the SOCP method.*

of this, the SOCP approach took roughly between one and two times the total time that the PDE approach took. This is because, in theory, the PDE method takes $O(n^2)$ CPU time per iteration for an $n \times n$ image, while the SOCP method (implemented



FIG. 15. *Original Barbara* 350×350.



FIG. 16. *Noisy Barbara* ($\sigma = 20$).



FIG. 17. *Denoised image by the SOCP method.*

TABLE 2
Numerical comparison between SOCP (MATLAB+Mosek) and PDE (C++) approaches (cont.).

Name	λ	SOCP			PDE ($\Delta t = 0.1$)		
		Total time	Itr.	Obj. value	Total time	Itr.	Obj. value
Books	0.0374718930	4.42	15	128927	3.22	950	128976
Dotplus	0.0180513895	13.68	16	592183	14.90	1875	592438
Scale	0.0337088230	157.86	17	3706529	81.21	1479	3706980
Barbara	0.0374657886	130.13	16	1691196	61.73	1166	1692180
Barbara	0.0374969349	283.56	16	2905191	151.71	1464	2907500
Barbara	0.0375681804	806.70	16	5895359	456.25	1900	5900690

as described in section 4) takes $O(n^3)$ CPU time per iteration. In our numerical tests the SOCP approach took between $O(n^2 \log n)$ and $O(n^3)$ time per iteration due to the effect of memory references, which are slower than floating-point operations.

We also measured the running times on a PC with a Pentium IV 1.7GHz CPU and 512MB memory. We used the MS Windows versions of MATLAB, Mosek, and GUN g++ compiler. On this newer computer, the SOCP code ran 2–2.5 times faster and the PDE code ran about 4 times faster solving the same set of problems. Due to the PC's smaller amount of memory and cache, the SOCP approach, which performs more memory fetches, got a smaller speedup than the PDE approach, exhibiting a

running time somewhere between the $n \times n$-image storage cost of $O(n^2)$ and the nested dissection Cholesky factorization storage cost of $O(n^2 \log n)$.

5.2. Domain decomposition. Even with nested dissection the SOCP approach requires $O(n^2 \log n)$ memory. Hence, to denoise large (or three-dimensional) images, we consider here a [illegible] approach. Let $\tilde{n} \approx n/k$, where k is a positive integer and $\tilde{n} > n/k$. By subdividing an $n \times n$ image into k^2 $\tilde{n} \times \tilde{n}$ subimages with some overlap and applying the SOCP method to each of the subimages, we can reduce the storage needed by a factor of approximately $1/k^2$. Since, in applying the SOCP method to each subimage, the number of multiplications per iteration is reduced by a factor of approximately $1/k^3$ compared with applying the SOCP method to the entire image, and there are k^2 subimages, the total CPU time should also be reduced by a factor of approximately $1/k$. After obtaining the denoised subimages, we discarded the values of u for the rows (columns) of pixels in that half of any region of overlap closest to the boundary of the subimage. For example, Figure 20 illustrates a 2×2 domain decomposition with a 4-pixel overlap between subimages. After obtaining the denoised 65×65 image u for the top left block, we discarded the u values for the rightmost two columns and bottommost two rows of pixels. These values have indices (i, j) with $i = 64, 65$ and $j = 1, \ldots, 65$, and $j = 64, 65$ and $i = 1, \ldots, 65$. An analogous statement applies to the overlapped pixels of the other 3 denoised subimages.

Due to the different graphical characteristics of the subimages, including signal-to-noise ratio, contrast, and homogeneity, applying the SOCP method to all subimages using the same σ does not usually give the same output as applying the SOCP method to the entire image. In some cases the results obtained by using the former are better, while in other cases they are worse. The following numerical example illustrates this. Figure 18 is the noisy image obtained by selecting a rectangular area of the noisy image in Figure 13 and removing the outside portion of the image. Figure 19 is the result of the ROF method (3.2) separately applied to 4 equal subimages of size 65×65 with the same $\sigma = 18 \times 65^2$. In the numerical tests, we chose the size of all subimages so that all adjacent image pairs had overlapping regions that were 4 pixels wide. Figure 20 illustrates the 2×2 domain decomposition applied to the image in Figure 18. Figure 21 is the result of ROF applied to the entire image with $\sigma = 18 \times 65^2$. In Figure 21, Barbara's left eye is clearer than in Figure 19. This shows that the subimage at the bottom right is oversmoothed. After reducing $\sigma_{2,2}$ from 18×65^2 to 16×65^2 for this oversmoothed subimage, the ROF method yields Figure 22, in which Barbara's left eye is much clearer. Figure 23 gives the ten-times amplified differences between Figures 21 and 22. The differences in the regions of the four subimages vary, but we cannot see any big differences by comparing Figures 21 and 22. However, as we shall now demonstrate, we can obtain even closer results.

In [20], Strong and Chan give exact analytical solutions to the relaxed TV denoising problem (1.2) applied to one-, two-, and three-dimensional piecewise constant images and smooth radially symmetric ones. Their results establish an exact relationship between the Lagrange multiplier λ and the effect of TV regularization (1.2) on an image feature and show its localness property (i.e., in a piecewise constant or smooth radially symmetric region the solution of (1.2) depends only on λ and whether or not the region is on the image boundary, and is independent to the features of other regions). This can be extended approximately to complicated noisy images since they are composed of many tiny piecewise constant or smooth radially symmetric regions. Therefore, we can expect solving (1.2) with the same λ (or σ's for all subimages so that $z_0/(2\sigma) = \lambda$) as used for the entire image to give close results. Since TV



FIG. 18. *Noisy image* ($\sigma = 20 \times 65^2$).



FIG. 19. 2×2 *partitioned result* ($\sigma_{1,1} = \sigma_{1,2} = \sigma_{2,1} = \sigma_{2,2} = 18 \times 65^2$).

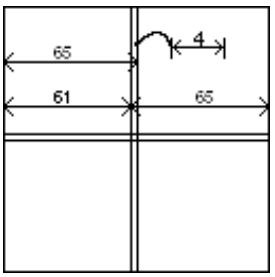


FIG. 20. 2×2 *domain decomposition applied to image of size* 126×126.



FIG. 21. *Nonpartition result* ($\sigma = 18 \times 65^2$).



FIG. 22. 2×2 *partitioned result* ($\sigma_{1,1} = \sigma_{1,2} = \sigma_{2,1} = 18 \times 65^2, \sigma_{2,2} = 16 \times 65^2$).



FIG. 23. *Ten-times amplified difference between Figures* 21 *and* 22.

regulation treats regions on boundaries differently, domain decomposition may give different results near subimage boundaries. Sharp edges are different from sharp noise by the surrounding geometry, e.g., two sides of sharp edges have very different brightness. Therefore, if a sharp edge is very close to the image boundaries, the difference between edges and noise is weaker. Moreover, every pixel in an image may affect the denoising results of the whole image, and their dependency is ambiguous. This effect, however, is limited to strips a few pixels wide when noise is present. Hence, in our tests, we discarded redundant information in these boundary strips of the subimages.

In our first test of the above approach we applied a 4×4 domain decomposition to a 512×512 image. The results are depicted in Figures 24–26. All subimages were of size 137×137, giving overlapping regions between adjacent subimages that were 12 pixels wide. The image values for the 6 rows (columns) of pixels in the overlapping regions closest to the boundaries of each subimage were discarded. The σ's for the full image and all of the subimages were chosen so that $z_0/(2\sigma) = 0.0375$. Table 3 gives the numerical results. The results obtained by the two approaches by more than 1 on the 256 gray-level scale was 0.11% of the total 512×512 pixels. The average difference per pixel was 0.07 and the maximum difference was $+0.97$ and -1.15. The running time 1280 seconds for denoising the entire image was estimated based on the $O(n^3)$ theoretical CPU time. The difference was more than 1 on the 256 gray-level scale for only 0.0008% of the pixels. The average difference per pixel was 0.04 and the maximum difference was $+0.75$ and -1.31. Clearly, SOCP with domain decomposition gave nearly identical results to SOCP applied to the full image.



FIG. 24. *Original* 512×512 *image.*



FIG. 25. *Noisy* 512×512 *image* ($\sigma = 20$).



FIG. 26. 4×4 *domain decomposition applied to Figure* 25.

TABLE 3

4×4 *domain decomposition applied to Figure* 24. *Blk. = block (subimage) index; itr. = number of SOCP iterations for denoising subimage; time = SOCP running time in seconds for denoising subimage.*

Blk.	Itr.	Time	Blk.	Itr.	Time	Blk.	Itr.	Time	Blk.	Itr.	Time
(1,1)	19	33.03	(1,2)	20	34.84	(1,3)	18	32.01	(1,4)	17	29.95
(2,1)	18	31.97	(2,2)	18	31.74	(2,3)	17	29.73	(2,4)	17	30.25
(3,1)	18	31.34	(3,2)	17	30.61	(3,3)	20	35.70	(3,4)	18	31.18
(4,1)	17	30.10	(4,2)	17	31.06	(4,3)	18	32.03	(4,4)	18	32.72
Sum of the total time (seconds) for denoising all subimages (Figure 26):											508.26
Total time for denoising the entire image:											806.70

In this example the total computation time for the domain decomposition approach was about 37 percent less than the time required by the SOCP method applied to the full image. Since $16 \cdot 137^3/512^3 \approx 0.33$, one would expect something close to a 67 percent reduction in running time. The reason this was not the case is explained by our remark in section 5.1 about execution time and memory references when cache is limited and slow.

We also applied an 8×8 domain decomposition to the noisy 512×512 image in

TABLE 4
8×8 domain decomposition applied to Figure 24. Time (itr.) = SOCP running time in seconds (iterations) for denoising subimage

Block index		Column 1	2	3	4	5	6	7	8
	1	6.86(16)	6.56(16)	8.37(19)	6.24(15)	5.80(14)	5.32(13)	5.67(14)	6.12(14)
	2	7.07(17)	6.90(16)	6.51(16)	5.70(14)	6.04(14)	6.02(14)	6.38(15)	5.83(14)
R	3	5.65(14)	6.64(16)	6.90(16)	7.53(17)	7.02(16)	5.86(14)	5.77(14)	6.02(15)
o	4	6.76(16)	7.17(17)	7.21(17)	5.79(14)	5.88(13)	5.84(14)	5.38(13)	6.29(15)
w	5	6.19(15)	7.19(17)	6.63(16)	6.17(14)	5.93(14)	5.75(14)	6.74(16)	6.97(17)
	6	5.99(14)	6.75(16)	6.88(17)	7.06(16)	6.80(15)	5.73(14)	6.50(15)	6.21(15)
	7	7.06(17)	7.22(17)	7.16(16)	6.59(15)	6.35(15)	6.35(15)	6.71(16)	6.14(15)
	8	7.04(17)	7.58(17)	7.10(17)	7.83(19)	7.25(17)	6.39(15)	7.13(17)	7.57(18)
Sum of the total time (seconds) for denoising all subimages:									418.06

Figure 25 after extending it to one with a size of 516×516 so that all subimages were of the same size (75×75) and the overlapping regions remained 12 pixels wide. As in the previous test, the σ's were chosen so that $z_0/(2\sigma) = 0.0375$ for all subimages. The numerical results given in Table 4 show that the reduction in the total computation time was further improved from 37 percent to 52 percent. Moreover, the total run time of 418.06 seconds was less than the 456.25-second run time of the PDE approach. The difference between the results obtained by applying the SOCP method with domain decomposition and applying it to the entire image was more than 1 on the 256 gray-level scale for 0.0015% of the pixels. The average difference per pixel increased to 0.08 and the maximum difference was +1.47 and −1.06. This difference was still not perceivable to the naked eye.

6. SOCP formulations of other restoration models. Meyer [15] introduced the following image restoration model based on TV and the space of oscillating functions:

$$\min \int_\Omega \|\nabla u\| \mathrm{d}x + \lambda \|v\|_* \quad \text{s.t. } f = u + v, \tag{6.1}$$

where $\|v\|_*$ is defined as the infimum of $\|\sqrt{g_1^2(x,y) + g_2^2(x,y)}\|_{L^\infty}$ over all $g_1, g_2 \in L^\infty(\mathbb{R}^2)$ satisfying $v = \partial_x g_1 + \partial_y g_2$. To handle $\|\sqrt{g_1^2(i,j) + g_2^2(i,j)}\|_{L^\infty}$ in the discretized version of (6.1), we introduce a three-dimensional quadratic cone

$$(g_0(i,j); g_1(i,j), g_2(i,j)) \in \mathcal{K}^3$$

for each i, j and let $w \geq g_0(i,j)$ for all i, j. The infimum of $\|\sqrt{g_1^2(i,j) + g_2^2(i,j)}\|_{L^\infty}$ is then equal to the minimum of w. Thus we can solve a discretized problem of (6.1) as an SOCP. It is important to note that because Meyer's model (6.1) is not amenable to the PDE approach, it has been primarily of theoretical interest. Thus, the SOCP approach can be applied to image restoration models that cannot be solved by a PDE approach.

In [22], Vese and Osher proposed an approximate version of Meyer's model and showed that their PDE-based method can decompose a given image f into u and v, where u and v represent cartoon and texture, respectively. Their method solves the minimization problem

$$\min \int_\Omega \|\nabla u\| \mathrm{d}x + \lambda \int |f - u - \partial_x g_1 - \partial_y g_2| + \mu \left[\int (\sqrt{g_1^2 + g_2^2})^p \right]^{1/p}, \tag{6.2}$$

where λ and μ are tuning parameters and $p \geq 1$. Inequalities with p-norm for p being rational can be represented as a system of second-order conic inequalities (refer to [1]). Therefore, (6.2), after discretization, can be formulated as an SOCP and solved by interior-point methods. In a forthcoming paper [24], we solve Meyer's model as an SOCP and compare the solutions with those obtained by the Vese–Osher approximation.

In [14], Lysaker, Lundervold, and Tai introduced a method for image restoration based on solving the following two-dimensional problem:

$$\begin{aligned} \min \quad & \int_\Omega |u_{x_1x_1}| + |u_{x_2x_2}| \, \mathrm{d}x \\ \text{s.t.} \quad & u + v = f, \\ & \int_\Omega |v|^2 \, \mathrm{d}x_1 \, \mathrm{d}x_2 \leq \sigma^2. \end{aligned} \tag{6.3}$$

The discretization of the second derivative of $u = \{u_{i,j}\}$ is linear in the $u_{i,j}$'s. To handle absolute values in minimization problems, say $\min |z|$, one can introduce an extra variable and transform $\min |z|$ into equivalent problems:

$$\min \quad |z| \tag{6.4}$$

$$\Longleftrightarrow \min \quad t \quad \text{s.t.} \quad z \leq t, \ -z \leq t \tag{6.5}$$

$$(s \stackrel{\text{def}}{=} t + z) \Longleftrightarrow \min(s - z) \ \text{s.t.} \quad 2z \leq s, \ s \geq 0. \tag{6.6}$$

Both (6.5) and (6.6) have a linear objective subject to linear constraints. In some cases, (6.6) is preferred since $s \geq 0$ is easier to handle. Therefore, expressing $\int_\Omega |v|^2 \, \mathrm{d}x \leq \sigma^2$ as a second-order conic constraint as before, it is clear that problem (6.3) can be formulated as an SOCP. This also works if the objective function is replaced by $\int_\Omega |u_{x_1x_1}| + 2|u_{x_1x_2}| + |u_{x_2x_2}| \, \mathrm{d}x$.

A variant of (1.1) was proposed by Osher, Solé, and Vese (OSV) in [16]. The model assumes that $f - u = \mathrm{div} g$, where $g = (g_1, g_2)$, and there exists a unique decomposition $g = \nabla P + Q$, where Q satisfies $\mathrm{div} Q = 0$. Therefore, $f - u = \mathrm{div} g = \Delta P$ or $\Delta^{-1}(f - u) = P$. The OSV model is

$$\begin{aligned} \min \quad & \int_\Omega \|\nabla u\| \, \mathrm{d}x \\ \text{s.t.} \quad & \int_\Omega \|\nabla(\Delta^{-1}(f - u))\| \, \mathrm{d}x \leq \sigma^2, \end{aligned} \tag{6.7}$$

or equivalently

$$\begin{aligned} \min \quad & \int_\Omega \|\nabla u\| \, \mathrm{d}x \\ \text{s.t.} \quad & \int_\Omega \|\nabla P\| \, \mathrm{d}x \leq \sigma^2, \\ & f - u = \Delta P. \end{aligned} \tag{6.8}$$

Clearly, by using a finite difference form of the Laplace operator and expressing the objective function and the constraint $\int_\Omega \|\nabla P\| \, \mathrm{d}x \leq \sigma^2$ as we have previously described, we can reformulate (6.8) as an SOCP.

In [8], the authors analyzed the ROF model with an L^1 fidelity term $\int_\Omega |v| \le \sigma$. Clearly, by discretizing $\int_\Omega |v|$ as the sum $\sum_{i,j} |v_{i,j}|$ and handling the absolute values $|v_{i,j}|$ using (6.5) or (6.6), this problem can be formulated as an SOCP.

It is not difficult to see that various TV-based deblurring models can also be formulated as SOCPs.

Appendix. In this appendix, we present a detailed analysis of Cholesky factorization of a matrix M whose rows and columns have been symmetrically reordered by applying nested dissection to the adjacency graph of M, assuming that the latter is an $n \times n$ grid graph of the type shown in Figure 1. We follow an approach similar to the one given in [18]. First, let us introduce some graph notation.

A graph G is defined as the pair (V_G, E_G), where V_G is the set of nodes and E_G is the set of edges of G. In Figure 27, each square or bullet is a node, and each line segment connecting a pair of nodes is an edge. Figure 27 has 8 rows and 7 columns corresponding to the C^2 constraints, and 7 rows and 8 columns corresponding to the C^1 constraints. For convenience, we number node $C^k_{i,j}$ by $(i,j)^k$. In this way, $C^1_{3,1}$ corresponds to node $(3,1)^1$ and $C^2_{8,1}$ to node $(8,1)^2$. Due to the boundary conditions, the row of nodes $(8,\cdot)^1$ and the column of nodes $(\cdot,8)^2$, corresponding to $(\partial^+_x u)_{8,j} = 0$ and $(\partial^+_y u)_{i,8} = 0$ for $1 \le i,j \le 8$, do not appear in Figure 27. Let us denote by $G_{2n,2n}$ a graph of the form of Figure 27 with nodes $(i,j)^k$ for $k = 1,2$ and $i,j = 1,\dots,n-1$, $(i,n)^1$ for $i = 1,\dots,n$, and $(n,j)^2$ for $j = 1,\dots,n$. For example, Figure 27 depicts a $G_{16,16}$ mesh.

Given the graph $G_{2n,2n}$, we define three extended graphs $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$ as follows:

$$A_{2n,2n} \stackrel{\text{def}}{=} (V_A, E_A), \quad \text{where} V_A = V_G \bigcup L_{left} \bigcup L_{bottom}, \tag{A.1}$$

$$B_{2n,2n} \stackrel{\text{def}}{=} (V_B, E_B), \quad \text{where} V_B = V_G \bigcup L_{left} \bigcup L_{bottom} \bigcup L_{right}, \tag{A.2}$$

$$C_{2n,2n} \stackrel{\text{def}}{=} (V_C, E_C), \quad \text{where} V_C = V_G \bigcup L_{top} \bigcup L_{left} \bigcup L_{bottom} \bigcup L_{right}, \tag{A.3}$$

where

$$L_{left} = \{(i,0)^2 \mid i = 1,2,\dots,n\}, \tag{A.4}$$

$$L_{right} = \{(i,n)^2 \mid i = 1,2,\dots,n\}, \tag{A.5}$$

$$L_{top} = \{(0,j)^1 \mid j = 1,2,\dots,n\}, \tag{A.6}$$

$$L_{bottom} = \{(n,j)^1 \mid j = 1,2,\dots,n\}, \tag{A.7}$$

and E_A, E_B, and E_C are the sets of edges that connect V_A, V_B, and V_C by a grid of the type shown in Figure 27. For example, for $n = 8$, $A_{16,16}$, $B_{16,16}$, and $C_{16,16}$ are depicted in Figures 28, 29, and 30, in which extended boundaries are distinguished by being enclosed in ellipses.

The reason that we need these extended graphs in our analysis can be illustrated as follows. As we described in section 4, the graph $G_{16,16}$ can be separated into four subgraphs, each of which is a $G_{8,8}$, by separators $(4,j)^1$, $j = 1,\dots,8$, followed by $(i,4)^2$, $i = 5,\dots,8$, and $(i,4)^2$, $i = 1,\dots,4$. The order of the graph separators is just the reverse of the order assigned to the rows and columns in the matrix. It is well known that, in Cholesky factorization (CF), eliminating a node of degree d (i.e., the number of nodes directly connected to it) costs $K_m = d(d+3)/2$ multiplies and requires $K_s = d+1$ storage. In the next stage the case is different from that of $G_{8,8}$

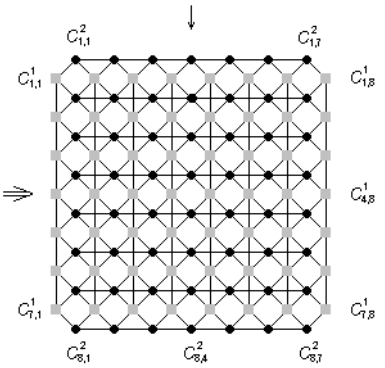


FIG. 27. $G_{16,16}$.

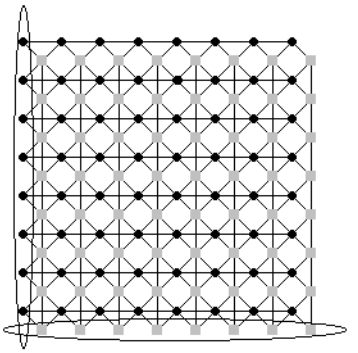


FIG. 28. $A_{16,16}$.

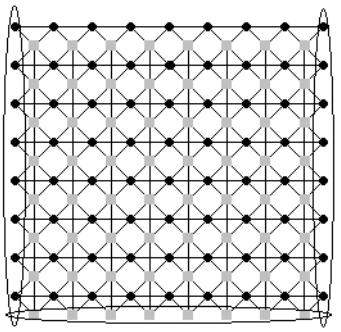


FIG. 29. $B_{16,16}$.

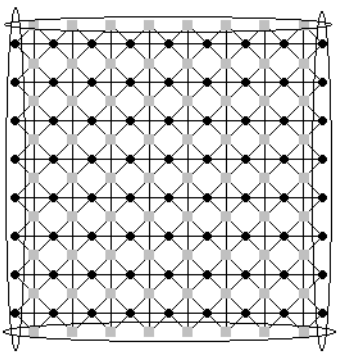


FIG. 30. $C_{16,16}$.

since, for example, the subgraph $G_{8,8}$ in the upper right corner of $G_{16,16}$ is connected to the nodes $(4,j)^1$, $j = 5,\ldots,8$, and $(i,4)^2$, $i = 1,\ldots,4$, which are the bottom and left extended boundaries of $G_{8,8}$. Therefore, we need to consider separating $A_{8,8}$ instead of $G_{8,8}$. Since applying any rotation or reflection to $A_{8,8}$ does not change the multiplication and storage costs of CF, separating any of the other three subgraphs of $G_{16,16}$ is essentially the same as separating $A_{8,8}$. Similar remarks apply to any rotation or reflection of $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$.

Let $\mu(\cdot)$ denote a function of a graph whose value gives the complexity of applying CF to a matrix corresponding to that graph. Specifically, we use $\mu_m(\cdot)$ and $\mu_s(\cdot)$ to present the multiplication cost and the storage cost, respectively. In the divide and conquer approach, the complexity of applying CF to $G_{2n,2n}$ can be calculated by the recursion

$$\mu(G_{2n,2n}) = 4\,\mu(A_{n,n}) + \mu(S^G_{2n,2n}), \tag{A.8}$$

where $S^G_{2n,2n}$ is defined as the separator of $G_{2n,2n}$.

Following similar arguments, one can divide each of $A_{2n,2n}$, $B_{2n,2n}$, and $C_{2n,2n}$ into 4 subgraphs and derive the following recursions:

$$\mu(A_{2n,2n}) = \mu(A_{n,n}) + 2\,\mu(B_{n,n}) + \mu(C_{n,n}) + \mu(S^A_{2n,2n}), \tag{A.9}$$

$$\mu(B_{2n,2n}) = 2\,\mu(B_{n,n}) + 2\,\mu(C_{n,n}) + \mu(S^B_{2n,2n}), \tag{A.10}$$

$$\mu(C_{2n,2n}) = 4\,\mu(C_{n,n}) + \mu(S^C_{2n,2n}). \tag{A.11}$$

The next step is to calculate $\mu(S^G_{2n,2n})$, $\mu(S^A_{2n,2n})$, $\mu(S^B_{2n,2n})$, and $\mu(S^C_{2n,2n})$. Take $\mu(S^G_{16,16})$, for example. Recall that we eliminate $(i,4)^2$, $i = 1,\ldots,4$, followed by $(i,4)^2$, $i = 5,\ldots,8$, and $(4,j)^1$, $j = 1,\ldots,8$. At the time we eliminate $(1,4)^2$, all nodes except for those in $S^G_{2n,2n}$ are eliminated. Therefore, $(1,4)^2$ is connected to $(i,4)^2$, $i = 2,\ldots,4$, and $(4,j)^1$, $j = 1,\ldots,8$. Therefore, $d_{(1,4)^2} = 3 \times (n/2) - 1$, where $n = 8$. Next, $(2,4)^2$ is eliminated with $d_{(2,4)^2} = 3 \times (8/2) - 2$. When we eliminate $(4,4)^2$, it has degree of $3 \times (8/2) - 4 = 8$ since it is connected to eight nodes $(4,j)^1$, $j = 1,\ldots,8$. Hence, the sums of d and d^2 for $G_{2n,2n}$'s separator $(i,n/2)^2$, $i = 1,\ldots,n/2$, are given by $\sum_{j=n}^{3n/2-1} j$ and $\sum_{j=n}^{3n/2-1} j^2$, respectively. By symmetry, this is also true for the separator $(i,n/2)^2$ for $i = n/2+1, n/2+2,\ldots,n$. Last, we eliminate the n nodes $(n/2,j)$ for $j = 1,2,\ldots,n$. Whenever we eliminate one of them, it is connected to all the nodes to be eliminated. Therefore, for these last n nodes, the sums of d and d^2 are given by $\sum_{j=0}^{n-1} j$ and $\sum_{j=0}^{n-1} j^2$, respectively. Altogether,

$$\sum_{v \in S^G_{2n,2n}} d_v^2 = 2 \sum_{j=n}^{3n/2-1} j^2 + \sum_{j=0}^{n-1} j^2 = \frac{23}{12} n^3 + O(n^2), \tag{A.12}$$

$$\sum_{v \in S^G_{2n,2n}} d_v = 2 \sum_{j=n}^{3n/2-1} j + \sum_{j=0}^{n-1} j = \frac{7}{4} n^2 + O(n), \tag{A.13}$$

where the first summand is for nodes $(i,n/2)^2$ for $i = 1,\ldots,n$, and the second one for nodes $(n/2,j)^1$ for $j = 1,\ldots,8$. Consequently,

$$\mu_m(S^G_{2n,2n}) = \sum_{v \in S^A_{2n,2n}} d_v(d_v+3)/2 = \frac{23}{24} n^3 + O(n^2), \tag{A.14}$$

$$\mu_s(S^G_{2n,2n}) = \sum_{v \in S^A_{2n,2n}} (d_v+1) = \frac{7}{4} n^2 + O(n). \tag{A.15}$$

We skip the detailed analysis but just list the results for A, B, and C below:

$$\sum_{v \in S_A} d_v = \sum_{j=3n/2}^{2n-1} j + \sum_{j=2n}^{5n/2-1} j + \sum_{j=2n}^{3n-1} j, \tag{A.16}$$

$$\mu_m(S^A_{2n,2n}) = \frac{125}{24} n^3 + O(n^2), \tag{A.17}$$

$$\mu_s(S^A_{2n,2n}) = \frac{9}{2} n^2 + O(n); \tag{A.18}$$

$$\sum_{v \in S_B} d_v = 2 \sum_{j=5n/2}^{3n-1} j + \sum_{j=3n}^{4n-1} j, \tag{A.19}$$

$$\mu_m(S^B_{2n,2n}) = \frac{239}{24} n^3 + O(n^2), \tag{A.20}$$

$$\mu_s(S^B_{2n,2n}) = \frac{25}{4} n^2 + O(n); \tag{A.21}$$

$$\sum_{v\in S_C} d_v = 2\sum_{j=3n}^{7n/2-1} j + \sum_{j=4n}^{5n-1} j, \tag{A.22}$$

$$\mu_m(S^C_{2n,2n}) = \frac{371}{24}n^3 + o(n^3), \tag{A.23}$$

$$\mu_s(S^C_{2n,2n}) = \frac{31}{4}n^2 + o(n^2). \tag{A.24}$$

While $S^G_{2n,2n}$ and $S^C_{2n,2n}$ are eliminated in the order of the column nodes $(\cdot, n/2)^2$ followed by the row nodes $(n/2,\cdot)^1$, we eliminate $S^B_{2n,2n}$ in the order of the row nodes $(n/2,\cdot)^1$ followed by the column nodes $(\cdot, n/2)^2$. Moreover, we eliminate $S^A_{2n,2n}$ by

$$\begin{aligned}
&(i, n/2)^2 \quad \text{for } i = 1,\dots,n/2,\\
&(n/2, j)^1 \quad \text{for } i = n/2+1,\dots,n,\\
&(n/2, j)^1 \quad \text{for } i = 1,\dots,n/2,\\
&(i, n/2)^2 \quad \text{for } i = n/2+1,\dots,n,
\end{aligned}$$

in this order. This approach gives a better result.

Given $\mu(S^G_{2n,2n})$, $\mu(S^A_{2n,2n})$, $\mu(S^B_{2n,2n})$, and $\mu(S^C_{2n,2n})$, we can solve from the above recursions (see [18]) that the multiplication cost of applying CF to $G_{2n,2n}$ is $943n^3/84 + O(n^2\log_2 n)$ and the storage cost is $(31/4)n^2\log_2 n + O(n^2)$. Similar results can be derived for matrices whose adjacency graphs have the form shown in Figure 2.

Acknowledgments. We would like to thank Stanley Osher for introducing us to total variation image restoration and for many helpful and stimulating discussions, and Jinjun Xu for making his PDE codes available to us.

REFERENCES

[1] F. Alizadeh and D. Goldfarb, *Second-order cone programming*, Math. Program., 95 (2003), pp. 3–51.

[2] F. Alizadeh and S. Schmieta, *Optimization with Semidefinite, Quadratic, and Linear Constraints*, Technical report, RUTCOR, Rutgers University, Piscataway, NJ, 1997.

[3] K.D. Andersen, E. Christiansen, A.R. Conn, and M.L. Overton, *An efficient primal-dual interior-point method for minimizing a sum of Euclidean norms*, SIAM J. Sci. Comput., 22 (2000), pp. 243–262.

[4] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing*, Appl. Math. Sci. 147, Springer, New York, 2002.

[5] A. Chambolle, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vis., 20 (2004), pp. 89–97.

[6] A. Chambolle and P.-L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[7] R. Chan, T. Chan, and H.-M. Zhou, *Continuation Method for Total Variation Denoising Problems*, CAM Report 95-18, UCLA, 1995.

[8] T. Chan and S. Esedoglu, *Aspects of Total Variation Regularized L^1 Function Approximation*, CAM Report 04-07, UCLA, 2004.

[9] T.F. Chan, G.H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[10] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[11] D. Goldfarb and K. Scheinberg, *Numerically Stable Product-Form Cholesky (LDL^T) Factorization Based Implementations of Interior Point Methods for Second-Order Cone Programming*, CORC Report TR-2003-05, IEOR Department, Columbia University, New York, 2003.

[12] M. Hintermüller and K. Kunisch, *Total bounded variation regularization as a bilaterally constrained optimization problem*, SIAM J. Appl. Math., 64 (2004), pp. 1311–1333.
[13] M. Hintermüller and G. Stadler, *An Infeasible Primal-Dual Algorithm for TV-Based inf-Convolution-type Image Restoration*, Technical report TR04-15, CAAM Department, Rice University, Houston, TX, 2004.
[14] M. Lysaker, A. Lundervold, and X.-C. Taj, *Noise removal using fourth-order partial differential equations with applications to medical magnetic resonance images in space and time*, IEEE Trans. Image Process., 12 (2003), pp. 1579–1590.
[15] Y. Meyer, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, Univ. Lecture Ser. 22, AMS, Providence, RI, 2002.
[16] S. Osher, A. Solé, and L. Vese, *Image decomposition and restoration using total variation minimization and the H^{-1} norm*, Multiscale Model. Simul., 1 (2003), pp. 349–370.
[17] I. Pollak, A. Willsky, and Y. Huang, *Nonlinear evolution equations as fast and exact solvers of estimation problems*, IEEE Trans. Signal Process., 53 (2005), pp. 484–498.
[18] D. Rose and G. Whitten, *A recursive analysis of dissection strategies*, in Sparse Matrix Computations, Academic Press, New York, 1976, pp. 59–84.
[19] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[20] D. Strong and T. Chan, *Edge-preserving and scale-dependent properties of total variation regularization*, Inverse Problems, 19 (2003), pp. 165–187.
[21] L. Vese, *A study in the BV space of a denoising-deblurring variational problem*, Appl. Math. Optim., 44 (2001), pp. 131–161.
[22] L. Vese and S. Osher, *Modeling textures with total variation minimization and oscillating patterns in image processing*, J. Sci. Comput., 19 (2003), pp. 553–572.
[23] C.R. Vogel and M.E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.
[24] W. Yin, D. Goldfarb, and S. Osher, *Total Variation-Based Image Cartoon-Texture Decomposition*, CORC Report, TR-2005-01, IEOR Department, Columbia University, New York, 2005.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 646–666
© 2005 Society for Industrial and Applied Mathematics

FAST ITERATIVE SOLVERS FOR DISCRETE STOKES EQUATIONS*

JÖRG PETERS†, VOLKER REICHELT†, AND ARNOLD REUSKEN†

Abstract. We consider saddle point problems that result from the finite element discretization of stationary and nonstationary Stokes equations. Three efficient iterative solvers for these problems are treated, namely, the preconditioned conjugate gradient method introduced by Bramble and Pasciak, the preconditioned MINRES method, and a method due to Bank et al. We give a detailed overview of algorithmic aspects and theoretical convergence results. For the method of Bank et al. a new convergence analysis is presented. A comparative study of the three methods for a three-dimensional Stokes problem discretized by the Hood–Taylor $P_2 - P_1$ finite element pair is given.

Key words. Stokes equations, inexact Uzawa methods, preconditioned MINRES, multigrid

AMS subject classifications. 65N30, 65F10

DOI. 10.1137/040606028

1. Introduction.

We consider a class of Stokes equations on a bounded connected polyhedral Lipschitz domain Ω in d-dimensional Euclidean space. We use the notation $\mathbf{V} := H_0^1(\Omega)^d$ and $M = L_0^2(\Omega) := \{\, p \in L^2(\Omega) \mid \int_\Omega p(x)\,dx = 0 \,\}$ for the velocity space and the pressure space, respectively. The variational problem is as follows: Given $\mathbf{f} \in L^2(\Omega)^d$, find $\{\mathbf{u}, p\} \in \mathbf{V} \times M$ such that

$$\begin{cases} (\nabla \mathbf{u}, \nabla \mathbf{v}) + \xi(\mathbf{u}, \mathbf{v}) - (\operatorname{div} \mathbf{v}, p) = (\mathbf{f}, \mathbf{v}) & \text{for } \mathbf{v} \in \mathbf{V}, \\ (\operatorname{div} \mathbf{u}, q) = 0 & \text{for } q \in M \end{cases} \tag{1.1}$$

with a constant $\xi \geq 0$. Here and in what follows the L^2 scalar product and associated norm are denoted by $(\cdot,\cdot)$, $\|\cdot\|$, respectively. The zeroth order term $\xi(\mathbf{u}, \mathbf{v})$ is included in view of implicit time integration methods applied to nonstationary Stokes equations. For the discretization of this problem we use a pair of conforming Ladyzhenskaya–Babuška–Brezzi (LBB) stable finite element spaces [7, 13]. This results in a saddle point problem of the form

$$\begin{pmatrix} \mathrm{A} & \mathrm{B}^T \\ \mathrm{B} & 0 \end{pmatrix} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix} = \begin{pmatrix} \mathrm{f} \\ 0 \end{pmatrix}. \tag{1.2}$$

Many different iterative methods for solving this discrete problem are known. One can apply multigrid techniques to the whole coupled system in (1.2); cf. [15, 27, 28]. Most other approaches are based on the prominent classical Uzawa method. This Uzawa method requires that $\mathrm{A}^{-1}\mathrm{x}$ can be computed exactly [1]. In many variants of this method, which are often called inexact Uzawa methods, one tries to avoid the exact inversion by using an inner iterative method, for example, a Jacobi-like iteration (Arrow–Hurwicz algorithm [1]) or a multigrid method; cf. [24]. We also mention (variants of) the preconditioned minimal residual (MINRES) method [21, 22, 23, 26] and the method from [2]. A different approach is presented in [3], where the indefinite problem (1.2) is reformulated as a symmetric positive definite problem. For most of

*Received by the editors March 31, 2004; accepted for publication (in revised form) January 25, 2005; published electronically October 31, 2005.
http://www.siam.org/journals/sisc/27-2/60602.html
†Institut für Geometrie und Praktische Mathematik, RWTH-Aachen, D-52056 Aachen, Germany (peters@igpm.rwth-aachen.de, reichelt@igpm.rwth-aachen.de, reusken@igpm.rwth-aachen.de). The work of the second author was supported by the German Research Foundation through SFB 540.

these methods theoretical convergence analyses are known; cf. [2, 3, 5, 12, 21, 23, 24, 26, 29]. In [11] the performance of a few of these methods is compared by means of systematic numerical experiments for a stationary two-dimensional Stokes problem.

In this paper we consider three representative methods from the large class of inexact Uzawa methods, namely, the preconditioned conjugate gradient method from [3] (denoted by BPCG), the preconditioned MINRES method from [21, 23, 26] (denoted by PMINRES), and the method from [2] (denoted by MGUZAWA). The topics treated in this paper are the following:

- For these three methods we discuss costs per iteration, known theoretical convergence results, and some implementation issues. This makes it possible to make a fair comparison of these methods.
- For the MGUZAWA method we present a convergence analysis. This analysis is much simpler than the analyses presented in [2, 29]. The result that we obtain is different from the ones in [2, 29] and gives a better explanation of the observation that if one uses a very good preconditioner for A (like multigrid), then even with a very low accuracy in the inner iteration the MGUZAWA method converges (cf. Remark 8).
- We present a comparative study of the performance of the three methods. For this we consider a Stokes problem as in (1.1) in three dimensions. We treat both the stationary ($\xi = 0$) and nonstationary ($\xi > 0$) cases. For the discretization we apply the popular Hood–Taylor $P_2 - P_1$ finite element pair. As a preconditioner for A a standard multigrid method is used. For the Schur complement preconditioner we use the mass matrix for the stationary case and a more sophisticated preconditioner analyzed in [4] for the nonstationary case.

The main results of this paper are a detailed comparative study of the three fast iterative solvers for a three-dimensional Stokes problem and a new convergence analysis for the MGUZAWA method.

The paper is organized as follows. In section 2 we formulate the standard Galerkin discretization of the Stokes problem. In section 3.1 we consider the BPCG method. An efficient implementation of this method is discussed and a main convergence result known from the literature is formulated. In section 3.2 the PMINRES method is treated. Also for this method a known convergence result is given. In section 3.3 we discuss the MGUZAWA method and give an efficient implementation of this method. In section 4 a convergence analysis of this method is given. Theorem 4.3 contains the main theoretical result of this paper. In section 5 we discuss the preconditioners for the A-block and for the Schur complement. In section 6 results of numerical experiments are presented that illustrate the performance of the three methods. Finally, in section 7 we give some conclusions.

2. The discrete Stokes equations. For the discretization of the Stokes problem (1.1) we assume a family of triangulations $\{\mathcal{T}_h\}$ in the sense of [9, 10] and a pair of finite element spaces $\mathbf{V}_h \subset \mathbf{V}$ and $M_h \subset M$ that is LBB stable with a constant $\hat{\beta}$ independent of h:

$$\inf_{q_h \in M_h} \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{(\operatorname{div} \mathbf{v}_h, q_h)}{\|\nabla \mathbf{v}_h\| \|q_h\|} \geq \hat{\beta} > 0. \tag{2.1}$$

The Galerkin discretization is as follows: Find $\{\mathbf{u}_h, p_h\} \in \mathbf{V}_h \times M_h$ such that

$$\begin{cases} (\nabla \mathbf{u}_h, \nabla \mathbf{v}_h) + \xi(\mathbf{u}_h, \mathbf{v}_h) - (\operatorname{div} \mathbf{v}_h, p_h) = (\mathbf{f}, \mathbf{v}_h) & \text{for } \mathbf{v}_h \in \mathbf{V}_h, \\ (\operatorname{div} \mathbf{u}_h, q_h) = 0 & \text{for } q_h \in M_h. \end{cases} \tag{2.2}$$

In practice the discrete space M_h for the pressure is constructed by taking a standard finite element space, which we denote by M_h^+ (for example, continuous piecewise linear functions), and then adding an orthogonality condition:

$$M_h := \{\, p_h \in M_h^+ \mid (p_h, 1) = 0 \,\}.$$

Note that $\dim(M_h) = \dim(M_h^+) - 1$. Let $n := \dim(\mathbf{V}_h)$, $m := \dim(M_h^+)$. We assume standard (nodal) bases in $\mathbf{V}_h$ and M_h^+ and corresponding isomorphisms

$$J_V : \mathbb{R}^n \to V_h \ , \quad J_M : \mathbb{R}^m \to M_h^+.$$

Let the stiffness matrices $\mathrm{D} \in \mathbb{R}^{n\times n}$, $\mathrm{B} \in \mathbb{R}^{m\times n}$ and the mass matrices $\tilde{\mathrm{M}} \in \mathbb{R}^{n\times n}$, $\mathrm{M} \in \mathbb{R}^{m\times m}$ be given by

$$\begin{aligned}
\langle \mathrm{Dx}, \mathrm{y}\rangle &= (\nabla J_V \mathrm{x}, \nabla J_V \mathrm{y}) && \text{for all } \ \mathrm{x}, \mathrm{y} \in \mathbb{R}^n, \\
\langle \mathrm{Bx}, \mathrm{y}\rangle &= (\operatorname{div} J_V \mathrm{x}, J_M \mathrm{y}) && \text{for all } \ \mathrm{x} \in \mathbb{R}^n,\ \mathrm{y} \in \mathbb{R}^m, \\
\langle \tilde{\mathrm{M}}\mathrm{x}, \mathrm{y}\rangle &= (J_V \mathrm{x}, J_V \mathrm{y}) && \text{for all } \ \mathrm{x}, \mathrm{y} \in \mathbb{R}^n, \\
\langle \mathrm{Mx}, \mathrm{y}\rangle &= (J_M \mathrm{x}, J_M \mathrm{y}) && \text{for all } \ \mathrm{x}, \mathrm{y} \in \mathbb{R}^m.
\end{aligned} \tag{2.3}$$

Here $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean scalar product. The discrete problem has a matrix-vector representation of the form

$$\begin{pmatrix} \mathrm{A} & \mathrm{B}^T \\ \mathrm{B} & 0 \end{pmatrix} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix} = \begin{pmatrix} \mathrm{f} \\ 0 \end{pmatrix} \ , \qquad \mathrm{A} := \mathrm{D} + \xi \tilde{\mathrm{M}}, \tag{2.4}$$

with f such that $\langle \mathrm{f}, \mathrm{y}\rangle = (\mathbf{f}, J_V \mathrm{y})$ for all $\mathrm{y} \in \mathbb{R}^n$. We introduce the notation

$$\mathrm{K} := \begin{pmatrix} \mathrm{A} & \mathrm{B}^T \\ \mathrm{B} & 0 \end{pmatrix} \ , \quad \mathrm{S} := \mathrm{B}\mathrm{A}^{-1}\mathrm{B}^T.$$

Define the constant vector $\mathrm{e} := J_M^{-1} 1 = (1, \ldots, 1)^T \in \mathbb{R}^m$. Note that $\mathrm{B}^T \mathrm{e} = 0$ and that both the Schur complement S and the matrix K are singular and have a one-dimensional kernel. For the Schur complement this kernel is $\ker(\mathrm{S}) = \operatorname{span}\{\mathrm{e}\}$. Note that

$$(J_M \mathrm{y}, 1) = 0 \ \Leftrightarrow \ (J_M \mathrm{y}, J_M \mathrm{e}) = 0 \ \Leftrightarrow \ \langle \mathrm{My}, \mathrm{e}\rangle = 0 \ .$$

Hence, with $\mathrm{e}^{\perp_M} := \{\, \mathrm{y} \in \mathbb{R}^m \mid \langle \mathrm{y}, \mathrm{Me}\rangle = 0 \,\}$ we have $M_h = \{\, J_M \mathrm{y} \mid \mathrm{y} \in \mathrm{e}^{\perp_M} \,\}$ and we get the following matrix-vector representation of the discrete problem (2.2):

$$\text{Find } \ \mathrm{x} \in \mathbb{R}^n, \ \mathrm{y} \in \mathrm{e}^{\perp_M} \ \text{ such that (2.4) holds.} \tag{2.5}$$

3. Iterative solvers. In this section we describe three iterative solvers for problem (2.5). The first two methods, a preconditioned conjugate gradient method (BPCG) and a preconditioned MINRES method (PMINRES) are known from the literature and often used in practice. The third method, which may be less known, is a simple variant of the Uzawa method as presented in [2]. In section 4 a convergence analysis of this method is given.

In all three solvers we need symmetric positive definite preconditioners Q_A of A and Q_S of S. The quality of these preconditioners is described by the following spectral equivalences. Let $\gamma_A > 0$, Γ_A, $\gamma_S > 0$, Γ_S be such that

$$\gamma_A \mathrm{Q}_A \le \mathrm{A} \le \Gamma_A \mathrm{Q}_A, \tag{3.1}$$

$$\gamma_S \mathrm{Q}_S \le \mathrm{S} \le \Gamma_S \mathrm{Q}_S \quad \text{on } \ \mathrm{e}^{\perp_M}. \tag{3.2}$$

Here and in what follows, for symmetric matrices E, F we write $E \le F$ ($E < F$) if and only if $F - E$ is positive semidefinite (positive definite). In section 5 we will discuss concrete choices for the preconditioners Q_S and Q_A.

3.1. Preconditioned conjugate gradient method. We describe the preconditioned conjugate gradient (BPCG) method introduced in [3]. For this method we need the following assumption on the preconditioner Q_A:

$$0 < \mathrm{Q}_A < \mathrm{A}; \tag{3.3}$$

i.e., there exists $\gamma_A > 1$ such that (3.1) holds. Clearly, using some suitable scaling one can always satisfy the condition (3.3). Premultiplication of (2.4) by the matrix

$$\mathrm{G} := \begin{pmatrix} \mathrm{Q}_A^{-1} & 0 \\ \mathrm{B}\mathrm{Q}_A^{-1} & -\mathrm{I} \end{pmatrix}$$

yields an equivalent system:

$$\hat{\mathrm{K}} \begin{pmatrix} \mathrm{x} \\ \mathrm{y} \end{pmatrix} = \mathrm{G} \begin{pmatrix} \mathrm{f} \\ 0 \end{pmatrix}, \quad \hat{\mathrm{K}} := \mathrm{G}\mathrm{K} = \begin{pmatrix} \mathrm{Q}_A^{-1}\mathrm{A} & \mathrm{Q}_A^{-1}\mathrm{B}^T \\ \mathrm{B}\mathrm{Q}_A^{-1}\mathrm{A} - \mathrm{B} & \mathrm{B}\mathrm{Q}_A^{-1}\mathrm{B}^T \end{pmatrix}. \tag{3.4}$$

Due to the assumption (3.3) the bilinear form

$$\left[\begin{pmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \end{pmatrix}, \begin{pmatrix} \mathrm{y}_1 \\ \mathrm{y}_2 \end{pmatrix} \right] := \langle (\mathrm{A} - \mathrm{Q}_A)\mathrm{x}_1, \mathrm{y}_1 \rangle + \langle \mathrm{x}_2, \mathrm{y}_2 \rangle \tag{3.5}$$

defines an inner product. In [3] it is shown that the matrix $\hat{\mathrm{K}}$ is [illegible] on $\mathbb{R}^n \times \mathrm{e}^{\perp_M}$ with respect to this inner product. The matrix

$$\tilde{\mathrm{K}} := \begin{pmatrix} \mathrm{I} & 0 \\ 0 & \mathrm{Q}_S \end{pmatrix}$$

is also symmetric positive definite with respect to this scalar product. Hence we can apply the preconditioned conjugate gradient method in the scalar product $\left[\cdot,\cdot\right]$ and with preconditioner $\tilde{\mathrm{K}}$ to the linear system in (3.4). The algorithm is as follows:

$$\begin{cases} \text{Choose a starting vector } \mathrm{v}^0 = \begin{pmatrix} \mathrm{x}^0 \\ \mathrm{y}^0 \end{pmatrix}; \quad \bar{\mathrm{r}}^0 := \begin{pmatrix} \mathrm{f} \\ 0 \end{pmatrix} - \mathrm{K}\mathrm{v}^0, \quad \mathrm{r}^0 := \mathrm{G}\bar{\mathrm{r}}^0. \\ \text{For } k \geq 0 \quad (\text{if } \mathrm{r}^k \neq 0): \\ \text{Solve } \mathrm{z}^k \text{ from } \tilde{\mathrm{K}}\mathrm{z}^k = \mathrm{r}^k. \\ \beta_k^{(n)} := \left[\mathrm{z}^k, \mathrm{r}^k\right], \quad \beta_k := \beta_k^{(n)}/\beta_{k-1}^{(n)} \quad (\beta_0 := 0), \\ \mathrm{p}^k := \mathrm{z}^k + \beta_k \mathrm{p}^{k-1} \quad (\mathrm{p}^0 := \mathrm{z}^0), \\ \alpha_k^{(d)} := \left[\hat{\mathrm{K}}\mathrm{p}^k, \mathrm{p}^k\right], \quad \alpha_k := \beta_k^{(n)}/\alpha_k^{(d)}, \\ \mathrm{v}^{k+1} := \mathrm{v}^k + \alpha_k \mathrm{p}^k, \\ \mathrm{r}^{k+1} := \mathrm{r}^k - \alpha_k \hat{\mathrm{K}}\mathrm{p}^k. \end{cases} \tag{3.6}$$

This is the standard preconditioned conjugate gradient algorithm. We now discuss an efficient implementation of this algorithm based on ideas from [11]. In particular we want to avoid the evaluation of Q_A. We use indices 1 and 2 to denote the first and second vector components, respectively (as in (3.5)). First note that due to the structure of $\tilde{\mathrm{K}}$ we have $\mathrm{z}_1^k = \mathrm{r}_1^k$. We introduce $\bar{\mathrm{r}}_1^k := \mathrm{Q}_A \mathrm{r}_1^k$ and $\mathrm{d} := \mathrm{A}\mathrm{r}_1^k$. We then have

$$\begin{aligned} \left[\mathrm{z}^k, \mathrm{r}^k\right] &= \langle (\mathrm{A} - \mathrm{Q}_A)\mathrm{z}_1^k, \mathrm{r}_1^k \rangle + \langle \mathrm{z}_2^k, \mathrm{r}_2^k \rangle \\ &= \langle (\mathrm{A} - \mathrm{Q}_A)\mathrm{r}_1^k, \mathrm{r}_1^k \rangle + \langle \mathrm{z}_2^k, \mathrm{r}_2^k \rangle \\ &= \langle \mathrm{d}, \mathrm{r}_1^k \rangle - \langle \bar{\mathrm{r}}_1^k, \mathrm{r}_1^k \rangle + \langle \mathrm{z}_2^k, \mathrm{r}_2^k \rangle \ . \end{aligned}$$

Also define $s^k := Ap_1^k$. From $p^k = z^k + \beta_k p^{k-1}$ it follows that for s^k we have the relation

$$s^k = Ar_1^k + \beta_k s^{k-1} = d + \beta_k s^{k-1}.$$

With

$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix} := Kp^k = \begin{pmatrix} s^k + B^T p_2^k \\ Bp_1^k \end{pmatrix}, \quad \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} := \hat{K}p^k = \begin{pmatrix} Q_A^{-1} q_1 \\ BQ_A^{-1} q_1 - q_2 \end{pmatrix},$$

we obtain

$$\begin{aligned} \left[\hat{K}p^k, p^k\right] &= \langle (A - Q_A) w_1, p_1^k \rangle + \langle w_2, p_2^k \rangle \\ &= \langle w_1, Ap_1^k \rangle - \langle q_1, p_1^k \rangle + \langle w_2, p_2^k \rangle = \langle w_1, s^k \rangle - \langle q_1, p_1^k \rangle + \langle w_2, p_2^k \rangle. \end{aligned}$$

Finally note that for $\bar{r}_1^k$ we have the recursion

$$\bar{r}_1^{k+1} := \bar{r}_1^k - \alpha_k (Kp^k)_1 = \bar{r}_1^k - \alpha_k q_1.$$

Summarizing, we then obtain the following efficient implementation of the BPCG algorithm:

$$\text{(3.7)} \quad \left\{ \begin{array}{l} \text{Choose a starting vector } v^0 = \begin{pmatrix} x^0 \\ y^0 \end{pmatrix}; \quad \bar{r}^0 := \begin{pmatrix} f \\ 0 \end{pmatrix} - Kv^0 \ , \quad r^0 := G\bar{r}^0. \\ \text{For } k \geq 0 \quad (\text{if } r^k \neq 0): \\ \text{Solve } z_2^k \text{ from } Q_S z_2^k = r_2^k, \quad z_1^k := r_1^k. \\ d := Ar_1^k, \\ \beta_k^{(n)} := \langle d, r_1^k \rangle - \langle \bar{r}_1^k, r_1^k \rangle + \langle z_2^k, r_2^k \rangle, \quad \beta_k := \beta_k^{(n)} / \beta_{k-1}^{(n)} \quad (\beta_0 := 0), \\ p^k := z^k + \beta_k p^{k-1} \quad (p^0 := z^0), \quad s^k = d + \beta_k s^{k-1} \quad (s^0 := d), \\ \begin{pmatrix} q_1 \\ q_2 \end{pmatrix} := \begin{pmatrix} s^k + B^T p_2^k \\ Bp_1^k \end{pmatrix}, \quad \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} := \begin{pmatrix} Q_A^{-1} q_1 \\ BQ_A^{-1} q_1 - q_2 \end{pmatrix}, \\ \alpha_k^{(d)} := \langle w_1, s^k \rangle - \langle q_1, p_1^k \rangle + \langle w_2, p_2^k \rangle, \quad \alpha_k := \beta_k^{(n)} / \alpha_k^{(d)}, \\ v^{k+1} := v^k + \alpha_k p^k, \\ r^{k+1} := r^k - \alpha_k \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \quad \bar{r}_1^{k+1} := \bar{r}_1^k - \alpha_k q_1. \end{array} \right.$$

In this algorithm we need Q_A, but we need evaluation of Q_A^{-1} per iteration (by "evaluation" we mean the computation of $Q_A^{-1} q_1$ for a $q_1 \in \mathbb{R}^n$). Furthermore, per iteration we need one evaluation of Q_S^{-1} and two matrix-vector products with B, one with B^T and one with A.

We want to use this algorithm to solve problem (2.5). Then it is desirable that the iterates $v^k = (x^k, y^k)$ are such that $y^k \in e^{\perp_M}$ holds. Therefore we assume that the preconditioner Q_S satisfies the consistency condition

$$\text{(3.8)} \qquad Q_S e = \alpha Me, \quad \alpha \in \mathbb{R}.$$

From this assumption it follows that $y^0 \in e^{\perp_M}$ implies $y^k \in e^{\perp_M}$ for all $k \geq 1$. Moreover, if (3.8) holds, then $\tilde{K}^{-1}\hat{K}$ is bijective on $\mathbb{R}^n \times e^{\perp_M}$. We now formulate a

main convergence result for the BPCG algorithm due to [3, 29]. Let $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}})$ be the spectral condition number of the matrix $\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}$ on the subspace $\mathbb{R}^n \times \mathrm{e}^{\perp_M}$. This quantity plays a key role in the convergence analysis of the preconditioned conjugate gradient method.

THEOREM 3.1. [illegible] (3.1) (3.2) (3.3) [illegible] (3.8) [illegible] $x \in \mathbb{R}$ [illegible]

$$b(x) := \frac{1}{2}(1+x) + \frac{1}{2}\sqrt{(1+x)^2 - 4x\Gamma_A^{-1}}.$$

[illegible]

$$\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) \le \frac{\Gamma_S}{\gamma_S}\frac{\Gamma_A b(\Gamma_S) b(\gamma_S)}{\Gamma_S} \ . \tag{3.9}$$

[illegible] We can apply Theorem 4.1 in [29] in the subspace $\mathbb{R}^n \times \mathrm{e}^{\perp_M}$. This yields $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) \le (1-\rho_1)/(1-\rho_2)$, with

$$\rho_1 := \frac{2-(1+\Gamma_S)\Gamma_A}{2} - \sqrt{\frac{(2-(1+\Gamma_S)\Gamma_A)^2}{4} + \Gamma_A - 1}\ ,$$
$$\rho_2 := \frac{2-(1+\gamma_S)\Gamma_A}{2} + \sqrt{\frac{(2-(1+\gamma_S)\Gamma_A)^2}{4} + \Gamma_A - 1}\ .$$

Elementary manipulations show that $(1-\rho_1)/(1-\rho_2) = \frac{\Gamma_S}{\gamma_S}\frac{\Gamma_A b(\Gamma_S) b(\gamma_S)}{\Gamma_S}$ holds. ∎

[illegible] 1. Note that $\Gamma_A \ge 1$ and thus $\frac{1}{2}(1+x) + \frac{1}{2}|1-x| \le b(x) \le \frac{1}{2}(1+x) + \frac{1}{2}|1+x|$. Hence, $1 \le b(x) \le 1+x$ if $x \in [0,1]$ and $x \le b(x) \le 1+x$ if $x \ge 1$. From this it follows that for the factor $C(\gamma_S, \Gamma_S) := \frac{b(\Gamma_S)b(\gamma_S)}{\Gamma_S}$ in (3.9) we have

$$\frac{1}{\Gamma_S} \le C(\gamma_S,\Gamma_S) \le \frac{4}{\Gamma_S} \quad \text{if } \ \Gamma_S \le 1,$$
$$1 \le C(\gamma_S,\Gamma_S) \le 2 + \frac{2}{\Gamma_S} \le 4 \quad \text{if } \ \gamma_S \le 1 \le \Gamma_S,$$
$$\gamma_S \le C(\gamma_S,\Gamma_S) \le 3 + \gamma_S \le 4\gamma_S \quad \text{if } \ 1 \le \gamma_S.$$

From this we see that although the condition number of the preconditioned Schur complement $\kappa_*(\mathrm{Q}_S^{-1}\mathrm{S}) \le \Gamma_S/\gamma_s$ is [illegible] of the scaling of the preconditioner Q_S, the bound for the condition number $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}})$ in (3.9) does depend on the scaling of Q_S. A poor scaling (i.e., $\Gamma_S \ll 1$ or $\gamma_S \gg 1$) leads to a large upper bound. We consider an example, which shows that this bound is sharp. Take $\mathrm{Q}_A = \Gamma_A^{-1}\mathrm{A}$ and $\mathrm{Q}_S = \Gamma^{-1}\mathrm{S}$ (on $\mathrm{e}^{\perp}$) with $\Gamma_A > 1$ and $\Gamma > 0$. Then (3.2) is fulfilled with $\gamma_S = \Gamma_S = \Gamma$. A simple computation shows that $\sigma_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) = \{\Gamma/b(\Gamma), \Gamma_A, \Gamma_A b(\Gamma)\}$ and thus $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) = \Gamma_A\frac{b(\Gamma)^2}{\Gamma}$, which agrees with the bound in (3.9). Hence $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) \ge \frac{\Gamma_A}{\Gamma}$ if $\Gamma \le 1$ and $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}}) \ge \Gamma_A\Gamma$ if $\Gamma \ge 1$. Note, however, that although $\kappa_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}})$ is large if $\Gamma \ll 1$ or $\Gamma \gg 1$ the BPCG method will converge in three iterations because $\sigma_*(\tilde{\mathrm{K}}^{-1}\hat{\mathrm{K}})$ contains only three elements.

3.2. Preconditioned minimal residual method. In this section we consider a preconditioned minimal residual (PMINRES) method for solving the discretized Stokes problem. This class of methods has been developed in [20, 21, 23]. We consider a symmetric positive definite preconditioner

$$\tilde{\mathrm{K}} := \begin{pmatrix} \mathrm{Q}_A & 0 \\ 0 & \mathrm{Q}_S \end{pmatrix} \tag{3.10}$$

for K. Define the norm $\|v\|_{\tilde{K}} := \langle \tilde{K}v, v\rangle^{\frac{1}{2}}$ for $v \in \mathbb{R}^{n+m}$. Given a starting vector v^0 with corresponding error $e^0 := v^* - v^0$, then in the PMINRES method one computes the vector $v^k \in v^0 + \operatorname{span}\{\tilde{K}^{-1}Ke^0, \ldots, (\tilde{K}^{-1}K)^k e^0\}$, which minimizes the preconditioned residual $\|\tilde{K}^{-1}K(v^* - v)\|_{\tilde{K}}$ over this subspace.

An efficient implementation of this method can be derived using the Lanczos algorithm and Givens rotations. For such an implementation we refer to the literature; see, e.g., [19, 14]. In an efficient implementation of this method one needs per iteration one evaluation of Q_A^{-1}, one evaluation of Q_S^{-1}, and one matrix-vector product with K.

We assume that the preconditioner Q_S satisfies the consistency condition (3.8). Then it follows that the approximations y^k of the discrete pressure are in the subspace $e^{\perp_M}$ if the starting vector y^0 is in this subspace.

Note that for this algorithm one does not need the scaling condition (3.3) for Q_A. We give two main theoretical convergence results on the PMINRES algorithm from the literature (cf. [14, 21, 23]).

THEOREM 3.2. $v^0 = (x^0, y^0)$ $y^0 \in e^{\perp_M}$ v^k $k \geq 0$ $\tilde{r}^k := \tilde{K}^{-1}\big((f, 0) - Kv^k\big)$

$$\begin{aligned}\|\tilde{r}^k\|_{\tilde{K}} &= \min_{p_k \in \mathcal{P}_k; p_k(0)=1} \|p_k(\tilde{K}^{-1}K)\tilde{r}^0\|_{\tilde{K}} \\ &\leq \min_{p_k \in \mathcal{P}_k; p_k(0)=1} \max_{\lambda \in \sigma_*(\tilde{K}^{-1}K)} |p_k(\lambda)|\, \|\tilde{r}^0\|_{\tilde{K}} ,\end{aligned}$$

$\sigma_*(\cdot)$ $\mathbb{R}^n \times e^{\perp_M}$

THEOREM 3.3. (3.1) (3.2) (3.8) $\tilde{K}^{-1}K$

$$\begin{aligned}\sigma_*(\tilde{K}^{-1}K) \subset &\Big[\frac{1}{2}\big(\gamma_A - \sqrt{\gamma_A^2 + 4\Gamma_S\Gamma_A}\big)\,,\ \frac{1}{2}\big(\gamma_A - \sqrt{\gamma_A^2 + 4\gamma_S\gamma_A}\big)\Big] \\ &\bigcup \Big[\gamma_A\,,\ \frac{1}{2}\big(\Gamma_A + \sqrt{\Gamma_A^2 + 4\Gamma_S\Gamma_A}\big)\Big].\end{aligned} \tag{3.11}$$

From the results in these theorems it follows that the rate of convergence of the PMINRES method is robust with respect to variation in the parameters h and ξ if the spectral constants γ_S, Γ_S, γ_A, Γ_A do not depend on these parameters. Moreover, we can expect faster convergence if we have better preconditioners.

2. Related to the scaling we consider the same example as in Remark 1: $Q_A = \Gamma_A^{-1}A$, $Q_S = \Gamma^{-1}S$ (on $e^{\perp}$). We then have $\gamma_A = \Gamma_A$, $\gamma_S = \Gamma_S = \Gamma$. A simple computation yields $\sigma_*(\tilde{K}^{-1}K) = \{\frac{1}{2}\big(\Gamma_A - \sqrt{\Gamma_A^2 + 4\Gamma\Gamma_A}\big), \Gamma_A, \frac{1}{2}\big(\Gamma_A + \sqrt{\Gamma_A^2 + 4\Gamma\Gamma_A}\big)\}$. This shows that for this example the result in (3.11) is sharp. Furthermore, the scaling of Q_S has a significant influence on the endpoints of the intervals in (3.11). However, for this example, the PMINRES method will converge in three iterations since $\sigma_*(\tilde{K}^{-1}K)$ contains only three elements.

3.3. An inexact Uzawa method. A basic method for saddle point problems is the Uzawa method. This method is closely related to the block factorization

$$K = \begin{pmatrix} A & 0 \\ B & -I \end{pmatrix} \begin{pmatrix} I & A^{-1}B^T \\ 0 & S \end{pmatrix}.$$

Solving the problem $\mathrm{K}\binom{\mathrm{x}}{\mathrm{y}} = \binom{\mathrm{f}}{0}$ by block forward-backward substitution yields the equivalent problem:

(3.12) 1. Solve $\mathrm{Az} = \mathrm{f}$.

(3.13) 2. Solve $\mathrm{Sy} = \mathrm{Bz}, \quad \mathrm{y} \in \mathrm{e}^{\perp_M}$.

(3.14) 3. Solve $\mathrm{Ax} = \mathrm{z} - \mathrm{B}^T\mathrm{y}$.

In the Uzawa method one applies an iterative solver (e.g., conjugate gradient) to the Schur complement system in step 2. The A-systems that occur in each iteration of this method and in steps 1 and 3 are solved sufficiently accurately using some fast Poisson solver.

Here we reconsider a simple variant of this method which was introduced in [2]. Let Q_A be a preconditioner of A as in (3.1). We use this preconditioner in steps 1 and 3 and also for the approximation of the Schur complement in step 2. For this we introduce the notation

$$\hat{\mathrm{S}} := \mathrm{BQ}_A^{-1}\mathrm{B}^T.$$

Note that $\hat{\mathrm{S}} : \mathrm{e}^{\perp_M} \to \mathrm{e}^{\perp}$ is bijective. We use a (nonlinear) approximate inverse of $\hat{\mathrm{S}}$ denoted by $\Psi : \mathrm{e}^{\perp} \to \mathrm{e}^{\perp_M}$. For each $\mathrm{w} \in \mathrm{e}^{\perp}$, $\Psi(\mathrm{w})$ is an approximation to the solution $\mathrm{z}^* \in \mathrm{e}^{\perp_M}$ of $\hat{\mathrm{S}}\mathrm{z} = \mathrm{w}$. We assume that

$$\|\Psi(\mathrm{w}) - \mathrm{z}^*\|_{\hat{\mathrm{S}}} \le \delta \|\mathrm{z}^*\|_{\hat{\mathrm{S}}} \quad \text{for all} \quad \mathrm{w} \in \mathrm{e}^{\perp} \tag{3.15}$$

holds with some $\delta < 1$. In our experiments Ψ will be the preconditioned conjugate gradient method.

Let $(\mathrm{x}^k, \mathrm{y}^k)$ be a given approximation to the solution (x, y). Note that using the block factorization of K gives us

$$\begin{pmatrix}\mathrm{x}\\ \mathrm{y}\end{pmatrix} = \begin{pmatrix}\mathrm{x}^k\\ \mathrm{y}^k\end{pmatrix} + \begin{pmatrix}\mathrm{I} & -\mathrm{A}^{-1}\mathrm{B}^T\mathrm{S}^{-1}\\ 0 & \mathrm{S}^{-1}\end{pmatrix}\begin{pmatrix}\mathrm{A}^{-1} & 0\\ \mathrm{BA}^{-1} & -\mathrm{I}\end{pmatrix}\left(\begin{pmatrix}\mathrm{f}\\ 0\end{pmatrix} - \mathrm{K}\begin{pmatrix}\mathrm{x}^k\\ \mathrm{y}^k\end{pmatrix}\right). \tag{3.16}$$

With $\mathrm{A}^{-1} \approx \mathrm{Q}_A^{-1}$, $\mathrm{S}^{-1}\mathrm{w} \approx \hat{\mathrm{S}}^{-1}\mathrm{w} \approx \Psi(\mathrm{w})$, and $\mathrm{r}_1^k := \mathrm{f} - \mathrm{Ax}^k - \mathrm{B}^T\mathrm{y}^k$ we obtain the (nonlinear) iterative method

$$\begin{aligned}\mathrm{x}^{k+1} &= \mathrm{x}^k + \mathrm{Q}_A^{-1}\mathrm{r}_1^k - \mathrm{Q}_A^{-1}\mathrm{B}^T\Psi\big(\mathrm{B}(\mathrm{Q}_A^{-1}\mathrm{r}_1^k + \mathrm{x}^k)\big),\\ \mathrm{y}^{k+1} &= \mathrm{y}^k + \Psi\big(\mathrm{B}(\mathrm{Q}_A^{-1}\mathrm{r}_1^k + \mathrm{x}^k)\big).\end{aligned} \tag{3.17}$$

We obtain the following algorithmic structure:

Choose a starting vector $\mathrm{v}^0 = \begin{pmatrix}\mathrm{x}^0\\ \mathrm{y}^0\end{pmatrix}$, with $\mathrm{y}^0 \in \mathrm{e}^{\perp_M}$; $\mathrm{r}_1^0 := \mathrm{f} - \mathrm{Ax}^0 - \mathrm{B}^T\mathrm{y}^0$.

For $k \ge 0$:

$$\mathrm{w} := \mathrm{x}^k + \mathrm{Q}_A^{-1}\mathrm{r}_1^k, \tag{3.18}$$

$$\mathrm{z} := \Psi(\mathrm{Bw}) \in \mathrm{e}^{\perp_M}, \tag{3.19}$$

$$\mathrm{x}^{k+1} := \mathrm{w} - \mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{z}, \tag{3.20}$$

$$\mathrm{y}^{k+1} := \mathrm{y}^k + \mathrm{z}, \tag{3.21}$$

$$\mathrm{r}_1^{k+1} := \mathrm{r}_1^k - \mathrm{A}(\mathrm{x}^{k+1} - \mathrm{x}^k) - \mathrm{B}^T\mathrm{z}. \tag{3.22}$$

This is the same method as the one presented in section 4 in [2]. An obvious choice for $\Psi(\mathrm{Bw})$, which is also used in our experiments, is the following:

$$\Psi(\mathrm{Bw}) = \begin{cases} \text{Result of } \ell \text{ preconditioned conjugate gradient iterations with startvector 0 and preconditioner } \mathrm{Q}_S \\ \text{applied to } \hat{\mathrm{S}}\mathrm{y} = \mathrm{Bw}. \end{cases} \tag{3.23}$$

3. We comment on some implementation issues and on the arithmetic costs per iteration of the algorithm (3.18)–(3.23). In each iteration of the preconditioned conjugate gradient algorithm in (3.23) one needs an evaluation of Q_S^{-1} and of $\hat{\mathrm{S}} = \mathrm{B}\mathrm{Q}_A^{-1}\mathrm{B}^T$ (i.e., the computation of a matrix-vector product with these matrices). Thus per preconditioned conjugate gradient iteration we need one evaluation of Q_S^{-1}, one of Q_A^{-1}, and a matrix-vector multiplication with B and with B^T. In the preconditioned conjugate gradient method one computes approximations $\mathrm{z}^0 = 0, \mathrm{z}^1, \ldots, \mathrm{z}^\ell =: \mathrm{z}$. For these approximations we have a recursion of the form $\mathrm{z}^{i+1} = \mathrm{z}^i + \alpha_i \mathrm{p}^i$ (cf. algorithm (3.6)) and the vectors $\hat{\mathrm{S}}\mathrm{p}^i = \mathrm{B}(\mathrm{Q}_A^{-1}(\mathrm{B}^T\mathrm{p}^i))$ are computed. Hence, in the inner preconditioned conjugate gradient loop (3.19), (3.23) the vectors $\mathrm{B}^T\mathrm{p}^i$, $\mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{p}^i$, $i = 0, \ldots, \ell-1$, are available. For $\bar{\mathrm{z}}^i := \mathrm{B}^T\mathrm{z}^i$, $\hat{\mathrm{z}}^i := \mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{z}^i$ we have recursions $\bar{\mathrm{z}}^{i+1} = \bar{\mathrm{z}}^i + \alpha_i \mathrm{B}^T\mathrm{p}^i$ and $\hat{\mathrm{z}}^{i+1} = \hat{\mathrm{z}}^i + \alpha_i \mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{p}^i$, and thus $\bar{\mathrm{z}}^i$, $\hat{\mathrm{z}}^i$, $i = 1, \ldots, \ell$, can be computed with low costs (AXPY operations). Using this approach we can obtain $\bar{\mathrm{z}}^\ell = \mathrm{B}^T\mathrm{z}^\ell = \mathrm{B}^T\mathrm{z}$ and $\hat{\mathrm{z}}^\ell = \mathrm{Q}_A^{-1}\bar{\mathrm{z}}^\ell = \mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{z}$ with negligible computational costs. Hence for the computation of $\mathrm{B}^T\mathrm{z}$ and $\mathrm{Q}_A^{-1}\mathrm{B}^T\mathrm{z}$ in (3.20), (3.22) we do not need significant arithmetic work. Summarizing, per outer iteration of the algorithm (3.18)–(3.23) we need $\ell+1$ evaluations of Q_A^{-1}, ℓ evaluations of Q_S^{-1}, $\ell+1$ matrix-vector multiplications with B, ℓ matrix-vector multiplications with B^T, and one matrix-vector multiplication with A.

Clearly an important issue in the algorithm is the stopping criterion for the inner iteration. From the analysis presented in section 4 it follows that for an efficient method this stopping criterion should depend on the accuracy of the preconditioner Q_A. A very efficient Poisson solver is the multigrid method. Thus for the preconditioner Q_A we propose the following:

$$\mathrm{Q}_A^{-1}\mathrm{w} = \begin{cases} \text{One symmetric multigrid V-cycle iteration, using one symmetric Gauss–Seidel iteration for pre- and post-} \\ \text{smoothing, with startvector 0 applied to } \mathrm{Ax} = \mathrm{w}. \end{cases} \tag{3.24}$$

The convergence factor per iteration of this method typically is between 0.1 and 0.2. From the analysis (cf. Remark 4) we deduce the following two possibilities for the stopping criterion of the preconditioned conjugate gradient method:

- Let r^i be the residual in the preconditioned conjugate gradient algorithm. We stop the preconditioned conjugate gradient iteration if

$$\|\mathrm{r}^\ell\|_2 \leq \sigma_{\mathrm{acc}} \|\mathrm{r}^0\|_2 \quad \text{is satisfied, with a given } \sigma_{\mathrm{acc}} \in [0.2, 0.6]. \tag{3.25}$$

- We use a fixed low number of iterations: $\ell \in [1, 4]$.

4. Analysis of the inexact Uzawa algorithm. The inexact Uzawa method has been analyzed in [2, 29]. Here we present an alternative analysis. This analysis is much simpler than those presented in [2] and [29] and the result is different. In Remark 8 we discuss the main differences to the results from [2, 29].

For simplicity we assume that for the preconditioner Q_A we have

$$\mathrm{A} \leq \mathrm{Q}_A, \quad \text{i.e., } \gamma_A \mathrm{Q}_A \leq \mathrm{A} \leq \mathrm{Q}_A. \tag{4.1}$$

We make the scaling assumption $A \leq Q_A$ because, opposite to the scaling condition in (3.3), it is fulfilled for the multigrid preconditioner in (3.24). Moreover, it improves the presentation of the analysis. However, this [illegible]

In the analysis we use the following natural norms:

$$\|x\|_Q := \|Q_A^{\frac{1}{2}} x\| = \langle Q_A x, x\rangle^{\frac{1}{2}} \quad \text{for} \quad x \in \mathbb{R}^n, \quad \|y\|_{\hat{S}} := \|\hat{S}^{\frac{1}{2}} y\| = \langle \hat{S} y, y\rangle^{\frac{1}{2}} \quad \text{for} \quad y \in e^{\perp_M}.$$

For the error in algorithm (3.18)–(3.22) we use the notation

$$e^k = \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x^k \\ y^k \end{pmatrix} =: \begin{pmatrix} e_1^k \\ e_2^k \end{pmatrix}.$$

Note that $e_2^k \in e^{\perp_M}$ holds.

LEMMA 4.1. [illegible] (3.18) [illegible]

$$\langle \hat{S}^{-1} Bw, Bw\rangle^{\frac{1}{2}} \leq (1 - \gamma_A)\|e_1^k\|_Q + \|e_2^k\|_{\hat{S}}.$$

[illegible]. For $y \in \text{range}(\hat{S}) = \text{range}(B)$ define $\|y\|_{\hat{S}^{-1}} = \langle \hat{S}^{-1} y, y\rangle^{\frac{1}{2}}$. Note that for the exact discrete solution x we have $Bx = 0$. Using this and the definition in (3.18) we get

$$Bw = Bx^k - Bx + BQ_A^{-1}(Ae_1^k + B^T e_2^k) = -B(I - Q_A^{-1}A)e_1^k + \hat{S}e_2^k.$$

Hence,

$$\begin{aligned} \langle \hat{S}^{-1} Bw, Bw\rangle^{\frac{1}{2}} &= \|Bw\|_{\hat{S}^{-1}} \leq \|B(I - Q_A^{-1}A)e_1^k\|_{\hat{S}^{-1}} + \|\hat{S}e_2^k\|_{\hat{S}^{-1}} \\ &\leq \|\hat{S}^{-\frac{1}{2}} B(I - Q_A^{-1}A)Q_A^{-\frac{1}{2}}\| \|Q_A^{\frac{1}{2}} e_1^k\| + \|\hat{S}^{\frac{1}{2}} e_2^k\|. \end{aligned}$$

Now note that

$$\|\hat{S}^{-\frac{1}{2}} B(I - Q_A^{-1}A)Q_A^{-\frac{1}{2}}\| \leq \|\hat{S}^{-\frac{1}{2}} BQ_A^{-\frac{1}{2}}\| \|Q_A^{\frac{1}{2}}(I - Q_A^{-1}A)Q_A^{-\frac{1}{2}}\|,$$

and

$$\begin{aligned} \|\hat{S}^{-\frac{1}{2}} BQ_A^{-\frac{1}{2}}\|^2 &= \rho(\hat{S}^{-\frac{1}{2}} BQ_A^{-1} B^T \hat{S}^{-\frac{1}{2}}) = \rho(I) = 1, \\ \|Q_A^{\frac{1}{2}}(I - Q_A^{-1}A)Q_A^{-\frac{1}{2}}\| &= \rho(I - Q_A^{-\frac{1}{2}} AQ_A^{-\frac{1}{2}}) \leq 1 - \gamma_A. \end{aligned}$$

This completes the proof. □

We now formulate the main convergence result.

THEOREM 4.2. [illegible] (3.18) (3.22) [illegible] Ψ [illegible] (3.15) [illegible] $e^k = (e_1^k, e_2^k)$ [illegible]

$$\|e_1^{k+1}\|_Q \leq (1 - \gamma_A)\|e_1^k\|_Q + \|e_2^{k+1}\|_{\hat{S}}, \tag{4.2}$$

$$\|e_2^{k+1}\|_{\hat{S}} \leq (1 - \gamma_A)(1 + \delta)\|e_1^k\|_Q + \delta\|e_2^k\|_{\hat{S}} \tag{4.3}$$

[illegible] γ_A [illegible] (4.1) [illegible] δ [illegible] (3.15)

[illegible]. For the error component e_1^{k+1} we have the relations

$$\begin{aligned} e_1^{k+1} &= x - x^{k+1} = x - w + Q_A^{-1} B^T z \\ &= x - x^k - Q_A^{-1}(Ae_1^k + B^T e_2^k - B^T z) \\ &= (I - Q_A^{-1}A)e_1^k - Q_A^{-1} B^T (e_2^k - z) \\ &= (I - Q_A^{-1}A)e_1^k - Q_A^{-1} B^T e_2^{k+1}. \end{aligned}$$

Hence,

$$\|e_1^{k+1}\|_Q \le \|I - Q_A^{-\frac{1}{2}} A Q_A^{-\frac{1}{2}}\| \|e_1^k\|_Q + \|Q_A^{-\frac{1}{2}} B^T \hat{S}^{-\frac{1}{2}}\| \|e_2^{k+1}\|_{\hat{S}}.$$

In combination with

$$\|I - Q_A^{-\frac{1}{2}} A Q_A^{-\frac{1}{2}}\| \le 1 - \gamma_A \ , \quad \|Q_A^{-\frac{1}{2}} B^T \hat{S}^{-\frac{1}{2}}\| = 1,$$

this proves the inequality in (4.2). For the error component e_2^{k+1} we obtain

$$e_2^{k+1} = y - y^{k+1} = e_2^k - z = (e_2^k - \hat{S}^{-1} B w) + (\hat{S}^{-1} B w - z). \tag{4.4}$$

Using $Bx = 0$ we get

$$\begin{aligned} \|e_2^k - \hat{S}^{-1} B w\|_{\hat{S}} &= \|e_2^k - \hat{S}^{-1} B(x^k + Q_A^{-1} A e_1^k + Q_A^{-1} B^T e_2^k)\|_{\hat{S}} \\ &= \|e_2^k + \hat{S}^{-1} B(I - Q_A^{-1} A) e_1^k - \hat{S}^{-1} B Q_A^{-1} B^T e_2^k\|_{\hat{S}} \\ &= \|\hat{S}^{-1} B(I - Q_A^{-1} A) e_1^k\|_{\hat{S}} \\ &\le \|\hat{S}^{-\frac{1}{2}} B Q_A^{-\frac{1}{2}}\| \|I - Q_A^{-\frac{1}{2}} A Q_A^{-\frac{1}{2}}\| \|e_1^k\|_Q \\ &\le (1 - \gamma_A) \|e_1^k\|_Q. \end{aligned} \tag{4.5}$$

Furthermore, using (3.15) and Lemma 4.1 we also have

$$\begin{aligned} \|\hat{S}^{-1} B w - z\|_{\hat{S}} &= \|\hat{S}^{-1} B w - \Psi(B w)\|_{\hat{S}} \le \delta \, \|\hat{S}^{-1} B w\|_{\hat{S}} \\ &= \delta \, \langle \hat{S}^{-1} B w, B w \rangle^{\frac{1}{2}} \le \delta \big((1 - \gamma_A) \|e_1^k\|_Q + \|e_2^k\|_{\hat{S}} \big). \end{aligned} \tag{4.6}$$

Combining the results in (4.4), (4.5), (4.6) proves the inequality in (4.3). ∎

As a simple consequence of this theorem we obtain the following convergence result.

THEOREM 4.3. [illegible]

$$\mu_A := 1 - \gamma_A, \quad g(\mu_A, \delta) := 2\mu_A + \delta(1 + \mu_A).$$

[illegible] (3.18) (3.22) [illegible] Ψ [illegible] (3.15) [illegible] $e^k = (e_1^k, e_2^k)$ [illegible]

$$\max \big\{ \|e_1^{k+1}\|_Q, \|e_2^{k+1}\|_{\hat{S}} \big\} \le g(\mu_A, \delta) \, \max \big\{ \|e_1^k\|_Q, \|e_2^k\|_{\hat{S}} \big\}, \tag{4.7}$$

$$\|e_1^k\|_Q + \|e_2^k\|_{\hat{S}} \le 3 \frac{1}{2} \Big(\frac{g(\mu_A, \delta) + \sqrt{g(\mu_A, \delta)^2 - 4\mu_A \delta}}{2} \Big)^k (\|e_1^0\|_Q + \|e_2^0\|_{\hat{S}}). \tag{4.8}$$

[illegible]. Define the matrix

$$C = \begin{pmatrix} \mu_A(2 + \delta) & \delta \\ \mu_A(1 + \delta) & \delta \end{pmatrix}.$$

Due to Theorem 4.2 we obtain

$$\begin{pmatrix} \|e_1^{k+1}\|_Q \\ \|e_2^{k+1}\|_{\hat{S}} \end{pmatrix} \le C \begin{pmatrix} \|e_1^k\|_Q \\ \|e_2^k\|_{\hat{S}} \end{pmatrix}.$$

From $\|\mathrm{C}\|_\infty = g(\mu_A, \delta)$ we obtain the result in (4.7). A simple (MAPLE) computation yields the eigenvector decomposition $\mathrm{C} = \mathrm{V}\mathrm{D}\mathrm{V}^{-1}$ with a diagonal matrix D. From this we get

$$\rho(\mathrm{C}) = \frac{1}{2}\Big(g(\mu_A,\delta) + \sqrt{g(\mu_A,\delta)^2 - 4\mu_A\delta}\Big),$$
$$\max_{0<\mu_A,\delta<1} \|\mathrm{V}\|_1 \|\mathrm{V}^{-1}\|_1 \le 3\frac{1}{2}.$$

From this the result in (4.8) follows. □

COROLLARY 4.4. [illegible] (4.7) [illegible] (4.8) [illegible] μ_A [illegible] δ [illegible] $\mu_A \to 0$ [illegible] $g(0,\delta) = \delta$ [illegible] $g(\mu_A,\delta) \ge \frac{1}{2}\big(g(\mu_A,\delta) + \sqrt{g(\mu_A,\delta)^2 - 4\mu_A\delta}\big)$

$$g(\mu_A,\delta) < 1 \quad \text{for} \quad 0 \le \delta < \frac{1-2\mu_A}{1+\mu_A}, \tag{4.9}$$

$$\frac{1}{2}\big(g(\mu_A,\delta) + \sqrt{g(\mu_A,\delta)^2 - 4\mu_A\delta}\big) < 1 \quad \text{for} \quad 0 \le \delta < 1-2\mu_A. \tag{4.10}$$

[illegible] $\mu_A < \frac{1}{2}$ [illegible] δ [illegible] (4.9), (4.10) [illegible]

4. We comment on the important special case where we take for Q_A^{-1} a symmetric multigrid V-cycle as in (3.24). Then μ_A is the contraction number (with respect to the norm $\|\cdot\|_A$) of this multigrid method. It is known from numerical experiments that typically we have $0.1 \le \mu_A \le 0.2$. The result in (4.10) shows that for $\mu_A \approx 0.15$ we have a convergent method if for the accuracy of the inner method we take $\delta \lesssim 0.7$. Clearly it is not efficient to use a very small value for δ. We comment on the choice of this parameter δ. We consider the case $\mu_A = 0.1$. For the contraction number in (4.7) we then have $g(0.1,\delta) = 0.2 + 1.1\delta$. For a given eps $\ll 1$ let k be such that

$$g(0.1,\delta)^k \approx \text{eps}.$$

Then

$$k \approx \frac{\ln(\text{eps})}{\ln(0.2 + 1.1\delta)}$$

holds. We use the ansatz (cf. (3.23)) $\delta = \beta^\ell$ with some $\beta < 1$. From Remark 3 it follows that the arithmetic costs are dominated by $k(\ell+1)$ evaluations of Q_A^{-1}. Hence, one wants to minimize

$$k(\ell+1) \approx \frac{\ln(\text{eps})}{\ln\beta}\frac{\ln\delta + \ln\beta}{\ln(0.2+1.1\delta)} =: k_{\text{Uzawa}} K(\beta,\delta),$$

where $k_{\text{Uzawa}} := \ln(\text{eps})/\ln\beta$ is the number of iterations of the exact Uzawa method ($\mu_A = 0$). For different β values the function $\delta \to K(\beta,\delta)$ is given in Figure 1. We see that an almost optimal value for δ is obtained in a very broad range. For other μ_A values (< 0.2) one observes a similar behavior.

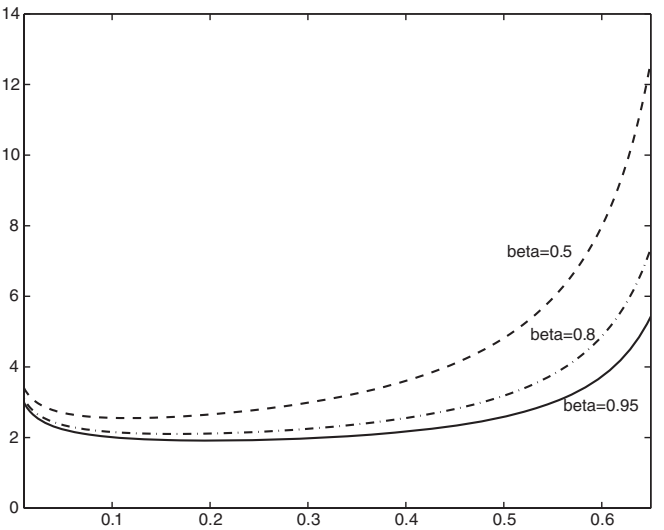


FIG. 1. *Function* $\delta \to K(\beta, \delta)$ *for* $\beta = 0.5, 0.8, 0.95$.

In the preconditioned conjugate gradient method (3.23) we use a preconditioner Q_S of $\hat{S}$. We assume that (3.2) and (3.8) hold. Using (4.1) it follows that

$$\gamma_A \gamma_S Q_S \leq \hat{S} \leq \Gamma_S Q_S \tag{4.11}$$

holds.

REMARK 5. Note that opposite to the BPCG method treated in section 3.1 and the PMINRES method treated in section 3.2, the scaling of the preconditioner Q_S does influence the rate of convergence of the inexact Uzawa method (cf. Remarks 1 and 2).

REMARK 6. If we consider the case with $\mu_A \approx 1$, $\delta \approx 1$, i.e., both the A-system and the $\hat{S}$-system are solved inaccurately, then $g_B(\mu_A, \delta) > 1$ and $g(\mu_A, \delta) > 1$. Hence, these convergence analyses do not yield convergence of the inexact Uzawa method. In our numerical experiments, however, we do observe convergence for this case; cf. Remark 9. To our knowledge, an analysis which covers the whole range μ_A, $\delta \in [0, 1)$ is not yet available.

In the convergence analysis we use the norm

$$\sqrt{\|x\|_Q^2 + \|y\|_{\hat{S}}^2} =: |||(x, y)||| \quad \text{on} \quad \mathbb{R}^n \times e^{\perp_M}. \tag{4.12}$$

REMARK 7. If we consider a stationary Stokes problem, then the norm in (4.12) is very natural. In this setting one usually takes $Q_S = M$ (pressure mass matrix as in (2.3)). Then the constants $\gamma_A, \gamma_S, \Gamma_S$ can be shown to be independent of h, and from this one obtains

$$\begin{aligned} &c_1 \big(\|\nabla J_V x\|^2 + \|J_M y\|^2 \big)^{\frac{1}{2}} \leq |||(x, y)||| \\ &\leq c_2 \big(\|\nabla J_V x\|^2 + \|J_M y\|^2 \big)^{\frac{1}{2}} \quad \text{for all } (x, y) \in \mathbb{R}^n \times e^{\perp_M} \end{aligned}$$

with constants $c_1 > 0$ and c_2 independent of h. Now note that the norm $(\mathbf{u}_h, p_h) \to \big(\|\nabla \mathbf{u}_h\|^2 + \|p_h\|^2 \big)^{\frac{1}{2}}$ is the canonical one on $\mathbf{V}_h \times M_h$.

We indicate how the norm of the error, $|||\mathrm{e}^k||| = (\|\mathrm{e}_1^k\|_Q + \|\mathrm{e}_2^k\|_S^2)^{\frac{1}{2}}$, can be estimated from quantities that are available in the inexact Uzawa algorithm (3.18)–(3.22). Let $\tilde{\mathrm{K}}$ be the symmetric positive definite block diagonal matrix as in (3.10) with Q_A such that (4.1) holds. From (4.11) we get

$$\gamma_A\gamma_S\|\mathrm{v}\|_{\tilde{\mathrm{K}}}^2 \le |||\mathrm{v}|||^2 \le \Gamma_S\|\mathrm{v}\|_{\tilde{\mathrm{K}}}^2 \quad \text{for all } \mathrm{v} \in \mathbb{R}^n \times \mathrm{e}^{\perp_M}. \tag{4.13}$$

Note that $\tilde{\mathrm{K}} : \mathbb{R}^n \times \mathrm{e}^{\perp_M} \to \mathbb{R}^n \times \mathrm{e}^{\perp}$ is bijective. From this and from Theorem 3.3 with $\Gamma_A = 1$ we obtain

$$\begin{aligned}
\|\tilde{\mathrm{K}}^{-\frac{1}{2}}\mathrm{K}\tilde{\mathrm{K}}^{-\frac{1}{2}}\mathrm{w}\| &\le \frac{1}{2}\big(1+\sqrt{1+4\Gamma_S}\big)\|\mathrm{w}\| =: c_1(\Gamma_S)\|\mathrm{w}\| \quad \text{for all } \mathrm{w} \in \mathbb{R}^n \times \mathrm{e}^{\perp},\\
\|\tilde{\mathrm{K}}^{-\frac{1}{2}}\mathrm{K}\tilde{\mathrm{K}}^{-\frac{1}{2}}\mathrm{w}\| &\ge \min\left\{\gamma_A, \frac{1}{2}\left(\sqrt{\gamma_A^2+4\gamma_S\gamma_A}-\gamma_A\right)\right\}\|\mathrm{w}\|\\
&=: c_2(\gamma_S,\gamma_A)\|\mathrm{w}\| \quad \text{for all } \mathrm{w} \in \mathbb{R}^n \times \mathrm{e}^{\perp}.
\end{aligned}$$

This yields

$$\begin{aligned}
|||\mathrm{v}|||_{\tilde{\mathrm{K}}}^2 &= \frac{\langle\tilde{\mathrm{K}}\mathrm{v},\mathrm{v}\rangle}{\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle}\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle\\
&\le c_2(\gamma_S,\gamma_A)^{-2}\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle \quad \text{for all } \mathrm{v} \in \mathbb{R}^n \times \mathrm{e}^{\perp_M},\\
|||\mathrm{v}|||_{\tilde{\mathrm{K}}}^2 &= \frac{\langle\tilde{\mathrm{K}}\mathrm{v},\mathrm{v}\rangle}{\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle}\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle \ge c_1(\Gamma_S)^{-1}\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle \quad \text{for all } \mathrm{v} \in \mathbb{R}^n \times \mathrm{e}^{\perp_M}.
\end{aligned}$$

In combination with (4.13) we then obtain

$$c_2(\gamma_S,\gamma_A)^2\Gamma_S^{-1}|||\mathrm{v}||| \le \langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{v},\mathrm{K}\mathrm{v}\rangle \le c_1(\Gamma_S)^2\gamma_A^{-1}\gamma_S^{-1}|||\mathrm{v}||| \quad \text{for all } \mathrm{v} \in \mathbb{R}^n \times \mathrm{e}^{\perp_M}.$$

Hence we can use $\langle\tilde{\mathrm{K}}^{-1}\mathrm{K}\mathrm{e}^k,\mathrm{K}\mathrm{e}^k\rangle$ as a measure for the error $|||\mathrm{e}^k|||^2$. Let $(\mathrm{r}_1^k,\mathrm{r}_2^k) = \mathrm{r}^k = \mathrm{K}\mathrm{e}^k$ be the residual. Then we have

$$|||\mathrm{e}^k|||^2 \sim \langle\tilde{\mathrm{K}}^{-1}\mathrm{r}^k,\mathrm{r}^k\rangle = \langle\mathrm{Q}_A^{-1}\mathrm{r}_1^k,\mathrm{r}_1^k\rangle + \langle\mathrm{Q}_S^{-1}\mathrm{r}_2^k,\mathrm{r}_2^k\rangle.$$

Note that $\mathrm{Q}_A^{-1}\mathrm{r}_1^k$ is available from (3.18) and $\mathrm{Q}_S^{-1}\mathrm{r}_2^k$ can be computed (or approximated accurately) with relatively low costs.

. . . 8. We discuss the main differences between our convergence result and those in [2, 29]. In [2, Theorem 10] and [29, Theorem 5.5], an inequality of the form

$$|||\mathrm{e}^{k+1}|||_* \le g_B(\mu_A,\delta)|||\mathrm{e}^k|||_*$$

is proved. The norms $|||\cdot|||_*$ used in [2] and [29] are different and also differ from the norm $|||\cdot|||$ in (4.12) that is used in our analysis. The norm $|||\cdot|||_*$ used in [29] depends on the (nonlinear) function Ψ. In case of the preconditioned conjugate gradient method as in (3.23) the norm then depends on the iteration index ℓ. In [2] a bound of the form

$$g_B(\mu_A,\delta) = \max\left\{\mu_A, \frac{2\delta}{1-\delta}\right\} \tag{4.14}$$

is proved. In [29] a more general setting is considered. However, in the special case of the inexact Uzawa method (3.18)–(3.22) the bound has a form similar to the one

in (4.14). If we compare $g_B(\mu_A, \delta)$ with our bound $g(\mu_A, \delta) = 2\mu_A + \delta(1 + \mu_A)$, then it is clear that the bounds are quite different. For $g_B(\mu_A, \delta) < 1$ one needs $\mu_A < 1$ and $\delta < \frac{1}{3}$, whereas for $g(\mu_A, \delta) < 1$ we need $\mu_A < \frac{1}{2}$ and $\delta < \frac{1-2\mu_A}{1+\mu_A}$. In the extreme case $\delta = 0$ we have $g_B(\mu_A, 0) = \frac{1}{2} g(\mu_A, 0)$, whereas if $\mu_A = 0$, we get $g(0, \delta) = \frac{1}{2}(1-\delta) g_B(0, \delta)$. We are particularly interested in the case where for Q_A we take a multigrid method. Therefore, as an example we consider $\mu_A = 0.1$. Then we have

$$\begin{aligned} g(0.1, \delta) < 1 \quad &\text{if} \quad \delta < 0.72, \\ g_B(0.1, \delta) < 1 \quad &\text{if} \quad \delta < \frac{1}{3}, \\ g(0.1, \delta) < g_B(0.1, \delta) \quad &\text{if} \quad \delta > 0.16. \end{aligned}$$

The bound $g_B(\mu_A, \delta)$ is better for "small" δ-values, whereas the bound $g(\mu_A, \delta)$ is better for "larger" δ-values. In our opinion, the bound $g(\mu_A, \delta)$ gives a better explanation of the observation (that has been made in many numerical experiments) that if one uses a very good preconditioner Q_A (such as multigrid), then even with (very) low accuracy in the inner iteration the inexact Uzawa method converges.

5. Preconditioners. In this section we discuss the choice of the preconditioners Q_A and Q_S. In particular, for the preconditioners we want the constants γ_S, Γ_S, γ_A, Γ_A in the spectral equivalences (3.2) and (3.1) to be independent of the parameters h and ξ.

It is known that if for Q_A we take a symmetric multigrid V-cycle, $\mathrm{Q}_A = \mathrm{Q}_{MG}$, then we have (cf. [6, 15, 16])

$$(1 - \mu_{MG})\mathrm{Q}_{MG} \leq \mathrm{A} \leq \mathrm{Q}_{MG}$$

with a constant $\mu_{MG} < 1$ independent of h. In [18] it is shown that for H^2-regular problems μ_{MG} is also independent of ξ. Note that if we use a scaling $\mathrm{Q}_{MG} \to \alpha \mathrm{Q}_{MG}$ with $\alpha < 1 - \mu_{MG}$, then we obtain a preconditioner that satisfies (3.3). In the numerical experiments in section 6 we will use a (scaled) symmetric multigrid V-cycle as in (3.24).

It is well known that for the case $\xi = 0$ the pressure mass matrix M is an appropriate preconditioner for S. From the discrete LBB property (2.1) and the fact that for arbitrary $\mathrm{y} \in \mathbb{R}^m$, we have (if $\xi = 0$)

$$\langle \mathrm{Sy}, \mathrm{y} \rangle = \sup_{\mathbf{u}_h \in \mathbf{V}_h} \frac{(\operatorname{div} \mathbf{u}_h, J_M \mathrm{y})^2}{\|\nabla \mathbf{u}_h\|^2} ,$$

it follows that M is uniformly (with respect to h) spectrally equivalent to S on the subspace $\mathrm{e}^{\perp_M}$:

$$\hat{\beta}^2 \mathrm{M} \leq \mathrm{S} \leq d\,\mathrm{M} \quad \text{on} \quad \mathrm{e}^{\perp_M}$$

with $\hat{\beta}$ the LBB constant from (2.1). The [illegible] mass matrix $\bar{\mathrm{M}}$ is the diagonal matrix with diagonal entries $\bar{\mathrm{M}}_{ii} = \sum_{j=1}^{m} \mathrm{M}_{ij}$. In [25] it is proved that this matrix is uniformly in h spectrally equivalent to M and thus also (for $\xi = 0$) to S on the subspace $\mathrm{e}^{\perp_M}$. A further elementary observation is

$$\bar{\mathrm{M}}\mathrm{e} = \mathrm{M}\mathrm{e}.$$

Hence, both for $\mathrm{Q}_S = \mathrm{M}$ and $\mathrm{Q}_S = \bar{\mathrm{M}}$ the consistency condition (3.8) is fulfilled.

For the case $\xi \gg 1$ the (lumped) mass matrix is not an appropriate preconditioner for the Schur complement. In [4, 8, 17] robust (with respect to h and ξ) preconditioning strategies are introduced. We briefly explain the method from [4].

For $g \in L^2(\Omega)$ consider the Neumann problem: Find $w \in H^1(\Omega) \cap M$ such that

$$(\nabla w, \nabla \phi) = (g, \phi) \quad \text{for all} \quad \phi \in H^1(\Omega) \cap M. \tag{5.1}$$

Let T_h be the stiffness matrix of the Galerkin discretization of this problem in $M_h^+ \subset H^1(\Omega)$:

$$\langle \mathrm{T}_h \mathrm{x}, \mathrm{y} \rangle = (\nabla J_M \mathrm{x}, \nabla J_M \mathrm{y}) \quad \text{for all} \quad \mathrm{x}, \mathrm{y} \in \mathbb{R}^m.$$

Note that $\ker(\mathrm{T}_h) = \operatorname{span}(\mathrm{e})$ and that $\mathrm{T}_h : \mathrm{e}^{\perp_M} \to \mathrm{e}^{\perp}$ is bijective. We define $\tilde{\mathrm{Q}}_S^{-1} : \mathrm{e}^{\perp} \to \mathrm{e}^{\perp_M}$ by

$$\tilde{\mathrm{Q}}_S^{-1} = \begin{cases} \mathrm{M}^{-1} + \xi \mathrm{T}_h^{-1} & \text{if} \quad \xi \le h^{-2}, \\ \xi h^2 \mathrm{M}^{-1} + \xi \mathrm{T}_h^{-1} & \text{if} \quad h^{-2} \le \xi. \end{cases} \tag{5.2}$$

In the analysis it is convenient to extend $\tilde{\mathrm{Q}}_S^{-1}$ on $\mathbb{R}^m$ by defining $\tilde{\mathrm{Q}}_S^{-1}\mathrm{Me} := \mathrm{e}$. In practice, however, this extension is not needed. Note that for $\xi = 0$ we get $\tilde{\mathrm{Q}}_S = \mathrm{M}$.

In [4] it is shown that $\tilde{\mathrm{Q}}_S$ is uniformly in h and ξ spectrally equivalent to S. We rephrase the main theorem (Theorem 4.1) of [4] (for the special case $S_h = W_h$ in [4]).

THEOREM 5.1. [illegible] $\xi = 0$ [illegible] $\mathbf{f} \in L_2(\Omega)^d$ [illegible] $(\mathbf{u}, p)^T$ [illegible] (1.1) [illegible] $\|\mathbf{u}\|_2 + \|p\|_1 \le C\|\mathbf{f}\|$ [illegible] C [illegible] $\mathbf{f}$ [illegible] (5.1) [illegible] H^2 [illegible] $(\mathbf{V}_h, M_h)$ [illegible] (2.1) [illegible] $M_h^+ \subset H^1(\Omega)$ $1 \in M_h^+$ [illegible] C [illegible] h

$$\inf_{\mathbf{w} \in \mathbf{V}_h} \|\mathbf{v} - \mathbf{w}\|_1 \le Ch\|\mathbf{v}\|_2 \qquad \text{[illegible]} \quad \mathbf{v} \in (H^2(\Omega) \cap H_0^1(\Omega))^d,$$

$$\inf_{q \in M_h^+} \|p - q\| \le Ch\|p\|_1 \qquad \text{[illegible]} \quad p \in H^1(\Omega),$$

$$\|\mathbf{v}\|_s \le Ch^{-1}\|\mathbf{v}\|_{s-1} \qquad \text{[illegible]} \quad \mathbf{v} \in \mathbf{V}_h,\ s \in \{0, 1\},$$

$$\|p\| \le Ch^{-1}\|p\|_{-1} \qquad \text{[illegible]} \quad p \in M_h.$$

[illegible] $C_0 > 0$ C_1 [illegible] h [illegible] ξ [illegible]

$$C_0 \langle y, \tilde{\mathrm{Q}}_S^{-1} y \rangle \le \langle \mathrm{S}\tilde{\mathrm{Q}}_S^{-1} y, \tilde{\mathrm{Q}}_S^{-1} y \rangle \le C_1 \langle y, \tilde{\mathrm{Q}}_S^{-1} y \rangle \quad \text{[illegible]} \quad y \in \mathrm{e}^{\perp}. \tag{5.3}$$

Note that many pairs of finite element spaces satisfy the assumptions in this theorem. In our experiments we use such a pair, namely, the Hood–Taylor $P_2 - P_1$ pair.

Although the Laplace problem for the pressure that has to be solved in each application of $\tilde{\mathrm{Q}}_S^{-1}$ is much smaller than the problems involving A or Q_A, often it is inefficient to solve it exactly. Let Q_T be a preconditioner for T_h that is spectrally equivalent to T_h, uniformly in h. Instead of $\tilde{\mathrm{Q}}_S^{-1}$ in (5.2) one can then use

$$\mathrm{Q}_S^{-1} := \begin{cases} \mathrm{M}^{-1} + \xi \mathrm{Q}_T^{-1} & \text{if} \quad \xi \le h^{-2}, \\ \xi h^2 \mathrm{M}^{-1} + \xi \mathrm{Q}_T^{-1} & \text{if} \quad h^{-2} \le \xi. \end{cases} \tag{5.4}$$

In [4] (Remark 4.1) it is shown that for this preconditioner a result as in (5.3) holds with constants $C_0 > 0$ and C_1 independent of h and ξ. Hence, the preconditioner Q_S as in (5.4) is spectrally equivalent to S (on $\mathrm{e}^{\perp_M}$) uniformly in h and ξ.

In the numerical experiments in section 6 for Q_T we use a symmetric multigrid V-cycle method.

6. Numerical experiments. In this section we present a comparison of the three iterative solvers discussed above. These solvers are denoted by BPCG (algorithm (3.7)), PMINRES (section 3.2), and MGUZAWA (algorithm (3.18)–(3.23)). We consider a problem as in (1.1) on the unit cube $\Omega = (0,1)^3$. For the discretization we start with a uniform tetrahedral grid with $h = \frac{1}{2}$, and we apply regular refinements to this starting triangulation. For the finite element discretization we used the LBB stable pair of Hood–Taylor $P_2 - P_1$, i.e., continuous piecewise quadratics for the velocity and continuous piecewise linears for the pressure. We performed computations for the cases $h = 1/16$, $h = 1/32$. Note that for $h = 1/32$ we have approximately $7.5{\cdot}10^5$ velocity unknowns and $3.3{\cdot}10^4$ pressure unknowns ($n \approx 7.5{\cdot}10^5$, $m \approx 3.3{\cdot}10^4$). We consider the linear system as in (2.4) with solution $(\mathrm{x}, \mathrm{y}) = 0$. We take a fixed arbitrary starting vector $(\mathrm{x}^0, \mathrm{y}^0)$, with $\mathrm{y}^0 \in \mathrm{e}^{\perp_M}$. The iteration is stopped after a reduction of the Euclidean norm of the starting residual by a factor 10^6, i.e., $\|\mathrm{r}^k\| \leq 10^{-6}\|\mathrm{r}^0\|$, with $\mathrm{r}^k = (\mathrm{f}, 0) - \mathrm{K}(\mathrm{x}^k, \mathrm{y}^k)$.

Experiments for the case $\xi = 0$. We apply the BPCG, PMINRES, and MGUZAWA methods described above. For the preconditioner Q_A we use one multigrid V-cycle iteration as in (3.24). For the preconditioner Q_S for the Schur complement we use one multigrid V-cycle iteration as in (3.24), but now applied to the mass matrix M.

For the BPCG method we need a suitable scaling of Q_A. To determine the scaling parameter, we compute $\tilde{\lambda} \approx \lambda\max(I - \mathrm{QMG}^{-1}\mathrm{A})$ by a few power-iterations and use $\mathrm{Q}_A := (1 - \alpha\,\tilde{\lambda})\mathrm{Q}_{MG}$, $\alpha \in [1, \tilde{\lambda}^{-1})$, as scaled preconditioner. The approximation $\tilde{\lambda}$ satisfies $\tilde{\lambda} \leq \lambda\max(I - \mathrm{QMG}^{-1}\mathrm{A})$, and hence we take $\alpha \geq 1$. In our experiments we use $\alpha = 1.1$. Although not negligible, the arithmetic costs for determining this scaling parameter are not taken into account in the tables presented below. The PMINRES method is parameter-free. In the MGUZAWA method we have to choose a stopping criterion for the inner iteration. We use the one in (3.25) with $\sigma_{\mathrm{acc}} = 0.5$. For all three methods the costs are dominated by the evaluations of Q_A^{-1}. Therefore to measure the arithmetic costs we count the number of Q_A^{-1} evaluations, which is denoted by $\#\mathrm{Q}_{MG}$. Results are presented in Table 1.

TABLE 1
$\#\mathrm{Q}_{MG}$ for the three methods.

	$h = 1/16$	$h = 1/32$
BPCG	29	29
PMINRES	49	49
MGUZAWA	33	30

In Figure 2 we give the convergence behavior of $\|\mathrm{r}^k\|$ for the case $h = 1/32$. Note that in the MGUZAWA method, in each iteration we have an inner iteration. In Table 2 we give the number of inner iterations per outer iteration.

The behavior of the BPCG method depends on the scaling parameter α in $(1 - \alpha\,\tilde{\lambda})\mathrm{Q}_{MG}$. In Table 3 we show the dependence of $\#\mathrm{Q}_{MG}$ on this scaling parameter for $\tilde{\lambda} = \lambda\max(I - \mathrm{Q}_{MG}^{-1}\mathrm{A})$. The convergence of the MGUZAWA method

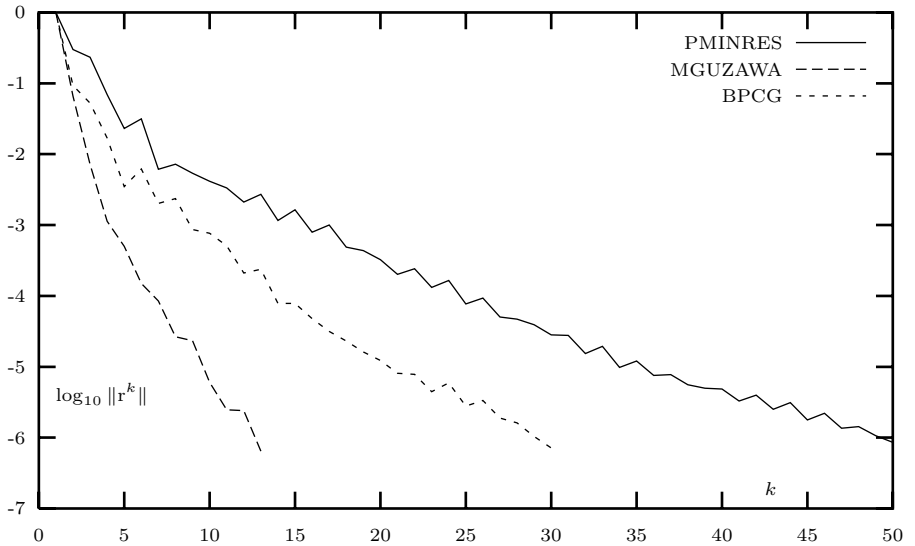


FIG. 2. *Convergence behavior of* $\|r^k\|$ *for the case* $h = 1/32$.

TABLE 2
Number of inner iterations in MGUZAWA; $h = 1/32$.

k	1	2	3	4	5	6	7	8	9	10	11	12
# inner it.	1	1	1	2	1	2	1	2	1	1	3	2

TABLE 3
Dependence of BPCG method on scaling parameter; $h = 1/32$.

α	0.7	0.8	0.9	1.0	1.1	1.2	1.3	1.5	2.0
$\#Q_{MG}$	div	div	28	28	29	30	31	32	36

TABLE 4
Dependence of MGUZAWA method on σ_{acc}*;* $h = 1/32$.

σ_{acc}	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$\#Q_{MG}$	38	33	34	30	30	29	30	30	52

TABLE 5
Dependence of $\#Q_{MG}$ *on the scaling of* Q_S*;* $h = 1/32$.

ρ	10^{-4}	10^{-2}	1.0	10^2	10^4
BPCG	110	58	29	44	45
PMINRES	135	91	49	30	40
MGUZAWA	30	30	30	30	30

depends on σ_{acc}. In Table 4 we show the dependence of $\#Q_{MG}$ on this parameter.

According to Remarks 1, 2, 5 the bounds for the convergence rates of BPCG and PMINRES depend on the scaling parameter ρ if we use ρQ_S ($\rho > 0$) as a preconditioner for the Schur complement. The convergence rate of MGUZAWA does not depend on the scaling of Q_S. Table 5 shows $\#Q_{MG}$ for different values of ρ.

. . .. 9. We also performed experiments with the MGUZAWA method with

TABLE 6
#Q_{MG} for the three methods; $\xi = h^{-1}$.

	$h = 1/16$	$h = 1/32$
BPCG	29	28
PMINRES	48	48
MGUZAWA	26	29

TABLE 7
#Q_{MG} for the three methods; $\xi = h^{-2}$.

	$h = 1/16$	$h = 1/32$
BPCG	26	24
PMINRES	44	41
MGUZAWA	27	25

inaccurate approximations of the A-block and of the $\hat{S}$-system (cf. Remark 6). For Q_A we use one SSOR iteration and we apply only one preconditioned conjugate gradient iteration (with the same preconditioner as above) for solving the $\hat{S}$-system. It turns out that for both $h = 1/16$ and $h = 1/32$ the MGUZAWA method converges. For the case $h = 1/16$ we need 417 (outer) iterations to satisfy the stopping criterion and for $h = 1/32$ we need 1485 iterations.

Experiments for the case $\xi > 0$. We again apply the three methods described above. For the preconditioner Q_A we use the same multigrid V-cycle iteration as for the case $\xi = 0$. For the preconditioner for the Schur complement we use the one in (5.4). For Q_T we apply one multigrid V-cycle iteration with one symmetric Gauss–Seidel pre- and postsmoothing iteration. For the BPCG method we take the same scaling of Q_A as we used for the case $\xi = 0$. Note that this may not be the optimal scaling if $\xi > 0$. In the MGUZAWA method we take $\sigma_{\text{acc}} = 0.6$. In Tables 6 and 7 we present results for $\xi = h^{-1}$ and $\xi = h^{-2}$.

7. Concluding remarks. In this paper we considered three fast iterative methods for discretized Stokes equations. The BPCG method is a preconditioned conjugate gradient method applied to a transformed system and with a special scalar product. The PMINRES method is a standard preconditioned minimal residual method. The MGUZAWA method results from an approximation of an exact block factorization of the matrix K (cf. (3.16), (3.17)). The MGUZAWA algorithm has the structure of an outer-inner iteration. Thus for this method a stopping criterion (tolerance parameter) for the inner iteration is needed. The PMINRES method is parameter-free. In the BPCG method a scaling parameter for the preconditioner Q_A has to be determined. If for Q_A we use a multigrid solver, then for all three methods the costs per iteration are dominated by the evaluations of Q_A^{-1}. In the BPCG and the PMINRES methods one needs one Q_A^{-1} evaluation per iteration. In the MGUZAWA method $\ell + 1$ Q_A^{-1} evaluations per iteration are needed (ℓ: # inner iterations). For all three methods convergence analyses are available in which bounds for the rate of convergence are derived that depend only on the constants in the spectral equivalences $\gamma_A Q_A \le A \le \Gamma_A Q_A$, $\gamma_S Q_S \le S \le \Gamma_S Q_S$. For the MGUZAWA method we presented a convergence analysis that shows that if one uses a multigrid preconditioner for A, then for the inner iteration a very low accuracy is already sufficient to have a convergent and efficient method. In this convergence analysis we use a natural norm.

In the numerical experiments for the stationary case, all three methods have a rate of convergence independent of h. In our experiments the BPCG and MGUZAWA

methods have almost the same efficiency. The PMINRES method is less efficient. The latter method, however, is the most robust one. It is parameter-free and converges even without preconditioning. From Table 3 we conclude that the performance of the BPCG method depends only mildly on α (provided $\alpha \geq 1$). Thus, we need only a rough approximation $\tilde{\lambda} \approx \lambda\max(I - \mathrm{Q}_A^{-1}\mathrm{A})$ to obtain an efficient solver. The results in Table 4 show that the MGUZAWA method is efficient for a broad range of σacc-values. Note that the method converges even for large σ_{acc} values. This confirms the theory in section 4.

The rate of convergence of both BPCG and PMINRES depend on the scaling of Q_S, whereas for the MGUZAWA method this is not the case. For particular choices of this scaling it may happen that the PMINRES method is more efficient than the BPCG method (cf. Table 5).

The experiments for the nonstationary case demonstrate the effectiveness of the preconditioner in (5.4). The rate of convergence is independent of h for $\xi = h^{-1}$ and $\xi = h^{-2}$. If instead of the preconditioner in (5.4) one uses $\mathrm{Q}_S = \bar{\mathrm{M}}$, for $\xi = h^{-2}$ all three solvers would need several hundred iterations.

REFERENCES

[1] K. Arrow, L. Hurwicz, and H. Uzawa, *Studies in Nonlinear Programming*, Stanford University Press, Stanford, CA, 1958.

[2] R. E. Bank, B. D. Welfert, and H. Yserentant, *A class of iterative methods for solving saddle point problems*, Numer. Math., 56 (1990), pp. 645–666.

[3] J. H. Bramble and J. E. Pasciak, *A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems*, Math. Comp., 50 (1988), pp. 1–17.

[4] J. H. Bramble and J. E. Pasciak, *Iterative techniques for time dependent Stokes problems*, Comput. Math. Appl., 33 (1997), pp. 13–30.

[5] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev, *Analysis of the inexact Uzawa algorithm for saddle point problems*, SIAM J. Numer. Anal., 34 (1997), pp. 1072–1092.

[6] J. H. Bramble, *Multigrid Methods*, Pitman Res. Notes Math. Ser. 294, John Wiley, New York, 1993.

[7] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer, New York, 1991.

[8] J. Cahouet and J. P. Chabard, *Some fast 3D finite element solvers for the generalized Stokes problem*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 869–895.

[9] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, North-Holland, Amsterdam, 1978; Classics Appl. Math. 40, SIAM, Philadelphia, 2002 (reprint).

[10] P. G. Ciarlet, *Basic Error Estimates for Elliptic Problems*, in Handbook of Numerical Analysis, Vol. II, P. G. Ciarlet and J. L. Lions, eds., North-Holland, Amsterdam, 1991, pp. 17–351.

[11] H. C. Elman, *Multigrid and Krylov subspace methods for the discrete Stokes equations*, Internat. J. Numer. Methods Fluids, 227 (1996), pp. 755–770.

[12] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[13] V. Girault and P.-A. Raviart, *Finite Element Methods for Navier–Stokes Equations*, Springer, Berlin, 1986.

[14] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[15] W. Hackbusch, *Multigrid Methods and Applications*, Springer, Berlin, Heidelberg, 1985.

[16] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer, Berlin, Heidelberg, 1994.

[17] G. M. Kobelkov and M. A. Olshanskii, *Effective preconditioning of Uzawa type schemes for a generalized Stokes problem*, Numer. Math., 86 (2000), pp. 443–470.

[18] M. A. Olshanskii and A. Reusken, *On the convergence of a multigrid algorithm for linear reaction-diffusion problems*, Computing, 65 (2000), pp. 193–202.

[19] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[20] C. Powell and D. Silvester, *Black-box preconditioning for mixed formulation of self-adjoint elliptic PDEs*, in Challenges in Scientific Computing–CISC 2002, E. Bänsch, ed., Lecture Notes Comput. Sci. Engrg. 35, Springer, Berlin, 2003, pp. 268–285.

[21] T. Rusten and R. Winther, *A preconditioned iterative method for saddlepoint problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 887–904.

[22] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, J. Comput. Appl. Math., 128 (2001), pp. 261–279.

[23] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems. Part* II. *Using general block preconditioners*, SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

[24] R. Verfürth, *A combined conjugate gradient-multigrid algorithm for the numerical solution of the Stokes problem*, IMA J. Numer. Anal., 4 (1984), pp. 441–455.

[25] A. Wathen, *Realistic eigenvalues bounds for the Galerkin mass matrix*, IMA J. Numer. Anal., 7 (1987), pp. 449–457.

[26] A. Wathen and D. Silvester, *Fast iterative solution of stabilised Stokes systems. Part* I. *Using simple diagonal preconditioners*, SIAM J. Numer. Anal., 30 (1993), pp. 630–649.

[27] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley, Chichester, 1992.

[28] G. Wittum, *Multigrid methods for Stokes and Navier–Stokes equations. Transforming smoothers: Algorithms and numerical results*, Numer. Math., 54 (1989), pp. 543–563.

[29] W. Zulehner, *Analysis of iterative methods for saddle point problems: A unified approach*, Math. Comp., 71 (2001), pp. 479–505.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 667–686
© 2005 Society for Industrial and Applied Mathematics

AN IMPLICIT WAVELET SPARSE APPROXIMATE INVERSE PRECONDITIONER*

STUART C. HAWKINS† AND KE CHEN‡

Abstract. Wavelet-based sparse approximate inverse preconditioners are considered for the linear system $Ax = b$. The preconditioners are good sparse approximations to the inverse of A computed by taking advantage of the compression obtained by working in a wavelet basis. When the representation of A in a single scale basis (for example, a finite element basis) is available, the formulation presented obviates computation of the representation of A in the wavelet basis and removes the associated costs. Efficient application for both sparse and dense A is considered.

Key words. linear system, preconditioning, sparse approximate inverse, wavelet

AMS subject classifications. 65F10, 65F50, 65R20

DOI. 10.1137/S1064827503423500

1. Introduction.

We consider the fast solution of the linear system

$$Ax = b, \tag{1.1}$$

where A is a large $n \times n$ matrix. Such linear systems arise in finite element or finite difference discretizations, where typically A is sparse, and in boundary element discretizations, where typically A is dense. There is much interest in techniques for iteratively solving (1.1) using Krylov subspace methods such as GMRES [29, 30].

To improve performance of the iterative solver when A is ill conditioned (1.1) may be replaced by the left preconditioned linear system

$$MAx = Mb \tag{1.2}$$

or the right preconditioned linear system

$$AMy = b. \tag{1.3}$$

To improve performance of the iterative solver we desire $MA \approx I$ in (1.2) or $AM \approx I$ in (1.3). To ensure that matrix vector multiplication with the preconditioner is cheap we require that M be sparse.

Sparse approximate inverse techniques are widely used to produce preconditioners M. Sparse approximate inverses can be produced by iterative techniques, as described by Chow and Saad [13] and Benzi and Tůma [5], but we consider direct techniques based on direct solution of (reduced) least squares problems, as described by Benson and Frederickson [3], Kolotilina and Yeremin [24], Grote and Huckle [22], and Benzi

*Received by the editors March 3, 2003; accepted for publication (in revised form) August 8, 2004; published electronically November 15, 2005. This work was supported by EPSRC research grant GR/R22315/01.

http://www.siam.org/journals/sisc/27-2/42350.html

†Department of Mathematical Sciences, University of Liverpool, Liverpool, UK. Current address: School of Mathematics, The University of New South Wales, Sydney, NSW 2052, Australia (stuart @maths.unsw.edu.au).

‡Corresponding author. Department of Mathematical Sciences, University of Liverpool, Liverpool, UK (k.chen@liverpool.ac.uk, http://www.liv.ac.uk/~cmchenke).

and Tůma [5]. We present our work in the context of right preconditioning, where the aim is to minimize

$$\|AM - I\|_F \tag{1.4}$$

subject to a prescribed sparsity pattern for M. A sparse approximate inverse for left preconditioning is computed by minimizing $\|MA - I\|_F$. However, $\|MA - I\|_F = \|A^T M^T - I\|_F$ and so the theory and application is the same as for right preconditioning, though it looks more complicated.

To minimize $\|AM - I\|_F$ note that $\|AM - I\|_F^2 = \sum_{j=1}^{n} \|Am_j - e_j\|_2^2$, where m_j is the jth column of M. Thus minimizing $\|AM - I\|_F$ reduces to solving n independent least squares problems. This is easy to implement in parallel.

Good approximations to A^{-1} can be obtained from sparse approximate inverse techniques when A^{-1} is sparse, but not when A^{-1} is dense. Sparse matrices need not have a sparse inverse. For example, consider the inverse of the matrix

$$A = n^2 \begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}$$

obtained by finite difference discretization of the Laplacian. It can be seen in Figure 1.1(a) that A^{-1} is dense.

Chen [10] and Carpentieri, Duff, and Giraud [7] discuss sparse approximate inverse techniques for dense matrices arising from boundary element discretizations. The idea is to split $A = S + F$, where S is sparse and F is dense. Then A can be preconditioned with a sparse approximate inverse of S. The choice of S is crucial here.

When A is a discretization of a pseudodifferential operator one can view A^{-1} as the discretization of an integral operator whose kernel is the Green's function associated with A. A^{-1} is typically dense but representation in a wavelet basis compresses

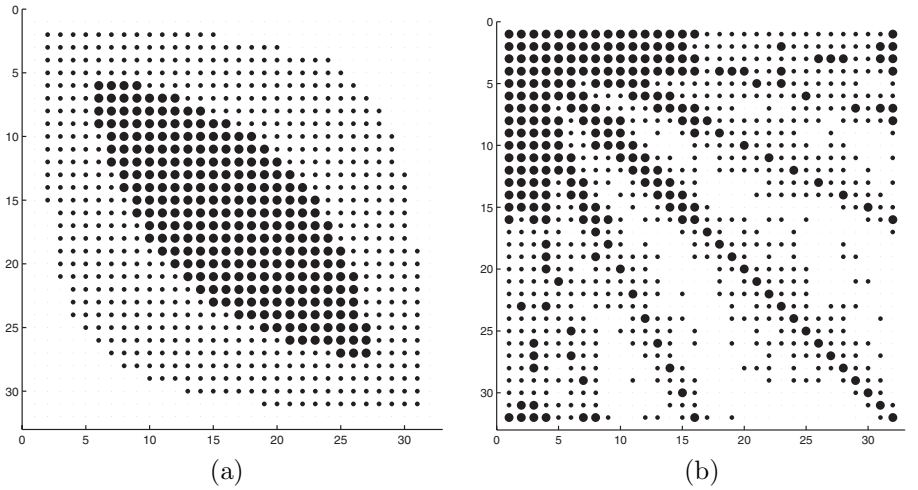


FIG. 1.1. (a) *Visualization of the inverse of a discretization of the Laplacian with* $n = 32$. (b) *Visualization of the same matrix represented in the Daubechies* 4 *wavelet basis. Radius of dots is based on matrix entry magnitudes.*

A^{-1} when the Green's function is smooth. If, additionally, the Green's function is smooth away from the diagonal, then A^{-1} has a special [illegible] in the wavelet basis. Off-diagonal fingers result from interactions between wavelets at different scales. This pattern can be seen in Figure 1.1(b), which is a visualization of the matrix in Figure 1.1(a) represented in the Daubechies 4 [17] wavelet basis.

The compression of the matrix A^{-1} in the wavelet basis allows it to be well approximated by a sparse matrix. Furthermore, the finger pattern of the large entries is predictable. Formally, denote representation in the wavelet basis with $\widetilde{\cdot}$ so that, in the wavelet basis, (1.1) becomes

$$\widetilde{A}\widetilde{x} = \widetilde{b}. \tag{1.5}$$

One can approximate $\widetilde{A^{-1}}$ with a finger-patterned sparse approximate inverse of $\widetilde{A}$ because $\widetilde{A^{-1}} = \widetilde{A}^{-1}$.

Chan, Tang, and Wan [9] use this sparse approximate inverse of $\widetilde{A}$ as a preconditioner for (1.5) but restrict the pattern of the sparse approximate inverse to block diagonal. This simplifies the computation, but interactions between wavelets at different scales are omitted in the calculation of the preconditioner. This is in line with the theory of Dahmen and Kunoth [16], who show that diagonal preconditioning yields a condition number that is independent of the number of unknowns.

However, as observed by Cohen and Masson [15], sometimes diagonal preconditioning is not sufficient for efficient iterative solution. Cohen and Masson [15] compute finger-patterned sparse approximate inverses for (1.5) in a slightly different context where (1.5) is obtained directly from the underlying partial differential equation (PDE) or boundary integral equation (BIE) with discretization using wavelet basis functions. Use of the full finger pattern for the sparse approximate inverse means that all interactions between scales are included in this preconditioner. This makes the preconditioner more robust but can sometimes make the computation of the sparse approximate inverse prohibitively expensive because a small number of dense columns or rows can be present in the finger pattern.

Bridson and Tang [6] compute a sparse approximate inverse of $\widetilde{A}$ where the sparsity pattern of the approximate inverse is determined adaptively. The adaptive sparse approximate inverse algorithm does not require the entries of $\widetilde{A}$ to be available, and allows second generation wavelets to be used efficiently. However, Chow [12] notes that adaptive sparse approximate inverse algorithms can be more expensive to implement than sparse approximate inverse algorithms for which the pattern of the approximate inverse is known in advance.

We remark that, outside the sparse approximate inverse framework, Chan and Chen [8] proposed a recursive preconditioner that does not require $\widetilde{A}$ to be computed.

We present a new preconditioner that takes advantage of the structure possessed by A^{-1} when it is represented in the wavelet basis. Like Cohen and Masson [15] we incorporate all of the interactions between wavelets at different scales, which is sometimes necessary to produce a robust preconditioner. We describe strategies that overcome the computational difficulties presented by the subsequent sparsity pattern of the preconditioner. Many of our computations involve A rather than $\widetilde{A}$ because our formulation applies the discrete wavelet transform implicitly. When A is sparse, it is often sparser than $\widetilde{A}$, and working with A rather than $\widetilde{A}$ reduces the cost of computing the sparse approximate inverse. The cost of computing $\widetilde{A}$ is also removed.

A typical discrete wavelet transform is applied to the vector $x \in \mathbb{R}^n$ as follows. For simplicity we assume $n = 2^N$ for some $N \in \mathbb{N}$, but the theory in this paper applies

whenever $n = s^N p$ for some $N, s, p \in \mathbb{N}$. Let $h_0, h_1, \ldots, h_{m-1}$ and $g_0, g_1, \ldots, g_{m-1}$ be the low-pass and high-pass filter coefficients of the transform. Then the level $L \leq N$ discrete wavelet transform of x is the vector $\widetilde{x} = (s^L, d^L, d^{L-1}, \ldots, d^1)$, where

$$s^0 = x, \tag{1.6}$$

$$s_j^k = \sum_{l=0}^{m-1} h_l s^{k-1}_{\langle l+2j-1\rangle_{n/2^{k-1}}}, \quad j = 1, 2, \ldots, n/2^k, \tag{1.7}$$

$$d_j^k = \sum_{l=0}^{m-1} g_l s^{k-1}_{\langle l+2j-1\rangle_{n/2^{k-1}}}, \quad j = 1, 2, \ldots, n/2^k, \tag{1.8}$$

where $\langle a\rangle_b$ denotes $c+1$, c being the remainder of $a-1$, modulo b. For example, $\langle 256\rangle_{256} = 256$ and $\langle 257\rangle_{256} = 1$. When $L = N$ we say that $\widetilde{x}$ is fully transformed. The discrete wavelet transform of a matrix is obtained by transforming each row of the matrix, and then transforming each column of the result.

Equation (1.7) can be rewritten as

$$s^k = U_k s^{k-1},$$

where

$$U_k = \begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{m-1} & & & \\ & & h_0 & h_1 & h_2 & \cdots & h_{m-1} & \\ & & & \vdots & & & & \\ h_2 & \cdots & h_{m-1} & & & & h_0 & h_1 \end{bmatrix}$$

is an $n/2^k \times n/2^{k-1}$ matrix. In the same way (1.8) can be rewritten as

$$d^k = V_k s^{k-1},$$

where

$$V_k = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{m-1} & & & \\ & & g_0 & g_1 & g_2 & \cdots & g_{m-1} & \\ & & & \vdots & & & & \\ g_2 & \cdots & g_{m-1} & & & & g_0 & g_1 \end{bmatrix}$$

is an $n/2^k \times n/2^{k-1}$ matrix. When $m > n/2^{k-1}$ the entries wrap around, for example, when $m = 4$ the 1×2 level N matrices are

$$U_N = \begin{bmatrix} h_0 + h_2 & h_1 + h_3 \end{bmatrix},$$
$$V_N = \begin{bmatrix} g_0 + g_2 & g_1 + g_3 \end{bmatrix}.$$

The level L discrete wavelet transform can be written as

$$\widetilde{x} = W^T x = W_L^T W_{L-1}^T \cdots W_1^T x \tag{1.9}$$

with

$$W_k^T = \left[\begin{array}{c|c} \begin{matrix} U_k \\ V_k \end{matrix} & 0 \\ \hline 0 & I_{n-n/2^{k-1}} \end{array}\right].$$

The paper is structured as follows. In section 2 we review the wavelet sparse approximate inverse preconditioner of Chan, Tang, and Wan [9]. In section 3 we introduce our new preconditioner that avoids transforming A. In section 4 we discuss how to implement our new algorithm and consider how to compute the entries of the wavelet basis vectors, which are required to implement the new method. We end the section with some comments on the relative costs of computing the Chan, Tang, and Wan [9] preconditioner and the new preconditioner. In section 5 we consider modifications to these preconditioners for the case when A is dense. In section 6 we present some theory on the conditioning of the new preconditioned iteration matrix. In section 7 we present some numerical experiments. We finish with some conclusions in section 8.

2. Wavelet-based sparse approximate inverse. Cohen and Masson [15] show that, for a wide class of PDE problems, A^{-1} is sparse in the wavelet basis. Noting this we review the preconditioning method of Chan, Tang, and Wan [9].

Let W be as in (1.9). Since $\widetilde{A^{-1}} = W^T A^{-1} W = (W^T A W)^{-1} = \widetilde{A}^{-1}$, Chan, Tang, and Wan [9] propose a sparse approximate inverse preconditioner $\widetilde{M}$ for $\widetilde{A}$ computed by solving a minimization problem analogous to (1.4), that is, by minimizing $\|\widetilde{A}\widetilde{M} - I\|_F$. Cohen and Masson [15] established that, for a wide class of pseudodifferential equations, both $\widetilde{A}$ and $\widetilde{A}^{-1}$ have a finger pattern. Thus a sparse approximate inverse of $\widetilde{A}$ is justified and a finger pattern for $\widetilde{M}$ can usually be prescribed. The Chan, Tang, and Wan [9] algorithm is described by Algorithm 1.

ALGORITHM 1.

1. Compute $\widetilde{A} = W^T A W$, $\widetilde{b} = W^T b$.
2. Compute $\widetilde{M}$, a right sparse approximate inverse of $\widetilde{A}$.
3. Solve $\widetilde{A}\,\widetilde{M}\,\widetilde{y} = \widetilde{b}$.
4. Compute $x = W\widetilde{M}\widetilde{y}$.

Step 1 involves computing $\widetilde{A}$, an operation that has complexity $O(n)$, depending on the level of the transform, the method of applying the transform, and the sparsity of A. The computation of $\widetilde{A}$ is a significant part of the total cost of implementing Algorithm 1.

When A is sparse, the matrix vector multiplications with $\widetilde{A}$ involved in step 3 should be applied in factored form with $W^T A W$ rather than with $\widetilde{A}$ itself.

Finger-patterned matrices, though sparse, can have a small number of dense columns and rows. As noted by Gould and Scott [20] the presence of dense columns in $\widetilde{M}$ require that care be taken in its computation. Reducing the sparsity pattern of $\widetilde{M}$ to block diagonal, as described by Chan, Tang, and Wan [9], is one way to solve this problem but relinquishes robustness because interactions between scales are disregarded. In this paper we present a new and cheaper way of using wavelets to give a sparse preconditioner with a predictable pattern but without computing $\widetilde{A}$. Interactions between wavelets at different scales are included.

3. A new implicit wavelet sparse approximate inverse preconditioner. We describe a new wavelet sparse approximate inverse preconditioner that is closely related to the preconditioner in Algorithm 1 but the requirement that $\widetilde{A}$ be available is removed and the wavelet transform is applied implicitly. This is facilitated by changing the way in which the sparse approximate inverse is computed. Additionally we use the whole finger pattern for the preconditioner to ensure that all interactions between scales are included.

Observe that

$$\begin{aligned}\|\widetilde{A}\widetilde{M} - I\|_F &= \|W^T AWW^T MW - I\|_F \\ &= \|W^T AMW - I\|_F \\ &= \|WW^T AMW - W\|_F \\ &= \|A\widehat{M} - W\|_F,\end{aligned}$$

where $\widetilde{M} = W^T MW$ and $\widehat{M} = MW = W\widetilde{M}$. It is not difficult to see that $\widehat{M}$ has a predictable sparsity pattern, because $\widetilde{M}$ has a predictable sparsity pattern. Minimizing $\|A\widehat{M} - W\|_F$ reduces to n independent least squares problems in the same way as minimizing $\|AM - I\|_F$. These observations lead to a new algorithm.

ALGORITHM 2.

1. Compute $\widehat{M}$, the minimizer of $\|A\widehat{M} - W\|_F$ subject to the prescribed sparsity pattern for $\widehat{M}$.
2. Solve $W^T A\widehat{M}\widetilde{y} = W^T b$.
3. Compute $x = \widehat{M}\widetilde{y}$.

$\widehat{M}$ is an approximation to A^{-1} represented from the wavelet basis to the standard, single scale basis. We establish in Theorem 6.1 that this matrix possesses a band pattern like that shown in Figure 3.1(b) and which is analogous to the finger pattern shown in Figure 3.1(a). In fact W possesses such a band pattern, and our practical experience has shown that the pattern of W provides a suitable pattern for $\widehat{M}$.

In fact we have implicitly applied a one-sided wavelet transform to M, compared with the two-sided wavelet transform of M in the Chan, Tang, and Wan [9] algorithm. In this way we achieve compression and structure in A^{-1} and like the Cohen and Masson [15] preconditioner, our preconditioner includes interactions between different scales. The one-sided application of the wavelet transform provides less compression than the two-sided application but our practical experience is that the compression achieved is sufficient.

In general the preconditioner $\widehat{M}$ is unsymmetric, even when A is symmetric. Symmetrization techniques such as the one presented for Algorithm 1 by Chan, Tang, and Wan [9] are precluded because the nonsymmetric pattern of $\widehat{M}$ is dictated by the decay of the inverse of A and the properties of the wavelet basis. Although in

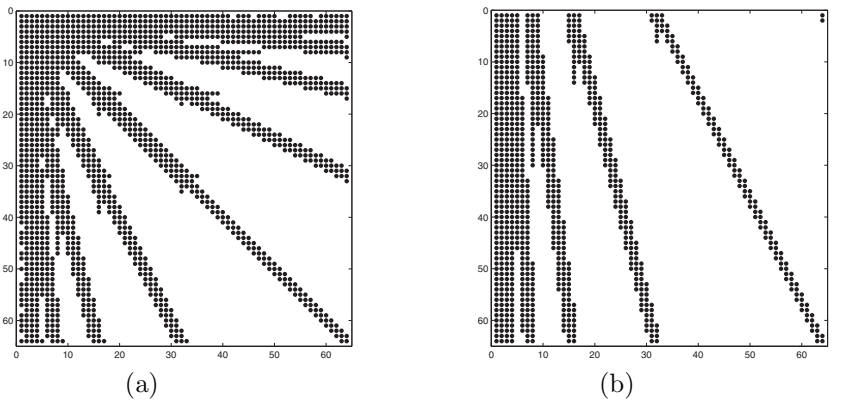


FIG. 3.1. *Visualizations of* (a) *a finger-patterned matrix and* (b) *a one-sided finger-patterned matrix.*

general it is more expensive to solve unsymmetric systems than symmetric systems, use of this new strategy for symmetric systems is justified because construction of the preconditioner in Algorithm 2 is cheaper than construction of the preconditioner in Algorithm 1.

4. Implementation details. We now discuss how to solve in practice the least squares problem in Algorithm 2. Analysis of the least squares problem reveals two important considerations for practical implementation of the algorithm. We show how to address these considerations by slightly modifying how the preconditioner is constructed, and by finding cheaper ways of computing the wavelet matrix W. We finish with some comments about the complexity of computing the matrices $\widetilde{A}$ and W.

Consider the problem of minimizing $\|A\widehat{M} - W\|_F$ at step 1 of Algorithm 2. Let S_j be the set of indices of the nonzero entries of $\widehat{m}_j$ where $\widehat{m}_j$ is the jth column of $\widehat{M}$. The classical approach to solving the minimization problem is to observe that

$$\|A\widehat{m}_j - w_j\|_2 = \|A(:, S_j)\widehat{m}_j(S_j) - w_j\|_2$$

and

$$\operatorname{argmin} \|A(:, S_j)\widehat{m}_j(S_j) - w_j\|_2 = \operatorname{argmin} \|A(T_j, S_j)\widehat{m}_j(S_j) - w_j(T_j)\|_2,$$

where T_j indexes the nonzero rows of $A(:, S_j)$ and w_j is the jth column of W. With the reduced QR factorization $A(T_j, S_j) = QR$ we have

$$\|A(T_j, S_j)\widehat{m}_j(S_j) - w_j(T_j)\|_2 = \|QR\,\widehat{m}_j(S_j) - w_j(T_j)\|_2, \tag{4.1}$$

which can be minimized by solving $R\widehat{m}_j(S_j) = Q^T w_j(T_j)$ by back substitution.

We make two observations. First, the entries $w_j(T_j)$ must be known. Second, the principal cost of minimizing (4.1) is that of computing the QR factorization of $A(T_j, S_j)$. This requires $O(|T_j||S_j|^2)$ arithmetic operations, becoming $O(|S_j|^3)$ when A is banded because then $|T_j| \sim |S_j|$. Algorithm 2 becomes impractical in its basic form when $L = N$ because then $|S_j| = n$ for some columns j. The following modifications address the second observation.

1. Use a wavelet transform of level $L < N$. Note that $\max_{j=1,n} |S_j| \approx 2^L$.
2. Compute $\widehat{m}_j$ by minimizing (4.1) whenever $|S_j|/n \le \rho$ for some parameter $\rho \in (0, 1)$. Otherwise compute $\widehat{m}_j$ by solving $A\widehat{m}_j = w_j$ using a small number of GMRES steps.
3. Restrict S_j. This approach is used by Chan, Tang, and Wan [9]. We do not use this approach because we aim to include the off-diagonal finger pattern into our preconditioner.

The ability to compute the necessary entries of W is crucial. Many discrete wavelet transforms are applied level by level, with

$$W = W_1 W_2 \cdots W_{L-1} W_L.$$

The matrices W_i represent single levels of the transform and are often readily available, although W may not be. Computation of W by multiplying the factors W_i is possible because W and its factors are sparse.

More generally, the kth column of W can be computed by inverse-transforming the kth Euclidean vector e_k using any transform algorithm. Only $O(L)$ wavelets need

to be computed in this way because the wavelets at each level are translations of each other—thus only one wavelet at each level must be computed.

It remains to comment on the relative costs of Algorithms 1 and 2. Sparse approximate inverse and setup are considered. In computing the sparse approximate inverse in Algorithm 1, the restricted pattern for $\widetilde{M}$ yields $|S_j| = c$ for some constant c. However, $|T_j|$ is difficult to estimate because it depends on the pattern of $\widetilde{A}$. Nevertheless, one can bound the costs of computing $\widetilde{M}$ by $O(n^2c^2)$. A realistic estimate is $O(nc^2)$ because there are few dense columns of $\widetilde{A}$.

In Algorithm 2, recall that $|T_j| \sim |S_j|$ when A is banded. The cost of computing $\widehat{M}$ is then bounded by $O(n[\max_{j=1,n} |S_j|]^3)$. Note that $\max_{j=1,n} |S_j| \approx 2^L$, yielding a cost $O(n2^{3L})$. When L is kept constant, the cost becomes linear in n.

Setup for Algorithm 1 involves computing $\widetilde{A}$. Setup for Algorithm 2 involves computing W. The former requires forward wavelet transforming $2n$ vectors. The latter requires inverse wavelet transforming $O(L) \leq O(\log_2 n)$ vectors. W may be stored and reused for other problems of the same size. We will show that $\widetilde{A}$ can be computed in $O(12mn\alpha)$ arithmetic operations when A has bandwidth α, and W can be computed in $O(4m(m-1)n)$ arithmetic operations. $\widetilde{A}$ and W may be computed naively by multiplication of sparse matrices. This is easily implemented but it introduces a computational overhead, although the arithmetic operation estimates will be realized. Computation of $\widetilde{A}$ remains more expensive than that of W.

Algorithm 3 is the typical pyramid algorithm for the forward discrete wavelet transform. Let $\alpha \in \mathbb{N}$ be large compared with $m/2$. Then the complexity of transforming a vector with α nonzeros is approximately $O(2m \times 2\alpha[1 - 1/2^L])$ and the transform of the vector contains approximately $2\alpha[1 - 1/2^L]$ nonzeros. It follows that transforming a banded matrix with α nonzeros in each column involves transforming n columns each with α nonzeros and then n rows each with $2\alpha[1 - 1/2^L]$ nonzeros. The complexity of this transformation is approximately

$$O\left(2mn \times \left[2\alpha[1 - 1/2^L] + 2\left(2\alpha[1 - 1/2^L]\right)[1 - 1/2^L]\right]\right) \approx O(12mn\alpha).$$

Algorithm 4 is the typical pyramid algorithm for the inverse discrete wavelet transform. The complexity of inverse transforming the Euclidean vector whose nonzero entry appears in s^l or d^l is approximately $O(2m(m-1)2^l)$. Thus the complexity of transforming L Euclidean vectors whose nonzeros appear in $s^1, s^2, \ldots, s^L$, respectively, is approximately

$$O\left(\sum_{l=1}^{L} 2m(m-1)2^l\right) \approx O(4m(m-1)2^L).$$

When $n = 2^L$ the second estimate becomes $O(4m(m-1)n)$ and we see that computing $\widetilde{A}$ is approximately $12\alpha/4(m-1)$ times more expensive than computing W.

Algorithm 3. Forward transform.

1. Let $s^0 = x$.
2. For level $k = 1, 2, \ldots, L$
   (a) Let $s_j^k = \sum_{l=0}^{m-1} h_l s^{k-1}_{\langle l+2j-1\rangle_{n/2^{k-1}}}$, $j = 1, 2, \ldots, n/2^k$.
   (b) Let $d_j^k = \sum_{l=0}^{m-1} g_l s^{k-1}_{\langle l+2j-1\rangle_{n/2^{k-1}}}$, $j = 1, 2, \ldots, n/2^k$.
3. Let $\widetilde{x} = (s^L, d^L, d^{L-1}, \ldots, d^1)$.

ALGORITHM 4. INVERSE TRANSFORM.

1. Let $(s^L, d^L, d^{L-1}, \ldots, d^1) = \widetilde{x}$.
2. For level $k = L, L-1, \ldots, 1$
   (a) Let $s_j^{k-1} = \sum_{l=0}^{m/2-1} h_{2l} s^k_{\langle (j+1)/2-l \rangle n/2^k} + \sum_{l=0}^{m/2-1} g_{2l} d^k_{\langle (j+1)/2-l \rangle n/2^k}$ for $j = 1, 3, \ldots, n/2^{k-1} - 1$.
   (b) Let $s_j^{k-1} = \sum_{l=0}^{m/2-1} h_{2l+1} s^k_{\langle j/2-l \rangle n/2^k} + \sum_{l=0}^{m/2-1} g_{2l+1} d^k_{\langle j/2-l \rangle n/2^k}$ for $j = 2, 4, \ldots, n/2^{k-1}$.
3. Let $x = s^0$.

5. Dense problems. Although our primary purpose is to precondition sparse problems arising from PDEs, we consider in this section the application of Algorithms 1 and 2 to dense problems where A either is fully available or is partially available, for example, using the fast multipole method [21, 27].

Consider the general problem of finding M that minimizes $\|BM - C\|_F$ subject to a specified sparsity pattern for M. When $B = \widetilde{A}$ and $C = I$ this corresponds to the minimization problem at step 2 of Algorithm 1. When $B = A$ and $C = W$ this corresponds to the minimization problem at step 1 of Algorithm 2.

Let S_j index the nonzero entries of m_j, where m_j is the jth column of M. A generalization of the argument in section 4 shows that the minimization problem reduces to solving n problems of the form $Rm_j(S_j) = Q^T c_j(T_j)$, where we have the QR factorization $B(T_j, S_j) = QR$, where T_j indexes the nonzero rows of $B(:, S_j)$, and where c_j is the jth column of C.

The principal task in solving each least squares problem is the QR factorization of the $t \times s = |T_j| \times |S_j|$ matrix $B(T_j, S_j)$. This operation has complexity $O(ts^2)$. Note that s depends upon the chosen sparsity pattern for M and that t, loosely speaking, depends upon the sparsity of B.

When A is dense, then $\widetilde{A}$ in Algorithm 1 is also dense. However, if A is also smooth, then $\widetilde{A}$ will have a structure containing many small entries. Computation of a column of the sparse approximate inverse of a dense matrix using the least squares approach has complexity $O(ns^2)$ and for all but the smallest s is too expensive. Computation using the GMRES approach (for example, when the column is dense) is also expensive because it requires repeated matrix vector multiplication with a dense matrix.

We consider two adaptations to the method of computing the sparse approximate inverse. Write $B = S + F$, where S is sparse and F is dense.

1. B is replaced by S in the least squares minimization problems. Thus we minimize $\|SM - C\|_F$ instead of $\|BM - C\|_F$.

2. Dense columns of M are computed by solving $Sm_j = c_j$ instead of $Bm_j = c_j$ using GMRES.

We propose that adaptation 2 is always necessary for computing dense columns, but that adaptation 1 is required only when $s = |S_j|$ is not small.

In both cases the idea is to let S be a good sparse approximation to B so that M, which approximates S^{-1}, is a good preconditioner for B. A similar approach is taken by Chen [10] who shows that if the continuous operator $\mathcal{S}$ underlying S is bounded and the continuous operator $\mathcal{F}$ underlying F is compact, then $\mathcal{S}^{-1}(\mathcal{S}+\mathcal{F}) = \mathcal{I}+\mathcal{S}^{-1}\mathcal{F}$ and $\mathcal{S}^{-1}\mathcal{F}$ is compact. One expects the corresponding matrix $S^{-1}(S + F)$ to have the clustered eigenvalues associated with the discretization of such an operator.

The choice of S and F is crucial. Carpentieri, Duff, and Giraud [7] propose the following strategies.

1. [illegible] S is obtained from B by discarding entries whose absolute value falls below a specified threshold.

2. [illegible] When B is obtained by a finite element discretization, S is obtained by discarding those entries b_{ij} of B whose corresponding nodes i and j are not (at some given level) mesh neighbors.

3. [illegible] When B is obtained by a finite element discretization, S is obtained by discarding those entries of b_{ij} of B whose corresponding nodes i and j are separated by more than a specified distance.

For Algorithm 1 we propose strategy 1 because strategies 2 and 3 do not directly apply since the entries in $\widetilde{A}$ do not directly relate to connections between nodes in a mesh. However, strategies analogous to 2 and 3 can be developed by replacing relationships between nodes with relationships between supports of wavelet basis functions; see, for example, Hawkins and Chen [23].

For Algorithm 2 one could apply any of these strategies. In some circumstances details of the mesh may be available but the entries of A are not explicitly known. Then $\widetilde{A}$ cannot easily be computed and our method is particularly useful. Under this context Carpentieri, Duff, and Giraud [7] propose strategies 2 and 3. This situation arises, for example, in particle simulations or integral equation formulations for Laplace's equation when they are solved using the fast multipole method [21, 27]. Here, matrix vector multiplication with A can be performed cheaply and to a specified accuracy but A is not computed explicitly. One way to obtain columns of A for the purposes of computing a least squares sparse approximate inverse is to multiply Euclidean vectors by A. This naive approach may be very expensive, because all columns of A will be needed, though they will not be needed or stored simultaneously when the preconditioner is sparse. A better way to compute the sparse approximate inverse in this case might be to use Algorithm 2 with adaptation 1, replacing A in the sparse approximate inverse computations by a sparse approximation S to A. The entries of S might be obtained using only near field contributions associated with the singularity.

6. Theoretical details. In this section we justify the banded pattern chosen for $\widehat{M}$ and present a theoretical bound on $\|W^T A\widehat{M} - I\|_F$. This provides a measure of the conditioning of the iteration matrix $W^T A\widehat{M}$.

The following theorem establishes that $A^{-1}W$, which we approximate by $\widehat{M}$, has the banded pattern illustrated in Figure 3.1(b). Let $\{\psi_i\}_{i=1}^n$ be a wavelet basis in L_2 with m vanishing moments. Such properties of wavelets are discussed fully by Strang and Nguyen [31]. Let I_i denote the support of ψ_i, and suppose that $\|\psi_i\|_2 = 1$. Let $\{h_i\}_{i=1}^n$ be a single scale basis in L_2, with $\bar{I}_i$ denoting the support of h_i, and with $\|h_i\|_2 = 1$.

THEOREM 6.1. [illegible] $K(x,y)$ [illegible]

$$\left|\frac{\partial^m K}{\partial x^m}(x,y)\right| \leq \frac{C_m}{|x-y|^{m+1}} \quad x \neq y \tag{6.1}$$

[illegible] C_m [illegible] $\widehat{N}$ [illegible] $K(x,y)$ [illegible] $\{\psi_j\}_{j=1}^n$ [illegible] $\{h_i\}_{i=1}^n$ [illegible]

$$|\widehat{N}_{ij}| \leq b(I_j, \bar{I}_i)$$

[illegible]

$$b(I_j, \bar{I}_i) = C|I_j|^{(2m+1)/2} \frac{1}{\operatorname{dist}(I_j, \bar{I}_i)^{m+1}}$$

[illegible] C

The proof of Theorem 6.1 is given in the appendix.

Choose $\epsilon > 0$ and let $S = \{(i,j) : |b(I_j, \bar{I}_i)| > \epsilon\}$. Denote by $\widehat{N}^S$ the matrix obtained by cutting $\widehat{N}$ to the pattern S. Then

$$\begin{aligned}\max_{(i,j)} \left|\widehat{N}^S_{ij} - \widehat{N}_{ij}\right| &\le \max_{(i,j)\notin S} |b(I_j, \bar{I}_i)| \\ &\le \epsilon.\end{aligned}$$

THEOREM 6.2. [illegible] S [illegible] $\widehat{M}$ [illegible]

$$\left\|W^T A\widehat{M} - I\right\|_F \le n\epsilon\|A\|_F.$$

[illegible] Denote the jth column of $\widehat{N}^S$ by $\widehat{n}^S_j$. Similarly denote the jth column of $\widehat{M}$ by $\widehat{m}_j$ and the jth column of W by w_j.

Then

$$\begin{aligned}\left\|W^T A\widehat{M} - I\right\|_F^2 &= \|A\widehat{M} - W\|_F^2 \\ &= \sum_{j=1}^{n} \|A\widehat{m}_j - w_j\|_2^2 \\ &\le \sum_{j=1}^{n} \|A\widehat{n}^S_j - w_j\|_2^2 \\ &= \|A\widehat{N}^S - W\|_F^2 \\ &\le \|\widehat{N}^S - A^{-1}W\|_F^2 \|A\|_F^2 \\ &\le \|\widehat{N}^S - \widehat{N}\|_F^2 \|A\|_F^2 \\ &\le n^2\epsilon^2\|A\|_F^2. \qquad \square\end{aligned}$$

[illegible] 6.3. These results give a guide to the effectiveness of $\widehat{M}$ for small ϵ. The eigenvalues of the preconditioned matrix will have a desirable distribution when ϵ is small, because they will be clustered close to 1.

Theorem 6.2 suggests that one should choose ϵ sufficiently small, so that the pattern S leads to a sufficiently small bound $\|W^T A\widehat{M} - I\|_F^2$ for the preconditioner $\widehat{M}$. In practice, good results can be obtained when ϵ is large.

A similar bound for $\|\widetilde{A}\widetilde{M} - I\|_F$ is given by Chan, Tang, and Wan [9].

7. Numerical results. We test variants of Algorithms 1 and 2 with GMRES [29, 30] for a range of finite difference discretizations of one-, two-, and three-dimensional model PDEs, for a range of sparse off-the-shelf matrices obtained from the Matrix Market [25] and representing finite difference and finite element discretizations of two- and three-dimensional problems, and for a dense problem arising from a boundary element discretization of a boundary integral equation. The test problems are as follows.

- 1 [illegible]. One-dimensional Poisson's equation

$$u_{xx} = \sin 2\pi x, \quad x \in [0,1],$$

with a Dirichlet boundary condition discretized with central differences.

- 2 . Two-dimensional Poisson's equation

$$u_{xx} + u_{yy} = -100x^2, \quad (x, y) \in [0, 1]^2,$$

with a Dirichlet boundary condition discretized with central differences.

- 2 . Two-dimensional PDE with discontinuous coefficients

$$(a(x, y)u_x)_x + (b(x, y)u_y)_y + u_x + u_y = \sin(\pi xy), \quad (x, y) \in [0, 1]^2,$$

with

$$a(x, y) = b(x, y) = \begin{cases} 10^{-3}, & (x, y) \in [0, 0.5] \times [0.5, 1], \\ 10^{3}, & (x, y) \in [0.5, 1] \times [0, 0.5], \\ 1 & \text{otherwise}, \end{cases}$$

with a Dirichlet boundary condition discretized with central differences.

- 3 . Three-dimensional Poisson's equation

$$u_{xx} + u_{yy} + u_{zz} = -100x^2, \quad (x, y, z) \in [0, 1]^3,$$

with a Dirichlet boundary condition discretized with central differences.

- 3 . Three-dimensional PDE with variable coefficients

$$\big((1 + x^2)u_x\big)_x + ((1 + y)u_y)_y + \big((2 - z^2)u_z\big)_z = -100x^2, \quad (x, y) \in [0, 1]^2,$$

with a Dirichlet boundary condition discretized with mixed forward and backward differences.

- 3. Matrix Market [25] problem arising from finite difference discretization of a three-dimensional oil reservoir model.
- 4. Matrix Market [25] problem arising from finite difference discretization of a three-dimensional oil reservoir model.
- 1. Matrix Market [25] problem arising in petroleum engineering.
- 2048 . Matrix Market [25] problem arising from finite difference discretization of a reaction diffusion Brusselator model.
- . Two-dimensional PDE with variable coefficients

$$-\epsilon\Delta u + \frac{\partial}{\partial x}(e^{xy}u) + \frac{\partial}{\partial y}(e^{-xy}u) = g, \quad \epsilon = \frac{1}{1000}, \ (x, y) \in [0, 1]^2,$$

with a Dirichlet boundary condition discretized with central differences. The right-hand side is chosen so that the solution vector is $(1, 1, \ldots, 1)^T$. This is a widely used test problem; see, for example, [4, 28].

- . Two-dimensional PDE with variable coefficients

$$-\epsilon\Delta u + \frac{\partial}{\partial x}(e^{xy}u) + \frac{\partial}{\partial y}(e^{xy}u) = h, \quad \epsilon = \frac{1}{1000}, \ (x, y) \in [0, 1]^2,$$

with a Dirichlet boundary condition discretized with central differences. The right-hand side is chosen so that the solution vector is $(1, 1, \ldots, 1)^T$.

- . Two-dimensional PDE with variable coefficients

$$-\Delta u + 2P_1 u_x + 2P_2 u_y = f, \quad P_1 = P_2 = 50, \ (x, y) \in [0, 1]^2,$$

with a Dirichlet boundary condition discretized with central differences. The right-hand side is chosen following Elman [18] so that the solution is $u(x, y) = xe^{xy}\sin(\pi x)\sin(\pi y)$.

- . Dense matrix arising from a boundary element discretization of the boundary integral operator $-\frac{1}{2}I + M_k + iN_k$ arising in the Burton and Miller formulation for the exterior Neumann Helmholtz problem [2, 11].

Daubechies 4 wavelets [17] were used for all problems. For the BIE problem we additionally used multiwavelets [1]. Our method can be applied with other types of orthogonal wavelet and is easily generalized to work with biorthogonal wavelets [14], or second generation wavelets [32]. In all experiments we used a wavelet transform of the same dimension as the underlying PDE problem. Wavelet transforms in two and three dimensions are easily obtained using tensor products. The Sherman 3 and Sherman 4 problems have a dimension whose number of unknowns is not divisible by two and thus not amenable to wavelet transform. We can still use wavelets if we use a Schur complement method to extract a subsystem involving an appropriately sized submatrix. For the Sherman 3 and Sherman 4 problems we solve systems involving appropriate submatrices but do not implement the full Schur complement to obtain the right-hand side or the solution to the original system. Except where stated otherwise, results were obtained using software written in Fortran and C++ running on a 1 GHz Intel Pentium 3 machine.

The New preconditioner is of sparse approximate inverse type. As such, it complements incomplete factorization based preconditioners [29]. For illustration we include results for an ILUT preconditioner [29]. The fill-in parameter for the ILUT preconditioner is chosen so that the total number of nonzeros in the incomplete LU factors is comparable with the number of nonzeros in the New preconditioner. Incomplete factorization based preconditioning is known to be very effective for M-matrices [26] but can fail for some nonsymmetric problems [4, 18, 28, 29]. For test problems NONSYMA, NONSYMB, and NONSYMC we compute the ILUT preconditioner after applying a symmetric reverse Cuthill–McKee reordering of the matrix. Benzi, Szyld, and van Duin [4] show that for nonsymmetric problems such reorderings dramatically improve the effectiveness of incomplete factorization based preconditioners. Note that reordering is not required for the New preconditioner.

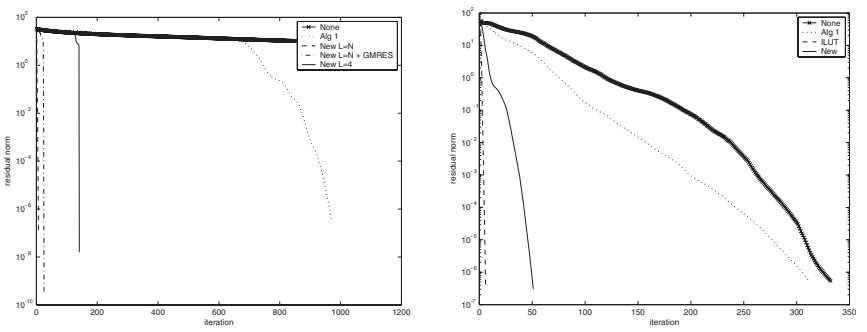
In Table 7.1 we present timings and iteration counts for solving the test problems using GMRES. Convergence histories for some of these problems are given in Figures 7.1, 7.2, and 7.3. The New preconditioner is Algorithm 2 implemented with a wavelet transform of level $L < N$. The choice of L varies between problems. For each of the finite difference problems we fix L, though N varies. The New preconditioner is very effective and gives substantial improvement over Algorithm 1 and over no preconditioning. In most cases the fewest iterations are required with ILUT preconditioning. Algorithm 1 and New compute and explicitly store the preconditioner, and application of the preconditioner at each iteration of GMRES involves only matrix-vector multiplication. In contrast, application of the ILUT preconditioner involves triangular solves, which are difficult to parallelize.

In Table 7.2 we present setup times and number of nonzeros for the preconditioning strategies. The setup time for Algorithm 1 includes the time t_{DWT} taken to compute $\widetilde{A}$ and the time t_{SPAI} taken to compute the sparse approximate inverse. The setup time for New includes the time t_{DWT} taken to compute W and the time t_{SPAI} taken to compute the sparse approximate inverse. In almost every case the setup time for New is less than that for Algorithm 1. The setup time t_{FACT} for ILUT is less than that for the other preconditioners. However, the sparse approximate inverse components of Algorithm 1 and New are easy to implement in parallel. The discrete wavelet transform can also be implemented in parallel [19]. In contrast, the incomplete factorization in ILUT is difficult to implement in parallel.

TABLE 7.1

*GMRES iterations "Itns" and time "t_{GMRES}" required to reduce the residual norm by a factor of 10^{-8}. $\times$ denotes that 1000 steps of GMRES were insufficient to achieve this reduction of the residual norm. New denotes a variant of Algorithm 2. All times are measured in seconds. The superscript * denotes that the iteration matrix is a submatrix of the original matrix. ILUT inverts the matrix exactly for the 1D Laplacian. ILUT fails for the nonsymmetric problems NONSYMB and NONSYMC when the lexicographical ordering is used. The superscript † denotes results obtained for a reverse-Cuthill–McKee reordering of the original system.*

		None		ILUT		Alg. 1		New	
Experiment	Size	Itns	t_{GMRES}	Itns	t_{GMRES}	Itns	t_{GMRES}	Itns	t_{GMRES}
1D Laplacian	256	256	0.16	—	—	155	0.11	23	0.00
	512	512	1.36	—	—	282	0.67	40	0.03
	1024	×	×	—	—	520	3.89	74	0.19
	2048	×	×	—	—	970	24.67	140	0.89
2D Laplacian	256	44	0.00	8	0.00	29	0.01	28	0.02
	1024	90	0.09	12	0.02	51	0.19	58	0.16
	4096	179	1.59	23	0.17	95	1.94	118	1.76
2D Disc	256	188	0.08	6	0.01	249	0.31	28	0.00
	1024	645	4.19	11	0.02	×	×	60	0.17
	4096	×	×	21	0.16	×	×	128	1.89
3D Laplacian	512	27	0.01	5	0.00	20	0.09	17	0.08
	4096	54	0.23	9	0.16	32	1.33	34	1.06
3D PDE	512	39	0.02	5	0.01	22	0.09	18	0.08
	4096	79	0.42	9	0.16	35	1.50	38	1.19
Sherman 3	2048*	331	2.56	5	0.03	311	8.27	50	0.73
Sherman 4	1104	127	0.23	57	0.06	115	0.33	92	0.17
	768*	97	0.11	15	0.00	86	0.23	54	0.11
Watt 1	1856	403	3.41	24	0.05	154	1.08	115	0.58
RDB2048L	2048	134	0.53	8	0.09	201	2.53	69	1.09
NONSYMA	1024	299	1.05	11†	0.02†	672	6.95	80	0.25
NONSYMB	1024	299	1.00	16†	0.03†	906	11.72	76	0.22
NONSYMC	1024	392	1.73	×†	×†	×	×	155	0.59



(a) (b)

FIG. 7.1. (a) *Variants of the New preconditioner perform well for the 1D Laplacian. Using cheaper variants such as $L = 4$ and $L = N +$* GMRES *impairs the performance.* (b) *GMRES with ILUT preconditioning and New preconditioning converge quickly for the Sherman 3 problem.*

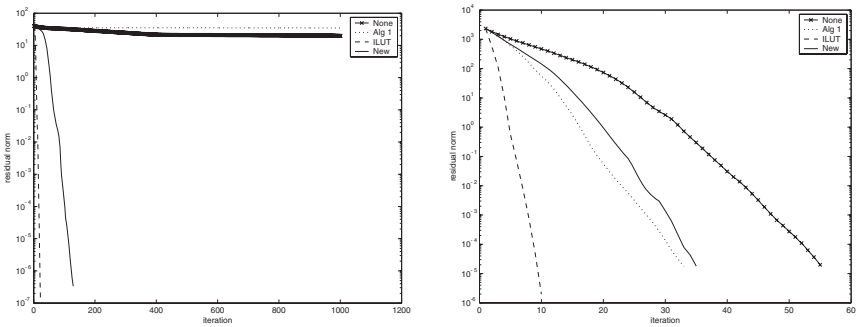


FIG. 7.2. (a) *GMRES with ILUT preconditioning converges very quickly for the* 2*D Disc problem* [9]. *The New preconditioning strategy is also very effective.* (b) *GMRES with ILUT preconditioning converges very quickly for the* 3*D Laplacian. This is an easy problem for ILUT but a challenging problem for sparse approximate inverse based preconditioners. Nevertheless the New and Algorithm* 1 *preconditioners perform well.*

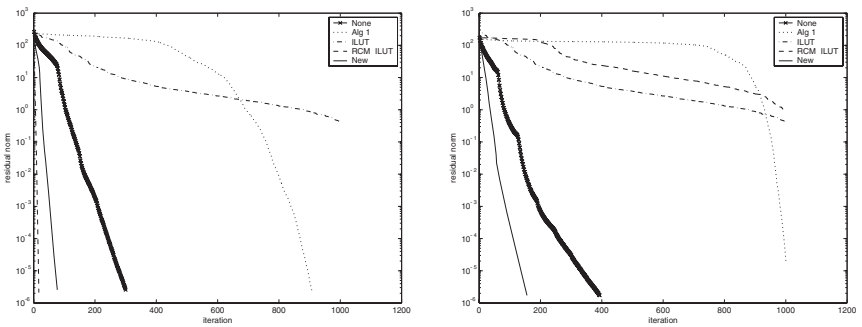


FIG. 7.3. *The nonsymmetric problems* (a) *NONSYMB and* (b) *NONSYMC are challenging for all of the preconditioners. For both problems New is effective but ILUT fails. ILUT can be made effective for NONSYMB by preordering the system with a reverse-Cuthill–McKee ordering (denoted RCM-ILUT).*

. . 7.1.

1. In section 4 we showed that the sparse approximate inverse and discrete wavelet transform components of New have linear complexity in n. This linear complexity can be observed in Table 7.2.

2. Most of the setup time for New is the time taken to compute the sparse approximate inverse in serial. However this task is easily parallelized, especially when A is banded, because then load balancing is easy even though the number of nonzeros in each column of $\widehat{M}$ varies. We expect the time taken to compute the sparse approximate inverse to be of order n/p when there are p processors used in parallel.

3. Improved iteration counts and t_{GMRES} can be achieved by increasing the level L of the wavelet transform. Corresponding increases in t_{SETUP} can in some part be alleviated by using GMRES to compute dense rows of the preconditioner.

4. The two- and three-dimensional problems yield better conditioned matrices than the one-dimensional problems because the ill conditioning of the matrix grows with decreasing mesh size rather than the size of the matrix. In fact the condition number for the d-dimensional Laplacian problem grows like $n^{2/d}$.

TABLE 7.2

Setup times and numbers of nonzeros in the preconditioner for the experiments in Table 7.1. New denotes a variant of Algorithm 2. Although the setup time t_{FACT} for ILUT is typically less than the setup time $t_{\mathrm{DWT}}+t_{\mathrm{SPAI}}$ for the New strategy, both the discrete wavelet transform and the sparse approximate inverse components of the New strategy can be easily implemented in parallel. The fill-in parameters for ILUT ensure that the number of nonzeros "Nnz" in the preconditioner is comparable between ILUT and New. The superscript * denotes that the iteration matrix is a submatrix of the original matrix. The superscript † denotes results obtained for a reverse-Cuthill–McKee reordering of the original system.

		ILUT		Alg. 1			New		
Experiment	Size	Nnz	t_{FACT}	Nnz	t_{DWT}	t_{SPAI}	Nnz	t_{DWT}	t_{SPAI}
1D Laplacian	256	766	0.00	1330	0.00	0.00	3328	0.00	0.03
	512	1534	0.00	2616	0.01	0.03	6656	0.00	0.05
	1024	3070	0.00	5182	0.09	0.06	13312	0.00	0.11
	2048	6142	0.00	10308	0.20	0.22	26624	0.00	0.27
2D Laplacian	256	3896	0.00	1996	0.03	0.05	4096	0.00	0.02
	1024	16024	0.02	6764	0.38	0.39	16384	0.00	0.09
	4096	64856	0.09	24172	2.77	3.95	65536	0.02	0.66
2D Disc	256	3893	0.00	1996	0.03	0.05	4096	0.00	0.01
	1024	16021	0.00	6764	0.38	0.39	16384	0.00	0.09
	4096	64853	0.09	24172	2.75	4.36	65536	0.02	0.64
3D Laplacian	512	30386	0.05	4588	0.45	0.25	32768	0.00	1.63
	4096	256237	1.30	32236	18.38	23.53	262144	0.05	16.02
3D PDE	512	30376	0.05	4588	0.47	0.25	32768	0.00	1.66
	4096	256113	1.31	32236	19.36	24.11	262144	0.03	16.05
Sherman 3	2048*	66978	0.19	13382	4.58	3.13	131072	0.02	4.98
Sherman 4	1104	2698	0.00	7416	0.03	0.08	4416	0.00	0.03
	768*	6709	0.02	5816	0.19	0.28	12288	0.00	0.06
Watt 1	1856	17348	0.02	15114	0.16	0.30	18560	0.00	0.19
RDB2048L	2048	199378	0.25	12076	1.19	1.16	204800	0.05	107.41
NONSYMA	1024	16120†	0.01†	6764	0.38	0.41	16384	0.00	0.09
NONSYMB	1024	16131†	0.01†	6764	0.38	0.41	16384	0.02	0.09
NONSYMC	1024	15984†	0.00†	6764	0.41	0.39	16384	0.00	0.11

TABLE 7.3

Results from variants of Algorithm 2 applied to the 1D Laplacian problem. The best iteration counts "Itns" and times "t_{GMRES}" are obtained when the number of levels of wavelet transform L is large. The setup time "t_{SETUP}" increases with L but this can in some part be alleviated by using GMRES for computing dense columns in the preconditioner. All times are measured in seconds.

	$L=N$			$L=N$ with GMRES			$L=4$		
Size	Itns	t_{GMRES}	t_{SETUP}	Itns	t_{GMRES}	t_{SETUP}	Itns	t_{GMRES}	t_{SETUP}
256	6	0.00	0.38	18	0.00	0.05	23	0.00	0.03
512	6	0.00	2.56	20	0.03	0.34	40	0.03	0.05
1024	6	0.02	23.70	22	0.06	2.81	74	0.19	0.11
2048	6	0.05	210.25	24	0.17	23.20	140	0.89	0.27

The effects of varying L can be seen in Table 7.3, where we present results obtained for the 1D Laplacian with $L=4$ and $L=N$. When $L=N$ the preconditioner has some full columns so that this strategy is impractical and we include it only for comparison. This strategy can be made more practical by computing column m_j using GMRES whenever $|S_j|/n > 1/2$. Results are presented for this modification.

TABLE 7.4

Results for a dense problem. The New preconditioning strategies produce large reductions in the number of GMRES iterations required to reduce the residual norm by a factor of 10^{-8}. *For New strategies S is* ^{3}*bandwidth* 3, ^{4}*bandwidth* 7. *The superscripts* ^{1} *and* ^{2} *denote that dense columns of the preconditioner were computed using GMRES with A and S, respectively. For this experiment, which was performed in MATLAB, we use (D4) Daubechies* 4 *wavelets* [17] *and (A) multiwavelets* [1].

	None	Alg. 1		New $L = N$		New^1 $L = N$		New $L = N - 2$	New^2 $L = N$	
	–	D4	A	D4	A	D4	A	D4	D4	A
BIE^3	613	162	166	89	94	108	125	89	75	60
BIE^4	613	162	166	60	67	72	87	60	73	59

In Table 7.4 we present results for the dense problem BIE. The matrix is complex but the theory is the same as for the real case. These experiments were performed in MATLAB using the Daubechies 4 wavelets [17] and also a basis of supercompact orthogonal multiwavelets [1]. Algorithm 2 is implemented with the adaptations described in section 5 for dense problems. The variants of Algorithm 2 required the fewest iterations. Using the multiwavelets led to fewer iterations than using the Daubechies 4 wavelets. We believe this is due to the higher compression that can be achieved with the multiwavelets, which have more vanishing moments than the Daubechies 4 wavelets.

8. Conclusion. We have introduced a new wavelet sparse approximate inverse preconditioner. The new preconditioner, in common with the preconditioners of Chan, Tang, and Wan [9] and Cohen and Masson [15], takes advantage of the compression and structure created by working in a wavelet basis. Like the Cohen and Masson preconditioner [15] the new preconditioner takes into account interactions between scales, which is considered essential for robustness. However, the new preconditioner works with A rather than with the wavelet transform of A and this makes the computation of the sparse approximate inverse easier because, first, the wavelet transform of A need not be computed and, second, when A is sparse it is often sparser than its wavelet transform. We have considered efficient application of the new preconditioner and the Chan, Tang, and Wan preconditioner [9] to sparse and dense matrices.

To facilitate easy computation of our preconditioner we have (a) considered part wavelet transforms and developed a mixed least squares/GMRES sparse approximate inverse algorithm, and (b) considered efficient ways of computing the wavelet basis matrix W.

The new preconditioning strategy has been tested on a range of sparse and dense linear systems arising from discretization of partial differential equations and boundary integral equations. In all cases the preconditioner has been shown to be very useful in reducing the number of GMRES iterations required to solve the linear system.

Appendix. Compression theorem. We now state the proof of Theorem 6.1.

. We begin by expanding K as a Taylor series in x about $x_0 = \inf I_j$ to give

$$K(x,y) = K(x_0,y) + (x-x_0)\frac{\partial K}{\partial x}(x_0,y) + \cdots + \frac{(x-x_0)^{m-1}}{(m-1)!}\frac{\partial^{m-1}K}{\partial x^{m-1}}(x_0,y) + R(x,y),$$

where the remainder

$$R(x,y) = \frac{(x-x_0)^m}{m!}\frac{\partial^m K}{\partial x^m}(\xi,y)$$

for some $\xi \in I_j$. All terms in the Taylor series, except for the remainder term, are orthogonal to ψ_j because of the vanishing moments properties of the wavelets.

Now

$$\widehat{N}_{ij} = \int_{\bar{I}_i} \int_{I_j} K(x,y)\psi_j(x)dx\, h_i(y)dy$$

and

$$\begin{aligned}\int_{I_j} K(x,y)\psi_j(x)dx &= \int_{I_j} R(x,y)\psi_j(x)dx \\ &= \int_{I_j} \frac{(x-x_0)^m}{m!}\frac{\partial^m K}{\partial x^m}(\xi,y)\psi_j(x)dx \\ &= \frac{\partial^m K}{\partial x^m}(\xi,y)\int_{I_j}\frac{(x-x_0)^m}{m!}\psi_j(x)dx.\end{aligned}$$

It follows that

$$\begin{aligned}&\int_{\bar{I}_i}\int_{I_j} K(x,y)\psi_j(x)dx\, h_i(y)dy \\ &\quad= \int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)\int_{I_j}\frac{(x-x_0)^m}{m!}\psi_j(x)dx\, h_i(y)dy \\ &\quad= \left(\int_{I_j}\frac{(x-x_0)^m}{m!}\psi_j(x)dx\right)\int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)h_i(y)dx.\end{aligned}$$

The Cauchy–Schwarz inequality gives

$$\begin{aligned}&\left|\int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)h_i(y)dy\right| \\ &\quad\le \left(\int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)^2dy\right)^{1/2}\left(\int_{\bar{I}_i}h_i(y)^2dy\right)^{1/2} \\ &\quad\le \left(\int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)^2dy\right)^{1/2}\|h_i\|_2 \\ &\quad= \left(\int_{\bar{I}_i}\frac{\partial^m K}{\partial x^m}(\xi,y)^2dy\right)^{1/2} \\ &\quad\le |\bar{I}_i|^{1/2}\sup_{y\in\bar{I}_i}\left|\frac{\partial^m K}{\partial x^m}(\xi,y)\right|.\end{aligned}$$

A similar argument shows

$$\left|\int_{I_j}\frac{(x-x_0)^m}{m!}\psi_j(x)dx\right| \le \frac{|I_j|^{(2m+1)/2}}{m!(2m+1)^{1/2}}.$$

Combining these gives

$$\begin{aligned}\left|\int_{\bar{I}_i}\int_{I_j} K(x,y)\psi_j(x)dx h_i(y)dy\right| &\le \frac{|I_j|^{(2m+1)/2}}{m!(2m+1)^{1/2}}|\bar{I}_i|^{1/2}\sup_{y\in\bar{I}_i}\left|\frac{\partial^m K}{\partial x^m}(\xi,y)\right| \\ &\le \frac{|I_j|^{(2m+1)/2}}{m!(2m+1)^{1/2}}|\bar{I}_i|^{1/2}\frac{C_m}{\operatorname{dist}(I_j,\bar{I}_i)^{m+1}} \\ &\le C|I_j|^{(2m+1)/2}\frac{1}{\operatorname{dist}(I_j,\bar{I}_i)^{m+1}},\end{aligned}$$

where

$$C = C_m \frac{|\bar{I}_i|^{1/2}}{m!(2m+1)^{1/2}}.$$

Here we make use of the fact that the functions $\{h_i\}$ are a fixed scale basis and so $|\bar{I}_i|$ is constant. □

Acknowledgment. We thank M. Benzi and two anonymous referees, whose comments and suggestions greatly improved this paper.

REFERENCES

[1] B. Alpert, G. Beylkin, R. Coifman, and V. Rokhlin, *Wavelet-like bases for the fast solution of second-kind integral equations*, SIAM J. Sci. Comput., 14 (1993), pp. 159–184.

[2] S. Amini, P. J. Harris, and D. T. Wilton, *Coupled Boundary and Finite Element Methods for the Solution of the Dynamic Fluid-Structure Interaction Problem*, Springer-Verlag, Berlin, 1992.

[3] M. W. Benson and P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Util. Math., 22 (1982), pp. 127–140.

[4] M. Benzi, D. B. Szyld, and A. van Duin, *Orderings for incomplete factorization preconditioning of nonsymmetric problems*, SIAM J. Sci. Comput., 20 (1999), pp. 1652–1670.

[5] M. Benzi and M. Tůma, *A comparative study of sparse approximate inverse preconditioners*, Appl. Numer. Math., 30 (1999), pp. 305–340.

[6] R. Bridson and W. P. Tang, *Multiresolution approximate inverse preconditioners*, SIAM J. Sci. Comput., 23 (2001), pp. 463–479.

[7] B. Carpentieri, I. S. Duff, and L. Giraud, *Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism*, Numer. Linear Algebra Appl., 7 (2000), pp. 667–685.

[8] T. F. Chan and K. Chen, *On two variants of an algebraic wavelet preconditioner*, SIAM J. Sci. Comput., 24 (2002), pp. 260–283.

[9] T. F. Chan, W. P. Tang, and W. L. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, 37 (1997), pp. 644–660.

[10] K. Chen, *An analysis of sparse approximate inverse preconditioners for boundary integral equations*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1058–1078.

[11] K. Chen and P. J. Harris, *Efficient preconditioners for iterative solution of the boundary element equations for the three-dimensional Helmholtz equation*, Appl. Numer. Math., 36 (2001), pp. 475–489.

[12] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.

[13] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse interactions*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[14] A. Cohen, I. Daubechies, and J.-C. Feauveau, *Biorthogonal bases of compactly supported wavelets*, Comm. Pure Appl. Math., 45 (1992), pp. 485–560.

[15] A. Cohen and R. Masson, *Wavelet methods for second-order elliptic problems, preconditioning, and adaptivity*, SIAM J. Sci. Comput., 21 (1999), pp. 1006–1026.

[16] W. Dahmen and A. Kunoth, *Multilevel preconditioning*, Numer. Math., 63 (1992), pp. 315–344.

[17] I. Daubechies, *Orthonormal bases of compactly supported wavelets*, Comm. Pure Appl. Math., XLI (1988), pp. 909–996.

[18] H. C. Elman, *A stability analysis of incomplete ILU factorizations*, Math. Comp., 47 (1986), pp. 191–217.

[19] J. Ford, K. Chen, and N. J. Ford, *Flexible parallelization of fast wavelet transforms*, Parallel Algorithms Appl., 18 (2003), pp. 155–169.

[20] N. I. M. Gould and J. A. Scott, *Sparse approximate-inverse preconditioners using norm-minimization techniques*, SIAM J. Sci. Comput., 19 (1998), pp. 605–625.

[21] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[22] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[23] S. C. Hawkins and K. Chen, *New wavelet preconditioner for solving boundary integral equations over nonsmooth boundaries*, Int. J. Comput. Math., 81 (2004), pp. 353–360.
[24] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings I. Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[25] *Matrix Market*, http://math.nist.gov/MatrixMarket/.
[26] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.
[27] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, J. Comput. Phys., 60 (1983), pp. 197–207.
[28] Y. Saad, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 24 (1988), pp. 89–105.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, PA, 2003.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[31] G. Strang and T. Nguyen, *Wavelets and Filter Banks*, Wellesley–Cambridge Press, Wellesley, MA, 1996.
[32] W. Sweldens, *The lifting scheme: A construction of second generation wavelets*, SIAM J. Math. Anal., 29 (1998), pp. 511–546.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 687–713
© 2005 Society for Industrial and Applied Mathematics

PARALLEL LAGRANGE–NEWTON–KRYLOV–SCHUR METHODS FOR PDE-CONSTRAINED OPTIMIZATION. PART I: THE KRYLOV–SCHUR SOLVER*

GEORGE BIROS† AND OMAR GHATTAS‡

Abstract. Large-scale optimization of systems governed by partial differential equations (PDEs) is a frontier problem in scientific computation. Reduced quasi-Newton sequential quadratic programming (SQP) methods are state-of-the-art approaches for such problems. These methods take full advantage of existing PDE solver technology and parallelize well. However, their algorithmic scalability is questionable; for certain problem classes they can be very slow to converge. In this two-part article we propose a new method for steady-state PDE-constrained optimization, based on the idea of using a full space Newton solver combined with an approximate reduced space quasi-Newton SQP preconditioner. The basic components of the method are Newton solution of the first-order optimality conditions that characterize stationarity of the Lagrangian function; Krylov solution of the Karush–Kuhn–Tucker (KKT) linear systems arising at each Newton iteration using a symmetric quasi-minimum residual method; preconditioning of the KKT system using an approximate state/decision variable decomposition that replaces the forward PDE Jacobians by their own preconditioners, and the decision space Schur complement (the reduced Hessian) by a BFGS approximation initialized by a two-step stationary method. Accordingly, we term the new method *Lagrange–Newton–Krylov–Schur* (LNKS). It is fully parallelizable, exploits the structure of available parallel algorithms for the PDE forward problem, and is locally quadratically convergent. In part I of this two-part article, we investigate the effectiveness of the KKT linear system solver. We test our method on two optimal control problems in which the state constraints are described by the steady-state Stokes equations. The objective is to minimize dissipation or the deviation from a given velocity field; the control variables are the boundary velocities. Numerical experiments on up to 256 Cray T3E processors and on an SGI Origin 2000 include scalability and performance assessment of the LNKS algorithm and comparisons with reduced SQP for up to $1,000,000$ state and 50,000 decision variables. In part II of the article, we address globalization and inexactness issues, and apply LNKS to the optimal control of the steady incompressible Navier–Stokes equations.

Key words. sequential quadratic programming, adjoint methods, PDE-constrained optimization, optimal control, Lagrange–Newton–Krylov–Schur methods, Navier–Stokes, finite elements, preconditioners, indefinite systems, nonlinear equations, parallel algorithms

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65J22, 65N55, 65W10, 65Y05, 65Y20, 76D05, 76D07, 76D55, 90C52, 90C55, 90C90, 93C20

DOI. 10.1137/S106482750241565X

1. Introduction.

refers to the optimization of systems governed by partial differential equations (PDEs). The or is to solve the PDEs for the , given appropriate data, i.e., geometry, coefficients, boundary conditions, initial conditions, and source functions. The seeks to determine some of these data—the —given an (e.g., performance goals) and (possibly inequality) constraints

*Received by the editors October 9, 2002; accepted for publication (in revised form) November 17, 2004; published electronically November 15, 2005. This work is a part of the Terascale Algorithms for Optimization of Simulations (TAOS) project at CMU, with support from NASA grant NAG-1-2090, NSF grant ECS-9732301 (under the NSF/Sandia Life Cycle Engineering Program), and the Pennsylvania Infrastructure Technology Alliance. Computing services on the Pittsburgh Supercomputing Center's Cray T3E were provided under PSC grant BCS-960001P.

http://www.siam.org/journals/sisc/27-2/41565.html

†Courant Institute of Mathematical Sciences, Department of Computer Science, New York University, New York, NY 10012 (biros@cs.nyu.edu).

‡Ultrascale Simulation Laboratory, Departments of Biomedical Engineering and Civil & Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 (oghattas@cs.cmu.edu).

on the behavior of the system. Since the behavior of the system is modeled by the PDEs, they appear as constraints in the optimization problem.

PDE-constrained optimization problems arise in several contexts, including optimal design, optimal control, and parameter estimation. They share the common difficulty that PDE solution is just a subproblem associated with optimization, and that the optimization problem can be ill-posed even when the forward problem is well-posed. Thus the optimization problem is often significantly more difficult to solve than the forward problem.

To illustrate the main issues, let us consider a model problem of optimal distributed control of a steady Navier–Stokes flow:

$$\min \mathcal{F}(\boldsymbol{u}, p, \boldsymbol{b}) = \frac{1}{2}\int_\Omega (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) d\Omega + \frac{1}{2}\int_\Omega \boldsymbol{b} \cdot \boldsymbol{b}\, d\Omega$$

subject to

$$\begin{aligned} -\nu \nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) + (\nabla \boldsymbol{u})\boldsymbol{u} + \nabla p + \boldsymbol{b} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \Gamma. \end{aligned}$$

Here $\boldsymbol{u}$ is the fluid velocity field, p the pressure field, $\boldsymbol{b}$ the body force control function, $\boldsymbol{g}$ the prescribed Dirichlet boundary conditions, and ν the inverse of the Reynolds number. The objective is to minimize the rate of dissipation of viscous energy without applying an excessively large body force. The constraints are the stationary incompressible Navier–Stokes equations with Dirichlet boundary conditions. The decision variable is the body force function $\boldsymbol{b}$.

For this problem a Lagrangian functional is given by

$$\begin{aligned} &\frac{1}{2}\int_\Omega (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) d\Omega + \frac{1}{2}\int_\Omega \boldsymbol{b} \cdot \boldsymbol{b}\, d\Omega \\ &\qquad + \int_\Omega \boldsymbol{\lambda} \cdot (-\nu \nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) + (\nabla \boldsymbol{u})\boldsymbol{u} + \nabla p + \boldsymbol{b}) d\Omega + \int_\Omega \mu \nabla \cdot \boldsymbol{u}\, d\Omega. \end{aligned}$$

By requiring its stationarity with respect to the state ($\boldsymbol{u}$,p), decision ($\boldsymbol{b}$), and Lagrange multiplier (i.e., adjoint) variables ($\boldsymbol{\lambda}$,μ), taking the respective variations, and invoking the appropriate Green's identities, we arrive at the (strong form of) the so-called [illegible], or the [illegible] conditions. [illegible]:

$$\begin{aligned} -\nu \nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) + (\nabla \boldsymbol{u})\boldsymbol{u} + \nabla p + \boldsymbol{b} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{u} &= \boldsymbol{g} \quad \text{on } \Gamma. \end{aligned}$$

[illegible]:

$$\begin{aligned} -\nu \nabla \cdot (\nabla \boldsymbol{\lambda} + \nabla \boldsymbol{\lambda}^T) + (\nabla \boldsymbol{u})^T \boldsymbol{\lambda} - (\nabla \boldsymbol{\lambda})\boldsymbol{u} + \nabla \mu - \nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) &= \boldsymbol{0} \quad \text{in } \Omega, \\ \nabla \cdot \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{in } \Omega, \\ \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{on } \Gamma. \end{aligned}$$

[illegible]:

$$\boldsymbol{b} + \boldsymbol{\lambda} = \boldsymbol{0} \quad \text{in } \Omega.$$

The first set of equations are just the original Navier–Stokes PDEs. The [illegible] equations, which result from stationarity with respect to state variables, are themselves PDEs, and are linear in the Lagrange multipliers $\boldsymbol{\lambda}$ and μ. Finally, the [illegible] are (in this case) algebraic.

The above optimality system is a large, nonlinear, coupled, unstructured system of PDEs (or at least larger, more coupled, and less structured than usually seen by the forward PDE solver). Solving this system presents significant challenges beyond the forward PDE. One way to mitigate some of the difficulties is to eliminate state variables and Lagrange multipliers—and correspondingly, the state equations and adjoint equations—to reduce the system to a manageable one in just the decision variables $\boldsymbol{b}$. This can be done as follows: given $\boldsymbol{b}$ at some iteration, we solve the state equations for the state variables $\boldsymbol{u}, p$. Knowing the state variables then permits us to solve the adjoint equations for the Lagrange multipliers $\boldsymbol{\lambda}, \mu$. Finally, with the states and multipliers known, we can update $\boldsymbol{b}$ by taking a Newton-like iteration on the control equation. The entire process is repeated until convergence. This elimination procedure is termed a [illegible] method, in contrast to a [illegible] method, in which one solves for the states, controls, and multipliers simultaneously. The variable reduction can be done either prior to or after the Newton linearization. The former is a type of [illegible] and is known as a [illegible], and the latter is a [illegible] or [illegible] (SQP).

Reduced space methods are attractive for several reasons. Solving the subsets of equations in sequence imparts some structure to the optimization problem. State equation solvers build on years of development of large-scale parallel PDE solvers. The strong similarities between the state and adjoint operators suggest that an existing PDE solver for the state equations can often be modified readily to handle the adjoint system. Finally, the control equations are often reasonably straightforward, at least to evaluate. Another advantage of reduction is that the full space system is often very ill-conditioned, whereas the three subsystems are typically better conditioned individually.

Given these advantages, the overwhelming popularity of reduced space methods, for optimization of systems governed by PDEs is not surprising. We will not attempt to survey such methods, but rather we give a sample of the literature on applications to various PDE problems: applications to compressible flow airfoil design [33, 40]; heat equation boundary control [28]; inverse parameter estimation [15, 27]; Navier–Stokes flow control [19]; inverse acoustic scattering [16]; chemically reacting flows [39]; and structural optimization [34, 36, 37]. In addition, parallel implementations of reduced sequential quadratic programming (RSQP) methods exhibiting high parallel efficiency and good scalability have been developed [16, 20, 33, 35, 29].

On the other hand, the major disadvantage of reduced methods is the requirement to solve the (possibly linearized) state and adjoint equations [illegible] of the reduced system—which is a direct consequence of the reduction onto the decision variable space. Because of this difficulty it is natural to return to the full space and ask if it is possible to solve simultaneously the entire optimality system but retain the structure-inducing, condition-improving advantages of reduced space methods—while avoiding their disadvantages.

In this two-part article we present such a method. The key idea is to solve in the full space using a Newton method, but precondition with a quasi-Newton reduced space method. We refer to the (nonlinear) Newton iterations as [illegible] iterations, and use the term [illegible] to describe the (linear) Krylov iterations for the Karush–Kuhn–Tucker (KKT) system that arises at each Newton iteration. This linearized KKT

system is solved using a Krylov iterative method (symmetric QMR), and it is this system to which the preconditioner is applied. The reduced space preconditioner is very effective in deflating the spectrum of the KKT matrix and in reducing the number of Krylov iterations—applying it captures the favorable structure of reduced methods. On the other hand, since the reduction is used just as a preconditioner, we can approximate the state and adjoint solves, replacing them for example by their own preconditioners. So we arrive at a method that combines rapid convergence in the outer Newton iteration (typically mesh-independent) with fast convergence of the inner Krylov iteration (which can be as good as mesh-independent).

The new method is termed . It is common in the PDE-solver literature to use the phrase to refer to Newton methods for solving PDEs that employ Krylov linear solvers, with as the preconditioner for the Krylov method. Since is used sometimes to describe a Newton method for solving the optimality system (i.e., an SQP method), and since a reduced space method can be viewed as a Schur complement method for the KKT system, we arrive at the concatenation .

The LNKS method is inspired by domain-decomposed Schur complement PDE solvers. In such techniques, reduction onto the interface space requires exact subdomain solves, so one often prefers to iterate within the full space while using a preconditioner based on approximate subdomain solution [26]. In our case, the decomposition is into states and decisions, as opposed to subdomain and interface spaces.

An early version of an LNKS method was presented in [9] and [10]. Here we extend the method in a number of directions. In part I, we analyze and compare several different variants of the KKT preconditioner. Scalability to larger problems and processor counts is studied. In part II, important globalization and inexactness issues, which were not considered in the earlier work, are introduced and analyzed. We also apply the method to some viscous flow control problems.

Haber and Asher [24] have proposed a KKT preconditioner similar to [9] and [10]. The main differences are in the reduced Hessian preconditioner and the application to inverse problems. Battermann and Heinkenschloss [7] have presented a related KKT-system preconditioner that also makes use of state and decision space decompositions. However, their preconditioner is based on congruence transformations of the original system and not on its exact factorization. The resulting preconditioned system has both positive and negative eigenvalues and its spectrum is less favorably distributed. Another important difference is that we precondition in the reduced space. We also consider parallelism and scalability issues.

Part I of the article is organized as follows: in section 2 we discuss SQP methods and in particular reduced space variants; in section 3 we introduce the LNKS method and present several approaches for preconditioning the KKT matrix; in section 4 we examine the formulation of a Stokes flow problem; and in section 5 we conclude with numerical results, and study parallel and algorithmic scalability of the method.

. We use boldface characters to denote vector-valued functions and vector-valued function spaces. We use roman characters to denote discretized quantities and italics for their continuous counterparts. For example, $\boldsymbol{u}$ will be the continuous velocity field and $\mathbf{u}$ will be its discretization. Greek letters are overloaded and whether we refer to the discretization or the continuous fields should be clear from context. We also use + as a subscript or superscript to denote variable updates within an iterative algorithm.

2. Reduced space methods. We begin with a constrained optimization problem formulated as follows:

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \text{subject to } \mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^N$ are the optimization variables, $f : \mathbb{R}^N \to \mathbb{R}$ is the objective function, and $\mathbf{c} : \mathbb{R}^N \to \mathbb{R}^n$ are the constraints. We assume that the constraints consist of just the (discretized) state equations.[1] In order to exploit the structure of the PDEs, we partition $\mathbf{x}$ into the state variables $\mathbf{x}_s \in \mathbb{R}^n$ and the decision variables $\mathbf{x}_d \in \mathbb{R}^m$,

$$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{x}_s \\ \mathbf{x}_d \end{array} \right\}, \tag{2.2}$$

so that $N = m + n$. By introducing the Lagrangian function $\mathcal{L}$ one can derive first- and higher-order optimality conditions. The Lagrangian is defined by

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{2.3}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ are the Lagrange multipliers (or adjoint variables). The first-order optimality conditions state that at a local minimum the Lagrangian gradient must vanish:

$$\left\{ \begin{array}{c} \partial_x \mathcal{L} \\ \partial_\lambda \mathcal{L} \end{array} \right\} (\mathbf{x}, \boldsymbol{\lambda}) = \left\{ \begin{array}{c} \partial_x f(\mathbf{x}) + (\partial_x \mathbf{c}(\mathbf{x}))^T \boldsymbol{\lambda} \\ \mathbf{c}(\mathbf{x}) \end{array} \right\} = \mathbf{0}. \tag{2.4}$$

Points at which the gradient of the Lagrangian vanishes are often called [illegible]. Equations (2.4) are typically known as the KKT optimality conditions. To simplify the notation further, let us define

$$\begin{array}{lll} \mathbf{A}(\mathbf{x}) := \partial_x \mathbf{c}(\mathbf{x}) & \in \mathbb{R}^{n\times N} & \text{Jacobian matrix of the constraints,} \\ \mathbf{W}(\mathbf{x}, \boldsymbol{\lambda}) := \partial_{xx} f(\mathbf{x}) + \sum_i \lambda_i \partial_{xx} \mathbf{c}_i(\mathbf{x}) & \in \mathbb{R}^{N\times N} & \text{Hessian matrix of the Lagrangian,} \\ \mathbf{g}(\mathbf{x}) := \partial_x f(\mathbf{x}) & \in \mathbb{R}^N & \text{gradient vector of the objective.} \end{array}$$

Consistent with the partitioning of the optimization variables into states and decisions, we logically partition $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$ as follows:[2]

$$\mathbf{g} = \left\{ \begin{array}{c} \mathbf{x}_s \\ \mathbf{x}_d \end{array} \right\}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d \end{bmatrix}, \quad \text{and} \quad \mathbf{W} = \begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} \\ \mathbf{W}_{ds} & \mathbf{W}_{dd} \end{bmatrix}.$$

At each iteration of the SQP method a quadratic programming problem (QP) is solved to generate a new search direction $\mathbf{p}_x$. When SQP is derived from a Newton method for solving the optimality conditions (2.4), the resulting QP is of the form

$$\min_{\mathbf{p}_x\in\mathbb{R}} \frac{1}{2} \mathbf{p}_x^T \mathbf{W} \mathbf{p}_x + \mathbf{g}^T \mathbf{p}_x \quad \text{subject to } \mathbf{A}\mathbf{p}_x + \mathbf{c} = \mathbf{0}. \tag{2.5}$$

[1] However, the methodology can be extended to problems that include additional equality or inequality constraints, e.g., through the use of a partially reduced SQP method [41] as a preconditioner.

[2] All vectors and matrices depend on the optimization variables $\mathbf{x}$; in addition, the Hessian $\mathbf{W}$ depends also on the Lagrange multipliers $\boldsymbol{\lambda}$. For clarity, we suppress identification of these dependencies.

Reduced space methods for solving (2.5) eliminate the linearized constraint by using the null and range space decomposition of the search direction,

$$\mathbf{p}_x = \mathbf{Z}\mathbf{p}_z + \mathbf{Y}\mathbf{p}_y, \tag{2.6}$$

where the columns of $\mathbf{Z} \in \mathbb{R}^{n\times m}$ form a basis for the null space of $\mathbf{A}$ (so that $\mathbf{AZ} = \mathbf{0}$). Note that $\mathbf{Y}$ is often not orthogonal to the null space of the constraints (and hence not a true range space basis for $\mathbf{A}^T$). However, $\mathbf{AY}$ should have full rank to ensure solvability for the (not strictly) range-space component $\mathbf{p}_y$.[3] The decomposition (2.6) permits the $\mathbf{p}_z$ and $\mathbf{p}_y$ components to be computed separately, as shown below. Let us define the reduced gradient of the objective function $\mathbf{g}_z$, the reduced Hessian $\mathbf{W}_z$, and an auxiliary matrix $\mathbf{W}_y$:

$$\begin{aligned} \mathbf{g}_z &:= \mathbf{Z}^T\mathbf{g}, \\ \mathbf{W}_z &:= \mathbf{Z}^T\mathbf{W}\mathbf{Z}, \\ \mathbf{W}_y &:= \mathbf{Z}^T\mathbf{W}\mathbf{Y}. \end{aligned} \tag{2.7}$$

Then the reduced space SQP (RSQP) method (without line search) is given by Algorithm 1. The "+" subscript signifies an updated variable.

Algorithm 1. Reduced space sequential quadratic programming (RSQP).

```
Choose x
loop
   Evaluate c, g, A, W
   g_z = Z^T g
   if ||g_z|| ≤ tol and ||c|| ≤ tol then
      Converged
   end if
   (AY)p_y = −c                                   solve for p_y
   W_z p_z = −(g_z + W_y p_y)                     solve for p_z
   p_x = Z p_z + Y p_y
   x_+ = x + p_x
   (AY)^T λ_+ = −Y^T (g + W p_x)                  solve for λ_+
end loop
```

Choices on how to approximate $\mathbf{W}_z$ and on how to construct the bases $\mathbf{Z}$ and $\mathbf{Y}$ determine some of the different RSQP variants. An important step of this algorithm is "inverting" $\mathbf{W}_z$. The condition number of the reduced Hessian is affected by the choice of $\mathbf{Z}$, and ideally $\mathbf{Z}$ would come from an orthogonal factorization of $\mathbf{A}$. This approach is not possible for discretized PDE constraints, particularly in three dimensions, due to the high computational cost of dense linear algebra operations. A convenient form of $\mathbf{Z}$ that is easy to compute and takes advantage of the structure of the PDE constraints is given by

$$\mathbf{Z} := \begin{bmatrix} -\mathbf{A}_s^{-1}\mathbf{A}_d \\ \mathbf{I} \end{bmatrix}. \tag{2.8}$$

[3] One obvious choice for $\mathbf{Y}$ is $\mathbf{A}^T$ but the computation of $\mathbf{p}_y$ becomes prohibitively expensive since this approach squares the condition number of the PDE Jacobian.

This form is motivated by the fact that $\mathbf{A}_s$ is the (linearized) forward PDE operator and expressing the null basis in terms of the (formal) inverse of $\mathbf{A}_s$ builds on a large body of PDE solver technology. $\mathbf{Y}$ can then be defined by

$$\mathbf{Y} := \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}. \tag{2.9}$$

The expression for the null space basis $\mathbf{Z}$ implies that the reduced gradient is given by

$$\mathbf{g}_z = \mathbf{g}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{g}_s, \tag{2.10}$$

and the reduced Hessian is given by

$$\mathbf{W}_z = \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{ss} \mathbf{A}_s^{-1} \mathbf{A}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{sd} - \mathbf{W}_{ds} \mathbf{A}_s^{-1} \mathbf{A}_d + \mathbf{W}_{dd}. \tag{2.11}$$

Algorithm 2 states a variant of RSQP particularized to the special form of $\mathbf{Z}$ and $\mathbf{Y}$ given above, to which we refer as [illegible] (N-RSQP). The algorithm can be decomposed into three main steps: inverting the reduced Hessian for the decision search direction (step 8), inverting the (linearized) forward operator for the state search direction (step 9), and inverting the adjoint of the (linearized) forward operator for the Lagrange multipliers (step 10).

Algorithm 2. Newton RSQP.

1: Choose $\mathbf{x}_s$, $\mathbf{x}_d$, $\boldsymbol{\lambda}$
2: **loop**
3: Evaluate $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$
4: $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T \boldsymbol{\lambda}$
5: **if** $\|\mathbf{g}_z\| \leq$ tol and $\|\mathbf{c}\| \leq$ tol **then**
6: Converged
7: **end if**
8: $\mathbf{W}_z \mathbf{p}_d = -\mathbf{g}_z + (\mathbf{W}_{ds} - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{W}_{ss}) \mathbf{A}_s^{-1} \mathbf{c}$ solve for $\mathbf{p}_d$ (**Decision step**)
9: $\mathbf{A}_s \mathbf{p}_s = -\mathbf{A}_d \mathbf{p}_d - \mathbf{c}$ solve for $\mathbf{p}_s$ (**State step**)
10: $\mathbf{A}_s^T \boldsymbol{\lambda}_+ = -(\mathbf{g}_s + \mathbf{W}_{ss} \mathbf{p}_s + \mathbf{W}_{sd} \mathbf{p}_d)$ solve for $\boldsymbol{\lambda}_+$ (**Adjoint Step**)
11: $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
12: **end loop**

There are two ways to solve the decision equation in step 8. We can either compute and store the reduced Hessian $\mathbf{W}_z$ for factorization with a direct solver; or we can use an iterative method, which requires only matrix-vector products with $\mathbf{W}_z$. Direct solvers are very effective, but computing the reduced Hessian requires m linearized forward solves (one for each column of $\mathbf{A}_d$ in the product $\mathbf{A}_s^{-1} \mathbf{A}_d$). If we count the solves for the right-hand side of the decision equation and the adjoint and forward equations, then we have a total of $m + 4$ linearized PDE solves [illegible] optimization iteration.

If an iterative method is used to solve the decision equation in step 8, then a matrix-vector multiplication with $\mathbf{W}_z$ requires one solve with each of $\mathbf{A}_s$ and $\mathbf{A}_s$. A Krylov method such as conjugate gradients (CG) will converge in m steps (in exact arithmetic) and thus the number of forward/adjoint solves will not exceed $2m$. In practice the number of iterations depends on the spectrum of the reduced Hessian. With an optimal preconditioner the iteration count can be independent of m.

However, it is not obvious how to devise optimal preconditioners for the reduced Hessian for general problems. Furthermore, we still require four additional forward (and adjoint) solves for each SQP iteration, and two solves per CG iteration.

Even when the number of decision variables is small, it is advantageous to choose an RSQP variant that avoids computing $\mathbf{W}_z$. The main reason is that second derivatives are often difficult to compute. Moreover, Newton's method is not globally convergent and far from the solution the quality of the decision step $\mathbf{p}_z$ is questionable.

Motivated by these difficulties, quasi-Newton RSQP (QN-RSQP) methods have been developed that replace $\mathbf{W}_z$ by a quasi-Newton approximation, which drops the convergence rate from quadratic to superlinear. The QN-RSQP method is summarized in Algorithm 3. In addition, the second derivative terms are dropped from the right-hand sides of the decision and adjoint steps, at the expense of a reduction from one-step to two-step superlinear convergence [8]. An important advantage of this variant of quasi-Newton method is that only two linearized forward/adjoint problems need to be solved at each iteration, as opposed to the m forward solves required by N-RSQP for constructing $\mathbf{A}_s^{-1}\mathbf{A}_d$ in $\mathbf{W}_z$ [19]. Furthermore, no second derivatives are required. Under assumptions on the existence and uniqueness of a local minimum, one can show that the combination of a sufficiently accurate line search and an appropriate quasi-Newton update guarantees a descent search direction and thus a globally convergent algorithm [12].

Algorithm 3. Quasi-Newton RSQP.

1: Choose $\mathbf{x}_s$, $\mathbf{x}_d$, $\boldsymbol{\lambda}$, $\mathbf{B}_z$
2: **loop**
3: Evaluate $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$
4: $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T\boldsymbol{\lambda}$
5: **if** $\|\mathbf{g}_z\| \leq$ tol and $\|\mathbf{c}\| \leq$ tol **then**
6: Converged
7: **end if**
8: $\mathbf{B}_z\mathbf{p}_d = -\mathbf{g}_z$ solve for $\mathbf{p}_d$ (**Decision step**)
9: $\mathbf{A}_s\mathbf{p}_s = -\mathbf{A}_d\mathbf{p}_d - \mathbf{c}$ solve for $\mathbf{p}_s$ (**State step**)
10: $\mathbf{A}_s^T\boldsymbol{\lambda}_+ = -\mathbf{g}_s$ solve for $\boldsymbol{\lambda}_+$ (**Adjoint Step**)
11: $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
12: Update $\mathbf{B}_z$
13: **end loop**

QN-RSQP has been parallelized efficiently for moderate numbers of decision variables [29]; the parallelism is fine-grained across the state and adjoint fields. Unfortunately, the number of iterations taken by quasi-Newton methods often increases as the number of decision variables grows,[4] rendering large-scale problems intractable. Additional processors will not help since the bottleneck is in the iteration dimension.

On the other hand, convergence of the nonlinear iterations of N-RSQP method can be independent of the number of decision variables m, particularly when these variables are mesh related. But unless an optimal $\mathbf{W}_z$ preconditioner can be used, the large number of forward/adjoint solves per iteration preclude its use, particularly on a parallel machine, where iterative methods for the forward problem must be used.

[4]For example, for the limiting case of a quadratic programming problem, the popular BFGS quasi-Newton method is equivalent to conjugate gradients, which scales with the square root of the condition number of the reduced Hessian.

However, there is a way to exploit the fast convergence of the Newton method and avoid solving the PDEs exactly, and this method is proposed in the next section. The basic idea is to solve the full space system (2.4) simultaneously for states, adjoints, and decision variables but invoke QN-RSQP as a preconditioner. Since it is just a preconditioner, we can avoid exact solution of the linearized state and adjoint equations in favor of applications of state and adjoint preconditioners.

3. LNKS method. The KKT optimality conditions (2.4) define a system of nonlinear equations. The Jacobian $\mathbf{K}$ of this system is termed the [illegible]. A Newton step on the optimality conditions is given by

$$\begin{bmatrix} \mathbf{W} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_x \\ \mathbf{p}_\lambda \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} \\ \mathbf{c} \end{Bmatrix} \quad (\text{or } \mathbf{K}\mathbf{v} = -\mathbf{h}), \tag{3.1}$$

where $\mathbf{p}_x$ and $\mathbf{p}_\lambda$ are the updates of $\mathbf{x}$ and $\boldsymbol{\lambda}$ from current to next iterations. Assuming sufficient smoothness, and that the initial guess is sufficiently close to a solution, Newton steps obtained by the above system will converge quadratically to a solution of the KKT system [17]. Thus, the forward solves required for reduced methods can be avoided by remaining in the full space of state and decision variables, since it is the reduction onto the decision space that necessitates the solves with $\mathbf{A}_s^{-1}$. Nevertheless, the full space approach presents difficulties too: a descent direction is not guaranteed, second derivatives are required, and the KKT system itself can be difficult to solve. The size of the KKT matrix is more than twice that of the forward problem, and it is expected to be very ill-conditioned. Ill-conditioning results not only from the ill-conditioning of the forward problem, but also from the different scales between first and second derivatives submatrices. Moreover, the system is indefinite; mixing negative and positive eigenvalues is known to slow down Krylov solvers. Therefore, a good preconditioner is essential to make the method efficient, and that is the main focus of Part I of this two-part article.

The LNKS method we present here uses a proper Newton method to solve the KKT optimality conditions. To compute the Newton step we solve the KKT system using an appropriate Krylov method. At the core of the algorithm lies the preconditioner $\mathbf{P}$ for the Krylov method, which in our case is an [illegible] version of the QN-RSQP algorithm. An outline of the local LNKS algorithm is given by Algorithm 4. In part II, we discuss and analyze globalization and inexactness aspects of LNKS.

Algorithm 4. Lagrange–Newton–Krylov–Schur.

1: Choose $\mathbf{x}$, $\boldsymbol{\lambda}$
2: **loop**
3: Check for convergence
4: Compute $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$
5: Solve $\mathbf{P}^{-1}\mathbf{K}\mathbf{v} = -\mathbf{P}^{-1}\mathbf{h}$
6: Update $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
7: Update $\boldsymbol{\lambda}_+ = \boldsymbol{\lambda} + \mathbf{p}_\lambda$
8: **end loop**

This algorithm will produce the same steps taken by N-RSQP (in exact arithmetic, with exact solves and without globalization). It is easier to see how RSQP can be used to precondition LNKS if we rewrite the Newton step (3.1) in a block-partitioned

form:

$$\begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} & \mathbf{A}_s^T \\ \mathbf{W}_{ds} & \mathbf{W}_{dd} & \mathbf{A}_d^T \\ \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_s \\ \mathbf{p}_d \\ \mathbf{p}_\lambda \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g}_s + \mathbf{A}_s^T \boldsymbol{\lambda} \\ \mathbf{g}_d + \mathbf{A}_d^T \boldsymbol{\lambda} \\ \mathbf{c} \end{Bmatrix}. \tag{3.2}$$

RSQP is equivalent to a block-row elimination; given $\mathbf{p}_d$, solve the last block of equations for $\mathbf{p}_s$, then solve the first to find $\mathbf{p}_\lambda$, and finally solve the middle one for $\mathbf{p}_d$, the search direction for the decision variables. Therefore RSQP can be written as a particular block factorization of the KKT matrix:

$$\mathbf{K} = \begin{bmatrix} \mathbf{W}_{ss}\mathbf{A}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds}\mathbf{A}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T\mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{sd} - \mathbf{W}_{ss}\mathbf{A}_s^{-1}\mathbf{A}_d & \mathbf{A}_s^T \end{bmatrix}. \tag{3.3}$$

Note that these factors are permutable to block triangular form (and therefore this is equivalent to a block-LU factorization) and that $\mathbf{W}_z$ is the Schur complement for $\mathbf{p}_d$.

This block factorization suggests a preconditioner created by replacing the reduced Hessian $\mathbf{W}_z$ with an approximation $\mathbf{B}_z$, for example a quasi-Newton approximation. However, we still require four forward solves per Krylov iteration. One way to restore the two solves per iteration of QN-RSQP is to, in addition, drop second-order information from the preconditioner, just as one does when going from N-RSQP to QN-RSQP. A further simplification of the preconditioner is to replace the exact forward operator $\mathbf{A}_s$ by an approximation $\tilde{\mathbf{A}}_s$, which could be any appropriate forward problem preconditioner. With these changes, $\tilde{\ }$ forward solves need to be performed at each inner (Krylov) iteration. Thus, the work per inner iteration becomes linear in the state and adjoint variable dimension (e.g., when $\tilde{\mathbf{A}}_s$ is a constant-fill domain decomposition approximation). Furthermore, when $\mathbf{B}_z$ is based on a limited-memory quasi-Newton update, the work per inner iteration is also linear in the decision variable dimension. Since all of the steps involved in an inner iteration not only require linear work but are also readily parallelized at the fine-grained level, we conclude that each inner (Krylov) iteration will execute with high parallel efficiency and scalability.

Scalability of the entire method additionally requires mesh-independence of both inner and outer iterations. Newton methods (unlike quasi-Newton) are often characterized by a number of iterations that is independent of problem size [1]. With an "optimal" forward preconditioner and a good $\mathbf{B}_z$ approximation, we can expect that the number of inner iterations is also insensitive to the problem size. Scalability with respect to both state and decision variables would then result. Several KKT preconditioning variations on this theme are discussed in the next subsection, and some choices for the reduced Hessian preconditioner $\mathbf{B}_z$ are presented in the following subsection.

3.1. Preconditioning variants for the KKT system. Below we present several preconditioners for the KKT matrix. They are based on the block-factorization of the KKT matrix (3.3) and are therefore indefinite matrices. To examine separately the effects of discarding the Hessian terms and approximating the forward solver, we define four different variations. The subscript denotes the number of the forward PDE solves per Krylov iteration, and a tilde mark on (top of) the preconditioner symbol means that a forward or adjoint solve is replaced by a single application of its preconditioner.

• Preconditioner $\mathbf{P}_4$ replaces $\mathbf{W}_z$ by $\mathbf{B}_z$ and retains the four exact linearized PDE solves per iteration. The preconditioner is

$$\mathbf{P}_4 = \begin{bmatrix} \mathbf{W}_{ss}\mathbf{A}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds}\mathbf{A}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T\mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_y & \mathbf{A}_s^T \end{bmatrix}, \tag{3.4}$$

and the preconditioned KKT matrix is

$$\mathbf{P}_4^{-1}\mathbf{K} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{3.5}$$

• Preconditioner $\mathbf{P}_2$ replaces $\mathbf{W}_z$ by $\mathbf{B}_z$ and discards all other Hessian terms, resulting in two exact linearized PDE solves per iteration. The preconditioner is

$$\mathbf{P}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_d^T\mathbf{A}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_s^T \end{bmatrix}, \tag{3.6}$$

and the preconditioned KKT matrix is

$$\mathbf{P}_2^{-1}\mathbf{K} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_y^T\mathbf{A}_s^{-1} & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{W}_{ss}\mathbf{A}_s^{-1} & \mathbf{W}_y\mathbf{B}_z^{-1} & \mathbf{I} \end{bmatrix}. \tag{3.7}$$

Note that the spectrum of the preconditioned KKT matrix is unaffected by dropping the second derivative terms.

• Preconditioner $\tilde{\mathbf{P}}_4$ replaces $\mathbf{W}_z$ by $\mathbf{B}_z$ and $\mathbf{A}_s$ by $\tilde{\mathbf{A}}_s$, and retains all other Hessian terms, resulting in four preconditioner applications but no forward/adjoint solves. The preconditioner is

$$\tilde{\mathbf{P}}_4 = \begin{bmatrix} \mathbf{W}_{ss}\tilde{\mathbf{A}}_s^{-1} & \mathbf{0} & \mathbf{I} \\ \mathbf{W}_{ds}\tilde{\mathbf{A}}_s^{-1} & \mathbf{I} & \mathbf{A}_d^T\tilde{\mathbf{A}}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{A}}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{W}}_y & \tilde{\mathbf{A}}_s^T \end{bmatrix}, \tag{3.8}$$

and the preconditioned KKT matrix is

$$\tilde{\mathbf{P}}_4^{-1}\mathbf{K} = \begin{bmatrix} \mathbf{I}_s & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_s^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathcal{O}(\mathbf{E}_s) & \mathbf{0} \\ \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) \\ \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) & \mathbf{0} \end{bmatrix}, \tag{3.9}$$

where $\mathbf{E}_s := \mathbf{A}_s^{-1} - \tilde{\mathbf{A}}_s^{-1}$ and $\mathbf{I}_s := \mathbf{A}_s\tilde{\mathbf{A}}_s^{-1}$. The "cross-Hessian" $\tilde{\mathbf{W}}_y$ is given by (2.7) with the exception that the exact PDE solve $\mathbf{A}_s^{-1}$ is replaced by a preconditioner application $\tilde{\mathbf{A}}_s^{-1}$ in the definition of $\mathbf{Z}$. Clearly, $\mathbf{E}_s = \mathbf{0}$ and $\mathbf{I}_s = \mathbf{I}$ for exact forward PDE solves.

• Preconditioner $\tilde{\mathbf{P}}_2$ replaces $\mathbf{W}_z$ by $\mathbf{B}_z$ and $\mathbf{A}_s$ by $\tilde{\mathbf{A}}_s$, and drops all other Hessian terms, resulting in two preconditioner applications and no forward/adjoint solves. The preconditioner is

$$\tilde{\mathbf{P}}_2 = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{I} & \mathbf{A}_d^T\tilde{\mathbf{A}}_s^{-T} \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{A}}_s & \mathbf{A}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_z & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \tilde{\mathbf{A}}_s^T \end{bmatrix}, \tag{3.10}$$

and the preconditioned KKT matrix is

$$\tilde{\mathbf{P}}_2^{-1}\mathbf{K} = \begin{bmatrix} \mathbf{I}_s & \mathbf{0} & \mathbf{0} \\ \tilde{\mathbf{W}}_y^T & \mathbf{W}_z\mathbf{B}_z^{-1} & \mathbf{0} \\ \mathbf{W}_{ss}\tilde{\mathbf{A}}_s^{-1} & \tilde{\mathbf{W}}_y^T & \mathbf{I}_s^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathcal{O}(\mathbf{E}_s) & \mathbf{0} \\ \mathbf{0} & \mathcal{O}(\mathbf{E}_s) & \mathcal{O}(\mathbf{E}_s) \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.11}$$

We review some basic properties of the Krylov methods that will allow us to understand the effectiveness of the preconditioners. The performance of a Krylov method depends highly on the preconditioned operator spectrum [22, 38]. In most Krylov methods the number of iterations required to obtain the solution is at most equal to the number of distinct eigenvalues (in exact arithmetic). When solving $\mathbf{Kv} = -\mathbf{h}$ such methods satisfy the following relation for the residual at the ith iteration:

$$\mathbf{r}_i = \text{span}\left\{\mathbf{r}_0 + \mathbf{K}\mathbf{r}_0 + \mathbf{K}^2\mathbf{r}_0 + \cdots + \mathbf{K}^i\mathbf{r}_0\right\} = \psi(\mathbf{K})\mathbf{r}_0, \quad \psi \in \mathcal{P}_i.$$

Here $\mathbf{v}_0$ is the initial guess, $\mathbf{r}$ is defined as $\mathbf{r} = \mathbf{h} + \mathbf{K}\mathbf{v}$, $\mathbf{r}_0 = \mathbf{h} + \mathbf{K}\mathbf{v}_0$, and $\mathcal{P}_i$ is the space of monic polynomials of degree at most i. Minimum residual methods like MINRES and GMRES determine a polynomial ψ so that[5]

$$\|\mathbf{r}_i\| = \min_{\psi \in \mathcal{P}_i} \|\psi(\mathbf{K})\mathbf{r}_0\|.$$

If $\mathbf{K}$ is diagonalizable (with spectrum $\mathcal{S}(\mathbf{K})$ and $\mathbf{X}$ the matrix or its eigenvectors), then[6]

$$\frac{\|\mathbf{r}_i\|}{\|\mathbf{r}_0\|} \leq \text{cond}(\mathbf{X}) \min_{\lambda \in \mathcal{S}(\mathbf{K})} |\psi(\lambda)|. \tag{3.12}$$

Additionally, if the spectrum lies on the positive real axis, it can be shown that

$$\frac{\|\mathbf{r}_i\|}{\|\mathbf{r}_0\|} \leq \text{cond}(\mathbf{X}) \frac{\lambda_{\max}}{\lambda_{\min}}. \tag{3.13}$$

From (3.12) we can see why clustering the eigenvalues is important. Indeed, if the eigenvalues are clustered, then the solution polynomial in the right-hand side of (3.12) can be approximated by the zeros of a low-degree polynomial (resulting in fewer Krylov iterations). When $\mathbf{K}$ is normal (unitarily diagonalizable) then $\text{cond}(\mathbf{X}) = 1$; in this case the estimates (3.12) and (3.13) are known to be sharp [22].

The preconditioned KKT matrix $\mathbf{P}_4^{-1}\mathbf{K}$ is a block-diagonal matrix with two unit eigenvalues and the m eigenvalues of the preconditioned reduced Hessian. It is immediate that GMRES takes $\mathcal{O}(\text{cond}(\mathbf{B}_z^{-1}\mathbf{W}_z))$ or at most $m + 2$ steps to converge for preconditioner $\mathbf{P}_4$. This is similar to the complexity estimates for a Krylov-based solution of the $\mathbf{B}_z$-preconditioned reduced Hessian in N-RSQP. Preconditioner $\mathbf{P}_2$ has the same effect on the spectrum of the KKT matrix, but the preconditioned KKT system is no longer a normal operator. Yet, if $\mathbf{P}_2^{-1}\mathbf{K}$ is diagonalizable and its eigenvector matrix well conditioned, relation (3.13) is still a good indicator of the effectiveness of the preconditioner.

A more efficient approach, of course, is to replace the exact forward operators $\mathbf{A}_s$ by their forward preconditioners $\tilde{\mathbf{A}}_s$, as in $\tilde{\mathbf{P}}_4$ or $\tilde{\mathbf{P}}_2$. In this case, the preconditioned KKT matrix assumes a more complicated structure. We write $\tilde{\mathbf{P}}_4^{-1}\mathbf{K}$ as the sum

[5] Throughout this paper we use the vector 2-norm for norms and condition numbers.

[6] Here λ is temporary overloaded to denote an eigenvalue of $\mathbf{K}$.

of two matrices; the second matrix in this sum includes terms that approach $\mathbf{0}$ as the forward preconditioner improves. The spectrum of the first matrix in (3.9) is given by $\mathcal{S}_n = \mathcal{S}(\mathbf{B}_z^{-1}\mathbf{W}_z) \cup \mathcal{S}(\tilde{\mathbf{A}}_s^{-1}\mathbf{A}_s)$. If $\tilde{\mathbf{A}}_s^{-1}\mathbf{A}_s$ is normal, then by the Bauer–Fike theorem [21] the eigenvalues are not sensitive to small KKT matrix perturbations and $\mathcal{S}(\tilde{\mathbf{P}}_4^{-1}\mathbf{K}) \cong \mathcal{S}_n$. Hence, a good preconditioner for the forward problem would bring the spectrum of $\tilde{\mathbf{P}}_4^{-1}\mathbf{K}$ close to the spectrum of $\mathbf{P}_4^{-1}\mathbf{K}$. Thus, we would expect a similar iteration count to N-RSQP, but at a much reduced cost per iteration—since the PDE solves would be replaced by preconditioner applications. For preconditioner $\tilde{\mathbf{P}}_2$, the left matrix in preconditioner (3.11) is not normal and may have ill-conditioned eigenvalues. In our numerical experiments, however, the number of Krylov iterations was unaffected by switching from $\tilde{\mathbf{P}}_4$ to $\tilde{\mathbf{P}}_2$.

Finally, we comment on our choice of a Krylov method for solving the symmetric indefinite KKT system (3.3) that has been preconditioned with any of the four preconditioners described above. The most popular method for large symmetric indefinite systems is the MINRES method. However, MINRES requires positive definite preconditioners, and the KKT preconditioners defined above are indefinite. Indefinite preconditioners are allowed in the quasi-minimum residual (QMR) method [18]. The version commonly used is the transpose-free QMR algorithm, which is designed for general nonsymmetric problems and requires two matrix-vector multiplications per Krylov iteration. In our implementation we use another variant, described in [18], that exploits symmetry and uses only one matrix-vector multiplication per iteration.

3.2. Preconditioners for the reduced Hessian. A very important component of the LNKS preconditioners is an approximation to the inverse of the reduced Hessian. In the preceding sections we suggested that a quasi-Newton approximation be used in place of the exact reduced Hessian $\mathbf{W}_z$, in the spirit of using QN-RSQP as a preconditioner. Below, we elaborate upon this choice and describe a two-step stationary iterative preconditioner that we use either stand-alone or to initialize the quasi-Newton preconditioner.

Limited memory quasi-Newton updates. Quasi-Newton methods are popular methods for approximating the reduced Hessian matrix in constrained optimization. Thus, they are natural candidates to precondition the exact reduced Hessian. In particular, we use the Broyden–Fletcher–Goldberg–Shanno (BFGS) update [32, section 8.1]. At each outer iteration we compute

$$\mathbf{s} = \mathbf{p}_d^{(+)} - \mathbf{p}_d,$$
$$\mathbf{y} = \mathbf{g}_z^{(+)} - \mathbf{g}_z$$

and update the approximation to the inverse of the reduced Hessian by

$$\mathbf{B}_+^{-1} = (I - \omega \mathbf{y}\mathbf{s}^T)^T \mathbf{B}^{-1} (I - \omega \mathbf{y}\mathbf{s}^T) + \omega \mathbf{s}\mathbf{s}^T, \qquad \omega = \frac{1}{\mathbf{y}^T\mathbf{s}}. \tag{3.14}$$

If we encounter an iterate that has $\mathbf{y}^T\mathbf{s} \leq \kappa\|\mathbf{y}\|$ we skip the update. $\mathbf{B}^{-1}$ is dense; thus for large decision-space problems, storing or operating with this matrix can be expensive or even impossible. Furthermore, applying the dense $\mathbf{B}^{-1}$ to a vector in parallel involves significant all-to-all communication and may be too expensive for problems that require large numbers of processors. Thus, we have chosen a limited memory variant of BFGS (L-BFGS), which uses a limited number of vectors to approximate the inverse of the reduced Hessian. When updating we do not compute (3.14) but instead we simply store $\mathbf{s}$ and $\mathbf{y}$. Then the action of $\mathbf{B}^{-1}$ on a vector $\mathbf{x}$ is given in

ALGORITHM 5. LIMITED MEMORY BFGS.

$\mathbf{q} = \mathbf{x}$, and $l =$ number of stored vectors
for $i = k-1, k-2, \ldots, k-l$ **do**
  $\alpha_i = \omega_i \mathbf{s}_i^T \mathbf{q}$
  $\mathbf{q} = \mathbf{q} - \alpha_i \mathbf{y}_i$
end for
$\mathbf{z} = \mathbf{B}_{z,0}^{-1} \mathbf{q}$
for $i = k-l, k-l+1, \ldots, k-1$ **do**
  $\beta = \omega_i \mathbf{y}_i^T \mathbf{z}$
  $\mathbf{z} = \mathbf{z} + \mathbf{s}_i(\alpha_i - \beta)$
end for

Algorithm 5. Analysis and details of the convergence properties of this algorithm can be found in [32, sections 9.1–9.2]. We generate the initial inverse reduced Hessian approximation, $\mathbf{B}_{z,0}$, using the stationary iterative preconditioner presented below.

Two-step stationary method. As an initialization for the L-BFGS approximation above, which can also be used as a stand-alone preconditioner, we turn to stationary iterative methods. Such methods provide us with a constant preconditioner as required by most Krylov methods, while effecting an approximate solution of the reduced Hessian system. To guarantee convergence, the most popular methods (such as Jacobi and SOR) require the iteration matrix to have spectral radius less than one, which is difficult to guarantee in the general case. Furthermore, most stationary iterative methods require some kind of splitting, which is not convenient for matrix-free treatment of operators like the reduced Hessian. A method that does not have the above restrictions is a two-step stationary iterative method, first presented by Frankel in 1950 [3, section 5.2]. The method is suitable for positive-definite matrices but requires an accurate estimate of the minimum and maximum eigenvalues. The preconditioner based on this method can be arbitrarily effective, depending on the number of iterations L it is carried to. If $\lambda_1 \leq \lambda_m$ are estimates for the extreme eigenvalues of $\tilde{\mathbf{W}}_z$, then the application of the preconditioner to a vector $\mathbf{d}_{\text{in}}$ is given by Algorithm 6. Step 4 requires the action of the reduced Hessian on a vector. To avoid exact forward solves, we use the approximate reduced Hessian $\tilde{\mathbf{W}}_z$ instead of $\mathbf{W}_z$. We obtain $\tilde{\mathbf{W}}_z$ from (2.11) if we replace the linearized forward PDE operators $\mathbf{A}_s^{-1}$ and $\mathbf{A}_s^{-T}$ with their preconditioners $\tilde{\mathbf{A}}_s^{-1}$ and $\tilde{\mathbf{A}}_s^{-T}$. Like CG, the method's convergence rate depends on the square root of the condition number of the (approximate) reduced Hessian. It parallelizes well because it does not require any inner products, and because in our context the matrix-vector products require just two forward preconditioner applications. To obtain estimates of the extremal eigenvalues, we use a few iterations of the Lanczos method (once per Newton iteration).

Other possibilities for reduced Hessian preconditioning. In addition to the quasi-Newton and two-step stationary iterative preconditioners we use, there are several other possibilities, which we comment on below.

- [illegible]. Incomplete factorizations are popular and robust preconditioners, but an assembled matrix (modulo some exceptions for structured grids) is required. Not only is the reduced Hessian usually prohibitive to assemble (for large-scale PDE optimization problems), but it is in general dense and thus impossible to store for a large number of decision variables. An incomplete factorization would be feasible only if the exact solves are replaced by the forward preconditioner and if some kind of sparsity is enforced (perhaps via element-by-element computations and

ALGORITHM 6. TWO-STEP STATIONARY ITERATIVE REDUCED SPACE PRECONDITIONER.

1: $\rho = \frac{1-\lambda_1/\lambda_n}{1+\lambda_1/\lambda_n}$, $\alpha = \frac{2}{1+(1-\rho^2)^{1/2}}$, $\beta_0 = \frac{1}{\lambda_1+\lambda_n}$, $\beta = \frac{2\alpha}{\lambda_1+\lambda_n}$,
2: $\mathbf{r} = -\mathbf{d}_{\text{in}}$, $\mathbf{d}_0 = \mathbf{0}$, $\mathbf{d}_1 = \beta_0\mathbf{r}$
3: **for** $i = 1 \cdots L$ **do**
4: $\quad \mathbf{r} = \tilde{\mathbf{W}}_z\mathbf{d}_1 - \mathbf{d}_{\text{in}}$
5: $\quad \mathbf{d} = \alpha\mathbf{d}_1 + (1-\alpha)\mathbf{d}_0 - \beta\mathbf{r}$
6: $\quad \mathbf{d}_0 = \mathbf{d}_1$, $\mathbf{d}_1 = \mathbf{d}$
7: **end for**

threshold-ILU methods).

• [illegible]. SPAI techniques are attractive since they require just matrix-vector multiplications and a given sparsity pattern. In [14] an analysis for CFD-related Schur complements shows that this method is promising.

• [illegible]. In [31] two different reduced Hessian preconditioners are presented, one based on a state-Schur complement factorization and the second on power series expansions. It is assumed that a (nonsingular approximation) $\tilde{\mathbf{W}}^{-1}$ is available. The basic component of these preconditioners is the application of $\mathbf{Z}^T\tilde{\mathbf{W}}^{-1}\mathbf{Z}$ on a vector. We did not test this method because in our problems $\mathbf{W}$ has thousands of zero eigenvalues and it is not clear how to construct an approximation to $\mathbf{W}^{-1}$. If $\mathbf{W}_{dd}$ is nonsingular, then a block-Jacobi-ILU technique could be used to approximate $\mathbf{W}_z^{-1}$ with $\mathbf{W}_{dd}^{-1}$ (or an approximation $\tilde{\mathbf{W}}_{dd}^{-1}$).

• [illegible]. Another option is to take a few CG iterations in the reduced space at each preconditioner application. Since we want to avoid solving the forward problem exactly, we replace $\mathbf{A}_s^{-1}$ with $\tilde{\mathbf{A}}_s^{-1}$ in (2.11). We experimented with this approach but found that full convergence of CG was needed to avoid loss of orthogonality for the Krylov vectors. This slowed down the method significantly. We have experimented also with flexible GMRES for the KKT solver, but the nice properties of a symmetric Krylov solver are lost and the algorithm was slow to converge.

• [illegible]. For PDE-constrained optimization problems one may be able to derive an approximate representation of the infinite-dimensional reduced Hessian. A discretization of this expression can then be used to precondition the reduced Hessian, e.g., [2].

Notwithstanding the above alternatives, we feel the choice of a limited memory quasi-Newton approximation, initialized by Frankel's two-step stationary iterative method, strikes the best balance between effectiveness, generality, and parallel scalability. In the next section we present numerical experiments that demonstrate the effectiveness of the stationary preconditioner. To isolate the effects of the (inner) Krylov–Schur solver on LNKS, we study problems with linear constraints and quadratic objective functions. For this class of problems, the KKT system is linear, Newton's method trivially requires just one step to converge, and globalization is not an issue. In part II of this article, we discuss how the limited memory quasi-Newton scheme can be used to safeguard the Newton step.

4. Numerical results. The LNKS method has been tested on two Stokes optimal flow control problems that give rise to quadratic programming problems. Both problems include the three-dimensional (3D) Stokes equations as PDE constraints, and Dirichlet boundary conditions on some portion of the boundary as decision variables. They differ in the choice of objective function: the first is a matching velocity

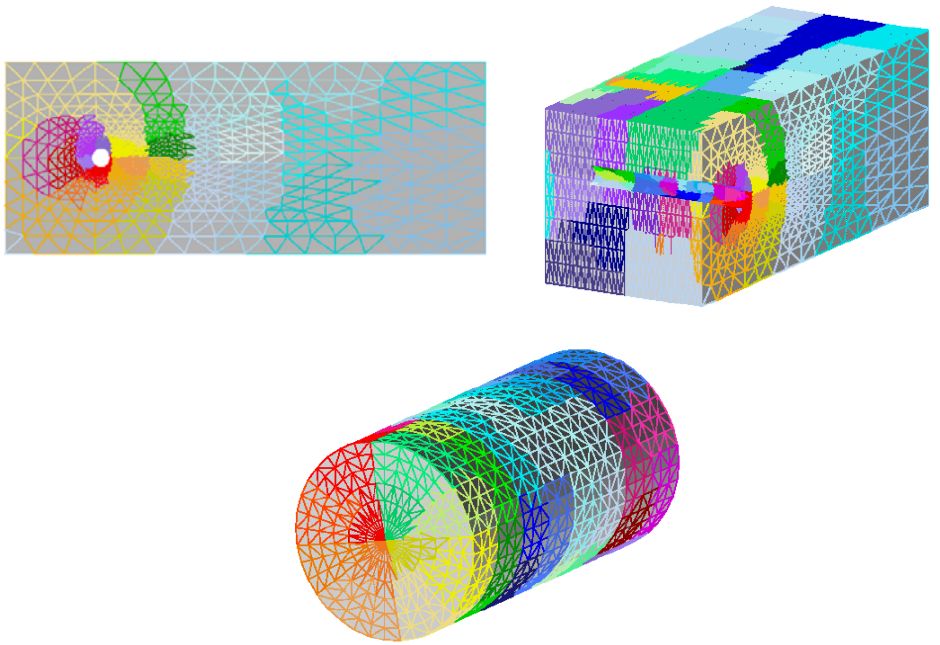


FIG. 1. *This figure depicts the (partitioned) computational meshes used in our numerical experiments. We have solved two different optimal flow control problems: that of Poiseuille flow in a pipe (bottom) and that of a flow around a cylinder embedded inside a rectangular duct (top). Both flows are interior flow problems with nonslip conditions everywhere except the outflow boundary. In the case of the Poiseuille flow the decision variables are the boundary conditions on the circumferential walls of the pipe. In the cylinder problem the decision variables are the downstream cylinder surface boundary conditions.*

problem for a Poiseuille flow, and the second is an energy dissipation minimization problem for a flow around a cylinder. Below, we define the two test problems, discuss a forward preconditioner (which as we have seen plays a key role in the effectiveness of the KKT preconditioner), and provide numerical results for both problems.

4.1. Optimal control of Stokes flows. An optimal flow control problem can be stated as follows:

$$\min_{\boldsymbol{u},\boldsymbol{u}_d,p} \mathcal{J}(\boldsymbol{u},\boldsymbol{u}_d,p) \tag{4.1}$$

$$\begin{aligned} \text{subject to} \quad & -\nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)+\nabla p=\boldsymbol{b} \text{ in } \Omega, \\ & \nabla\cdot\boldsymbol{u}=0 \quad \text{in } \Omega, \\ & \boldsymbol{u}=\boldsymbol{u}^* \quad \text{on } \Gamma_u, \\ & \boldsymbol{u}=\boldsymbol{u}_d \quad \text{on } \Gamma_d, \\ & -p\,\boldsymbol{n}+\nu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)\boldsymbol{n}=\boldsymbol{0} \quad \text{on } \Gamma_N. \end{aligned} \tag{4.2}$$

Here, $\boldsymbol{u}$ is the fluid velocity, p the pressure, ν a nondimensional viscosity, $\boldsymbol{b}$ a body force, and $\boldsymbol{n}$ the outward unit normal vector on the boundary. The first problem is a velocity matching optimal control problem. A quadratic velocity profile is prescribed on the inflow boundary Γ_u, and we specify a traction-free outflow boundary Γ_N. The velocities $\boldsymbol{u}_d$, defined on Γ_d, are the decision variables. In this example, $\boldsymbol{u}^*$ is taken as a Poiseuille flow inside a pipe, and the decision variables correspond to boundary velocities on the circumferential surface of the pipe (Figure 1). The objective functional

is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{u}_d, p) = \frac{1}{2}\int_\Omega (\boldsymbol{u}^* - \boldsymbol{u})^2 d\Omega + \frac{\rho}{2}\int_{\Gamma_d} \boldsymbol{u}_d^2 d\Omega,$$

where ρ penalizes the "cost" of the controls. Notice that the size of ρ effects the condition number of the reduced Hessian. Here we keep ρ constant and equal to 1.

In the second problem we control the flow around a cylinder that is anchored inside a rectangular duct. A quadratic velocity profile is used as an inflow Dirichlet condition and we prescribe a traction-free outflow. The decision variables are defined to be the velocities on the downstream portion of the cylinder surface. In this problem the objective functional is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{u}_d, p) = \frac{1}{2}\int_\Omega |\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T|^2 d\Omega + \frac{\rho}{2}\int_{\Gamma_d} \boldsymbol{u}_d^2 d\Omega.$$

In this problem we also choose $\rho = 1$.

The Stokes equations are discretized by the Galerkin finite element method, using tetrahedral Taylor–Hood elements [23]. We implemented the reduced-space algorithm as well as the LNKS method with the four different preconditioners. Our code is based on the PETSc library [4, 5, 6] and makes use of PETSc parallel domain-decomposition preconditioners for the approximate forward solves. We use the symmetric QMR for both the KKT and Stokes linear systems. The two flow control problems have quadratic objective functions and linear constraints and thus Newton's method takes only one iteration to converge. Hence, it is not possible to build a quasi-Newton approximation (as described above) to use within the KKT preconditioner (but see [30] for other options). Instead, we use the two-step stationary algorithm to precondition the reduced Hessian.

4.2. The Stokes forward preconditioner. It is evident that one of the two major components of the LNKS method is the forward solver preconditioner (the choice of which is, of course, problem-dependent). The Stokes problem, in algebraic form, is given by

$$\begin{bmatrix} \mathbf{V} & \mathbf{P}^T \\ \mathbf{P} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{u} \\ \mathbf{p} \end{Bmatrix} = \begin{Bmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{Bmatrix}. \tag{4.3}$$

In our earlier numerical experiments (results are reported in [9, 10]) we used the following preconditioner:

$$\begin{bmatrix} \mathbf{V}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^{-1} \end{bmatrix}. \tag{4.4}$$

PETSc's block-Jacobi preconditioners with local ILU(0) were used to provide a domain decomposition approximation of $\mathbf{V}^{-1}$ and $\mathbf{M}^{-1}$, which are the discrete Laplacian and discrete pressure "mass" matrices, respectively. We found the performance of this preconditioner somewhat unsatisfactory, especially when it was part of the KKT preconditioner. For this reason we have switched to a preconditioner that is based on the following exact factorization of the inverse of (4.3),

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}^{-1}\mathbf{P}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P}\mathbf{V}^{-1} & \mathbf{I} \end{bmatrix}, \tag{4.5}$$

where $\mathbf{S} := -\mathbf{P}\mathbf{V}^{-1}\mathbf{P}^T$ is the Schur complement for the pressure. Based on this factorization, the Stokes preconditioner is defined by replacing the exact solves $\mathbf{V}^{-1}$ with domain decomposition approximations $\tilde{\mathbf{V}}^{-1}$. For the pressure Schur complement block, we use the two-step stationary method described in Algorithm 6. Performance statistics are presented in Table 4.1. We can see that the factorization-based preconditioner reduces significantly solution times compared to the earlier block-Jacobi variant. Nevertheless, the efficiency of the new preconditioner is still mesh-dependent. It is known that block-Jacobi-ILU preconditioners require work that is superlinear in problem size. This can be overcome with an additive Schwarz domain-decomposition preconditioner, with generous overlap and a coarse grid component [13].

TABLE 4.1

Effectiveness of various preconditioners for Stokes forward solver with increasing number of processors. The first two columns list the number of processors (PEs) *and the problem size* (n). *The remaining columns give the number of Krylov iterations required to satisfy* $\|\mathbf{r}\|/\|\mathbf{r}_0\| \leq 1 \times 10^{-8}$*;* none *signifies no preconditioner;* bj *is a block-Jacobi preconditioner, with each diagonal block mapped to a processor and approximated by ILU*(0)*;* psc3 *is the factorization-based preconditioner, with three (two-step) stationary iterations for the Schur complement block* $\mathbf{S}$*;* psc7 *employs seven stationary iterations. In parentheses is wall-clock time in seconds on the Pittsburgh Supercomputing Center's Cray T3E-900.*

Poiseuille flow

PEs	n	None	bj	psc3	psc7
16	64,491	18,324 (250)	1,668 (86)	140 (30)	128 (39)
32	114,487	24,334 (620)	2,358 (221)	197 (39)	164 (45)
64	280,161	27,763 (1410)	3,847 (515)	294 (71)	249 (84)
128	557,693	32,764 (1297)	5,010 (620)	446 (109)	357 (123)
256	960,512	49,178 (2272)	6,531 (780)	548 (122)	389 (128)

Flow around a cylinder

PEs	n	None	bj	psc3	psc7
16	50,020	24,190 (720)	2,820 (206)	251 (72)	234 (103)
32	117,048	35,689 (1284)	4,512 (405)	363 (120)	327 (176)
64	389,440	41,293 (1720)	7,219 (1143)	581 (332)	497 (456)
128	615,981	52,764 (2345)	10,678 (1421)	882 (421)	632 (512)
256	941,685	71,128 (3578)	13,986 (1578)	1,289 (501)	702 (547)

4.3. Numerical results for Poiseuille optimal flow control problem. Results for the Poiseuille flow control problem are presented in Table 4.2. We have solved for five different problem sizes on up to 256 processors in order to assess the performance and scalability of the LNKS algorithm. Our first and most important finding is that the fastest variant is LNKS-II (which uses $\tilde{\mathbf{P}}_2$ as a preconditioner; i.e., it performs two approximate forward solves per KKT iteration), which is approximately 30 [illegible] faster than QN-RSQP. Another observation is that LNKS-I (which uses $\mathbf{P}_2$, i.e., two exact forward solves), despite its discarding second-order terms and approximating the reduced Hessian, is very effective in reducing the number of iterations—note the difference in KKT iterations between LNK (unpreconditioned) and LNKS-I for each problem instance. Indeed, the mesh-independence of LNKS-I iterations suggests that the two-step stationary preconditioner is very effective for the reduced Hessian,[7] and that if a forward preconditioner that is spectrally equivalent to the forward operator is available, one can expect mesh-independence for the entire KKT preconditioner. Even if a suboptimal forward preconditioner is used (witnessed

[7] Also, the weak increase in quasi-Newton iterations suggests that the reduced Hessian for this problem is well-conditioned.

TABLE 4.2

Performance of LNKS and comparisons with QN-RSQP for the Poiseuille flow problem as a function of increasing number of state and decision variables and number of processors. Here, QN-RSQP *is quasi-Newton reduced-space SQP;* LNK *is the full-space Lagrange–Newton–Krylov method with no preconditioning for the KKT system;* LNKS-I *uses* $\mathbf{P}_2$ *KKT preconditioner—which requires two Stokes solves—combined with the two-step stationary preconditioner for the reduced Hessian* (3) *iterations);* LNKS-II *uses the* $\tilde{\mathbf{P}}_2$ *preconditioner with the same reduced Hessian preconditioner;* LNKS-III *is the same as* LNKS-II *except that no reduced Hessian preconditioning is used, i.e.,* $\mathbf{B}_z = \mathbf{I}$*;* QN iter *reports the number of quasi-Newton iterations (the number of outer iterations for the LNKS variants is just one, since this is a quadratic optimization problem);* KKT iter *is the number of Krylov (symmetric QMR) iterations for the KKT system to converge; finally* time *is wall-clock time in seconds on a Cray T3E-900. Both QN-RSQP and LNKS methods are converged so that* $\|\mathbf{c}\|/\|\mathbf{c}_0\| \leq 1 \times 10^{-6}$ *and* $\|\mathbf{g}\|/\|\mathbf{g}_0\| \leq 1 \times 10^{-6}$ *in all cases but the unpreconditioned KKT case, in which the Krylov method was terminated when the number of iterations exceeded* 500,000.

States Decisions	Method	QN iter	KKT iter	Time
64,491	QN-RSQP	221	—	15,365
7,020	LNK	—	274,101	19,228
(16 PEs)	LNKS-I	—	27	1,765
	LNKS-II	—	170	482
	LNKS-III	—	1,102	969
114,487	QN-RSQP	224	—	19,493
10,152	LNK	—	499,971	39,077
(32 PEs)	LNKS-I	—	26	2,089
	LNKS-II	—	258	519
	LNKS-III	—	1,525	1225
280,161	QN-RSQP	228	—	33,167
18,144	LNK	—	>500,000	—
(64 PEs)	LNKS-I	—	29	4,327
	LNKS-II	—	364	934
	LNKS-III	—	1,913	1,938
557,693	QN-RSQP	232	—	53,592
28,440	LNK	—	>500,000	—
(128 PEs)	LNKS-I	—	29	6,815
	LNKS-II	—	603	1,623
	LNKS-III	—	2,501	3,334
960,512	QN-RSQP	241	—	63,865
40,252	LNK	—	>500,000	—
(256 PEs)	LNKS-I	—	32	8,857
	LNKS-II	—	763	1,987
	LNKS-III	—	2,830	3,735

by the growth in KKT iterations with increasing problem size for LNKS-II), one can trade off increasing KKT iterations for reduced work per iteration to realize a significant reduction in overall work. For example, in the largest problem size case, we give up a factor of 24 in KKT iterations, but we gain a factor of 4.5 in execution time.

The number of unpreconditioned KKT iterations illustrates the severe ill-conditioning of the KKT matrix (even for a problem as simple as controlling a Poiseuille flow). The comparison between LNKS-I and QN-RSQP simply illustrates the better convergence properties of a true Newton, as opposed to quasi-Newton, method. The comparison between LNKS-II and LNKS-III reveals the significance of a good preconditioner for the reduced space. When the two-step preconditioner is used, LNKS runs approximately twice as quickly as with no preconditioning (and requires four times fewer iterations).

The scalability of LNKS is studied by tracking the execution time for LNKS as

TABLE 4.3

*Isogranular scalability results for LNKS-*I. PEs *is the number of processors;* max Mflop/s/PE *is the per-processor maximum (across PEs) sustained Mflop/s;* Mflop/s/PE *is the average sustained Mflop/s per processor. A comparison between the second and third columns is an indication of load imbalance. Implementation efficiency* (η_i) *is based on Mflop rate; optimization work efficiency* (η_a) *is based on number of optimization iterations; forward solver work efficiency* (η_f) *is based on the number of forward solver Krylov iterations* psc3 *(Table* 4.1*); overall efficiency* (η) *is based on execution time;* (η') *is an estimate of the overall efficiency given by* $\eta' = \eta_f \times \eta_i \times \eta_a$.

PEs	Max Mflop/s/PE	Mflop/s/PE	η_i	η_a	η_f	η	η'
16	76.5	52.0	1.00	1.00	1.00	1.00	1.00
32	74.8	51.1	0.98	1.04	0.71	0.84	0.72
64	74.9	49.2	0.96	0.93	0.48	0.41	0.43
128	71.2	47.8	0.91	0.93	0.31	0.26	0.26
256	69.8	45.1	0.87	0.84	0.26	0.18	0.19

the problem size and number of processors increase proportionately. The problem size per processor is held (relatively) constant, and execution time increases from about 8 minutes for 16 processors (65K states, 7K decisions) to about 30 minutes for 256 processors (960K states, 40K decisions). One may conclude that the method is not scalable. However, a glance at the KKT iterations column reveals that the number of optimization iterations when using the LNKS-I variant (so that the effect of inexact forward problem solves is factored out) is largely independent of problem size, and thus the work efficiency of LNKS should be very high. Furthermore, the Mflop rate drops (probably due to a poorer-quality mesh partition) only slowly as the problem size increases, suggesting good implementation efficiency. What, then, accounts for the increase in execution time?

Table 4.3 provides the answer. Following [25], the overall parallel efficiency η (based on execution time) has been decomposed into an implementation efficiency η_i (based on Mflop/s and which represents costs like latency, load imbalance, and interprocessor communication), and a work efficiency η_a (based on the number of optimization iterations), and the forward solver work efficiency η_f (based on Table 4.1). Whereas the optimization algorithm and the implementation are reasonably scalable (84% and 87% efficiency over a 16-fold increase in number of processors), the forward solver's parallel efficiency drops to near 26% for the largest problem size. The last column, η', gives an estimate of what the overall efficiency would be if we did not use the time measurement but instead set $\eta' = \eta_i * \eta_a * \eta_f$. The agreement between the last two columns makes apparent that the overall efficiency of the algorithm depends greatly upon the forward solver, which is essentially a question of forward preconditioning. The parallel inefficiency of the forward preconditioner can be addressed by switching to a more scalable approximation than the one we are currently using (nonoverlapping local ILU(0) block-Jacobi). With a better forward preconditioner, we anticipate good overall scalability.

Timings and flop measurements were performed by using PETSc logging routines, which were validated with native performance analyzers on the T3E and Origin platforms. At first glance CPU performance appears to be mediocre—less than 10% of the peak machine performance. Recall, however, that unstructured grid computations are not cache coherent and therefore the bottleneck is in memory bandwidth and not in the CPU flop rate.

. . . . 1. We do not advocate the use of exact forward solves within the LNKS context. If one can afford exact solves, then one should iterate in the reduced space.

If both exact forward solves and Hessian terms are retained, then 4 solves per KKT–Krylov iteration are required; iterating with N-RSQP requires only 2 solves per $\mathbf{W}_z$-Krylov iteration. Even when the Hessian terms are dropped (which is equivalent to retaining only $\mathbf{g}_z$ on the right-hand side of the decision equation (in Algorithm 2)), the positive definiteness of the reduced Hessian permits the use of CG in the reduced space, which seems more natural (in conjunction with a good reduced Hessian preconditioner) than iterating in the full space.

On the other hand, an order-of-magnitude argument can be used (assuming $\mathbf{A}_s$ is symmetric) to illustrate why it should be advantageous to stay in the full space when approximate solves are used. If the condition number of the preconditioned reduced Hessian is given by μ_m/μ_1 and the condition number of the preconditioned forward operator is given by λ_n/λ_1, the complexity for inverting the exact reduced Hessian is $\mathcal{O}(\sqrt{\mu_m/\mu_1} \times \sqrt{\lambda_n/\lambda_1} \times N)$, since the number of expected iterations is $\mathcal{O}(\sqrt{\mu_m/\mu_1})$ and for each iteration we need to solve the forward problem. (Recall that N is the number of states plus the number of decision variables.) Using the fact that (3.9) is block-diagonal we see that the complexity for inverting the preconditioned KKT system is given by $\mathcal{O}(\sqrt{\max(\mu_m, \lambda_n)/\min(\mu_1, \lambda_1)} \times N)$. If, in addition, the spectra of the preconditioned forward and reduced Hessian operators overlap, then a reduced method has an effective condition number that is approximately the square of the KKT system's condition number. (It is exactly equal to the square of the condition number of the reduced Hessian, if $\lambda_n = \mu_m$ and $\lambda_1 = \mu_1$.)

. . . 2. We have tested the four different preconditioners for the LNKS algorithm. Here we report results only for preconditioners $\mathbf{P}_2$ and $\tilde{\mathbf{P}}_2$. In our numerical experiments preconditioners $\mathbf{P}_4$ and $\mathbf{P}_2$ took approximately the same number of KKT iterations to converge. Since $\mathbf{P}_2$ requires two solves fewer, it is twice as fast as $\mathbf{P}_4$ and for this reason we report results only for $\mathbf{P}_2$. The same trend is evident when comparing the preconditioners $\tilde{\mathbf{P}}_4$ and $\tilde{\mathbf{P}}_2$ although the difference is less pronounced.

Numerical results for cylinder optimal flow control problem. In order to further test the KKT solver we performed additional numerical experiments for the flow around a cylinder. Figure 2 compares the controlled and the uncontrolled flow. Optimization drives the downstream surface of the cylinder to become a sink for the flow. As we can see in Figure 2, the locally controlled flow (around the cylinder) appears to be more irregular than the uncontrolled one. Yet the overall dissipation is reduced. The results can be explained if we consider the dissipation function as the sum along both the upstream and the downstream part of the duct. Flow suction results in reduced mass flow (i.e., velocity) on the downstream portion of the duct and thus minimizes dissipation losses.

Table 4.4 presents scalability results for this problem. We observe similar performance to the Poiseuille flow. Using exact second derivatives accelerates the algorithm by a factor of two. Switching to the inexact preconditioners makes the method another 6 (or more) times faster. For the 256 processor problem (941,685 states and 11,817 controls) quasi-Newton RSQP required 38 hours, whereas the LNKS algorithm with the $\tilde{\mathbf{P}}_2$ preconditioner required only 3 hours, almost 13 times faster. Notice that the fastest LNKS variant for this example (LNKS-III) does not precondition in the reduced space. Although LNKS-II takes fewer iterations to converge, the cost per KKT Krylov iteration is increased due to load imbalance—the Poiseuille flow problem has more uniform distribution of decision variables across processors.

The CPU performance has dropped compared to the Poiseuille flow results, but not significantly, as we can see in Table 4.5. The overall efficiency appears to be gov-

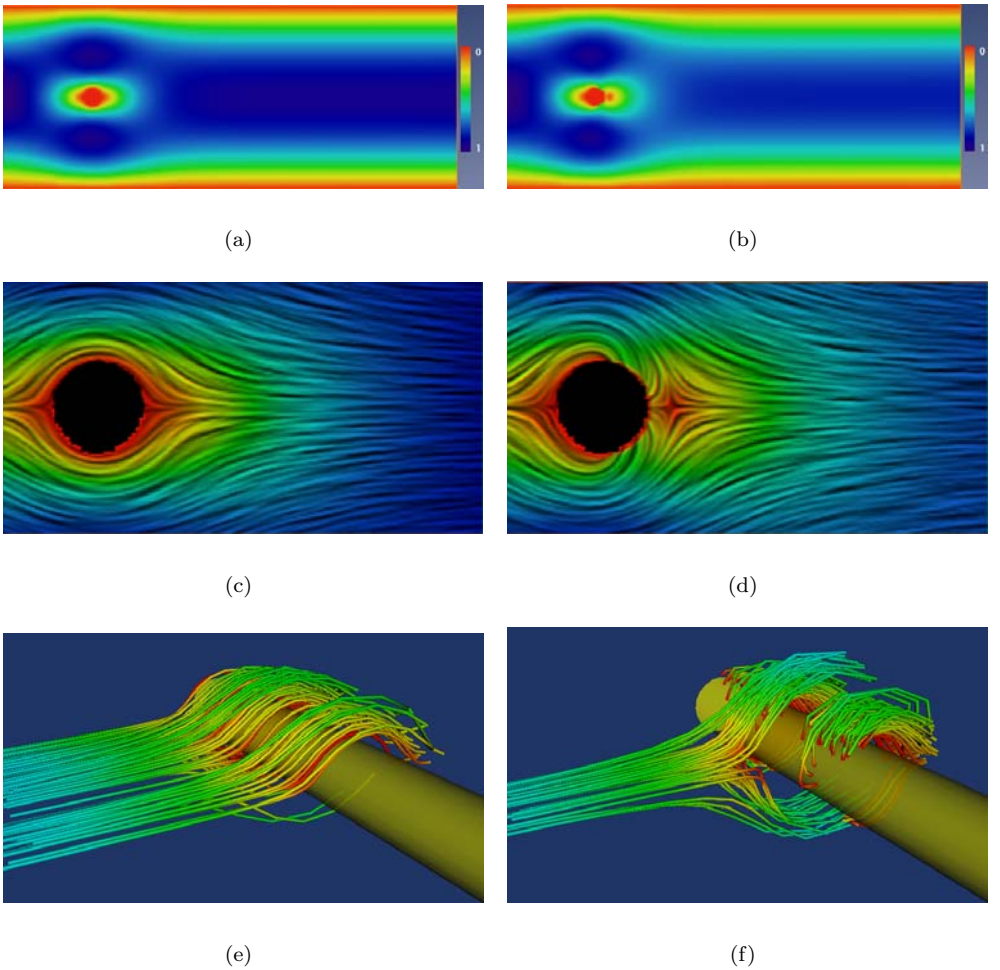


FIG. 2. *Optimal boundary control problem of a steady incompressible Stokes flow around a cylinder. The objective is to minimize the energy dissipation. The controls are injection points (velocity Dirichlet boundary conditions) on the downstream portion of the cylinder surface. The left column is the uncontrolled flow and the right column is the controlled one. Images* (a) *and* (b) *depict a color map of the velocity magnitude on a cross section at the middle of the duct. Comparing image* (a) *with* (b) *notice the lighter gray throughout the downstream region. This indicates smaller velocities and thus reduced dissipation. Images* (c) *and* (d) *show a closeup snapshot of the flow around the cylinder. The suction Dirichlet conditions (set by the optimizer) are clearly visible. Finally, the last two images show stream tubes of the flow around the cylinder. Although the flow appears to be more irregular (locally) for the controlled flow, overall the dissipation is reduced.*

erned by the forward problem preconditioner; however, there is a slight disagreement between the last two columns. An explanation can be found in η_a, which drops faster than in the Poiseuille flow case. The reason is that the faster LNKS variant for the cylinder example does not precondition in the reduced space.

Our performance analyses are based on isogranular scaling. More common are fixed-problem size scalability analyses. Although these kinds of tests are very good indicators of the performance of an algorithm, they do not capture the important issue

TABLE 4.4

Performance and scalability of the LNKS algorithm for the control of a 3D flow around a cylinder. Here the objective function is the energy dissipation and the constraints are the Stokes equations. QN-RSQP *is quasi-Newton reduced-space SQP;* LNK *is the full-space Lagrange–Newton–Krylov method with no preconditioning;* LNKS-I *uses the* $\mathbf{P}_2$ *preconditioner combined with the two-step stationary preconditioner for the reduced Hessian;* LNKS-II *uses the* $\tilde{\mathbf{P}}_2$ *preconditioner with the same reduced Hessian preconditioning;* LNKS-III *is the same as LNKS-*II *but the reduced Hessian is not preconditioned;* time *is wall-clock time in seconds on a T3E-*900*. The reduced gradient and the constraints were converged to a relative norm of* 1×10^{-6}.

States Controls	Method	N or QN iter	KKT iter	Time
50,020	QN-RSQP	116	—	9,320
1,653	LNK	1	390,456	37,670
(16 PEs)	LNKS-I	1	41	6.833
	LNKS-II	1	1,101	3,100
	LNKS-III	1	1,312	1,812
117,048	QN-RSQP	120	—	27,669
2,925	LNK	1	>500,000	—
(32 PEs)	LNKS-I	1	39	10,780
	LNKS-II	1	1,522	5,180
	LNKS-III	1	1,731	2,364
389,440	QN-RSQP	128	—	92,874
6,549	LNK	1	> 500,000	—
(64 PEs)	LNKS-I	1	50	34,281
	LNKS-II	1	2,987	18,451
	LNKS-III	1	3,637	15,132
615,981	QN-RSQP	132	—	113,676
8,901	LNK	1	> 500,000	—
(128 PEs)	LNKS-I	1	53	54,678
	LNKS-II	1	3,150	17,144
	LNKS-III	1	4,235	9,325
941,685	QN-RSQP	138	—	140,085
11,817	LNK	1	> 500,000	—
(256 PEs)	LNKS-I	1	52	59,912
	LNKS-II	1	4,585	20,384
	LNKS-III	1	5,687	11,028

of the dependence of number of Krylov iterations on the number of unknowns (mesh size). For completeness, we present a standard fixed-problem-size scalability analysis for the (117,048 states, 2,925 controls) cylinder problem for 4 different partitions and across two different platforms: a T3E-900 and an Origin 2000. Our experiments showed Origin to have superior performance, which is surprising since the T3E has a much faster interconnect. The fact that the Origin has bigger cache size is a possible explanation. Another observation is that the effectiveness of the LNKS algorithm degrades with the number of processors. The overall efficiency drops to 71% for the T3E and to 83% for the Origin. The work efficiency remains constant but part of it is hidden in η because the time per preconditioner application increases. Recall that the condition number of a linear system preconditioned with block-Jacobi is approximately $\mathcal{O}((pn)^{1/6})$ and therefore the flop count increases with the number of processors p. Our forward solver implementation uses such a preconditioner and this is why we observe the decrease in overall efficiency. We have not conducted any measurements on latency and bandwidth dependencies on the number of processors. The inferior performance of the T3E is somewhat surprising but this would probably

TABLE 4.5

Isogranular scalability results for LNKS-III ($\tilde{\mathbf{P}}_2$ KKT preconditioner with no preconditioning for the reduced Hessian). The Mflop rates given are sustained Mflop per processor; the first column is maximum across processors and the second column lists the average rate. The difference is an indication of imbalance. Efficiency (η_i) is based on Mflop rate; work efficiency (η_a) is based on number of optimization iterations; forward solver work efficiency (η_f) is based on Table 4.1; overall efficiency (η) is based on execution time; η' is an estimate of the overall efficiency given by $\eta' = \eta_f \times \eta_i \times \eta_a$.

PEs	Max Mflop/s/PE	Mflop/s/PE	η_i	η_a	η_f	η	η'
16	69.1	50.9	1.00	1.00	1.00	1.00	1.00
32	69.2	48.5	0.94	1.05	0.69	0.64	0.67
64	73.9	45.7	0.89	0.82	0.62	0.20	0.45
128	65.6	42.9	0.84	0.77	0.29	0.13	0.18
256	66.4	39.3	0.77	0.78	0.19	0.12	0.11

TABLE 4.6

Fixed-size scalability analysis for the 117,028 state variables problem. The experiments were performed on a SGI Origin 2000 and on a T3E-900. The 450 MHz Compaq/Alpha 21164 processor on the T3E is equipped with 8 KB L1 cache and 96 KB L2 cache. The Origin uses the 250 MHz MIPS R10000 processor with 32 KB L1 and 4MB L2 cache. Bisection bandwidth is 128GB/s for the T3E and 20.5 GB/s for the Origin (128 PEs).

CRAY T3E-900

Procs	Agr Gflop/s	Its	Time	Speedup	η_i	η_a	η
16	0.81	38	18,713	1.00	1.00	1.00	1.00
32	1.55	39	10,170	1.84	0.95	0.97	0.92
64	3 14	40	5,985	3.13	0.97	0.95	0.92
128	4.86	40	3,294	5.68	0.75	0.95	0.71

SGI ORIGIN 2000

Procs	Agr Gflop/s	Its	Time	Speedup	η_i	η_a	η
16	1.09	37	13,512	1.00	1.00	1.00	1.00
32	2.13	40	6,188	1.84	0.96	0.93	0.89
64	6.08	38	3,141	3.13	0.96	0.97	0.93
128	7.90	39	1,402	5.68	0.87	0.95	0.83

change had we increased the number of processors further.[8]

4.4. LNKS parallel scalability. In this section, we provide arguments for verifying the scalability of the LNKS method, with respect to increasing problem size and number of processors. For scalability, we require that the work increases near-linearly with problem size (work scalability) and that it parallelizes well. Let us examine the major components.

Formation of the KKT matrix-vector product. For PDE-constrained optimization, the Hessian of the Lagrangian function and the Jacobian of the constraints are usually sparse with structure dictated by the mesh (particularly when the decision variables are mesh-related). Thus, formation of the matrix-vector product at each QMR iteration is linear in both state and decision variables, and parallelizes well due to a high computation-to-communication ratio and minimal sequential bottlenecks characteristic of parallel matvecs for PDE-type problems.

[8] At the time these experiments were performed, the 256 partition on the Origin was not available.

Application of the KKT preconditioner. The main work involved is application of the PDE preconditioner $\tilde{\mathbf{A}}_s$ and its transpose, and "inversion" of an approximation to the reduced Hessian, $\mathbf{B}_z$. We can often make use of scalable, parallel PDE preconditioners that require $\mathcal{O}(n)$ work to apply (as in various domain decomposition preconditioners or multigrid for elliptic problems). The stationary preconditioner for the reduced Hessian involves only matrix-vector multiplications and Jacobian preconditioner applications, and therefore requires linear work and parallelizes well. The same can be said about the limited-memory quasi-Newton update. Therefore, we conclude that application of the KKT preconditioner requires linear work and parallelizes well.

The Krylov (inner) iteration. As argued above, with an "optimal" state preconditioner and a good $\mathbf{B}_z$ approximation, we can expect that the number of inner (Krylov) iterations will be relatively insensitive to the problem size.

The Lagrange–Newton (outer) iteration. The number of outer (Newton) iterations is often independent of problem size for PDE-type problems, and the PDE-constrained optimization problems we have solved exhibit this type of behavior as well. For an extended discussion and results from numerical experiments on the outer iteration we refer the reader to part II of this article [11].

This combination of linear work per Krylov iteration, weak dependence of Krylov iterations on problem size, and independence of Lagrange–Newton iterations on problem size suggests a method that scales well with increasing problem size and number of processors.

5. Conclusions. The main new idea offered by the LNKS algorithm for PDE-constrained optimization problems is the use of a reduced space quasi-Newton method, not as a driver, but rather as a preconditioner for the KKT system. The preconditioning exploits the PDE structure of the constraints by permitting approximate PDE solves at each KKT iteration along with preconditioning of the reduced Hessian. We argue that in order to achieve algorithmic scalability, a bona fide Newton method is necessary. LNKS requires second derivatives (but only in the form of matrix-vector multiplications, which allows for a directionally differenced approximation) and the adjoint operator of the forward problem, beyond the information required by a (forward sensitivity-based) quasi-Newton RSQP method. The most important result is that the problem of devising a good preconditioner for the KKT system is reduced to that of finding a good preconditioner for the forward PDE operator and the reduced Hessian operator.

The problems we have chosen to investigate are relatively simple, yet provide a reasonable testbed for algorithmic tuning and experimentation. The results obtained thus far are very encouraging: the full space LNKS method with reduced space preconditioning is by a factor of 10–30 faster than a popular reduced space method. We expect that other problems will exhibit similar behavior. The method can be parallelized efficiently, and work efficiency can be achieved provided a good forward preconditioner is available. Scalability then results from the combination of these two.

In part II, we extend our discussion to nonlinear PDE constraints and examine issues of robustness and globalization of the LNKS method.

Acknowledgments. We thank the PETSc development group at Argonne National Lab for making this work possible. We also thank Jonathan Shewchuk for providing the meshing and partitioning routines Pyramid and Slice. Finally, we thank David Keyes, David Young, and the other members of the TAOS project—Roscoe Bartlett, Larry Biegler, and Andreas Wächter—for their useful comments.

REFERENCES

[1] E. L. Allgower, K. Böhmer, F. A. Potra, and W. C. Rheinboldt, *A mesh-independence principle for operator equations and their discretizations*, SIAM J. Numer. Anal., 23 (1986), pp. 160–169.

[2] E. Arian and S. Ta'asan, *Analysis of the Hessian for Aerodynamic Optimization*, Technical report 96-28, Institute for Computer Applications in Science and Engineering, Hampton, VA, 1996.

[3] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[4] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, L. C. McInnes, and B. F. Smith, *PETSc home page*, http://www.mcs.anl.gov/petsc (2001).

[5] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *Efficient management of parallelism in object oriented numerical software libraries*, in Modern Software Tools in Scientific Computing, E. Arge, A. M. Bruaset, and H. P. Langtangen, eds., Birkhäuser, Boston, 1997, pp. 163–202.

[6] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc Users Manual*, Technical report ANL-95/11, Revision 2.1.1, Argonne National Laboratory, Argonne, IL, 2001.

[7] A. Battermann and M. Heinkenschloss, *Preconditioners for Karush-Kuhn-Tucker matrices arising in the optimal control of distributed systems*, in Optimal Control of Partial Differential Equations. Internat. Ser. Numer. Math. 126, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser-Verlag, 1998, pp. 15–32.

[8] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM J. Optim., 5 (1995), pp. 314–347.

[9] G. Biros and O. Ghattas, *Parallel Newton-Krylov algorithms for PDE-constrained optimization*, in Proceedings of SC99, The SCxy Conference Series, Portland, OR, 1999, ACM/IEEE.

[10] G. Biros and O. Ghattas, *Parallel preconditioners for KKT systems arising in optimal control of viscous incompressible flows*, in Parallel Computational Fluid Dynamics 1999, D. E. Keyes, A. Ecer, J. Periaux, and N. Satofuka, eds., North–Holland, Amsterdam, 1999.

[11] G. Biros and O. Ghattas, *Parallel Langrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* II*: The Lagrange–Newton solver and its application to optimal control of steady viscous flows*, SIAM J. Sci. Comput., 27 (2005), pp. 714–739.

[12] R. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Math. Program., 49 (1991), pp. 285–323.

[13] X.-C. Cai and O. B. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 243–258.

[14] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices*, SIAM J. Sci. Comput., 18 (1997), pp. 1657–1675.

[15] J. E. Dennis and R. M. Lewis, *A Comparison of Nonlinear Programming Approaches to an Elliptic Inverse Problem and a New Domain Decomposition Approach*, Technical report TR-9433, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1994.

[16] C. Farhat, R. Tezaur, and R. Djellouli, *On the Solution of Three Dimensional Inverse Obstacle Scattering Problems by a Regularized Newton Method*, Technical report CU-CAS-01-11, Center for Aerospace Structures, University of Colorado, Boulder, CO, 2001.

[17] R. Fletcher, *Practical Methods of Optimization*, 2nd ed. John Wiley and Sons, Chichester, UK, 1987.

[18] R. W. Freund and N. M. Nachtigal, *An implementation of the QMR method based on coupled two-term recurrences*, SIAM J. Sci. Comput., 15 (1994), pp. 313–337.

[19] O. Ghattas and J.-H. Bark, *Optimal control of two- and three-dimensional incompressible Navier-Stokes flows*, J. Comput. Phys., 136 (1997), pp. 231–244.

[20] O. Ghattas and C. E. Orozco, *A parallel reduced Hessian SQP method for shape optimization*, in Multidisciplinary Design Optimization: State of the Art, N. M. Alexandrov and M. Y. Hussaini, eds., SIAM, Philadelphia, 1997, pp. 133–152.

[21] G. H. Golub and C. H. V. Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[22] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, SIAM, Philadelphia, 1997.

[23] M. D. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, Boston, 1989.

[24] E. Haber and U. Ascher, *Preconditioned all-at-once methods for large sparse parameter estimation problems*, Inverse Problems, 17 (2000), pp. 1847–1864.

[25] D. E. Keyes, *How scalable is domain decomposition in practice?*, in Proceedings of the 11th International Conference on Domain Decomposition Methods, C.-H. Lai, M. Cross, and O. Widlund, eds., 1998.
[26] D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. S166–S202.
[27] K. Kunisch and E. W. Sachs, *Reduced SQP methods for parameter identification problems*, SIAM J. Numer. Anal., 29 (1992), pp. 1793–1820.
[28] F.-S. Kupfer and E. W. Sachs, *Numerical solution of a nonlinear parabolic control problem by a reduced SQP method*, Comput. Optim. Appl., 1 (1992), pp. 113–135.
[29] I. Malčević, *Large-Scale Unstructured Mesh Shape Optimization on Parallel Computers*, Master's thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.
[30] J. L. Morales and J. Nocedal, *Automatic preconditioning by limited memory quasi-Newton updating*, SIAM J. Optim., 10 (2000), pp. 1079–1096.
[31] S. G. Nash and A. Sofer, *Preconditioning reduced matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 47–68.
[32] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer, New York, 1999.
[33] C. E. Orozco and O. Ghattas, *Massively parallel aerodynamic shape optimization*, Comput. Systems Engrg., 1–4 (1992), pp. 311–320.
[34] C. E. Orozco and O. Ghattas, *A reduced SAND method for optimal design of nonlinear structures*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 2759–2774.
[35] J. Reuther, J. Alonso, M. J. Rimlinger, and A. J. Jameson, *Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on parallel computers*, Comput. & Fluids, 29 (1999), pp. 675–700.
[36] U. Ringertz, *Optimal Design of Nonlinear Shell Structures*, Technical report FFA TN 91-18, The Aeronautical Research Institute of Sweden, 1991.
[37] U. Ringertz, *An algorithm for optimization of nonlinear shell structures*, Internat. J. Numer. Methods Engrg., 38 (1995), pp. 299–314.
[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.
[39] A. Salinger, R. Pawlowski, J. Shadid, B. van Bloemen Waanders, R. Barlett, G. Itle, and L. Biegler, *RSQP optimization of large-scale reacting flow applications with MP-Salsa*, in Proceedings of First Sandia Workshop on PDE-Constrained Optimization, Santa Fe, NM, 2001.
[40] V. Schulz, *Solving discretized optimization problems by partially reduced SQP methods*, Comput. Vis. Sci., 1 (1997), pp. 83–96.
[41] V. H. Schulz and H. G. Bock, *Partially reduced SQP methods for large-scale nonlinear optimization problems*, in Proceedings of the Second World Congress of Nonlinear Analysis, Elsevier, 1997.

SIAM J. SCI. COMPUT.
Vol. 27, No. 2, pp. 714–739
© 2005 Society for Industrial and Applied Mathematics

PARALLEL LAGRANGE–NEWTON–KRYLOV–SCHUR METHODS FOR PDE-CONSTRAINED OPTIMIZATION. PART II: THE LAGRANGE–NEWTON SOLVER AND ITS APPLICATION TO OPTIMAL CONTROL OF STEADY VISCOUS FLOWS*

GEORGE BIROS† AND OMAR GHATTAS‡

Abstract. In part I of this article, we proposed a Lagrange–Newton–Krylov–Schur (LNKS) method for the solution of optimization problems that are constrained by partial differential equations. LNKS uses Krylov iterations to solve the linearized Karush–Kuhn–Tucker system of optimality conditions in the full space of states, adjoints, and decision variables, but invokes a preconditioner inspired by reduced space sequential quadratic programming (SQP) methods. The discussion in part I focused on the (inner, linear) Krylov solver and preconditioner. In part II, we discuss the (outer, nonlinear) Lagrange–Newton solver and address globalization, robustness, and efficiency issues, including line search methods, safeguarding Newton with quasi-Newton steps, parameter continuation, and inexact Newton ideas. We test the full LNKS method on several large-scale three-dimensional configurations of a problem of optimal boundary control of incompressible Navier–Stokes flow with a dissipation objective functional. Results of numerical experiments on up to 256 Cray T3E-900 processors demonstrate very good scalability of the new method. Moreover, LNKS is an order of magnitude faster than quasi-Newton reduced SQP, and we are able to solve previously intractable problems of up to 800,000 state and 5,000 decision variables at about 5 times the cost of a single forward flow solution.

Key words. sequential quadratic programming, adjoint methods, PDE-constrained optimization, optimal control, Lagrange–Newton–Krylov–Schur methods, Navier–Stokes, finite elements, preconditioners, indefinite systems, nonlinear equations, parallel algorithms

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65J22, 65N55, 65W10, 65Y05, 65Y20, 76D05, 76D07, 76D55, 90C52, 90C55, 90C90, 93C20

DOI. 10.1137/S1064827502415661

1. Introduction.

In part I of this two-part article [3] we proposed a Newton–Krylov method for solution of the optimality system stemming from the optimization of systems governed by partial differential equations (PDEs). We concentrated our discussion on the inner iteration: the solution of the linear system associated with a Newton step on the Karush–Kuhn–Tucker (KKT) optimality conditions. The algorithm is based on a Krylov solver combined with a family of Schur-type preconditioners that are equivalent to an approximate quasi-Newton reduced SQP (QN-RSQP) step. We termed the method [illegible] (LNKS), a concatenation of [illegible] for the outer iteration and [illegible] for the inner iteration. We also provided theoretical and numerical evidence that these preconditioners work very well by considering several linearly-constrained quadratic optimization problems, namely those involving boundary control of Stokes flows with flow-matching

*Received by the editors October 9, 2002; accepted for publication (in revised form) November 17, 2004; published electronically November 15, 2005. This work is a part of the Terascale Algorithms for Optimization of Simulations (TAOS) project at CMU, with support from NASA grant NAG-1-2090, NSF grant ECS-9732301 (under the NSF/Sandia Life Cycle Engineering Program), and the Pennsylvania Infrastructure Technology Alliance. Computing services on the Pittsburgh Supercomputing Center's Cray T3E were provided under PSC grant BCS-960001P.

http://www.siam.org/journals/sisc/27-2/41566.html

†Courant Institute of Mathematical Sciences, Department of Computer Science, New York University, New York, NY 10012 (biros@cs.nyu.edu).

‡Ultrascale Simulation Laboratory, Departments of Biomedical Engineering and Civil & Environmental Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 (oghattas@cs.cmu.edu).

and dissipation-type objectives.

In part II, we present algorithmic components of the LNKS method related to the (outer) Lagrange–Newton solver, including such globalization strategies as line search, quasi-Newton safeguarding, and parameter continuation, and inexactness in the inner linear solver and its interaction with the line search. We assess performance of the LNKS method on several more stringent test problems that contain many features of the most challenging PDE-constrained optimization problems: three-dimensional unstructured meshes, multicomponent coupling, large problem size, nonlinearity, and ill-conditioning. The underlying problem class is optimal control of a viscous incompressible fluid by boundary velocities, which is of both theoretical and industrial interest.

Following part I, we refer to the unknown PDE field quantities as the [illegible]; the PDE constraints as the [illegible]; solution of the PDE constraints as the [illegible]; the inverse, design, or control variables as the [illegible]; and the problem of determining the optimal values of the inverse, design, or control variables as the [illegible].

The paper is organized as follows. In section 2 we briefly review the problem formulation. We then discuss algorithmic issues related to Lagrange–Newton methods and in particular globalization methodologies. We give details on three globalization techniques to enhance robustness in the LNKS method: line search, safeguarding LNKS steps with (approximate) QN-RSQP steps, and continuation. We also discuss inexact Newton methods and how they interact with a merit function line search. In section 3 we present the full globalized LNKS algorithm. Section 4 formulates the optimal control problem for the Navier–Stokes equations and presents results for a Poiseuille flow, a flow around a cylinder, and a flow around a Boeing 707 wing.

[illegible]. We use boldface characters to denote vector-valued functions and vector-valued function spaces. We use roman characters to denote discretized quantities and italics for their continuous counterparts. For example, $\boldsymbol{u}$ will be the continuous velocity field and $\mathbf{u}$ will be its discretization. Greek letters are overloaded and whether we refer to the discretization or the continuous fields should be clear from context. We also use + as a subscript or superscript to denote variable updates within an iterative algorithm.

2. Globalization and inexactness of the LNKS method. In this section we consider strategies for globalizing the Newton iteration at the heart of the LNKS method, as well as opportunities for introducing inexactness in the solution for a Newton step. Let us revisit the constrained optimization problem formulation,

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \text{subject to } \mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^N$ are the optimization variables, $f : \mathbb{R}^N \to \mathbb{R}$ is the objective function, and $\mathbf{c} : \mathbb{R}^N \to \mathbb{R}^n$ are the constraints. In our context these constraints are discretizations of the state equations. The Lagrangian,

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{2.2}$$

is used to transform the constrained optimization problem into a system of nonlinear equations, the first-order optimality conditions:

$$\begin{Bmatrix} \partial_x \mathcal{L} \\ \partial_\lambda \mathcal{L} \end{Bmatrix} (\mathbf{x}, \boldsymbol{\lambda}) = \begin{Bmatrix} \mathbf{g}(\mathbf{x}) + \mathbf{A}(\mathbf{x})^T \boldsymbol{\lambda} \\ \mathbf{c}(\mathbf{x}) \end{Bmatrix} = \mathbf{0} \quad (\text{or } \mathbf{h}(\mathbf{q}) = \mathbf{0}), \tag{2.3}$$

where $\mathbf{q} := \{\mathbf{x}, \boldsymbol{\lambda}\}^T$ represents the optimization variables and Lagrange multipliers. As in part I, $\mathbf{g}$ is the gradient of the objective function, $\mathbf{A}$ the Jacobian of the constraints, and $\mathbf{W}$ the Hessian of the Lagrangian. Consistent with part I, we think of $\mathbf{x}$, $\mathbf{g}$, $\mathbf{A}$, and $\mathbf{W}$ as being partitioned into state (indicated by an s subscript) and decision variable (d subscript) components. We use Newton's method to solve for $\mathbf{x}$ and $\boldsymbol{\lambda}$. A Newton step on the optimality conditions is given by

$$\begin{bmatrix} \mathbf{W} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}_x \\ \mathbf{p}_\lambda \end{Bmatrix} = - \begin{Bmatrix} \mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} \\ \mathbf{c} \end{Bmatrix} \quad (\text{or } \mathbf{K}\mathbf{v} = -\mathbf{h}), \tag{2.4}$$

where $\mathbf{p}_x$ and $\mathbf{p}_\lambda$ are the updates of $\mathbf{x}$ and $\boldsymbol{\lambda}$ from the current to the next iteration. In part I we reviewed a popular algorithm for solving for a KKT point, RSQP and in particular its quasi-Newton variant (Algorithm 3 in part I). Although RSQP is efficient and robust, it often does not scale very well with the number of decision variables. It avoids solving (2.4) directly by reduction onto the decision space and in doing so requires repeated solution of linearized forward and adjoint problems. Thus, it can be very inefficient for large-scale PDE-constrained optimization. We argued that a better approach would be to remain in the full space of states, decisions, and multipliers by using a Krylov method to solve (2.4). For most problems, however, the KKT matrix is notoriously ill-conditioned. Motivated by this, part I proposed a family of efficient preconditioners. The key idea was to use an approximate RSQP method as a preconditioner. We showed that RSQP can be viewed as a block-LU factorization in which the reduced Hessian $\mathbf{W}_z$ is the Schur complement for the decision variables. It has the property that the only systems that have to be solved involve the reduced Hessian and the Jacobian of the state equations (and its transpose). Therefore, by using a limited memory quasi-Newton approximation of the reduced Hessian, and by replacing the linearized PDE solves by applications of their preconditioners, we obtain a method that requires no PDE solves within a Krylov iteration, yet maintains Newton convergence in the outer iteration. A sketch of the LNKS method for solving the KKT system (2.4) is given by Algorithm 1, in which $\mathbf{P}^{-1}$ is an application of the approximate RSQP preconditioner.

Algorithm 1. Lagrange–Newton–Krylov–Schur (LNKS).

1: Choose $\mathbf{x}$, $\boldsymbol{\lambda}$
2: **loop**
3: Check for convergence
4: Compute $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$
5: Solve $\mathbf{P}^{-1}\mathbf{K}\mathbf{v} = \mathbf{P}^{-1}\mathbf{h}$ (Newton step)
6: Update $\mathbf{x} = \mathbf{x} + \mathbf{p}_x$
7: Update $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \mathbf{p}_\lambda$
8: **end loop**

Nevertheless, there are two issues that should be addressed before we can claim a fast and robust general-purpose algorithm. The first is whether the LNKS algorithm can be made convergent for any initial guess $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$, and the second is whether we can utilize inexactness in the inner iterations to further accelerate LNKS. Within this framework we examine line search algorithms, reverting to a QN-RSQP step when a LNKS steps fails to make progress, continuation, and inexact Newton methods.

2.1. Line search methods. Algorithm 1 is only locally convergent. Popular methods to globalize Newton's method include line search and trust region

algorithms. Details can be found in [21]. Trust region methods, especially in combination with RSQP and inexact Newton methods, have been extended successfully to PDE-constrained optimization [14, 17, 19]. Global convergence proofs for these methods can be found in [5]. Trust region methods are often based on the Steihaug modification of the conjugate gradient (CG) algorithm [24]. However, this approach works well only with positive definite systems. It is not obvious how to use trust regions with an indefinite Krylov solver (which is required for the KKT system) and thus we have opted for a line search algorithm to help globalize LNKS.

An important component of a line search algorithm is the choice of a merit function: a scalar function (of the optimization variables) that monitors the progress of the algorithm. In contrast with unconstrained optimization, the choice of a merit function is not straightforward, since we are trying to balance minimization of the objective function with feasibility of the constraints. Two common choices are the l_1 merit function,

$$\phi(\mathbf{x}) := f + \rho_\phi \|\mathbf{c}\|_1, \tag{2.5}$$

and the augmented Lagrangian merit function,

$$\phi(\mathbf{x}, \boldsymbol{\lambda}) := f + \mathbf{c}^T \boldsymbol{\lambda} + \frac{\rho_\phi}{2} \mathbf{c}^T \mathbf{c}. \tag{2.6}$$

The scalar ρ_ϕ is the [illegible]—a weight chosen to balance minimization of the objective function with minimization of the residuals of the constraints. Both functions are "exact" provided the penalty parameter is large enough. By [illegible] we mean that if $\mathbf{x}^*$ is a minimizer for (2.1), then it is also an (unconstrained) minimizer for the merit function. A crucial property of a merit function is that it should accept unit step lengths close to a solution in order to allow full Newton steps and thus quadratic convergence. The l_1 merit function suffers from the "Maratos" effect; that is, sometimes it rejects good steps and slows down the algorithm. The augmented Lagrangian merit function does not exhibit such behavior but its drawback is that it requires accurate estimates of the Lagrange multipliers.

The outline of a general line search method is given in Algorithm 2. To simplify notation we use $\phi(\alpha)$ for $\phi(\mathbf{q} + \alpha\mathbf{v})$ and $\phi(0)$ for $\phi(\mathbf{q})$, and likewise for the derivative $\nabla\phi$. The algorithm used to compute the search direction $\mathbf{v}$ is intentionally left unspecified. All that matter to ensure global convergence are the properties of the merit function and the properties of $\mathbf{v}$. Step 3 in Algorithm 2 lists three conditions on $\mathbf{v}$: descent direction, sufficient angle, and sufficient step length size [9]. The condition in step 4 is often called the Armijo condition. If ϕ is bounded and has a minimum, and if $\mathbf{v}$ is bounded, Algorithm 2 is guaranteed to converge to a local minimum [20]. We use a simple backtracking line search, with a factor α of 0.5. The search is bounded so that $\alpha_{\min} \leq \alpha \leq 1$. As mentioned before, the choice of the penalty parameter has a great effect on the performance of the algorithm.

For a step computed by quasi-Newton RSQP (Algorithm 3 of part I), the update for the l_1-merit function is relatively straightforward. The directional derivative for a search direction $\mathbf{p}_x$ is given by

$$\nabla\phi^T \mathbf{p}_x = \mathbf{g}^T \mathbf{p}_x - \rho_\phi \|\mathbf{c}\|_1. \tag{2.7}$$

If $\mathbf{W}_z$ is positive definite, it can be shown that by setting

$$\rho_\phi = \|\boldsymbol{\lambda}\|_\infty + \delta, \quad \delta > 0, \tag{2.8}$$

ALGORITHM 2. LINE SEARCH.

1: Choose $\mathbf{q}$, $\delta_A > 0$ and κ_1, κ_2 arbitrary constants (strictly positive)
2: **while** Not converged **do**
3: Compute search direction $\mathbf{v}$ so that
$\mathbf{v}^T \nabla\phi(0) < 0$
$|\mathbf{v}^T \nabla\phi(0)| \geq \kappa_1 \|\mathbf{v}\| \, \|\nabla\phi(0)\|$ angle condition
$\|\mathbf{v}\| \geq \kappa_2 \|\nabla\phi(0)\|$ length condition
4: Compute α such that $\phi(\alpha) \leq \phi(0) + \alpha\delta_A \mathbf{v}^T \nabla\phi(0)$ Armijo condition
5: Set $\mathbf{q} = \mathbf{q} + \alpha\mathbf{v}$
6: **end while**

we obtain a descent direction. In our numerical experiments we have used the l_1-merit function with QN-RSQP and the augmented Lagrangian with LNKS. The l_1-merit function performed reasonably well. However, we did observe the Maratos effect. To overcome this obstacle we have implemented a second-order correction, in which an extra normal step toward feasibility is taken [21, p. 570].

When an augmented Lagrangian merit function is used, the penalty parameter should be chosen differently. The directional derivative of the augmented Lagrangian merit function is given by

$$\nabla\phi^T \mathbf{v} = (\mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} + \rho_\phi \mathbf{A}^T \mathbf{c})^T \mathbf{p}_x + \mathbf{c}^T \mathbf{p}_\lambda. \tag{2.9}$$

Lagrange multipliers slightly complicate the algorithm since we have to compute $\mathbf{p}_\lambda$. $\boldsymbol{\lambda}$ may be considered a function of $\mathbf{x}$ [4, 8] or an independent variable [5, 23] or may be simply ignored by setting $\mathbf{p}_\lambda = 0$ [25]. In LNKS we solve for $\boldsymbol{\lambda}$ simultaneously with $\mathbf{x}$ and it is natural to use the step $\mathbf{p}_\lambda$. On the other hand, RSQP uses $\boldsymbol{\lambda} = -\mathbf{A}_s^{-T} \mathbf{g}_s$ and it seems natural to consider $\boldsymbol{\lambda}$ a function of $\mathbf{x}$. In this case the last term in (2.9) is given by

$$\mathbf{c}^T \mathbf{p}_\lambda = \mathbf{c}^T (\partial_x \boldsymbol{\lambda}) \mathbf{p}_x,$$

where

$$\partial_x \boldsymbol{\lambda} := -\mathbf{A}_s^{-T} [\mathbf{W}_{ss} \ \mathbf{W}_{sd}].$$

(However, this formula cannot be used with the QN-RSQP method of Algorithm 3, part I, since second derivatives are not computed.) If we set

$$\rho_\phi = \frac{(\mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda}) \mathbf{p}_x + \mathbf{c}^T \mathbf{p}_\lambda + \delta}{\mathbf{c}^T \mathbf{A} \mathbf{p}_x}, \quad \delta > 0, \tag{2.10}$$

we obtain a descent direction.

2.2. Combining QN-RSQP with LNKS. For iterates far from the solution, relying solely on a line search algorithm to globalize LNKS is not sufficient, since the Newton step is likely to be of poor quality. Usually global convergence can be shown if the reduced (and not full) Hessian $\mathbf{W}_z$ is positive definite. If $\mathbf{W}_z$ is positive definite (and assuming the system (2.4) is solved exactly), then the resulting step $\mathbf{v}$ satisfies the Armijo sufficient decrease criterion. Far from the minimum, however, $\mathbf{W}_z$ can be singular or indefinite. On the other hand, certain quasi-Newton methods, like BFGS, are preferable for iterates far from the solution since they can guarantee positive definiteness. For this reason (and for preconditioning purposes) LNKS does maintain a BFGS approximation for $\mathbf{W}_z$: if a computed Newton search direction fails to satisfy the Armijo criterion, we discard it and revert to a QN-RSQP step.

2.3. Continuation. One of the standard assumptions in global convergence proofs is the full rank of the constraint Jacobian for all iterates. In LNKS, this translates to the nonsingularity of the state Jacobian $\mathbf{A}_s$, i.e., the PDE state operator. For highly nonlinear PDEs such as the Navier–Stokes equations, this can be an unrealistic assumption. Even if $\mathbf{A}_s$ is nonsingular, severe ill-conditioning may cause both QN-RSQP and LNKS algorithms to stall. Indeed, in our numerical experiments, the most difficult computation (for iterates far from the solution) was converging the $\mathbf{A}_s$-related linear solves. Krylov solvers often reached their maximum iteration counts without a significant decrease in the linear system residual. As a result, the iterates were of poor quality and the algorithm stagnated as it was impossible to compute a search direction, be it from QN-RSQP or LNKS.

One remedy for this problem is parameter continuation. This idea (in its simplest form) works when we can express the nonlinearity of the problem as a function of a single scalar parameter. Continuation is suitable for problems in which the underlying PDE has a parameter that scales (and thus determines the effects of) nonlinear terms. Examples of such parameters are the Reynolds number for viscous flow, the Mach number for compressible flow, the Weissenberg number for viscoelastic flow, load magnitude for buckling problems, and the Hartman number in magnetohydrodynamics. In problems where such a parameter cannot be found, an alternative approach is to use a pseudotransient continuation scheme [18], a homotopy method [1], or mesh-size continuation.

Continuation allows uphill steps to be taken (unlike monotone line search methods) and generates good initial guesses, not only for the optimization variables, but also for the penalty parameter in the merit function. An important feature of the continuation algorithm is that under certain conditions it globalizes trivially.[1] If the continuation step places the next iterate outside the attraction basin of the Newton method, then we simply reduce the continuation step size. In principle, the method globalizes LNKS without the need to use line search or some other globalization strategy. Nevertheless, taking a large number of continuation steps significantly slows down the algorithm. Experience from our numerical experiments suggests that the best strategy for globalizing LNKS is a combination of line searching, reverting to quasi-Newton steps, and continuation.

2.4. Inexact Newton method. Before we discuss the inexact Newton method in the context of LNKS, we briefly summarize a few results for a general nonlinear system of equations. Assume we want to solve $\mathbf{h}(\mathbf{q}) = \mathbf{0}$. Further assume the following: (1) $\mathbf{h}$ and $\mathbf{K} := \partial_q \mathbf{h}$ are sufficiently smooth in a neighborhood of a solution $\mathbf{q}^*$; (2) at each iteration an inexact Newton method computes a step $\mathbf{v}$ that satisfies

$$\|\mathbf{K}\mathbf{v} + \mathbf{h}\| \leq \eta_N \|\mathbf{h}\|, \tag{2.11}$$

where η_N is often called the [illegible]. It can be shown that if $\eta_N < 1$, then $\mathbf{q} \to \mathbf{q}^*$ linearly; if $\eta_N \to 0$, then $\mathbf{q} \to \mathbf{q}^*$ superlinearly; and if $\eta_N = \mathcal{O}(\|\mathbf{h}\|)$, then we recover the quadratic convergence rate of an exact Newton method. The forcing term is usually given by

$$\eta_N = \frac{\|\mathbf{h}_{(+)} - \mathbf{h} - \mathbf{K}\mathbf{v}\|}{\|\mathbf{h}\|}. \tag{2.12}$$

Other alternatives exist (for details see [7]).

[1] This is true only when the initial problem is a well-posed quadratic programming problem (like Stokes) and all iterates on the continuation path are far from turning and bifurcation points.

The extension of inexact Newton methods to unconstrained optimization problems is relatively easy. The extension becomes more complicated for problems with constraints. In [14] global convergence proofs are given for a trust region RSQP-based algorithm, but to our knowledge such an approach has not been extended to full space algorithms. Close to a KKT point the theory for Newton's method applies and one can use the analysis presented in [6] to show that the inexact version of the LNKS algorithm converges. However, the line search we are using is not based on the residual of the KKT equations but instead on the merit function discussed in the previous session. It is not obvious that an inexact Newton step (which simply reduces $\|\mathbf{h}\|$) will satisfy the merit function criteria. [illegible] Our analysis is based on the augmented Lagrangian merit function.[2] We assume that, locally, $\mathbf{A}$ and $\mathbf{K}$ are nonsingular and uniformly bounded. We define $\kappa_1 := \max \|\mathbf{K}^{-1}(\mathbf{q})\|$ for $\mathbf{q}$ in the neighborhood of the solution $\mathbf{q}^*$. We also define $\mathbf{v}$ as the exact solution of the (linearized) KKT system so that

$$\mathbf{K}\mathbf{v} + \mathbf{h} = \mathbf{0}, \tag{2.13}$$

and $\tilde{\mathbf{v}}$ the approximate solution so that

$$\mathbf{K}\tilde{\mathbf{v}} + \mathbf{h} = \mathbf{r}. \tag{2.14}$$

We also have $\|\mathbf{r}\| = \eta\|\mathbf{h}\|$, $0 < \eta \leq \eta_N$, from the inexact Newton stopping criterion (2.11). By (2.3) we get that $\|\mathbf{h}\|^2 = \|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\|^2 + \|\mathbf{c}\|^2$ and since $\mathbf{A}$ is uniformly bounded, there is constant κ_2 such that

$$\|\mathbf{A}^T\mathbf{c}\| \leq \kappa_2\|\mathbf{h}\|. \tag{2.15}$$

Besides the assumptions on $\mathbf{K}$ and $\mathbf{A}$ we also assume the following: (1) ρ_ϕ is sufficiently large so that the merit function is exact and $\|\nabla\phi\| \geq \kappa_3\|\mathbf{h}\|$ for some constant κ_3; (2) $\mathbf{v}$ is a descent direction and satisfies the angle and length conditions as well as the Armijo condition. From the latter it is immediate that $\mathbf{v}$ satisfies the Cauchy fraction condition:[3]

$$|\nabla\phi^T\mathbf{v}| \geq 2\kappa_4\|\nabla\phi\|^2. \tag{2.16}$$

We will show that if η is small enough, then the approximate step $\tilde{\mathbf{v}}$ satisfies the Cauchy fraction, angle, and length conditions. Then, we will use a theorem from [20] to conclude that the Armijo condition is satisfied with unit step lengths. Therefore if $\eta = \mathcal{O}(\|\mathbf{h}\|)$, quadratic convergence is preserved.

From (2.13) and (2.14) we have

$$\nabla\phi^T\tilde{\mathbf{v}} = \nabla\phi^T\mathbf{v} + \nabla\phi^T\mathbf{K}^{-1}\mathbf{r},$$

and thus by (2.16) we get

$$|\nabla\phi^T\tilde{\mathbf{v}}| \geq 2\kappa_4\|\nabla\phi\|^2 - |\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}|.$$

Therefore, to satisfy the Cauchy fraction condition

$$|\nabla\phi^T\tilde{\mathbf{v}}| \geq \kappa_4\|\nabla\phi\|^2, \tag{2.17}$$

[2] For brevity we drop the subscript from ϕ_L and just use the symbol ϕ for the augmented Lagrangian merit function.

[3] The Cauchy step is a steepest descent step for the merit function.

we need to show that

$$|\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}| \le \kappa_4\|\nabla\phi\|^2. \tag{2.18}$$

The gradient of the merit function is given by

$$\nabla\phi = \mathbf{h} + \rho_\phi \begin{Bmatrix} \mathbf{A}^T\mathbf{c} \\ \mathbf{0} \end{Bmatrix},$$

and thus

$$\begin{aligned} |\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}| &= \left|\mathbf{h}^T\mathbf{K}^{-1}\mathbf{r} + \rho_\phi \begin{Bmatrix} \mathbf{A}^T\mathbf{c} \\ \mathbf{0} \end{Bmatrix}^T \mathbf{K}^{-1}\mathbf{r}\right| \\ &\le \kappa_1(\|\mathbf{h}\|\,\|\mathbf{r}\| + \rho_\phi\|\mathbf{A}^T\mathbf{c}\|\,\|\mathbf{r}\|) \\ &\le \kappa_1\eta(\|\mathbf{h}\|^2 + \rho_\phi\|\mathbf{A}^T\mathbf{c}\|\,\|\mathbf{h}\|) \\ &\le \kappa_1\eta(1+\rho_\phi\kappa_2)\|\mathbf{h}\|^2 \\ &\le \kappa_1\eta(1+\rho_\phi\kappa_2)\frac{\|\nabla\phi\|^2}{\kappa_3^2}. \end{aligned}$$

If

$$\eta \le \frac{\kappa_4\ \kappa_3^2}{\kappa_1(1+\rho_\phi\kappa_2)}, \tag{2.19}$$

then (2.18) holds. If we choose a superlinearly convergent inexact Newton variant, then

$$\eta \le \eta_N \to 0,$$

and therefore, close to the solution (2.19) holds. We also have that

$$\begin{aligned} \tilde{\mathbf{v}} &= \mathbf{K}^{-1}(\mathbf{r}-\mathbf{h}), \\ \|\tilde{\mathbf{v}}\| &\le \kappa_1(1+\eta)\|\mathbf{h}\|, \\ \|\tilde{\mathbf{v}}\| &\le \kappa_1(1+\eta)\frac{\|\nabla\phi\|}{\kappa_3}, \\ \|\tilde{\mathbf{v}}\| &\le \kappa_5\|\nabla\phi\|. \end{aligned} \tag{2.20}$$

By combining (2.20) and (2.17) we get

$$|\nabla\phi^T\tilde{\mathbf{v}}| \ge \kappa_4\|\nabla\phi\|^2 \ge \kappa_4\kappa_5\|\nabla\phi\|\,\|\tilde{\mathbf{v}}\|$$

and

$$\|\nabla\phi\|\,\|\tilde{\mathbf{v}}\| \ge \kappa_6|\nabla\phi^T\tilde{\mathbf{v}}| \ge \kappa_6\kappa_4\|\nabla\phi\|^2 \Longrightarrow \|\tilde{\mathbf{v}}\| \ge \kappa_7\|\nabla\phi\|.$$

That is, the length and angle conditions are satisfied. It can be shown [20, Theorem 10.6] that there is an α, bounded below, so that the Armijo condition holds true. Thus by choosing δ_A small enough, the Armijo condition is satisfied with unit step length. Hence the quadratic convergence rate associated with Newton's method is observed; i.e., the inexactness does not interfere with the merit function. In addition it can be shown that the augmented Lagrangian merit function allows unit step length near the solution (see [8, 23] and the references therein). Finally, notice that convergence does not require that $\eta_N \to 0$; it only requires that η_N is small enough. This is in contrast with inexact reduced space methods which require the tolerances to become tighter as the iterates approach the solution [14].

2.5. The globalized inexact LNKS algorithm. In this section we present the complete LNKS method incorporating the globalization and inexactness strategies discussed above, and give a high-level description of implementation details and specific heuristics used to accelerate convergence. The basic steps of our method are given in Algorithm 3.

ALGORITHM 3. GLOBALIZED LNKS.

1: Choose $\mathbf{x}_s$, $\mathbf{x}_d$, ρ_ϕ, δ_A, set $\mathrm{Re} = \mathrm{Re}_{start}$, $\mathrm{tol} = \mathrm{tol}_0$
2: $\mathbf{A}_s^T\boldsymbol{\lambda} + \mathbf{g}_s \approx \mathbf{0}$ solve inexactly for $\boldsymbol{\lambda}$
3: **while** $\mathrm{Re} \neq \mathrm{Re}_{target}$ **do**
4: **loop**
5: Evaluate f, $\mathbf{c}$, $\mathbf{g}$, $\mathbf{A}$, $\mathbf{W}$
6: $\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T\boldsymbol{\lambda}$
7: Check convergence: $\|\mathbf{h}\| \leq \mathrm{tol}$
8: $\mathbf{P}^{-1}\mathbf{K}\mathbf{v} + \mathbf{P}^{-1}\mathbf{h} \approx \mathbf{0}$ solve inexactly for $\mathbf{v}$
9: Compute ρ_ϕ such that $\nabla\phi^T(0)\mathbf{v} \leq 0$
10: Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{v})$
11: **if** Line search failed **then**
12: Compute α s.t. $\|\mathbf{h}(\alpha)\| < (1 - \delta_A\alpha\,(\mathbf{v}^T\,\mathbf{K}\,\mathbf{h}(0)))$ Armijo for KKT residual
13: **end if**
14: **if** LNKS step failed **then**
15: $\mathbf{B}_z\mathbf{p}_d = -\mathbf{g}_z$ solve inexactly for $\mathbf{p}_d$
16: $\mathbf{A}_s\mathbf{p}_s + \mathbf{A}_d\mathbf{p}_d + \mathbf{c} \approx \mathbf{0}$ solve inexactly for $\mathbf{p}_s$
17: $\mathbf{A}_s^T\boldsymbol{\lambda}_+ + \mathbf{g}_s \approx \mathbf{0}$ solve inexactly for $\boldsymbol{\lambda}_+$
18: Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{v})$
19: **if** Line search on QN-RSQP step failed **then**
20: Reduce Re and go to step 5.
21: **end if**
22: **end if**
23: $\boldsymbol{\lambda}_+ = \boldsymbol{\lambda} + \mathbf{p}_\lambda$ (only for LNKS step)
24: $\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$
25: **end loop**
26: $\mathrm{Re} = \mathrm{Re} + \Delta\,\mathrm{Re}$
27: Tighten tol
28: **end while**

The algorithm uses a three-level iteration.

- In the [illegible] (lines 3–28) the continuation parameter is gradually increased until the target value is reached.
- The [illegible] (lines 4–25) corresponds to the Lagrange–Newton solver for the KKT optimality system for a fixed continuation number.
- The [illegible] (line 8 for LNKS, and lines 15–17 for QN-RSQP) refers to the solution of the linearized KKT system.

The outer iteration consists of two core branches: the computation of the LNKS search direction and the computation of a search direction with the limited-memory BFGS QN-RSQP method. The default branch is the LNKS step. If this step fails to satisfy the line search conditions, we switch to QN-RSQP. If the QN-RSQP search direction fails to satisfy the line search criteria as well, then we reduce the continuation parameter Re and return to the beginning of the continuation loop (line 3).

There are several possible instantiations of this framework. Below, we give some

additional implementation details on our algorithmic choices.

• The linear solves in lines 8, 16, and 17 are performed inexactly. We follow [7] in choosing the forcing term. For the LNKS step (line 8) we use $\|\mathbf{h}\|$ as the residual that drives the Newton solver; for the QN-RSQP step (lines 15–18) we use $\|\mathbf{c}\|$ and $\|\mathbf{g}_z\|$.

• We allow for nonmonotone line searches. If the LNKS step (line 8) is rejected by the merit function line search (line 10) we do not switch immediately to QN-RSQP. Instead we do a line search on the KKT residual (line 12) and if the step is accepted we use it to update the variables for the next iteration. However, we do store information ($\boldsymbol{\lambda}$ and $\mathbf{x}$, the merit function value and gradient) for the iterate for which the LNKS step failed, and we insist that a subsequent iterate satisfies the conditions of the merit line search (evaluated at the failure point) after a fixed number of iterations. Typically, we permit two iterations before we demand reduction of the merit function. If the Armijo criterion is still violated, we backtrack to the original point of failure and switch to a QN-RSQP method (lines 15–18).

• We use various heuristics to bound the penalty parameter and possibly reduce it. A new penalty parameter ρ_ϕ^+ is computed using the LNKS step and formula (2.10). If $\rho_\phi^+ > 4\rho_\phi$ we update the penalty parameter and switch to QN-RSQP. If $\rho_\phi^+ < \rho_\phi/4$ we [illegible] the penalty parameter and set $\rho_\phi^+ = 0.5\rho_\phi$. We also reduce the penalty parameter if there is a successful search on the KKT residual (step 12).

• A Lanczos algorithm can be used to (approximately) check the second-order optimality conditions. If an extremal eigenvalue of $\tilde{\mathbf{W}}_z$ is negative, then we abandon the full space and revert to a QN-RSQP step. The eigenvalue computation is frozen through a single continuation step, but if a negative direction is detected, they are recomputed at each SQP iteration.

• In line 6 we use the adjoint variables to update the reduced gradient. This is equivalent to $\mathbf{g}_z = \mathbf{g}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{g}_s$ if $\boldsymbol{\lambda}$ is computed by solving exactly $\mathbf{A}_s^T \boldsymbol{\lambda} + \mathbf{g}_s = \mathbf{0}$. When $\boldsymbol{\lambda}$ is taken from the LNKS step computation, it includes second-order terms (which tend to zero as we approach a stationary point). This introduces error in the computation of $\mathbf{g}_z$. When $\boldsymbol{\lambda}$ is taken from QN-RSQP it introduces additional error since we never solve the linear systems exactly.

• Typical values for the parameters are $\rho_\phi = 0.5$, $\delta_A = 10^{-4}$, and tol $= 10^{-5}\|\mathbf{h}_0\|$ (following [6]). Here $\mathbf{h}_0$ is the value of the KKT residual at the first iteration of each outer loop.

We have shown how it is possible to augment the basic LNKS method with line search and continuation, safeguard it with QN-RSQP steps, and further accelerate it with inexact Newton. We have shown that inexactness does not interfere with the line search algorithm, and we have described the globalized algorithm and various heuristics that are observed to improve performance.

In the next section we study an optimal control problem for the steady incompressible Navier–Stokes equations. We cite results on the existence and uniqueness of solutions and make comparisons between the discrete and continuous forms of the optimality conditions.

3. Formulation of an optimal control problem. We turn our attention to the formulation and well-posedness of a specific optimization problem: the Dirichlet control of the steady incompressible Navier–Stokes equations. We present the continuous form of the KKT optimality conditions and we cite convergence results for finite element approximations from [15] and [16]. A survey and articles on this topic can be found in [11]. More on the Navier–Stokes equations can be found in [10, 13].

We study problems in which we specify both Dirichlet and Neumann boundary conditions. The controls are restricted to be of Dirichlet type only, but the theory is similar for distributed and Neumann controls [15].

3.1. Continuous optimality conditions. We use the velocity-pressure $(\boldsymbol{u}, p)$ form of the incompressible steady Navier–Stokes equations. We begin by writing the following strong form of the flow equations:

$$\begin{aligned} -\nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)+(\nabla\boldsymbol{u})\boldsymbol{u}+\nabla p &= \boldsymbol{b} \quad \text{in } \Omega,\\ \nabla\cdot\boldsymbol{u} &= 0 \quad \text{in } \Omega,\\ \boldsymbol{u} &= \boldsymbol{u}_g \quad \text{on } \Gamma_u,\\ \boldsymbol{u} &= \boldsymbol{u}_d \quad \text{on } \Gamma_d,\\ -p\boldsymbol{n}+\nu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)\boldsymbol{n} &= \boldsymbol{0} \quad \text{on } \Gamma_N. \end{aligned} \tag{3.1}$$

Here $\nu = 1/\mathrm{Re}$ and the decision variables are the control velocities $\mathbf{u}_d$ on Γ_d. For a forward solve we need not distinguish between Γ_d and Γ_u. In the optimization problem, however, $\boldsymbol{u}_d$ is not known. We will present a mixed formulation that treats the tractions on the Dirichlet boundary Γ_d as additional unknown variables. The traction variables play the role of Lagrange multipliers (not to be confused with the Lagrange multipliers of the optimal control problem) and are used to enforce the Dirichlet boundary conditions [2].

By $L^2(\Omega)$ we denote the space of scalar functions that are square-integrable in Ω, and by $\boldsymbol{H}^1(\Omega)$ we denote the space of vector functions whose first derivatives are in $\boldsymbol{L}^2(\Omega)$. $\boldsymbol{H}^{1/2}(\Gamma)$ is the trace space (the restriction on Γ) of functions belonging to $\boldsymbol{H}^1(\Omega)$. Finally $\boldsymbol{H}^{-k}(D)$ is the set of bounded linear functionals on functions belonging to $\boldsymbol{H}^k(D)$, where D is some smooth domain in $\mathbb{R}^3$. We also define $\boldsymbol{V} := \{\boldsymbol{v} \in \boldsymbol{H}^1(\Omega) : \boldsymbol{v}|_{\Gamma_u} = \boldsymbol{0}\}$. We define the following bilinear and trilinear forms associated with the Navier–Stokes equations:

$$\begin{aligned} a(\boldsymbol{u},\boldsymbol{v}) &:= \int_\Omega (\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)\cdot(\nabla\boldsymbol{v}+\nabla\boldsymbol{v}^T)\,d\Omega \quad \forall \boldsymbol{u},\boldsymbol{v}\in\boldsymbol{H}^1(\Omega),\\ c(\boldsymbol{w},\boldsymbol{u},\boldsymbol{v}) &:= \int_\Omega (\nabla\boldsymbol{u})\boldsymbol{w}\cdot\boldsymbol{v}\,d\Omega \quad \forall \boldsymbol{u},\boldsymbol{v},\boldsymbol{w}\in\boldsymbol{H}^1(\Omega),\\ b(q,\boldsymbol{v}) &:= \int_\Omega -q\nabla\cdot\boldsymbol{v}\,d\Omega \quad \forall q\in L^2,\ \boldsymbol{v}\in\boldsymbol{H}^1(\Omega). \end{aligned}$$

We also use the notation $(\boldsymbol{x},\boldsymbol{y})_D$ for $\int_D \boldsymbol{x}\cdot\boldsymbol{y}\,dD$.

In the weak formulation of (3.1) we seek $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega)$, $p \in L^2(\Omega)$, and $\boldsymbol{\sigma} \in H^{-1/2}(\Gamma_d)$ such that

$$\begin{aligned} \nu a(\boldsymbol{u},\boldsymbol{v}) + c(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) + b(p,\boldsymbol{v}) - (\boldsymbol{\sigma},\boldsymbol{v})_{\Gamma_d} &= (\boldsymbol{f},\boldsymbol{v})_\Omega \quad && \forall \boldsymbol{v}\in\boldsymbol{V},\\ b(q,\boldsymbol{u}) &= 0 && \forall q\in L^2(\Omega),\\ -(\boldsymbol{t},\boldsymbol{u})_{\Gamma_d} &= -(\boldsymbol{t},\boldsymbol{u}_d)_{\Gamma_d} && \forall \boldsymbol{t}\in\boldsymbol{H}^{-1/2}(\Gamma_d). \end{aligned} \tag{3.2}$$

We also define $\boldsymbol{d}$ to be the decision field (so that $\boldsymbol{u}_d = \boldsymbol{d}$). Based on the above formulation we proceed in defining the Lagrangian function for the optimization problem. The objective function is given by

$$\mathcal{J}(\boldsymbol{u},\boldsymbol{d}) := \frac{\nu}{2}a(\boldsymbol{u},\boldsymbol{u}) + \frac{\rho}{2}(\boldsymbol{d},\boldsymbol{d})_{\Gamma_d}, \tag{3.3}$$

and (the weak form of) the constraints are given by (3.2). We define the Lagrangian function $\mathcal{L}$ by

$$\begin{aligned}\mathcal{L}(\boldsymbol{u},p,\boldsymbol{d},\boldsymbol{\sigma},\boldsymbol{\lambda},\mu,\boldsymbol{\tau}) &:= \mathcal{J}(\boldsymbol{u},\boldsymbol{d})\\ &+ \nu a(\boldsymbol{u},\boldsymbol{\lambda}) + c(\boldsymbol{u},\boldsymbol{u},\boldsymbol{\lambda}) - (\boldsymbol{\sigma},\boldsymbol{\lambda})_{\Gamma_d} - (\boldsymbol{f},\boldsymbol{\lambda})_\Omega + b(p,\boldsymbol{\lambda})\\ &+ b(\mu,\boldsymbol{u}) - (\boldsymbol{\tau},\boldsymbol{u}-\boldsymbol{d})_{\Gamma_d}\\ &\quad \forall \boldsymbol{u}\in \boldsymbol{H}^1(\Omega),\quad p\in L^2(\Omega),\quad \boldsymbol{\sigma}\in \boldsymbol{H}^{-1/2}(\Gamma_d),\quad \boldsymbol{d}\in \boldsymbol{H}^{1/2}(\Gamma_d),\\ &\quad \forall \boldsymbol{\lambda}\in \boldsymbol{V},\quad \mu\in L^2(\Omega),\quad \boldsymbol{\tau}\in \boldsymbol{H}^{-1/2}(\Gamma_d).\end{aligned} \tag{3.4}$$

Here $\boldsymbol{\lambda}$, μ, $\boldsymbol{\tau}$ are the Lagrange multipliers for the state variables $\boldsymbol{u}$, p, $\boldsymbol{\sigma}$. By taking variations with respect to the Lagrange multipliers we obtain (3.2) augmented with $\boldsymbol{u}_d = \boldsymbol{d}$ on Γ_d. Taking variations with respect to the states $\boldsymbol{u}$, p, $\boldsymbol{\sigma}$ we obtain the weak form of the adjoint equations

$$\begin{aligned}&\nu a(\boldsymbol{v},\boldsymbol{\lambda}) + c(\boldsymbol{v},\boldsymbol{u},\boldsymbol{\lambda}) + c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}) + b(\mu,\boldsymbol{v}) + (\boldsymbol{\tau},\boldsymbol{v})_{\Gamma_d} = -\nu a(\boldsymbol{u},\boldsymbol{v}) \quad \forall \boldsymbol{v}\in \boldsymbol{V},\\ &b(q,\boldsymbol{\lambda}) = 0 \quad \forall q\in L^2(\Omega),\\ &(\boldsymbol{t},\boldsymbol{\lambda})_{\Gamma_d} = 0 \quad \forall \boldsymbol{t}\in \boldsymbol{H}^{-1/2}(\Gamma_d).\end{aligned} \tag{3.5}$$

Finally, by taking variations with respect to $\boldsymbol{d}$ we obtain the decision equation

$$\rho(\boldsymbol{d},\boldsymbol{r})_{\Gamma_d} + (\boldsymbol{\tau},\boldsymbol{r})_{\Gamma_d} = 0 \quad \forall \boldsymbol{r}\in \boldsymbol{H}^{1/2}(\Gamma_d). \tag{3.6}$$

Equations (3.2), (3.5), (3.6) are the weak form of the first-order optimality conditions. Extensive discussion on the existence of a solution and the existence of the Lagrange multipliers can be found in [15, 16]. In [15] the existence of a local minimum for the optimization problem and the existence of Lagrange multipliers that satisfy the first-order optimality conditions are asserted.[4] Furthermore, uniqueness is shown for sufficiently small data. Note that in the absence of a Neumann condition ($\Gamma_N = \emptyset$) the controls have to satisfy the incompressibility condition $(\boldsymbol{d}\cdot\boldsymbol{n})_{\Gamma_d} = 0$.

The strong form of the adjoint and decision equations can be obtained by using the following integration by parts formulas:

$$\begin{aligned}a(\boldsymbol{u},\boldsymbol{v}) &= -(\boldsymbol{v},\Delta\boldsymbol{u})_\Omega + ((\nabla\boldsymbol{u})\boldsymbol{n},\boldsymbol{v})_\Gamma,\\ c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}) &= -c(\boldsymbol{u},\boldsymbol{\lambda},\boldsymbol{v}) - ((\nabla\cdot\boldsymbol{u})\boldsymbol{\lambda},\boldsymbol{v})_\Omega + ((\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{\lambda},\boldsymbol{v})_\Gamma,\\ b(\mu,\boldsymbol{v}) &= (\nabla\mu,\boldsymbol{v})_\Omega - (\mu\boldsymbol{n},\boldsymbol{v})_\Gamma.\end{aligned}$$

Upon sufficient smoothness we arrive at the strong form of the optimality conditions. Equation (3.1) is the strong form of the constraints. The strong form of the adjoint equations is given by

$$\begin{aligned}-\nu\nabla\cdot(\nabla\boldsymbol{\lambda}+\nabla\boldsymbol{\lambda}^T) + (\nabla\boldsymbol{u})^T\boldsymbol{\lambda} - (\nabla\boldsymbol{\lambda})\boldsymbol{u} + \nabla\mu &= \nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T) \quad \text{in } \Omega,\\ \nabla\cdot\boldsymbol{\lambda} &= 0 \quad \text{in } \Omega,\\ \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{on } \Gamma_u,\\ \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{on } \Gamma_d,\\ -\mu\boldsymbol{n} + \nu(\nabla\boldsymbol{\lambda}+\nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + (\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{\lambda} &= -\nu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)\boldsymbol{n} \quad \text{on } \Gamma_N,\end{aligned} \tag{3.7}$$

[4] The objective functional used in [15] is different from ours. An $\boldsymbol{L}^4$ functional is used for the matching problems and an $\boldsymbol{H}^{\frac{1}{2}}_{\Gamma_d}$ is used for the penalization of $\boldsymbol{u}_d$—resulting in a surface Laplacian equation for the decision variables.

and (equation for $\boldsymbol{\tau}$)

$$\nu(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + \nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\boldsymbol{n} - \boldsymbol{\tau} = \boldsymbol{0} \quad \text{on } \Gamma_d. \tag{3.8}$$

We may also determine that the strong form of the decision equation is given by

$$\boldsymbol{\tau} = \rho\boldsymbol{d} \quad \text{on } \Gamma_d. \tag{3.9}$$

In [15] estimates are given for the convergence rates of the finite element approximations to the exact solutions for the optimal control of steady viscous flow. For the case of boundary velocity control, the basic result is that if the exact solutions are smooth enough, then provided the Taylor–Hood element is used (for both adjoints and states), the solution error satisfies the following estimates:

$$\begin{aligned} \|\boldsymbol{u} - \boldsymbol{u}_h\|_0 &\le \mathcal{O}(h^3), \\ \|p - p_h\|_0 &\le \mathcal{O}(h^2), \\ \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_0 &\le \mathcal{O}(h^3), \\ \|\mu - \mu_h\|_0 &\le \mathcal{O}(h^2). \end{aligned} \tag{3.10}$$

Here h is the maximum element size, and $\|\cdot\|_0$ is the $L^2(\Omega)$ norm.

3.2. Discrete and discretized optimality conditions. In our implementation we have not discretized the continuous forms of the optimality conditions. Instead we have discretized the objective function and the Navier–Stokes equations and then used this discretization to form the optimality conditions. In general, discretization and differentiation (to obtain optimality conditions) do not commute. That is, if $\boldsymbol{A}$ is the infinite-dimensional (linearized) forward operator and $\boldsymbol{A}^*$ is its adjoint, then in general

$$(\boldsymbol{A}^*)_h \neq (\boldsymbol{A}_h)^T,$$

where the subscript h indicates discretization. We will show that for Galerkin approximation of the steady incompressible Navier–Stokes optimal control problem, discretization and differentiation do commute.

For the discretized equations we use the following notation:

$$\begin{aligned} a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{U}(\mathbf{u})\mathbf{u}, \\ a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{p}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{p}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{V}(\mathbf{u})\mathbf{p}, \\ a(\boldsymbol{u}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{Q}\mathbf{u}, \\ b(q_h, \boldsymbol{u}_h) &\dashrightarrow \mathbf{P}\mathbf{u}, \\ (\boldsymbol{t}_h, \boldsymbol{u}_h)_{\Gamma_d} &\dashrightarrow \mathbf{T}\mathbf{u}, \\ (d_h, r_h)_{\Gamma_d} &\dashrightarrow \mathbf{M}\mathbf{d}. \end{aligned}$$

The discretized form of the Navier–Stokes equations (3.2) is then given by

$$\begin{aligned} \mathbf{U}(\mathbf{u})\mathbf{u} + \mathbf{P}^T\mathbf{p} + \mathbf{T}^T\boldsymbol{\sigma} &= \mathbf{f}_1, \\ \mathbf{P}\mathbf{u} &= \mathbf{f}_2, \\ \mathbf{T}\mathbf{u} &= \mathbf{T}\mathbf{d}. \end{aligned} \tag{3.11}$$

The discrete Lagrangian function is given by

$$\begin{aligned}\frac{1}{2}\mathbf{u}^T\mathbf{Q}\mathbf{u}+\frac{\rho}{2}\mathbf{d}^T\mathbf{M}\mathbf{d}+\boldsymbol{\lambda}^T\{\mathbf{U}(\mathbf{u})\mathbf{u}+\mathbf{P}^T\mathbf{p}+\mathbf{T}^T\boldsymbol{\sigma}-\mathbf{f}_1\}\\ +\boldsymbol{\mu}^T\{\mathbf{P}\mathbf{u}-\mathbf{f}_2\}+\boldsymbol{\tau}^T\{\mathbf{T}\mathbf{u}+\mathbf{T}\mathbf{d}\}=\mathbf{0}.\end{aligned} \tag{3.12}$$

By taking derivatives with respect to the discrete Lagrange multiplier vectors $(\boldsymbol{\lambda},\boldsymbol{\mu},\boldsymbol{\tau})$, we recover the state equations (3.11). By taking derivatives with respect to the discrete state variables $(\mathbf{u},\mathbf{p},\boldsymbol{\sigma})$, we obtain the discrete adjoint equations

$$\begin{aligned}\mathbf{V}^T(\mathbf{u})\boldsymbol{\lambda}+\mathbf{P}^T\boldsymbol{\mu}+\mathbf{T}^T\boldsymbol{\tau}&=-\mathbf{Q}\mathbf{u},\\ \mathbf{P}\boldsymbol{\lambda}&=\mathbf{0},\\ \mathbf{T}\boldsymbol{\lambda}&=\mathbf{0}.\end{aligned} \tag{3.13}$$

These equations correspond to the discretization of (3.5) provided that $\mathbf{V}^T\boldsymbol{\lambda}$ is the discretization of $a(\boldsymbol{\lambda},\boldsymbol{u})+c(\boldsymbol{v},\boldsymbol{u},\boldsymbol{\lambda})+c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda})$. The bilinear form $a(\cdot,\cdot)$ is symmetric so we omit it from our discussion. If ϕ denotes the basis function for $\boldsymbol{u}$, ψ_v for $\boldsymbol{v}$, and ψ_λ for $\boldsymbol{\lambda}$, then the (3×3)-block elements for the linearized state and adjoint are

$$\int_\Omega \mathbf{I}\,(\nabla\phi\cdot\boldsymbol{u})\psi_v+(\nabla\boldsymbol{u})\phi\psi_v\;d\Omega \quad \text{state element matrix,}$$

$$\int_\Omega \mathbf{I}\,(\nabla\psi_v\cdot\boldsymbol{u})\psi_\lambda+(\nabla\boldsymbol{u})^T\psi_\lambda\psi_v\;d\Omega \quad \text{adjoint element matrix.}$$

Therefore, the transpose of the discretized (linearized) state equations coincides with the discretization of the adjoint equations, i.e.,

$$(\boldsymbol{A}^*)_h=(\boldsymbol{A}_h)^T.$$

One needs to be careful to use the weak form given by (3.5). If (3.7) were used without employing the reverse integration by parts on the term $c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda})$, this would result in a discretization which is incompatible with the discrete optimization problem (which is what is presented to the optimization algorithm). It would result in an nonsymmetric KKT matrix and would possibly prevent the optimizer from converging to a KKT point. A Petrov–Galerkin formulation would also be incompatible.

In our formulation we do not solve explicitly for $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. We approximate both tractions and velocity traces in $\boldsymbol{H}^1(\Gamma_d)$; in this case the stresses can be eliminated. For a discussion on choice of $H^1(\Gamma_d)$ for the stresses see [12]. The resulting equations are equivalent to the formulations described in this section. We use a standard Galerkin approximation scheme (no upwinding) with Taylor–Hood elements to approximate the velocities, the pressures, and their adjoints.

We conclude this section with a remark on continuation. To solve a Navier–Stokes control problem with large Reynolds number, some kind of continuation scheme is usually needed. We first solve a Stokes-flow optimal control problem ($\mathrm{Re}_0=0$) and then we progressively increase the Reynolds number by $\mathrm{Re}_{(+)}=\mathrm{Re}+\Delta\,\mathrm{Re}$. One could set $\mathbf{u}_{(+)}=\mathbf{u}+(\partial_{\mathrm{Re}}\mathbf{u})\Delta\,\mathrm{Re}$, where $\partial_{\mathrm{Re}}\mathbf{u}$ can be easily computed through a linearized forward solve. Since we consider only steady flows, we follow [10] and use a fixed $\Delta\,\mathrm{Re}$, and simply set $\mathbf{u}_{(+)}=\mathbf{u}$; i.e., the initial guess at the new Reynolds number is the solution from the previous optimization step. Additionally, quasi-Newton information is carried forward to the next Reynolds number.

4. Numerical results. In this section we investigate the accuracy, performance, and scalability of the LNKS method via four numerical examples. Numerical approximation and parallel implementation details are described in section 4.1 of part I of

this article. First we examine the finite element approximation convergence rates with a problem that has a closed form solution. Then we revisit the Poiseuille flow problem of part I and use it to study the effectiveness of the limited-memory BFGS method as a preconditioner for the reduced Hessian. In both cases we solve for the boundary conditions that reproduce the exact solution by minimizing a matching velocity functional.

We then consider the more challenging problem of the control of a flow around a cylinder. The objective function to be minimized is the rate of energy dissipation functional. We use this problem to study the LNKS line search algorithm, the effectiveness of the Krylov–Schur preconditioner for highly nonlinear problems, and the heuristics we introduced in section 2. The last test problem is the optimal control of a flow around a wing.

4.1. Finite element approximation error. In this section we use a model problem to verify the convergence rate estimates given in the previous section. The velocity and pressure are given by

$$\boldsymbol{u}^*(x,y,z) = \{1-(x^2+y^2)^2, x, -y\}^T, \qquad p^*(x,y,z) = x^2+y^2-z^2,$$

which satisfy the Navier–Stokes equations. We restrict this solution to a cylindrical domain and choose a part of its boundary as the control domain Γ_d. We define the velocity on the circumferential walls to be the decision variables. On Γ/Γ_d we set $\boldsymbol{u} = \boldsymbol{u}^*$. The objective function is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{u}_d, p) = \frac{1}{2}\int_\Omega (\boldsymbol{u}^* - \boldsymbol{u})^2 \, d\Omega.$$

Since the boundary conditions for $(\boldsymbol{u}, p)$ on Γ/Γ_d are compatible with $(\boldsymbol{u}^*, p^*)$, the values for the objective function and the Lagrange multipliers are equal to zero at the minimum.

Table 4.1 gives convergence rates for the state variables and the Lagrange multipliers. The results are in good agreement with the theoretical predictions. The convergence rate for the velocities and their adjoints is approximately 2.92 (comparing errors between the first and second rows) and 2.96 (comparing errors between the second and third rows). For the pressures and their adjoints the convergence rate is 1.96 and 1.97, respectively.

TABLE 4.1

Convergence rate of the finite element approximation for a matching velocity problem. Here $\mathbf{n}$ *is the number of elements and* h *is the cube root of the volume of the maximum inscribed sphere inside a tetrahedron of the finite element mesh. Near-optimal convergent rates can be observed for the state and adjoint variables.*

$\mathbf{n}$	h	$\|\boldsymbol{u}^* - \boldsymbol{u}_h\|_0$	$\|p^* - p_h\|_0$	$\|\boldsymbol{\lambda}^* - \boldsymbol{\lambda}_h\|_0$	$\|\mu^* - \mu_h\|_0$
124,639	0.80	1.34×10^{-4}	2.01×10^{-5}	3.88×10^{-4}	1.76×10^{-5}
298,305	0.53	4.40×10^{-5}	9.00×10^{-6}	1.19×10^{-4}	7.90×10^{-6}
586,133	0.40	1.70×10^{-5}	5.20×10^{-6}	5.00×10^{-5}	4.50×10^{-6}

4.2. Poiseuille flow. The Poiseuille flow is a stable solution of the Navier–Stokes equations for small Reynolds numbers. We use this example to study the effectiveness of the limited-memory BFGS method as a preconditioner for the reduced Hessian. Since the optimization problem is nonlinear, LNKS takes several iterations

and curvature information can be built up. Quasi-Newton theory predicts that $\mathbf{B}_z$ approaches $\mathbf{W}_z$ as the iterates get closer to the solution. Therefore, we expect the effectiveness of the preconditioner to improve as the optimization algorithm progresses. We store 30 vectors for the limited-memory BFGS preconditioner.

For this set of tests we fix the size and granularity of the problem. The target Reynolds number is 500. We start at Reynolds number 100 and use a continuation step $\Delta\,\mathrm{Re} = 200$. Continuation is used to initialize the state and control variables, but only to carry BFGS information forward to the next Reynolds number.

The forward problem preconditioner is based on a block factorization

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}^{-1}\mathbf{P}^T \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{V}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{P}\mathbf{V}^{-1} & \mathbf{I} \end{bmatrix},$$

where $\mathbf{S} := -\mathbf{P}\mathbf{V}^{-1}\mathbf{P}^T$ is the Schur complement for the pressure. Based on this factorization, the preconditioner is defined by replacing the exact solves $\mathbf{V}^{-1}$ with domain decomposition approximations $\tilde{\mathbf{V}}^{-1}$. To "invert" the pressure Schur complement $\mathbf{S}$ we use several iterations of the two-step stationary method described in part I (section 3.2).

In QN-RSQP we use QMR for the linearized Navier–Stokes solves, preconditioned with an overlapping additive Schwarz method with ILU(1) in each subdomain.[5] Results for a problem with 21,000 state and 3,900 design variables on 4 processors and for a sequence of three Reynolds numbers are presented in Table 4.2. The number of KKT iterations in LNKS-I reveals the efficacy of the BFGS preconditioner. This number drops from an average 48 iterations to 38. The effect of BFGS in LNKS-II is hidden since the KKT iterations are dominated by the ill-conditioning of the forward and adjoint operators (in LNKS-I these solves are exact in each iteration). Overall, we can observe that LNKS reduces significantly the execution time relative to QN-RSQP.

The Newton solver performed well, requiring just 3 iterations to converge. In these problems we did not use inexact Newton criteria to terminate the KKT solves early; instead they were fully converged at each iteration. No line search was used in the LNKS variants; we used the l_1-merit function for the QN-RSQP.

It is rather surprising that the quasi-Newton approximation works well as a preconditioner within the Newton method, whereas it stagnates when used alone to drive the QN-RSQP method. One explanation could be that the QN-RSQP method in these tests suffered the Maratos effect. In our subsequent tests we switched to a second-order correction method [21, p. 570].

4.3. Flow around a cylinder. All the problems examined so far were useful in verifying certain aspects of the LNKS method, but they are linear or mildly nonlinear. In order to test LNKS further we consider a highly nonlinear problem: that of flow around a cylinder with a dissipation-type objective function. The cylinder is anchored inside a rectangular duct, much like a wind tunnel. A quadratic velocity profile is used as an inflow Dirichlet condition and we prescribe a traction-free outflow. The decision variables are defined to be the velocities on the downstream portion of the cylinder surface. We have investigated flows in the laminar steady-state regime. For exterior problems the transition Reynolds number is 40 but for the duct problem we expect higher Reynolds numbers due to the dissipation from the duct walls.

Figures 1 and 2 illustrate optimal solutions for different Reynolds numbers. LNKS eliminates the recirculation region and secondary flows downstream of the cylinder. In

[5] For definitions of ILU(0) and ILU(1) see [22].

TABLE 4.2

*Work efficiency of the proposed preconditioners for a Poiseuille flow matching problem for fixed size and fixed granularity as a function of the Reynolds number. Recall that LNKS does not use any preconditioner for the KKT system, LNKS-*I *uses a KKT preconditioner that involves two linearized exact forward/adjoint solves per iteration, and LINKS-*II *uses a KKT preconditioner that involves just the application of the forward problem preconditioner.* Re *is the Reynolds number;* N/QN *denotes the number of outer iterations. The number of iterations for the KKT system is averaged across the optimization iterations. The problem has* 21,000 *state equations and* 3,900 *control variables; results are for* 4 *processors of the T3E-*900*. Wall-clock time is in hours.*

Re	Method	N/QN iter	KKT iter	$\|\mathbf{g}_z\|$	Time
100	QN-RSQP	262	—	1×10^{-4}	5.9
	LNK	3	186,000	9×10^{-6}	7.1
	LNKS-I	3	48	9×10^{-6}	3.2
	LNKS-II	3	4,200	9×10^{-6}	1.3
300	QN-RSQP	278	—	1×10^{-4}	6.4
	LNK	3	198,000	9×10^{-6}	7.6
	LNKS-I	3	40	9×10^{-6}	3.1
	LNKS-II	3	4,300	9×10^{-6}	1.4
500	QN-RSQP	289	—	1×10^{-4}	7.3
	LNK	3	213,000	9×10^{-6}	9.0
	LNKS-I	3	38	9×10^{-6}	3.0
	LNKS-II	3	4,410	9×10^{-6}	1.4

order to avoid the excessive suction that we observed in the Stokes case, we imposed Dirichlet boundary conditions on the outflow of the domain. The incompressibility condition prevents the optimizer from driving the flow inside the cylinder.[6]

Our experiments on the Stokes optimal control problem demonstrated the dependence of the performance of the Krylov–Schur iteration on the forward problem preconditioner. Thus before we discuss results on the LNKS algorithm we cite representative performance for the Navier–Stokes forward solver. We use an inexact Newton's method combined with the preconditioner we presented in part I. A block-Jacobi ILU(0) preconditioner is used for the velocity block as well as for the pressure mass matrix (scaled by 1/Re); the latter is used to precondition the pressure Schur complement block. ILU(1) would have been a better choice, but memory limitations prevented us from doing so.[7]

Table 4.3 gives statistics for three different Reynolds numbers and for three different problem sizes. We report the (aggregate) number or Krylov iterations required to converge the Newton solver, the number of Newton iterations, and the total execution time. For these runs we did not use continuation, but we did use an inexact Newton method. The time for a forward solve has increased almost sixfold in comparison with the linear Stokes solver (part I, Table 4.1). However the time , . Newton iteration is roughly the same as that taken in the linear Stokes case. For example, in the 128 processor problem and for Reynolds number 30, the average (Krylov) iteration count is 1005, whereas in the linear case it is 882. Similarly, the average time per Newton

[6]When Dirichlet conditions are specified everywhere on Γ, then $\int_\Gamma \boldsymbol{u} \cdot \boldsymbol{n}\, d\Gamma$ should be zero. The constraint should be imposed either explicitly or implicitly by using a proper function space. In our implementation we use a penalty approach by modifying the objective function.

[7]Our matrix storage includes the state Jacobian, the Hessian of the constraints, and the Hessian of the objective—substantially more than required for the forward solver. PSC's T3E-900 (which hosted the majority of our computations) has just 128 MB of memory per processor.

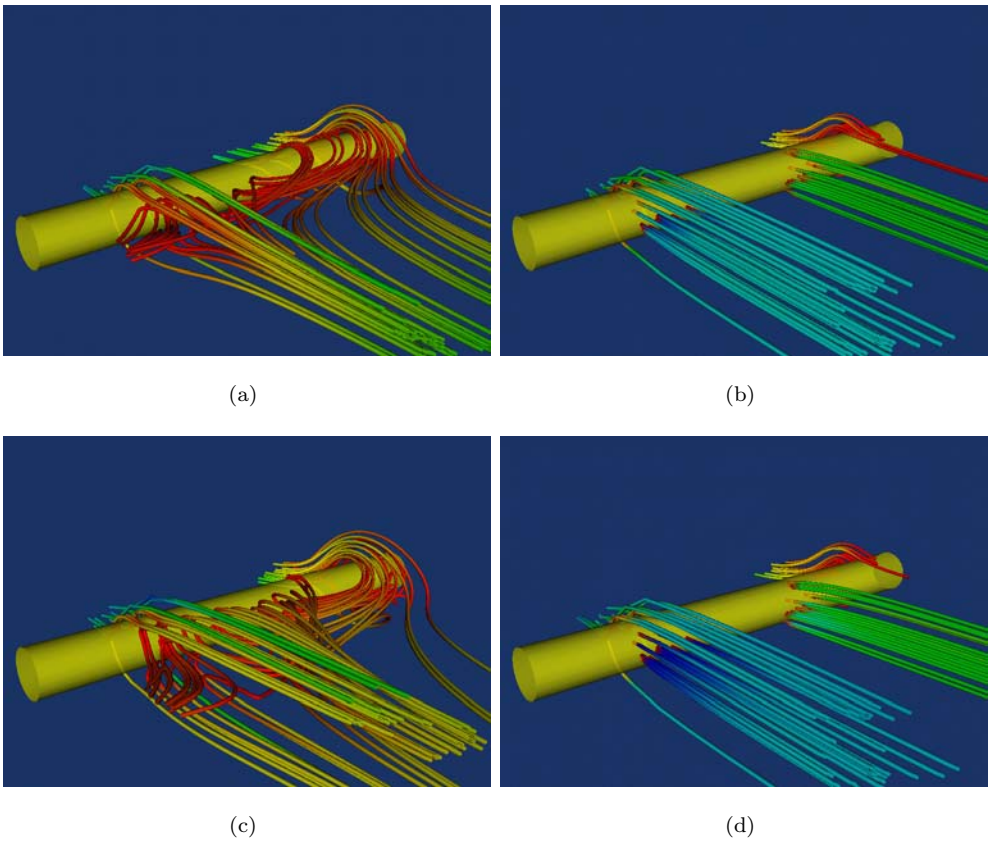


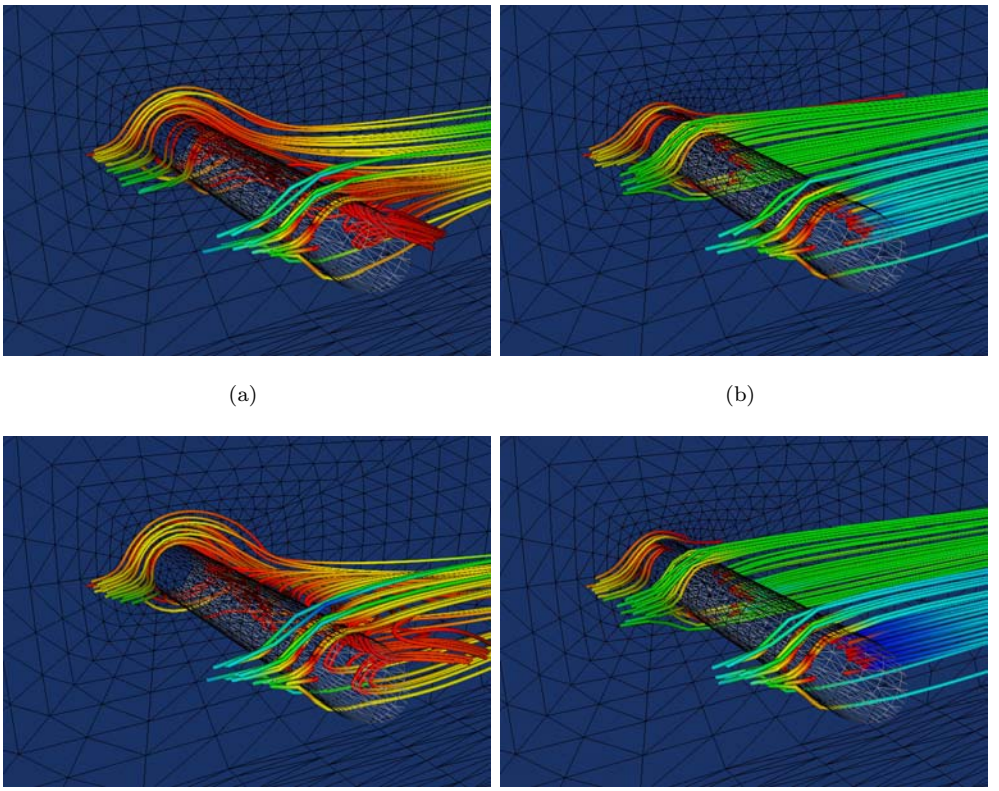
FIG. 1. *The top row depicts stream tubes of the flow for Reynolds number* 20 *and the bottom row for Reynolds number* 40. *The left column depicts the uncontrolled flow. The right column depicts the controlled flow. These images illustrate the flow pattern on the downstream side of the cylinder.*

step is 401 seconds. The time for the Stokes solver is little higher, 421 seconds.[8] We conclude that the forward preconditioner performs reasonably well.

Table 4.4 shows results for 32, 64, and 128 processors of a T3E-900 for increasing problem sizes. Results for two different preconditioning variants of LNKS are presented: the exact (LNKS-I) and inexact (LNKS-II) version of the Schur preconditioner. The globalized LNKS algorithm is compared with QN-RSQP. In LNKS-II-TR we activate the inexact Newton method. In this example we have used continuation to warm start the $\mathrm{Re} = 60$ problem. The reduced Hessian preconditioner is a combination of the BFGS and two-step preconditioners (as we described in part I, section 3.2). In the line search we use the augmented Lagrangian merit function.

Based on the overall wall-clock time reported in the rightmost column of Table 4.4, we observe that for this problem, QN-RSQP was able to converge, but only several days of wall-clock time. LNKS-I, although faster, does not reduce the required time significantly. LNKS-II, which avoids the exact forward/adjoint solves, is much

[8]The reason for this is related to the scaling between the velocity and pressure block of the forward problem. Increasing the Reynolds number improves this scaling and thus improves the eigenvalue distribution. Of course this is true up to certain Reynolds number. For higher values the Jacobian becomes highly nonsymmetric and ill-conditioned.



(a) (b)

(c) (d)

FIG. 2. *The top row depicts stream tubes of the flow for Reynolds number* 20 *and the bottom row for Reynolds number* 40. *The left column depicts the uncontrolled flow. The right column depicts the controlled flow. The decision variables are Dirichlet boundary conditions for the velocities on the downstream half of the cylinder surface. Here we see the flow from the upstream side of the cylinder. In* (c) *we can clearly identify the two standing vortices formed on the lower left corner of the image.*

TABLE 4.3

Forward solver efficiency in relation to problem size and Reynolds number for three-dimensional flow around a cylinder. PEs *is processor number;* n *is the problem size;* Re *is the Reynolds number;* qmr *is the number of aggregate Krylov iterations required to satisfy* $\|\boldsymbol{r}\|/\|\boldsymbol{r}_0\| \leq 1 \times 10^{-7}$*;* nw *is the number of Newton steps to satisfy* $\|\mathbf{c}\|/\|\boldsymbol{c}_0\| \leq 1 \times 10^{-6}$*; and* t *is time in seconds. The runs were performed on a T3E-900.*

		Re = 20			Re = 30			Re = 60		
PEs	n	qmr	nw	t	qmr	nw	t	qmr	nw	t
32	117.048	2,905	5	612	3,467	7	732	2,850	6	621
64	389,440	4,845	5	1,938	5,423	7	2,101	5,501	7	2,310
128	615,981	6,284	5	2,612	8,036	8	3,214	7,847	7	3,136

better—4 to 5 times faster than QN-RSQP. Even more dramatic is the acceleration achieved by using the inexact version of LNKS, i.e., LNKS-II-TR. The inexactness did not interfere at any point with the merit function and in all cases we observed quadratic convergence. Overall, LNKS-II-TR runs more than 10 times faster than

TABLE 4.4

The table shows results for 32, 64, *and* 128 *processors of a Cray T3E for a roughly doubling of problem size. Results for the QN-RSQP and LNKS algorithms are presented.* QN-RSQP *is quasi-Newton reduced-space SQP;* LNKS-I *requires two exact linearized forward/adjoint solves per Krylov step combined with the two-step-stationary-BFGS preconditioner for the reduced Hessian; in* LNKS-II *the exact solves have been replaced by approximate solves;* LNKS-II-TR *uses an exact Newton method that avoids fully converging the KKT system for iterates that are far from a solution.* Time *is wall-clock time in hours on a T3E-900. Continuation was used only for* Re = 60.

Re = 30

States Controls	Method	N or QN iter	Average KKT iter	Time (hours)
117,048	QN-RSQP	161	—	32.1
2,925	LNKS-I	5	18	22,8
(32 procs)	LNKS-II	6	1,367	5,7
	LNKS-II-TR	11	163	1.4
389,440	QN-RSQP	189	—	46.3
6,549	LNKS-I	6	19	27.4
(64 procs)	LNKS-II	6	2,153	15.7
	LNKS-II-TR	13	238	3.8
615,981	QN-RSQP	204	—	53.1
8,901	LNKS-I	7	20	33.8
(128 procs)	LNKS-II	6	3,583	16.8
	LNKS-II-TR	12	379	4.1

Re = 60

States Controls	Method	N or QN iter	Average KKT iter	Time (hours)
117,048	QN-RSQP	168	—	33.4
2,925	LNKS-I	6	20	31,7
(32 procs)	LNKS-II	7	1,391	6,8
	LNKS-II-TR	11	169	1.5
389,440	QN-RSQP	194	—	49.1
6,549	LNKS-I	8	21	44.2
(64 procs)	LNKS-II	7	2,228	18.9
	LNKS-II-TR	15	256	4.8
615,981	QN-RSQP	211	—	57.3
8,901	LNKS-I	8	22	45.8
(128 procs)	LNKS-II	8	3,610	13.5
	LNKS-II-TR	16	383	5.1

QN-RSQP (5.1 hours compared to 57.3 hours). This is consistent with the performance improvements we observed for the Stokes flow control problem.

Despite the high degree of nonlinearity of the external cylinder flow problem, the augmented Lagrangian globalization performed robustly and we did not have problems converging the optimality equations. In this example, the QN-RSQP safeguard was never activated due to detection of a negative curvature direction. Finally, it is worth noting that the optimization solution is found at a cost of 5 [illegible] — remarkable considering that there are thousands of control variables.

4.4. Flow around a Boeing 707 wing. For our last test we solved for the optimal flow control around a Boeing 707 wing. In this problem the control variables

TABLE 4.5

In this table we present results for the wing flow problem, which has 710,023 *state and* 4,984 *decision variables. The runs were performed on* 128 *processors on a T3E-900. Here* Re *is the Reynolds number;* iter *is the aggregate number of Lagrange–Newton iterations—the number in parentheses is the number of iterations in the last continuation step;* time *is the overall time in hours;* qn *is the number of QN-RSQP steps taken as a safeguard for Newton (the number in parentheses reports the number times a negative curvature was detected);* minc *is the number of nonmonotone line search iterations—in parentheses is the number of times this heuristic failed. The globalized LNKS-*II*-TR algorithm is used. The Lagrange–Newton solver was stopped after* 50 *iterations. In the last column* $\mathrm{Re} \times \Delta f$ *gives the reduction of the objective function (with the respect the uncontrolled flow). "no cont" means that continuation was not activated.*

Re	Iter	Time	qn	Minc	$\|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\|$	$\|\mathbf{c}\|$	$\mathrm{Re} \times \Delta f$
100 no cont	19	4.06	2	4	9×10^{-6}	9×10^{-6}	4.065
cont							
200 no cont	39	7.8	6(1)	2	9×10^{-6}	9×10^{-6}	5.804
cont	20(10)	4.6	0	3	9×10^{-6}	9×10^{-6}	5.805
300 no cont	48	11.8	16(3)	0	9×10^{-6}	9×10^{-6}	6.012
cont	29(11)	6.4	0	2	9×10^{-6}	9×10^{-6}	6.016
400 no cont	50	13.6	40(3)	0	2×10^{-4}	3×10^{-3}	3.023
cont	33(11)	7.36	0	6(1)	9×10^{-6}	9×10^{-6}	8.345
500 no cont	50	16.7	42(5)	0	4×10^{-2}	9×10^{-2}	1.235
cont	39(14)	9.09	1	5(1)	9×10^{-6}	9×10^{-6}	10.234

are the velocities (Dirichlet conditions) on the downstream half of the wing. The Reynolds number (based on the length of the wing root) was varied from 100 to 500, and the angle of attack was fixed at 12.5 degrees. The problem size is characterized by 710,023 state variables and 4,984 control variables.

Table 4.5 summarizes the results from this set of experiments. The main purpose of this analysis is to compare continuation with the other globalization techniques. In addition we employ the double inexactness idea; that is, we solve inexactly in both the continuation loop and the Lagrange–Newton loop. It is apparent that the use of continuation is crucial in this problem. Without it, and for Reynolds numbers larger than 300, LNKS was forced to terminate early (we set the Lagrange–Newton iteration bound to 50). In the last row (Re = 500) and when we did not use continuation, LNKS ends up switching to a QN-RSQP step 42 times out of a total of 50 iterations; a negative curvature direction was detected 5 times. As a result LNKS was terminated without satisfying the convergence criteria. Furthermore, the small reduction in the objective function and the residuals (last three columns) indicate small progress at each optimization step. Note that in these examples we did not activate backtracking in the continuation parameter.

It could be argued that a reason the algorithm stagnated was the early termination of the Krylov–Schur solver due to inexactness. We did not conduct exhaustive experiments to confirm or reject this. However, our experience on numerous problems suggests that it is the ill-conditioning and nonlinearity of these problems that leads to stagnation and not the inexactness. In our tests (systematic or during debugging and development) it was never the case that a run with exact solves converged in reasonable time, and the inexact version did not. On the contrary, inexactness significantly reduced execution times.

On the other hand, when we used continuation (consisting of relatively large steps on the Reynolds number), the algorithm converged successfully after 39 Lagrange–Newton iterations. Reverting to QN-RSQP was required just once. In the minc

column of Table 4.5 we monitor the nonmonotone line search criterion. Recall that if the merit function line search on the LNKS step fails, we perform a line search with a different merit—the KKT residual (i.e., the first-order optimality conditions). If the step gets accepted, via backtracking, we use it as an update direction. Eventually, we insist that the (augmented Lagrangian) merit gets reduced. This strategy was successful 20 times and failed only twice.[9]

Finally we conclude with some comments on the physics of this problem. Figures 3 and 4 depict snapshots of the uncontrolled and controlled flow for Reynolds number 500. The wing-tip vortices are eliminated by optimization. But at what cost? Figure 5 shows a snapshot of the (scaled) control variables—the velocity boundary conditions. The image illustrates that optimization has created a perforated wing, which means a significant reduction in lift (the plane will never leave the ground!). Changing the problem to include a penalty on reduction in lift would remedy the difficulty; this example demonstrates the importance of including all relevant desired goals in the objective, lest optimization defeat them.

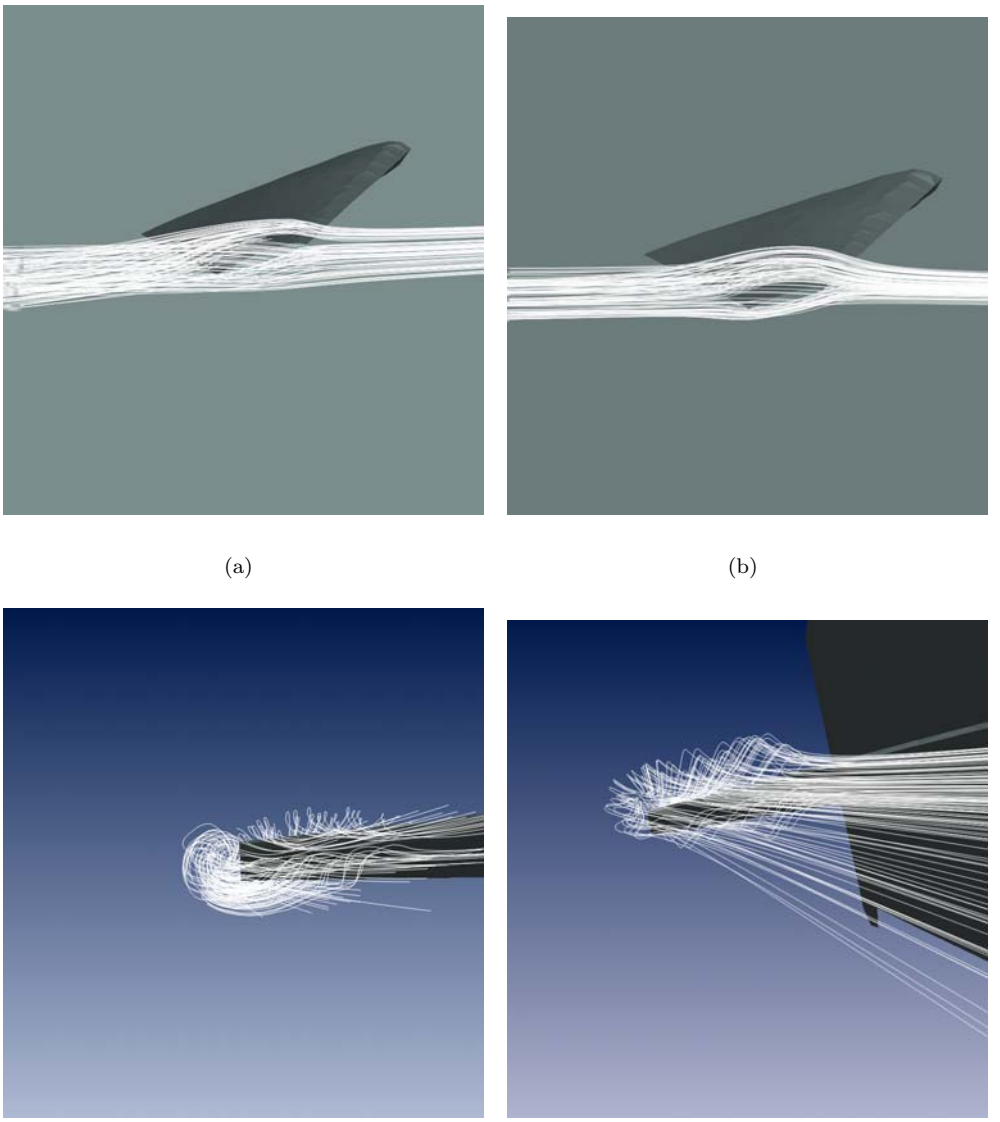
5. Conclusions. In this second part of our two-part article on the LNKS method for PDE-constrained optimization, we have presented the algorithmic components of the (outer) Lagrange–Newton solver, including such globalization strategies as line search, quasi-Newton safeguarding, and parameter continuation, and inexactness in the inner linear solver and its interaction with the line search. We studied the application of the LNKS method to a set of challenging viscous flow optimal boundary control problems that included such compounding factors as three-dimensional unstructured meshes, multicomponent coupling, large problem size, nonlinearity, and ill-conditioning. Our experiments demonstrate the efficacy and scalability of the LNKS method. The Krylov–Schur preconditioner maintained its effectiveness, the Lagrange–Newton solver exhibited mesh-independent convergence, and inexact Newton steps dramatically accelerated the method. A combination of line searching, quasi-Newton safeguarding, and continuation ensured global convergence.

The results reveal at least an order of magnitude improvement in solution time over popular quasi-Newton RSQP methods, rendering tractable some three-dimensional problems with close to a million state variables and several thousand control variables on parallel computers. Indeed, the optimum is often found in a , ,, multiple of the cost of a single simulation.

The LNKS method is more suitable for steady PDE constraints. Although the method is in principle applicable to time-dependent problems, it is not recommended for three-dimensional unsteady problems: because LNKS sees the entire space-time domain, it requires large memory for the state and adjoint variables. Moreover, the opportunities for economizing by hiding the nonlinear and linear forward iterations behind the optimization iterations are limited, since the nonlinear systems at each time step are usually mildly nonlinear and well-conditioned (for time-accurate integration). We are investigating various ways to address this issue. Another important extension of LNKS is the treatment of inequality constraints via interior point methods.

Acknowledgments. We thank the PETSc development group at Argonne National Lab for making this work possible. We also thank Jonathan Shewchuk for providing the meshing and partitioning routines Pyramid and Slice, and David Marcum for providing the meshing module AFLR. Finally, we thank David Keyes, David

[9] In general, using the residual of the KKT conditions to test a step can compromise robustness since the optimizer may become trapped in a saddle point or a local maximum.



(a) (b)

(c) (d)

FIG. 3. *The left column depicts streamlines of the uncontrolled flow, while the right depicts streamlines of the controlled flow. Top row gives a side snapshot of the flow; bottom row gives a front view. The Reynolds number (based on the length of the root of the wing) is* 500.

Young of Boeing, and the other members of the TAOS project—Roscoe Bartlett, Larry Biegler, and Andreas Wächter—for their useful comments.

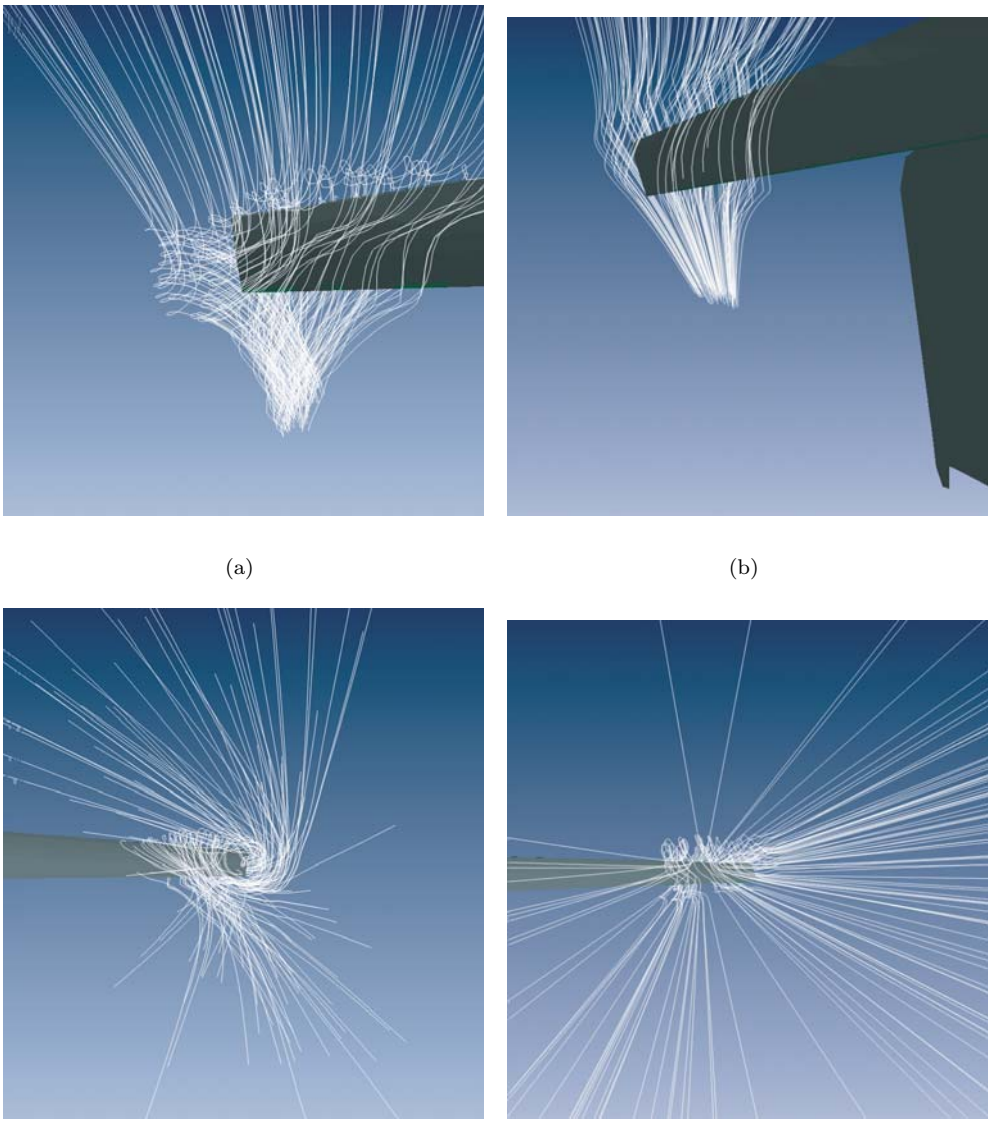


FIG. 4. *The left column depicts streamlines of the uncontrolled flow. The right column depicts streamlines of the controlled flow. Top row gives a snapshot of the flow from below; bottom row gives a rear view. Reynolds number is* 500*. We can clearly identify the wing tip vortices in the left images. The images in the right column depict the flow with the wing boundary conditions modified by optimization; the vorticity is eliminated (but so is the lift, since it does not enter into the objective function).*

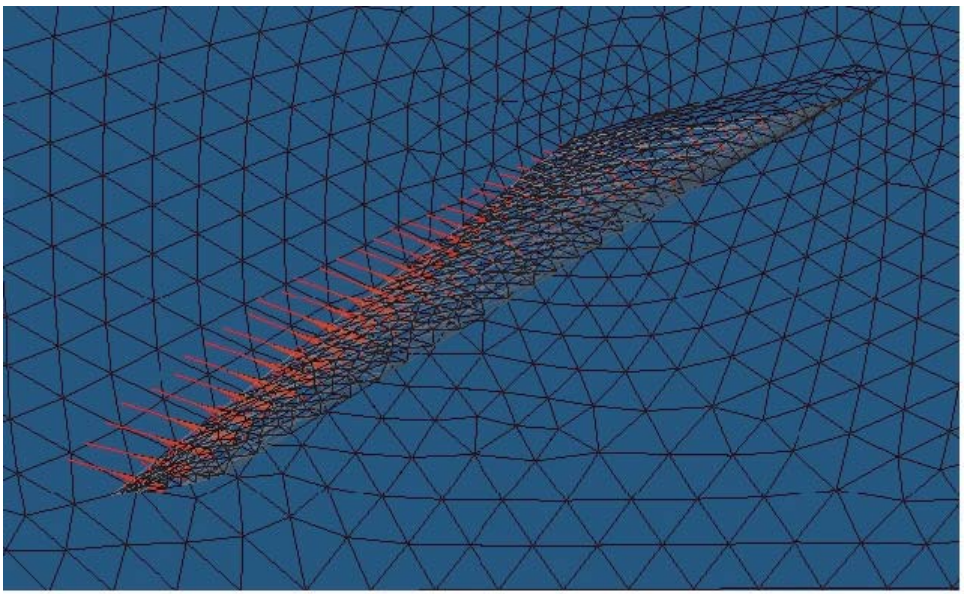


FIG. 5. *Snapshot of the (Dirichlet control) velocity field on the wing.*

REFERENCES

[1] E. L. ALLGOWER AND K. GEORG, *Introduction to Numerical Continuation Methods*, SIAM, Philadelphia, 2003.

[2] I. BABUŠKA, *The finite element method with Lagrangian multipliers*, Numer. Math., 20 (1973), pp. 179–192.

[3] G. BIROS AND O. GHATTAS, *Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* I: *The Krylov–Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687–713.

[4] P. T. BOGGS AND J. W. TOLLE, *A strategy for global convergence in a sequential quadratic programming algorithm*, SIAM J. Numer. Anal., 26 (1989), pp. 600–623.

[5] J. E. DENNIS, JR., M. EL-ALEM, AND M. C. MACIEL, *A global convergence theory for general trust-region-based algorithms for equality constrained optimization*, SIAM J. Optim., 7 (1997), pp. 177–207.

[6] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[7] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[8] R. FLETCHER, *Practical Methods of Optimization*, 2nd ed., John Wiley and Sons, Chichester, UK, 1987.

[9] P. E. GILL, W. MURRAY, AND M. H. WRIGHT, *Practical Optimization*, Academic Press, London, New York, 1981.

[10] M. D. GUNZBURGER, *Finite Element Methods for Viscous Incompressible Flows*, Academic Press, Boston, 1989.

[11] M. D. GUNZBURGER, ED., *Flow Control*, IMA Vol. Math. Appl. 68, Springer-Verlag, New York, 1995.

[12] M. D. GUNZBURGER AND S. L. HOU, *Treating inhomogeneous essential boundary conditions in finite element methods and the calculation of boundary stresses*, SIAM J. Numer. Anal., 29 (1992), pp. 390–424.

[13] M. D. GUNZBURGER AND R. A. NICOLAIDES, EDS., *Incompressible Computational Fluid Dynamics*, Cambridge University Press, Cambridge, UK, 1993.

[14] M. HEINKENSCHLOSS AND L. N. VICENTE, *Analysis of Inexact Trust-Region SQP Algorithms*, Technical report TR99-18, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1999.

[15] L. S. Hou, *Analysis and Finite Element Approximation of Some Optimal Control Problems Associated with the Navier-Stokes Equations*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1989.
[16] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results*, SIAM J. Sci. Comput., 20 (1999), pp. 1753–1777.
[17] C. T. Kelley and E. W. Sachs, *Truncated Newton methods for optimization with inaccurate functions and gradients*, J. Optim. Theory Appl., 116 (2003), pp. 83–98.
[18] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Numer. Anal., 35 (1998), pp. 508–523.
[19] F. Leibfritz and E. W. Sachs, *Inexact SQP interior point methods and large scale optimal control problems*, SIAM J. Control Optim., 38 (1999), pp. 272–293.
[20] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, New York, 1996.
[21] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.
[22] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.
[23] K. Schittkowski, *The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function*, Numer. Math., 38 (1981), pp. 83–114.
[24] T. Steihaug, *The conjugate gradient method and trust regions in large scale optimization*, SIAM J. Numer. Anal., 20 (1983), pp. 626–637.
[25] H. Yamashita, *A globally convergent constrained quasi-Newton method with an augmented Lagrangian type penalty function*, Math. Program., 23 (1982), pp. 75–86.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 741–758
© 2005 Society for Industrial and Applied Mathematics

OPTIMIZING THE EVALUATION OF FINITE ELEMENT MATRICES*

ROBERT C. KIRBY†, MATTHEW KNEPLEY‡, ANDERS LOGG§, AND L. RIDGWAY SCOTT¶

Abstract. Assembling stiffness matrices represents a significant cost in many finite element computations. We address the question of optimizing the evaluation of these matrices. By finding redundant computations, we are able to significantly reduce the cost of building local stiffness matrices for the Laplace operator and for the trilinear form for Navier–Stokes operators. For the Laplace operator in two space dimensions, we have developed a heuristic graph algorithm that searches for such redundancies and generates code for computing the local stiffness matrices. Up to cubics, we are able to build the stiffness matrix on any triangle in less than one multiply-add pair per entry. Up to sixth degree, we can do it in less than about two pairs. Preliminary low-degree results for Poisson and Navier-Stokes operators in three dimensions are also promising.

Key words. finite element, compiler, variational form

AMS subject classifications. 65D05, 65N15, 65N30

DOI. 10.1137/040607824

1. Introduction.

It has often been observed that the formation of the matrices arising from finite element methods over unstructured meshes takes a substantial amount of time and is one of the primary disadvantages of finite elements over finite differences. Here, we will show that the standard algorithm for computing finite element matrices by integration formulae is far from optimal and will present a technique that can generate algorithms with considerably fewer operations even than well-known precomputation techniques.

From fairly simple examples with Lagrangian finite elements, we will present a novel optimization problem and present heuristics for the automatic solution of this problem. We demonstrate that the stiffness matrix for the Laplace operator can be computed in about one multiply-add pair per entry in two dimensions up to cubics, and about two multiply-add pairs up to degree 6. Low-order examples in three dimensions suggest similar possibilities, which we intend to explore in the future. More important, the techniques are not limited to linear problems—in fact, we show the potential for significant optimizations for the nonlinear term in the Navier–Stokes equations. These results seem to have lower flop counts than even the best quadrature rules for simplices.

*Received by the editors May 4, 2004; accepted for publication (in revised form) April 22, 2005; published electronically November 15, 2005.
http://www.siam.org/journals/sisc/27-3/60782.html

†Department of Computer Science, University of Chicago, Chicago, IL 60637-1581 (kirby@cs.uchicago.edu).

‡Division of Mathematics and Computer Science, Argonne National Laboratory, 9700 Cass Ave., Argonne, IL 60439-4844 (knepley@mcs.anl.gov). This author's work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under contract W-31-109-Eng-38.

§Toyota Technological Institute at Chicago, University Press Building, 1427 East 60th St., Second Floor, Chicago, IL 60637 (logg@tti-c.org).

¶The Computation Institute and Departments of Computer Science and Mathematics, University of Chicago, Chicago, IL 60637-1581 (ridg@cs.uchicago.edu).

Our long-term goal is to develop a "form compiler" for finite element methods. Such a compiler will map high-level descriptions of the variational problem and finite element spaces into low-level code for building the algebraic systems. Currently, the Sundance project [21, 20] and the DOLFIN project [10] are developing run-time C++ systems for the assembly variational forms. Recent work in the PETSc project [18] is leading to compilation of variational forms into C code for building local matrices. Our work here complements these ideas by suggesting compiler optimizations for such codes that would greatly enhance the run-time performance of the matrix assembly.

Automating tedious but essential tasks has proven remarkably successful in many areas of scientific computing. Automatic differentiation of numerical code has allowed complex algorithms to be used reliably [5]. Indeed the family of automatic differentiation tools [1] that automatically produce efficient gradient, adjoint, and Hessian algorithms for existing code has been invaluable in enabling optimal control calculations and Newton-based nonlinear solvers.

2. Matrix evaluation by assembly. Finite element matrices are assembled by summing the constituent parts over each element in the mesh. This is facilitated through the use of a numbering scheme called the local-to-global index. This index, $\iota(e, \lambda)$, relates the local (or element) node number, $\lambda \in \mathcal{L}$, on a particular element, indexed by e, to its position in the global node ordering [4]. This local-to-global indexing works for Lagrangian finite elements, but requires generalizations for other families of elements. While this generalization is required for assembly, our techniques for optimizing the computation over each element can still be applied in those situations.

We may write a finite element function f in the form

$$\sum_e \sum_{\lambda \in \mathcal{L}} f_{\iota(e,\lambda)} \phi^e_\lambda, \tag{2.1}$$

where f_i denotes the "nodal value" of the finite element function at the ith node in the global numbering scheme and $\{\phi^e_\lambda \,:\, \lambda \in \mathcal{L}\}$ denotes the set of basis functions on the element domain T_e. By definition, the element basis functions, ϕ^e_λ, are extended by zero outside T_e. In many important cases, we can relate all of the "element" basis functions ϕ^e_λ to a fixed set of basis functions on a "reference" element, $\mathcal{T}$, via some mapping of $\mathcal{T}$ to T_e. This mapping could involve changing both the "x" values and the "ϕ" values in a coordinated way, as with the Piola transform [2], or it could be one whose Jacobian is nonconstant, as with tensor-product elements or isoparametric elements [4]. But for a simple example, we assume that we have an affine mapping, $\xi \to J\xi + x_e$, of $\mathcal{T}$ to T_e:

$$\phi^e_\lambda(x) = \phi_\lambda \left(J^{-1}(x - x_e) \right).$$

The inverse mapping $x \to \xi = J^{-1}(x - x_e)$ has as its Jacobian

$$J^{-1}_{mj} = \frac{\partial \xi_m}{\partial x_j},$$

and this is the quantity which appears in the evaluation of the bilinear forms. Of course, $\det J = 1/\det J^{-1}$.

The assembly algorithm utilizes the decomposition of a variational form as a sum over "element" forms

$$a(v, w) = \sum_e a_e(v, w),$$

where the "element" bilinear form for Laplace's equation is defined (and evaluated) via

$$
\begin{aligned}
a_e(v,w) &:= \int_{T_e} \nabla v(x)\cdot\nabla w(x)\,dx \\
&= \int_{\mathcal{T}} \sum_{j=1}^{d} \frac{\partial}{\partial x_j} v(J\xi + x_e) \\
&\quad \frac{\partial}{\partial x_j} w(J\xi + x_e)\det(J)\,d\xi \\
&= \int_{\mathcal{T}} \sum_{j,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial}{\partial \xi_m}\left(\sum_{\lambda\in\mathcal{L}} v_{\iota(e,\lambda)}\phi_\lambda(\xi)\right) \\
&\quad \times \frac{\partial \xi_{m'}}{\partial x_j}\frac{\partial}{\partial \xi_{m'}}\left(\sum_{\mu\in\mathcal{L}} w_{\iota(e,\mu)}\phi_\mu(\xi)\right)\det(J)\,d\xi \\
&= \begin{pmatrix} v_{\iota(e,1)} \\ \cdot \\ \cdot \\ v_{\iota(e,|\mathcal{L}|)} \end{pmatrix}^t \mathbf{K}^e \begin{pmatrix} w_{\iota(e,1)} \\ \cdot \\ \cdot \\ w_{\iota(e,|\mathcal{L}|)} \end{pmatrix}.
\end{aligned} \tag{2.2}
$$

Here, the [illegible], $\mathbf{K}^e$, is given by

$$
\begin{aligned}
K^e_{\lambda,\mu} &:= \sum_{j,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_{m'}}{\partial x_j}\det(J)\int_{\mathcal{T}} \frac{\partial}{\partial \xi_m}\phi_\lambda(\xi)\frac{\partial}{\partial \xi_{m'}}\phi_\mu(\xi)\,d\xi \\
&= \sum_{j,m,m'=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_{m'}}{\partial x_j}\det(J) K_{\lambda,\mu,m,m'} \\
&= \sum_{m,m'=1}^{d} G^e_{m,m'} K_{\lambda,\mu,m,m'},
\end{aligned} \tag{2.3}
$$

where

$$
K_{\lambda,\mu,m,m'} = \int_{\mathcal{T}} \frac{\partial}{\partial \xi_m}\phi_\lambda(\xi)\frac{\partial}{\partial \xi_{m'}}\phi_\mu(\xi)\,d\xi \tag{2.4}
$$

and

$$
G^e_{m,m'} := \det(J)\sum_{j=1}^{d} \frac{\partial \xi_m}{\partial x_j}\frac{\partial \xi_{m'}}{\partial x_j} \tag{2.5}
$$

for $\lambda,\mu\in\mathcal{L}$ and $m,m' = 1,\ldots,d$.

The matrix associated with a bilinear form is

$$
A_{ij} := a(\phi_i,\phi_j) = \sum_e a_e(\phi_i,\phi_j) \tag{2.6}
$$

for all i,j, where ϕ_i denotes a global basis function. We can compute this again by assembly.

First, set all the entries of A to zero. Then loop over all elements e and local element numbers λ and μ and compute

$$A_{\iota(e,\lambda),\iota(e,\mu)} += K^e_{\lambda,\mu} = \sum_{m,m'} G^e_{m,m'} K_{\lambda,\mu,m,m'}, \tag{2.7}$$

where $G^e_{m,m'}$ are defined in (2.5). One can imagine trying to optimize the computation of each

$$K^e_{\lambda,\mu} = \sum_{m,m'} G^e_{m,m'} K_{\lambda,\mu,m,m'}, \tag{2.8}$$

but each such term must be computed separately. We consider this optimization in section 3.

3. Computing the Laplacian stiffness matrix with general elements. The tensor $K_{\lambda,\mu,m,n}$ can be presented as an $|\mathcal{L}| \times |\mathcal{L}|$ matrix of $d \times d$ matrices, as presented in Table 3.1 for the case of quadratics in two dimension. The entries of resulting matrix K^e can be viewed as the dot (or Frobenius) product of the entries of K and G^e. That is,

$$K^e_{\lambda,\mu} = \mathbf{K}_{\lambda,\mu} : G^e. \tag{3.1}$$

The key point is that certain dependencies among the entries of K can be used to significantly reduce the complexity of building each K^e. For example, the four 2×2 matrices in the upper-left corner of Table 3.1 have only one nonzero entry, and six others in K are zero. There are significant redundancies among the rest. For example, $\mathbf{K}_{3,1} = -4\mathbf{K}_{4,1}$, so once $\mathbf{K}_{4,1} : G^e$ is computed, $\mathbf{K}_{3,1} : G^e$ may be computed by only one additional operation.

By taking advantage of these simplifications, we see that each K^e for quadratics in two dimensions can be computed with at most 18 floating point operations (see section 3.2) instead of 288 floating point operations using the straightforward definition, an improvement of a factor of 16 in computational complexity.

The tensor $K_{\lambda,\mu,m,n}$ for the case of linears in three dimensions is presented in Table 3.2. Each K^e can be computed by computing the three row sums of G^e, the three column sums, and the sum of one of these sums. We also have to negate all of the column and row sums, leading to a total of 20 floating point operations instead of 288 floating point operations using the straightforward definition, an improvement of a factor of nearly 15 in computational complexity. Using symmetry of G^e (row sums equal column sums) we can reduce the computation to only 10 floating point operations, leading to an improvement of nearly 29.

We leave it as an exercise to work out the reductions in computation that can be done for the case of linears in two dimensions.

3.1. Algorithms for determining reductions. Obtaining reductions in computation can be done systematically as follows. It may be useful to work in rational arithmetic and keep track of whether terms are exactly zero and determine common divisors. However, floating point representations may be sufficient in many cases. We can start by noting which submatrices are zero, which ones have only one nonzero element, and so forth. Next, we find arithmetic relationships among the submatrices, as follows.

Determining whether

$$\mathbf{K}_{\lambda,\mu} = c\mathbf{K}_{\lambda',mu'} \tag{3.2}$$

TABLE 3.1
The tensor K for quadratics represented as a matrix of 2×2 matrices.

3	0	0	-1	1	1	-4	-4	0	4	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
-1	0	0	3	1	1	0	0	4	0	-4	-4
1	0	0	1	3	3	-4	0	0	0	0	-4
1	0	0	1	3	3	-4	0	0	0	0	-4
-4	0	0	0	-4	-4	8	4	0	-4	0	4
-4	0	0	0	0	0	4	8	-4	-8	4	0
0	0	0	4	0	0	0	-4	8	4	-8	-4
4	0	0	0	0	0	-4	-8	4	8	-4	0
0	0	0	-4	0	0	0	4	-8	-4	8	4
0	0	0	-4	-4	-4	4	0	-4	0	4	8

TABLE 3.2
The tensor K (multiplied by four) for piecewise linears in three dimensions represented as a matrix of 3×3 matrices.

1	0	0	0	1	0	0	0	1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	-1	-1	-1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1	-1	-1	-1
-1	0	0	0	-1	0	0	0	-1	1	1	1
-1	0	0	0	-1	0	0	0	-1	1	1	1
-1	0	0	0	-1	0	0	0	-1	1	1	1

requires just simple linear algebra. We think of these as vectors in d^2-dimensional space and just compute the angle between the vectors. If this angle is zero, then (3.2) holds. Again, if we assume that we are working in rational arithmetic, then c could be determined as a rational number.

Similarly, a third vector can be written in terms of two others by considering its projection on the first two. That is,

$$\mathbf{K}_{\lambda,\mu} = c_1 \mathbf{K}_{\lambda^1,\mu^1} + c_2 \mathbf{K}_{\lambda^2,\mu^2} \tag{3.3}$$

if and only if the projection of $\mathbf{K}_{\lambda,\mu}$ onto the plane spanned by $\mathbf{K}_{\lambda^1,\mu^1}$ and $\mathbf{K}_{\lambda^2,\mu^2}$ is equal to $\mathbf{K}_{\lambda,\mu}$.

Higher-order relationships can be determined similarly by linear algebra as well. Note that any $d^2 + 1$ entries $\mathbf{K}_{\lambda,\mu}$ will be linearly dependent, since they are in d^2-dimensional space. Thus we might only expect lower-order dependencies to be useful in reducing the computational complexity.

We can then form graphs that represent the computation of K^e in (3.1). The graphs have d^2 nodes representing G^e as inputs and $|\mathcal{L}| \times |\mathcal{L}|$ nodes representing the entries of K^e as outputs. Internal edges and nodes represent computations and temporary storage. The inputs to a given computation come directly from G^e or indirectly from other internal nodes in the graph.

The computation represented in (2.7) would correspond to a dense graph in which each of the input nodes is connected directly to all $|\mathcal{L}| \times |\mathcal{L}|$ output nodes. It will be possible to attempt to reduce the complexity of the computation by finding sparse

graphs that represent equivalent computations.

We can generate interesting graphs by analyzing the entries of $\mathbf{K}_{\lambda,\mu}$ for relationships as described above. It would appear useful to consider

- entries $\mathbf{K}_{\lambda,\mu}$ which have only one nonzero element,
- entries $\mathbf{K}_{\lambda,\mu}$ which have $2 \leq k \ll d^2$ nonzero elements,
- entries $\mathbf{K}_{\lambda,\mu}$ which are scalar multiples of other elements,
- entries $\mathbf{K}_{\lambda,\mu}$ which are linear combinations of other elements,

and so forth. For each graph representing the computation of K^e, we have a precise count of the number of floating point operations, and we can simply minimize the total number over all graphs generated by the above heuristics. Our examples indicate that computational simplifications consisting of an order of magnitude or more can be achieved.

We have developed a code called `FErari` (Finite Element ReARrangements of Integrals or Finite Element Rearrangement to Automatically Reduce Instructions) to carry out such optimizations automatically. We describe this in detail in section 4.3. This code linearizes the graph representation discussed above, taking a particular evaluation path that is based on heuristics chosen to approximate optimal evaluation. We have used it to show that for classes of elements, significant improvements in computational efficiency are available.

3.2. Computing K for quadratics. Here we give a detailed algorithm for computing K for quadratics. Thus we have

(3.4)
$$6 * K^e = \begin{pmatrix} 3G_{11} & -G_{12} & \gamma_{11} & \gamma_0 & 4G_{12} & 0 \\ -G_{21} & 3G_{22} & \gamma_{22} & 0 & 4G_{21} & \gamma_1 \\ \gamma_{11} & \gamma_{22} & 3(\gamma_{11}+\gamma_{22}) & \gamma_0 & 0 & \gamma_1 \\ \gamma_0 & 0 & \gamma_0 & \gamma_2 & -\gamma_3 - 8G_{22} & \gamma_3 \\ 4G_{21} & 4G_{12} & 0 & -\gamma_3 - 8G_{22} & \gamma_2 & -\gamma_3 - 8G_{11} \\ 0 & \gamma_1 & \gamma_1 & \gamma_3 & -\gamma_3 - 8G_{11} & \gamma_2 \end{pmatrix},$$

where the G_{ij}'s are the inputs and the intermediate quantities γ_i are defined and computed from

(3.5)
$$\begin{aligned} \gamma_0 &= -4\gamma_{11}, \\ \gamma_1 &= -4\gamma_{22}, \\ \gamma_2 &= 4G_{1221} + 8G_{1122} = \gamma_3 + 8\gamma_{12} = 8(G_{12} + \gamma_{12}), \\ \gamma_3 &= 4G_{1221} = 4\gamma_{21} = 8G_{12}, \end{aligned}$$

where we use the notation $G_{ijk\ell} := G_{ij} + G_{k\ell}$, and finally, the γ_{ij}'s are

(3.6)
$$\begin{pmatrix} \gamma_{11} = G_{11} + G_{12} = G_{1112} & \gamma_{12} = G_{11} + G_{22} = G_{1122} \\ \gamma_{21} = G_{12} + G_{21} = G_{1221} & \gamma_{22} = G_{12} + G_{22} = G_{1222} \end{pmatrix}.$$

Let us distinguish different types of operations. The above formulas involve (a) negation, (b) multiplication of integers and floating point numbers, and (c) additions of floating point numbers. Since the order of addition is arbitrary, we may assume that the operations (c) are commutative (although changing the order of evaluation may change the result). Thus we have $G_{1222} = G_{2212}$ and so forth. The symmetry of G implies that $G_{1112} = G_{1121}$ and $G_{2122} = G_{1222}$. The symmetry of G implies that K^e is also symmetric, by inspection, as it must be from the definition.

The computation of the entries of K^e proceeds as follows. The computations in (3.6) are done first and require only four (c) operations, or three (c) operations and one (b) operation ($\gamma_{21} = 2G_{12}$). Next, the γ_i's are computed via (3.5), requiring four (b) operations and one (c) operation. Finally, the matrix K^e is completed, via three (a) operations, seven (b) operations, and three (c) operations. This makes a total of three (a) operations, twelve (b) operations, and three (c) operations. Thus only 18 operations are required to evaluate K^e, compared with 288 operations via the formula (2.8).

It is clear that there may be other algorithms with the same amount of work (or less) since there are many ways to decompose some of the submatrices in terms of others. Finding (or proving) the absolute minimum may be difficult. Moreover, the metric for minimization should be run time, not some arbitrary way of counting operations. Thus the right way to utilize the ideas we are presenting may be to identify sets of ways to evaluate finite element matrices. These could then be tested on different systems (architectures plus compilers) to see which is the best. It is amusing that it takes fewer operations to compute K^e than it does to write it down, so it may be that memory traffic should be considered in an optimization algorithm that seems to be the most efficient algorithm.

4. Evaluation of general multilinear forms. General multilinear forms can appear in finite element calculations. As an example of a trilinear form, we consider that arising from the first-order term in the Navier–Stokes equations. Though additional issues arise over bilinear forms, many techniques carry over to give efficient algorithms.

The local version of the form is defined by

$$
\begin{aligned}
c_e(\mathbf{u};\mathbf{v},\mathbf{w}) &:= \int_{T_e} \mathbf{u}\cdot\nabla\mathbf{v}(x)\cdot\mathbf{w}(x)\,dx \\
&= \int_{T_e} \sum_{j,k=1}^{d} u_j(x)\frac{\partial}{\partial x_j}v_k(x)w_k(x)\,dx \\
&= \int_{\mathcal{T}} \sum_{j,k=1}^{d} u_j(J\xi+x_e)\frac{\partial}{\partial x_j}v_k(J\xi+x_e)w_k(J\xi+x_e)\det(J)\,d\xi \\
&= \int_{\mathcal{T}} \sum_{j,k,m=1}^{d} \left(\sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)}\phi_\lambda(\xi)\right)\frac{\partial\xi_m}{\partial x_j}\left(\sum_{\mu\in\mathcal{L}} v_k^{\iota(e,\mu)}\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\right) \qquad (4.1)\\
&\quad\times\left(\sum_{\rho\in\mathcal{L}} w_k^{\iota(e,\rho)}\phi_\rho(\xi)\right)\det(J)\,d\xi \\
&= \sum_{j,k,m=1}^{d}\sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)}\frac{\partial\xi_m}{\partial x_j}v_k^{\iota(e,\mu)}w_k^{\iota(e,\rho)}\det(J) \\
&\quad\times\int_{\mathcal{T}}\phi_\lambda(\xi)\frac{\partial}{\partial\xi_m}\phi_\mu(\xi)\phi_\rho(\xi)\,d\xi.
\end{aligned}
$$

Thus we have found that

$$\begin{aligned} c_e(\mathbf{u},\mathbf{v},\mathbf{w}) &= \sum_{j,k=1}^{d} \sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)} \\ &\quad \times \sum_{m=1}^{d} \frac{\partial \xi_m}{\partial x_j} \det(J) \int_{\mathcal{T}} \phi_\lambda(\xi) \frac{\partial}{\partial \xi_m} \phi_\mu(\xi) \phi_\rho(\xi)\, d\xi \\ &= \sum_{j,k=1}^{d} \sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)} \sum_{m=1}^{d} \frac{\partial \xi_m}{\partial x_j} \det(J) N_{\lambda,\mu,\rho,m}, \end{aligned} \tag{4.2}$$

where

$$N_{\lambda,\mu,\rho,m} := \int_{\mathcal{T}} \phi_\lambda(\xi) \frac{\partial}{\partial \xi_m} \phi_\mu(\xi) \phi_\rho(\xi)\, d\xi. \tag{4.3}$$

To summarize, we have

$$\begin{aligned} c_e(\mathbf{u},\mathbf{v},\mathbf{w}) &= \sum_{j,k=1}^{d} \sum_{\lambda,\mu,\rho\in\mathcal{L}} u_j^{\iota(e,\lambda)} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)} N^e_{\lambda,\mu,\rho,j} \\ &= \sum_{k=1}^{d} \sum_{\mu,\rho\in\mathcal{L}} v_k^{\iota(e,\mu)} w_k^{\iota(e,\rho)} \sum_{j=1}^{d} \sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)} N^e_{\lambda,\mu,\rho,j}, \end{aligned} \tag{4.4}$$

where the element coefficients $N^e_{\lambda,\mu,\rho,j}$ are defined by

$$N^e_{\lambda,\mu,\rho,j} := \sum_{m=1}^{d} \frac{\partial \xi_m}{\partial x_j} \det(J) N_{\lambda,\mu,\rho,m}. =: \sum_{m=1}^{d} \widetilde{G}_{mj} N_{\lambda,\mu,\rho,m}, \tag{4.5}$$

where $\widetilde{G}_m j := \frac{\partial \xi_m}{\partial x_j} \det(J)$. Recall that J is the Jacobian above, and J^{-1} is its inverse, and

$$\left(J^{-1}\right)_{m,j} = \frac{\partial \xi_m}{\partial x_j}.$$

Note that both $N_{\lambda,\mu,\rho,(\cdot)}$ and $N^e_{\lambda,\mu,\rho,(\cdot)}$ can be thought of as d-vectors. Moreover

$$N^e_{\lambda,\mu,\rho,(\cdot)} = \det(J) N_{\lambda,\mu,\rho,(\cdot)} J^{-1}.$$

Also note that $N_{\lambda,\mu,\rho,(\cdot)} = N_{\rho,\mu,\lambda,(\cdot)}$, so that considerable storage reduction could be made if desired.

The matrix C defined by $C_{ij} = c(\mathbf{u}, \phi_i, \phi_j)$ can be computed using the assembly algorithm as follows. First, note that C can be written as a matrix of dimension $|\mathcal{L}| \times |\mathcal{L}|$ with entries that are $d \times d$ diagonal blocks. In particular, let I_d denote the $d \times d$ identity matrix. Now set C to zero, loop over all elements, and update the matrix by

$$\begin{aligned} C_{\iota(e,\mu),\iota(e,\rho)} +&= I_d \sum_{j=1}^{d} \sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)} N^e_{\lambda,\mu,\rho,j} \\ &= I_d \sum_{m,j=1}^{d} \widetilde{G}_{mj} \left(\sum_{\lambda\in\mathcal{L}} u_j^{\iota(e,\lambda)} N_{\lambda,\mu,\rho,m} \right) \\ &= I_d \sum_{m,\lambda\in\mathcal{L}} \gamma_{m\lambda} N_{\lambda,\mu,\rho,m} \end{aligned} \tag{4.6}$$

for all μ and ρ, where

$$\gamma_{m\lambda} = \sum_{j=1}^{d} \widetilde{G}_{mj} u_j^{\iota(e,\lambda)}. \tag{4.7}$$

It thus appears that the computation of C can be viewed as similar in form to (3.1), and similar optimization techniques applied. In fact, we can introduce the notation $K^{e,u}$, where

$$K^{e,u}_{\mu,\rho} = \sum_{m,\lambda\in\mathcal{L}} \gamma_{m\lambda} N_{\lambda,\mu,\rho,m}. \tag{4.8}$$

Then the update of C is done in the obvious way with $K^{e,u}$.

4.1. Trilinear forms with piecewise linears. For simplicity, we may consider the piecewise linear case. Here, the standard mixed formulations are not inf-sup stable, but the trilinear form is still an essential part of the well-known family of stabilized methods [8, 11] that do admit equal-order piecewise linear discretizations. Moreover, our techniques could as well be used with the nonconforming linear element [6], which does admit an inf-sup condition when paired with piecewise constant pressures.

In the piecewise linear case, (4.3) simplifies to

$$N_{\lambda,\mu,\rho,m} := \frac{\partial \phi_\mu}{\partial \xi_m} \int_{\mathcal{T}} \phi_\lambda(\xi)\phi_\rho(\xi)\, d\xi. \tag{4.9}$$

We can think of $N_{\lambda,\mu,\rho,m}$ defined from two matrices $N_{\lambda,\mu,\rho,m} = D_{\mu,m}F_{\lambda,\rho}$, where

$$D_{\mu,m} := \frac{\partial \phi_\mu}{\partial \xi_m} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} (d=2) \quad \text{and} \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix} (d=3) \tag{4.10}$$

and

$$F_{\lambda,\rho} := \int_{\mathcal{T}} \phi_\lambda(\xi)\phi_\rho(\xi)\, d\xi. \tag{4.11}$$

The latter matrix is easy to determine. In the piecewise linear case, we can compute integrals of products using the quadrature rule that is based on edge midpoints (with equal weights given by the area of the simplex divided by the number of edges). Thus the weights are $\omega = 1/6$ for $d = 2$ and $\omega = 1/24$ for $d = 3$. Each of the values $\phi_\lambda(\xi)$ is either $\frac{1}{2}$ or zero, and the products are equal to $\frac{1}{4}$ or zero. For the diagonal terms $\lambda = \rho$, the product is nonzero on d edges, so $F_{\lambda,\lambda} = 1/12$ for $d = 2$ and $1/32$ for $d = 3$. If $\lambda \neq \rho$, then the product $\phi_\lambda(\xi)\phi_\rho(\xi)$ is nonzero for exactly one edge (the one connecting the corresponding vertices), so $F_{\lambda,\rho} = 1/24$ for $d = 2$ and $1/96$ for $d = 3$. Thus we can describe the matrices F in general as having d on the diagonals, 1 on the off-diagonals, and scaled by $1/24$ for $d = 2$ and $1/96$ for $d = 3$. Thus

$$\begin{aligned} F &= \frac{1}{4(d+1)!} \begin{pmatrix} d & 1 & \cdots & 1 \\ 1 & d & \cdots & 1 \\ \cdot & \cdot & \cdots & \cdot \\ 1 & 1 & \cdots & d \end{pmatrix} \\ &= \frac{d-1}{4(d+1)!} I_{d+1} + \frac{1}{4(d+1)!} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \cdot & \cdot & \cdots & \cdot \\ 1 & 1 & \cdots & 1 \end{pmatrix} \end{aligned} \tag{4.12}$$

TABLE 4.1
The tensor N (multiplied by 96) for piecewise linears in three dimensions represented as a matrix of 4×3 matrices.

3	1	1	1	0	0	0	0	0	0	0	0	3	1	1	1
0	0	0	0	3	1	1	1	0	0	0	0	3	1	1	1
0	0	0	0	0	0	0	0	3	1	1	1	3	1	1	1
1	3	1	1	0	0	0	0	0	0	0	0	1	3	1	1
0	0	0	0	1	3	1	1	0	0	0	0	1	3	1	1
0	0	0	0	0	0	0	0	1	3	1	1	1	3	1	1
1	1	3	1	0	0	0	0	0	0	0	0	1	1	3	1
0	0	0	0	1	1	3	1	0	0	0	0	1	1	3	1
0	0	0	0	0	0	0	0	1	1	3	1	1	1	3	1
1	1	1	3	0	0	0	0	0	0	0	0	1	1	1	3
0	0	0	0	1	1	1	3	0	0	0	0	1	1	1	3
0	0	0	0	0	0	0	0	1	1	1	3	1	1	1	3

for $d = 2$ or 3, where I_d denotes the $d \times d$ identity matrix. Note that for a given d, the matrices in (4.12) are $d+1 \times d+1$ in dimension.

The tensor N for linears in three dimensions is presented in Table 4.1. We see now a new ingredient for computing the entries of $K^{e,u}$ from the matrix $\gamma_{m,\lambda}$. Define $\gamma_m = \sum_{\lambda=1}^{4} \gamma_{m,\lambda}$ for $m = 1, 2, 3$, and then $\tilde{\gamma}_{m,\lambda} = 2\gamma_{m,\lambda} + \gamma_m$ for $m = 1, 2, 3$ and $\lambda = 1, 2, 3, 4$. Then

$$K^{e,u} = \begin{pmatrix} \tilde{\gamma}_{11} & \tilde{\gamma}_{21} & \tilde{\gamma}_{31} & \tilde{\gamma}_{11} + \tilde{\gamma}_{21} + \tilde{\gamma}_{31} \\ \tilde{\gamma}_{12} & \tilde{\gamma}_{22} & \tilde{\gamma}_{32} & \tilde{\gamma}_{12} + \tilde{\gamma}_{22} + \tilde{\gamma}_{32} \\ \tilde{\gamma}_{13} & \tilde{\gamma}_{23} & \tilde{\gamma}_{33} & \tilde{\gamma}_{13} + \tilde{\gamma}_{23} + \tilde{\gamma}_{33} \\ \tilde{\gamma}_{14} & \tilde{\gamma}_{24} & \tilde{\gamma}_{34} & \tilde{\gamma}_{14} + \tilde{\gamma}_{24} + \tilde{\gamma}_{34} \end{pmatrix}. \tag{4.13}$$

However, note that the γ_m's are not computations that would have appeared directly in the formulation of $K^{e,u}$ but are intermediary terms that we have defined for convenience and efficiency. This requires 39 operations, instead of 384 operations using (4.8).

4.2. Algorithmic implications. The example in section 4 provides guidance for the general case. First of all, we see that the "vector" space of the evaluation problem (4.8) can be arbitrary in size. In the case of the trilinear form in Navier–Stokes equations considered there, the dimension is the spatial dimension times the dimension of the approximation (finite element) space. High-order finite element approximations [14] could lead to very high-dimensional problems. Thus we need to think about looking for a relationship among the "computational vectors" in high-dimensional spaces, e.g., up to several hundred in extreme cases. The example in section 4.1 is the lowest order case in three space dimensions, and it requires a twelve-dimensional space for the complexity analysis.

Second, it will not be sufficient just to look for simple combinations to determine optimal algorithms, as discussed in section 3.1. The example in section 4.1 shows that we need to think of this as an approximation problem. We need to look for vectors (matrices) which closely approximate a set of vectors that we need to compute. The vectors $\mathbf{V}_1 = (1, 1, 1, 1, 0, \ldots, 0)$, $\mathbf{V}_2 = (0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0)$, $\mathbf{V}_3 = (0, \ldots, 0, 1, 1, 1, 1)$ are each edit-distance one from four vectors we need to compute. The quantities γ_m represent the computations (dot-product) with $\mathbf{V}_m$. The quantities $\tilde{\gamma}_{m\lambda}$ are simple perturbations of γ which require only two operations to evaluate. A simple rescaling can reduce this to one operation.

TABLE 4.2
Summary of results for FErari on triangular Lagrange elements.

Order	Entries	Base MAPs	FErari MAPs
1	6	24	7
2	21	84	15
3	55	220	45
4	120	480	176
5	231	924	443
6	406	1624	867

Edit distance is a useful measure for approximating the computational complexity distance, since it provides an upper-bound on the number of computations it takes to get from one vector to another. Thus we need to add this type of optimization to the techniques listed in section 3.1.

4.3. The FErari system. The algorithms discussed in sections 3.1 and 4.2 have been implemented in a prototype system which we call FErari (Finite Element ReARrangements of Integrals or Finite Element Rearrangement to Automatically Reduce Instructions). We have used it to build optimized code for the local stiffness matrices for the Laplace operator using Lagrange polynomials of up to degree six. While there is no perfect metric to predict efficiency of implementations due to differences in architecture, we have taken a simple model to measure improvement due to FErari. A particular algorithm could be tuned by hand, common divisors in rational arithmetic could be sought, and more care could be taken to order the operations to maximize register usage. Before discussing the implementation of FErari more precisely, we point to Table 4.2 to see the levels of optimization detected. In the table, "Entries" refers to the number of entries in the upper triangle of the (symmetric) matrix. "Base MAPs" refers to the number of multiply-add pairs if we were not to detect dependencies at all, and "FErari MAPs" is the number of multiply-add pairs in the generated algorithm. Although we gain only about a factor of two for the higher-order cases, we are automatically generating algorithms with fewer multiply-add pairs than entries for linears, quadratics, and cubics.

FErari builds a graph of dependencies among the tensors $\mathbf{K}_{\lambda,\mu}$ in several stages. We now describe each of these stages, what reductions they produce in assembling K^e, and how much they cost to perform. We start by building the tensors $\{\mathbf{K}_{\lambda,\mu}\}$ for $1 \leq \lambda \leq \dim P_k$ and $\lambda \leq \mu \leq \dim P_k$ using FIAT (FInite element Automatic Tabulator) [15]. In discussing algorithmic complexity of our heuristics below, we shall let n be the size of this set. Throughout, we are typically using "greedy" algorithms that quit when they find a dependency. Also, the order in which these stages are performed matters, as if a dependency for a tensor found at one stage, will not mark it again later.

As we saw earlier, sometimes $\mathbf{K}_{\lambda,\mu} = 0$. In this case, $(K^e)_{\lambda,\mu} = 0$ as well for any e, so these entries cost nothing to build. Searching for uniformly zero tensors can be done in n operations. While we found that three tensors vanish for quadratics and cubics, none do for degrees four through six.

TABLE 4.3
Results of FErari on Lagrange elements of degrees 1 *through* 6.

Order	Entries	Zero	Eq	Eq t	1 Entry	Col	Ed1	Ed2	LC	Default	Cost
1	6	0	0	0	3	0	2	0	1	0	7
2	21	3	2	3	4	2	5	1	1	0	15
3	55	3	17	0	5	8	12	5	5	0	45
4	120	0	23	0	7	2	25	30	25	8	176
5	231	0	18	0	13	5	41	71	45	38	443
6	406	0	27	0	17	7	59	139	61	96	867

Next, if two tensors $\mathbf{K}_{\lambda,\mu}$ and $\mathbf{K}_{\lambda',\mu'}$ are equal, then their dot product into G^e will also be equal. So, we mark $\mathbf{K}_{\lambda',\mu'}$ as depending on $\mathbf{K}_{\lambda,\mu}$. Naively, searching through the nonzero tensors is an $O(n^2)$ process, but can be reduced to $O(n\log(n))$ operations by inserting the tensors into a binary tree using lexicographic ordering of the entries of the tensors. The fourth column of Table 4.3 titled "Eq" shows the number of such dependencies that FErari found.

As a variation on this theme, if $\mathbf{K}_{\lambda,\mu} = (\mathbf{K}_{\lambda',\mu'})^t$, then $\mathbf{K}_{\lambda,\mu} : G^e = (\mathbf{K}_{\lambda',\mu'} : G^e)$ since G^e is symmetric. Hence, once $(K^e)_{\lambda,\mu}$ is computed, $(K^e)_{\lambda',\mu'}$ is free. We may also search for these dependencies among nonzero tensors that are not already marked as equal to another tensor in $O(n\log(n))$ time by building a binary tree. In this case, we compare by lexicographically ordering the components of the symmetric part of each tensor (recall the symmetric part of K is $\frac{K+K^t}{2}$). Equality of symmetric parts is necessary but not sufficient for two matrices to be transposes of each other. So, if our insertion into the binary tree reveals an entry with the same symmetric part, we then perform an additional check to see if the two are indeed transposes. Unfortunately, we found such dependencies only for quadratics, where we found three. This is indicated in the fifth column of Table 4.3, titled "Eq t."

So far, we have focused on finding and marking tensors that can be dotted into G^e with no work (perhaps once some other dot product has been performed). Now, we turn to ways of finding tensors whose dot product with G^e can be computed cheaply. The first such way is to find unmarked tensors $\mathbf{K}_{\lambda,\mu}$ with only one nonzero entry. This may be trivially performed in $O(n)$ time. The sixth column of Table 4.3, titled "1 Entry," shows the number of such tensors for each polynomial degree.

If there is a constant α such that $\mathbf{K}_{\lambda',\mu'} = \alpha\mathbf{K}_{\lambda,\mu}$, then $\mathbf{K}_{\lambda',\mu'} : G^e = \alpha\mathbf{K}_{\lambda,\mu} : G^e$ and hence may be computed in a single multiply. Moreover, colinearity among the remaining tensors may be found in $O(n\log(n))$ time by a binary tree and lexicographic ordering of an appropriate normalization. To this end, we divide each (nonzero) tensor by its Frobenius norm and ensure that its first nonzero entry is positive. If insertion into the binary tree gives equality under this comparison, then we mark the tensors as colinear. The numbers of such dependencies found for each degree is seen in the "Col" column of Table 4.3.

Next, we seek tensors that are close together in edit distance, differing only in one or two entries. Then, the difference between the dot products can be computed cheaply. For each unmarked tensor, we look for a marked tensor that is edit-distance one away. We iterate this process until no more dependencies are found, then search for dependencies among the remainders. We repeat this process for things that are edit-distance two apart. In general, this process seems to be $O(n^2)$. Table 4.3 has two columns, titled "Ed1" and "Ed2," showing the number of tensors we were able to mark in such a way.

```
from Numeric import zeros
G=zeros(4,"d")
def K(K,jinv):
        detinv = 1.0/(jinv[0,0]*jinv[1,1] - jinv[0,1]*jinv[1,0])
        G[0] = ( jinv[0,0]**2 + jinv[1,0]**2 ) * detinv
        G[1] = ( jinv[0,0]*jinv[0,1]+jinv[1,0]*jinv[1,1] ) * detinv
        G[2] = G[1]
        G[3] = ( jinv[0,1]**2 + jinv[1,1]**2 ) * detinv
        K[1,1] = 0.5 * G[0]
        K[1,0] = -0.5 * G[1]- K[1,1]
        K[2,1] = 0.5 * G[2]
        K[2,0] = -0.5 * G[3]- K[2,1]
        K[0,0] = -1.0 * K[1,0] + -1.0 * K[2,0]
        K[2,2] = 0.5 * G[3]
        K[0,1] = K[1,0]
        K[0,2] = K[2,0]
        K[1,2] = K[2,1]
        return K
```

FIG. 4.1. *Generated code for computing the stiffness matrix for linear basis functions.*

If a tensor is a linear combination of two other tensors, then its dot product can be computed in two multiply-add pairs. This is an expensive search to perform ($O(n^3)$), since we have to search through pairs of tensors for each unmarked tensor. However, we do seem to find quite a few linear combinations, as seen in the "LC" column of Table 4.3. Any remaining tensors are marked as "Default" in which case they are computed by the standard four multiply-add pairs.

After building the graph of dependencies, we topologically sort the vertices. This gives an ordering for which, if vertex u depends on vertex v, then v occurs before u, an essential feature for generating code. We currently generate Python code for ease in debugging and integrating with the rest of our computational system, but we could just as easily generate C or Fortran. In fact, in the future we hope to generate not only particular code, but abstract syntax as in PETSc 3.0.

One interesting feature of the generated code is that it is completely unrolled—no loops are done. This leads to relatively large functions, but sets up the code to a point where the compiler really only needs to handle register allocation. In Figure 4.1, we present the generated code for computing linears.

4.4. Code verification. In general,the question of verifying a code's correctness is difficult. In this case, we have taken an existing Poisson solver and replaced the function to evaluate the local stiffness matrix with FErari-generated code. The correct convergence rates are still observed, and the stiffness matrices and computed solutions agree to machine precision. However, in the future, we hope to generate optimized code for new, complicated forms where we do not have an existing verified implementation, and the general question of verifying such codes is beyond the scope of this present work.

5. Computing a matrix via quadrature. The computations in equations (2.6)–(2.7) can be computed via quadrature as

$$
\begin{aligned}
A_{\iota(e,\lambda),\iota(e,\mu)} +&= \sum_{\xi\in\Xi} \omega_\xi \nabla\phi_\lambda(\xi)\cdot(\mathbf{G}^e\nabla\phi_\mu(\xi)) \\
&= \sum_{\xi\in\Xi} \omega_\xi \sum_{m,n=1}^{d} \phi_{\lambda,m}(\xi) G^e_{m,n}\phi_{\mu,n}(\xi) \\
&= \sum_{m,n=1}^{d} G^e_{m,n} \sum_{\xi\in\Xi} \omega_\xi \phi_{\lambda,m}(\xi)\phi_{\mu,n}(\xi) \\
&= \sum_{m,n=1}^{d} G^e_{m,n} K_{\lambda,\mu,m,n},
\end{aligned}
\tag{5.1}
$$

where the coefficients $K_{\lambda,\mu,m,n}$ are analogous to those defined in (2.3), but here they are defined by quadrature:

$$
K_{\lambda,\mu,m,n} = \sum_{\xi\in\Xi} \omega_\xi \phi_{\lambda,m}(\xi)\phi_{\mu,n}(\xi). \tag{5.2}
$$

(The coefficients are exactly those of (2.3) if the quadrature is exact.)

The right strategy for computing a matrix via quadrature would thus appear to be to compute the coefficients $K_{\lambda,\mu,m,n}$ first using (5.2), and then proceeding as before. However, there is a different strategy associated with quadrature when we want only to compute the [illegible] of the linear operator associated with the matrix and not the matrix itself (cf. [16]).

6. Other approaches. We have presented one approach to optimize the computation of finite element matrices. Other approaches have been suggested for optimizing code for scientific computation. The problem that we are solving can be represented as a sparse-matrix-times-full-matrix multiply, $\mathcal{K}\mathcal{G}$, where the dimensions of $\mathcal{K}$ are, for example, $|\mathcal{L}|^2 \times d^2$ for Laplace's equation in d dimensions using a (local) finite element space $\mathcal{L}$. For the nonlinear term in Navier–Stokes equations, the dimensions of $\mathcal{K}$ become $|\mathcal{L}|^2 \times d|\mathcal{L}|$. The first dimension of $\mathcal{G}$ of course matches the second of $\mathcal{K}$, but the second dimension of $\mathcal{G}$ is equal to the number of elements in the mesh. Thus it is worthwhile to do significant precomputation on $\mathcal{K}$.

At a low level, ATLAS [26] works with operations such as loop unrolling, cache blocking, etc. This type of optimization would not find the reductions in computation that FErari does, since the latter identifies more complex algebraic structures. In this sense, our work is more closely aligned with that of FLAME [7] which works with operations such as rank updates, triangular solves, etc. The BeBOP group has also worked on related issues in sparse matrix computation [22, 24, 23, 25, 12]. Our work could be described as utilizing a sparsity structure in a matrix representation, and it can be written as multiplying a sparse matrix times a very large set of (relatively small) vectors (the columns of $\mathcal{G}$). This is the reverse of the case considered in the SPARSITY system [13], where multiplying a sparse matrix times a small number of large vectors is considered.

Clearly the ideas we present here could be coupled with these approaches to generate improved code. The novelty of our work resides in the precomputation that is applied to evaluate the action of $\mathcal{K}$. In this way, it resembles the reorganization of computation done for evaluation of polynomials or matrix multiply [17].

7. Timing results. We performed several experiments for the Poisson problem with piecewise linear and piecewise quadratic elements. We set up a series of meshes using the mesh library in DOLFIN [10]. These meshes contained between 2048 and

524288 triangles. We timed computation of local stiffness matrices by several techniques; inserting them into a sparse PETSc matrix [3], multiplying the matrix onto a vector, and solving the linear system. All times were observed to be linear in the number of triangles, so we report all data as time per million triangles. All computations were performed on a Linux workstation with a 3 GHz Pentium 4 processor with 2 GB of RAM. All our code was compiled using `gcc` with "−O3" optimization.

Our goal is to assess the efficacy and relevance of our proposed optimizations. Solver technology has been steadily improving over the last several decades, with multigrid and other optimal strategies being found for wider classes of problems. When such efficient solvers are available, the cost of assembling the matrices (both computing local stiffness matrices and inserting them into the global matrix) is much more important. This is especially true in geometric multigrid methods in which a stiffness matrix can be built on each of a sequence of nested meshes, but only a few iterations are required for convergence. To factor out the choice of solver, we concentrate first on the relative costs of building local stiffness matrices, inserting them into the global matrix, and applying the matrix to a vector.

We used four strategies for computing the local stiffness matrices: numerical quadrature, tensor contractions (four flops per entry), tensor contractions with zeros omitted (this code was generated by the FEniCS form compiler [19]), and the FErari-optimized code translated into C. For both linear and quadratic elements, the cost of building all of the local stiffness matrices and inserting them into the global sparse matrix was comparable (after storage had been preallocated). In both situations, computing the matrix-vector product is an order of magnitude faster than computing the local matrices and inserting them into the global matrix. These costs, all measured as seconds to process one million triangles, are given in Table 7.1 and plotted in Figure 7.1 using log scale for time.

We may draw several conclusions from this. First, precomputing the reference tensors leads to a large performance gain over numerical quadrature. Beyond this, additional benefits are included by omitting the zeros, and more still by doing the FErari optimizations. We seem to have optimized the local computation to the point where it is constrained by read-write to cache rather than by floating point optimization. Second, these optimizations reveal a new bottleneck: matrix insertion. This motivates either studying whether we may improve the performance of insertion into the global sparse matrix or else implementing matrix-free methods.

While the FErari optimizations are highly successful for building the matrices, there is still quite a bit of solver overhead to contend with. We used GMRES (boundary conditions were imposed in a way that broke symmetry) preconditioned by the BoomerAMG method of HYPRE [9]. For both linear and quadratic elements, the Krylov solver converged with only three iterations. However, the cost of building and applying the AMG preconditioner is very large. Using numerical quadrature, building local stiffness matrices accounted for between five and nine percent of the total run time (building the local-to-global mapping, computing geometry tensors, computing local stiffness matrices, sorting and inserting local matrices into the global matrix, creating and applying the AMG preconditioner, and computing the Krylov iterations). Keeping everything else constant and switching to the FErari optimized code, building local stiffness matrices took less than one percent of the total time. Geometric multigrid algorithms tend to be much more efficient, and we conjecture that the cost of building local matrices would be even more important in this case.

TABLE 7.1
Seconds to process one million triangles: local stiffness matrices, global matrix insertion, and matrix-vector product.

	Quadrature	Tensor	FFC	FErari	Assemble	Matvec
Linear	0.3802	0.0725	0.0535	0.0513	0.4762	0.0177
Quadratic	2.0000	0.3367	0.1517	0.1506	1.9342	0.1035

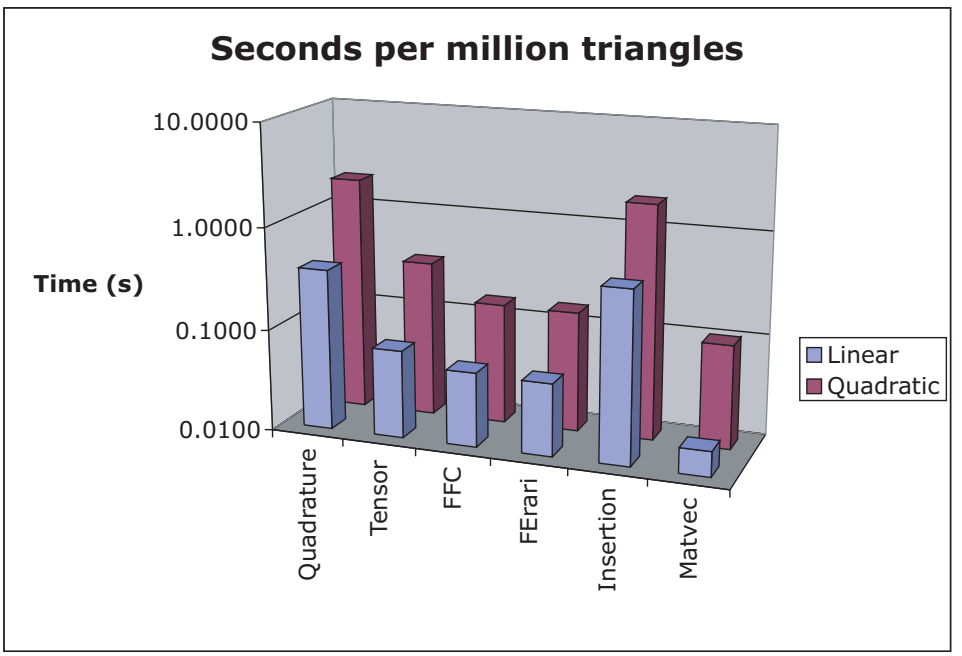


FIG. 7.1. *Seconds to process one million triangles: local stiffness matrices, global matrix insertion, and matrix-vector product.*

8. Conclusions. The determination of local element matrices involves a novel problem in computational complexity. There is a mapping from (small) geometry matrices to "difference stencils" that must be computed. We have demonstrated the potential speed-up available with simple low-order methods. We have suggested by examples that it may be possible to automate this to some degree by solving abstract graph optimization problems.

Acknowledgments. We thank the FEniCS team, and Todd Dupont, Johan Hoffman, and Claes Johnson in particular, for substantial suggestions regarding this paper.

REFERENCES

[1] A. A. ALBRECHT, P. GOTTSCHLING, AND U. NAUMANN, *Markowitz-type heuristics for computing Jacobian matrices efficiently*, in International Conference on Computational Science, Part II, Lecture Notes in Comput. Sci. 265 Springer, Berlin, 2003, pp. 575–584.

[2] D. N. ARNOLD, D. BOFFI, AND R. S. FALK, *Quadrilateral H(div) finite elements*, SIAM J. Numer. Anal., 42 (2005), pp. 2429–2491.

[3] S. Balay, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Users Manual*, Tech. Report ANL-95/11, Revision 2.1.5, Argonne National Laboratory, Argonne, IL, 2004.

[4] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, 2nd ed., Springer-Verlag, New York, 2002.

[5] K. A. Cliffe and S. J. Tavener, *Implementation of extended systems using symbolic algebra*, in Continuation Methods in Fluid Dynamics, D. Henry and A. Bergeon, eds., Notes Numer. Fluid Mech. 74, Vieweg, Braunschweig, 2000, pp. 81–92.

[6] M. Crouzeix and P.-A. Raviart, *Conforming and nonconforming finite element methods for solving the stationary Stokes equations.* I, Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge, 7 (1973), pp. 33–75.

[7] J. A. Gunnels, F. G. Gustavson, G. M. Henry, and R. A. van de Geijn, *FLAME: Formal linear algebra methods environment*, Trans. Math. Software, 27 (2001), pp. 422–455.

[8] P. Hansbo and A. Szepessy, *A velocity-pressure streamline diffusion finite element method for the incompressible Navier-Stokes equations*, Comput. Methods Appl. Mech. Engrg., 84 (1990), pp. 175–192.

[9] V. Henson and U. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner*, Appl. Numer. Math., 41 (2002), pp. 155–177.

[10] J. Hoffman and U. Yang, *DOLFIN: Dynamic Object Oriented Library for Finite Element Computation*, Tech. Report 2002-6, Chalmers Finite Element Preprint Series, Chalmers University, Göteborg, Sweden, 2002, http://www.fenics.org/dolfin.

[11] T. J. R. Hughes, L. P. Franca, and M. Balestra, *A new finite element formulation for computational fluid dynamics.* V. *Circumventing the Babuška-Brezzi condition: A stable Petrov-Galerkin formulation of the Stokes problem accommodating equal-order interpolations*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 85–99.

[12] E.-J. Im and K. A. Yelick, *Optimizing sparse matrix computations for register reuse in SPARSITY*, in Proceedings of the International Conference on Computational Science, (San Francisco, CA), Lecture Notes in Comput. Sci. 2073, Springer, New York, 2001, pp. 127–136.

[13] E.-J. Im, K. A. Yelick, and R. Vuduc, *SPARSITY: Framework for optimizing sparse matrix-vector multiply*, Int. J. High Performance Comput. Appl., 18 (2004), pp. 135–158.

[14] H. Juarez, L. R. Scott, R. Metcalfe, and B. Bagheri, *Direct simulation of freely rotating cylinders in viscous flows by high-order finite element methods*, Comput. & Fluids, 29 (2000), pp. 547–582.

[15] R. C. Kirby, *Algorithm* 839: *FIAT, a new paradigm for computing finite element basis functions*, ACM Trans. Math. Software, 30 (2004), pp. 502–516.

[16] R. C. Kirby, M. Knepley, and L. Scott, *Evaluation of the Action of Finite Element Operators*, Tech. report TR-2004-07, Department of Computer Science, University of Chicago, Chicago, IL, 2004.

[17] D. E. Knuth, *The Art of Computer Programming, Volume* 2: *Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading, MA, 1997.

[18] Argonne National Laboratory, *PETSc-CS Alphan*, http://www-unix.mcs.anl.gov/petsc/petsc-3/.

[19] A. Logg, *FFC: The FEniCS Form Compiler*, http://www.fenics.org/ffc.

[20] K. Long, *Sundance rapid prototyping tool for parallel PDE optimization*, in Large-Scale PDE-Constrained Optimization, Lecture Notes in Comput. Sci. Engrg. 30, Springer, Berlin, 2003, pp. 331–341.

[21] K. Long, *Sundance*, 2003, http://csmr.ca.sandia.gov/~krlong/sundance.html.

[22] R. Vuduc, J. Demmel, and J. Bilmes, *Statistical models for empirical search-based performance tuning*, Int. J. High Performance Comput. Appl., 18 (2004), pp. 65–94.

[23] R. Vuduc, J. W. Demmel, K. A. Yelick, S. Kamil, R. Nishtala, and B. Lee, *Performance optimizations and bounds for sparse matrix-vector multiply*, in Proceedings of Supercomputing, Baltimore, MD, IEEE, Piscataway, NJ, 2002, pp. 1–35.

[24] R. Vuduc, A. Gyulassy, J. W. Demmel, and K. A. Yelick, *Memory hierarchy optimizations and bounds for sparse* A^TAx, in Proceedings of the ICCS Workshop on Parallel Linear Algebra, P. M. A. Sloot, D. Abramson, A. V. Bogdanov, J. J. Dongarra, A. Y. Zomaya, and Y. E. Gorbachev, eds., (Melbourne, Australia), Lecture Notes in Comput. Sci. 2660, Springer, New York, 2003, pp. 705–714.

[25] R. Vuduc, S. Kamil, J. Hsu, R. Nishtala, J. W. Demmel, and K. A. Yelick, *Automatic performance tuning and analysis of sparse triangular solve*, presented at the Workshop on Performance Optimization via High-Level Languages and Libraries, in conjunction with the 16th International Conference on Supercomputing (ICS), ACM, New York, 2002. Available online at http://www.cs.berkeley.edu/~richie/research/pubs.html

[26] R. C. Whaley, A. Petitet, and J. J. Dongarra, *Automated empirical optimization of software and the ATLAS project*, Parallel Computing, 27 (2001), pp. 3–35. Also available online as University of Tennessee LAPACK Working Note 147, UT-CS-00-448 at www.netlib.org/lapack/lawns/downloads.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 759–773
© 2005 Society for Industrial and Applied Mathematics

EFFICIENT NUMERICAL CONTINUATION AND STABILITY ANALYSIS OF SPATIOTEMPORAL QUADRATIC OPTICAL SOLITONS*

D. J. B. LLOYD† AND A. R. CHAMPNEYS†

Abstract. A numerical method is set out which efficiently computes stationary (z-independent) two- and three-dimensional spatiotemporal solitons in second-harmonic-generating media. The method relies on a Chebyshev decomposition with an infinite mapping, bunching the collocation points near the soliton core. Known results for the type-I interaction are extended and a stability boundary is found by two-parameter continuation as defined by the Vakhitov–Kolokolov criteria. The validity of this criterion is demonstrated in (2+1) dimensions by simulation and direct calculation of the linear spectrum. The method has wider applicability for general soliton-bearing equations in (2+1) and (3+1) dimensions.

Key words. continuation, optical soliton, second harmonic generation, type-I interaction

AMS subject classifications. 37K50, 65N35, 65P40, 74S25

DOI. 10.1137/040604455

1. Introduction.

Optical spatiotemporal solitons in second-harmonic-generating media have received much attention due to the possibility of creating fully localized, self-supporting objects which freely propagate in a nonlinear medium [15]. Unlike Kerr media, second-harmonic-generating media have no wave collapse in (2+1) and (3+1) dimensions in quadratic (χ^2) media and have been observed experimentally; see Liu et al. [20, 19]. The equations governing the evolution of these quadratic solitons are nonintegrable so that analytical closed form stable solutions do not exist. For a review of solitons due to quadratic nonlinearities see [6].

Malomed et al. [23] studied general stationary, nonradially symmetric solitons found in the type-I interaction second harmonic generation (SHG) case. Their approach was to first construct a Gaussian variational approximation of solitons in (2+1) and (3+1) dimensions and analytically find an approximate stability boundary. They then carried out numerical simulations of the full z-dependent equations to ascertain the accuracy of the approximate stability boundary. Since this Gaussian approximation amounts to a one-mode Galerkin truncation, the stability boundary was found to be "rather fuzzy". Part of the problem is that the accuracy of this approximation deteriorates with increasing dimensionality due to the approximation "cutting off" the tails of the soliton.

A more thorough analysis of stationary solitons in type-I interaction was carried out by Mihalache et al. [24] who considered the numerical boundary value problem (BVP) for the stationary light bullets. Existence and stability of "elliptic" solitons was explored in (2+1) and (3+1) dimensions with and without "walk-off" [24, 25]. Later, Beckwitt et al. [2] and Towers, Malomed, and Wise [33] explored the region where the second harmonic dispersion is normal via solving the initial value problem

*Received by the editors February 25, 2004; accepted for publication (in revised form) April 26, 2005; published electronically November 15, 2005. This work was supported by an EPSRC Research Studentship.

http://www.siam.org/journals/sisc/27-3/60445.html

†Bristol Centre for Applied Nonlinear Mathematics, Department of Engineering Mathematics, University of Bristol, Queen's Building, University Walk, Bristol BS8 1TR, United Kingdom (david.lloyd@bris.ac.uk, a.r.champneys@bris.ac.uk).

starting from an initial Gaussian pulse. They also showed analytically that solitons with small oscillating tails at infinity, known as "quasi solitons", should exist in this region, though they were unable to find any numerically. One of the key problems with both numerical approaches is the high computational effort required to solve the equations with either finite difference or Fourier spectral methods. In both cases supercomputers were required to reduce the computational time.

The object of this work is to provide an [illegible] and [illegible] numerical method for calculating stationary spatiotemporal solitons, not necessarily just in the SHG equations, but also in other soliton bearing-equations in (2+1) and (3+1) dimensions. There are several benefits with using our method. First, we can easily move through parameter space and between (2+1) and (3+1) dimensions, assuming radial symmetry in (3+1) dimensions. Second, we can accurately compute light bullets with a mesh on the order of tens of modes rather than hundreds, greatly reducing the computational effort. Third, this approach also allows us to embed the method in a numerical continuation environment, e.g., AUTO [12], to compute branches of the solutions. Fourth, and perhaps most crucial, using criteria such as that due to Vakhitov and Kolokolov, which define soliton instabilities [30], one can then append the stability criterion to the numerical continuation problem. The Vakhitov–Kolokolov (VK) condition states that solitons will become unstable if the gradient of the energy of the solution (Q), as the effective propagation constant (κ) varies, becomes negative (i.e., $\partial_\kappa Q < 0$). A stability boundary of the solutions corresponds to when the gradient is zero (i.e., $\partial_\kappa Q = 0$). In a continuation environment this corresponds to a fold in the energy of the solution as the effective propagation constant is varied. Thus we can trace out stability boundaries automatically and accurately with the kind of computational effort that takes minutes on a Pentium 4, 2 GHz microprocessor.

Finally, criteria such as the VK conditions are necessary only for stability. The sufficiency of this condition can also be checked by computation of the spectrum linearized about the localized solution; see Mihalache et al. [24] for confirmation of the validity of the VK criteria in (2+1) and (3+1) dimensions. We demonstrate that an adaptation of our Chebyshev method can be used to calculate the linear spectrum. The description in terms of only tens of modes helps make such computations feasible.

The paper is organized in the following way. In section 2 we set out the problem. The numerical scheme for the computation of stationary solitons and their stability is stated in section 3. Our results for the type-I SHG interaction equations are presented in section 4. Section 5 draws conclusions and suggests avenues for future work.

2. Problem statement. We are interested in stationary (z-independent) soliton solutions to the scaled equations describing copropagation of the fundamental harmonic (FH) and second harmonic (SH) field in a quadratic optical medium,

$$\begin{aligned} iu_z + \nabla_\rho^2 u + u_{\tau\tau} - u + v\overline{u} &= 0, \\ 2iv_z + \nabla_\rho^2 v + \delta v_{\tau\tau} - \gamma v + \frac{1}{2}u^2 &= 0 \end{aligned} \tag{2.1}$$

(see [14] for their derivation and physical meaning).

Here $u = \sqrt{2}\Phi_1 e^{-iz/z_0} z_0 \omega_1^2 \chi^{(2)}/(k_1 c^2)$ and $v = \Phi_2 e^{-2iz/z_0} z_0 \omega_1^2 \chi^{(2)}/(k_1 c^2)$, $\Phi_{1,2}$ are complex electric-field envelopes at the frequencies ω_1 and $2\omega_1$, and $k_1 \equiv k(\omega_1)$ is the FH carrier wave number. The normalized propagation distance and reduced time are $\xi = z/z_0$ and $\tau = (t - z/\nu_g)/t_0$, respectively, with ν_g and t_0 being the group velocity and time scale at the FH. $z_0 = 2t_0^2/k_1''$, where k_1'' is the group velocity dispersion (GVD) coefficient at the FH, and the nondimensional transverse radial

coordinate is $\rho = \sqrt{x^2+y^2}/r_0$, where $r_0^2 = (z_0/2k_1)$. The diffraction operator, $\nabla_\rho^2 \equiv \partial^2/\partial\rho^2 + [(D-1)/\rho]\partial/\partial\rho$, where $D+1 = 2$ or 3 is the spatial dimension, acts on the transverse spatial coordinate(s) (we shall assume cylindrical symmetry in (3+1) dimensions). The phase mismatch between the harmonics $\gamma = 4+2z_0[2k(\omega_1)-k(2\omega_1)]$ and the ratio of the SH and FH GVD coefficients is $\delta = k_2''/k_1''$.

Under the substitution, $u(z,\rho,\tau) = u(\rho,\tau)e^{i\kappa z}$, $v(z,\rho,\tau) = v(\rho,\tau)e^{2i\kappa z}$, where κ is the effective propagation constant, we seek stationary, z-independent soliton solutions defined on the infinite domain, $-\infty < \rho < \infty$ and $-\infty < \tau < \infty$, given by the following equations for the , functions u and v:

$$
\begin{aligned}
u_{\rho\rho} + (d-1)\rho^{-1}u_\rho + u_{\tau\tau} - (1+\kappa)u + vu = 0,\\
v_{\rho\rho} + (d-1)\rho^{-1}v_\rho + \delta v_{\tau\tau} - (\gamma+4\kappa)v + \frac{1}{2}u^2 = 0.
\end{aligned}
\tag{2.2}
$$

Equations (2.2) are known to support localized solutions, known as solitary waves, that are able to survive strong perturbations and remain intact after collisions with other solitary waves [5]. Exact solitary wave solutions to (2.1) are not available, except in special degenerate limits, since the equations are nonintegrable and so numerical methods provide the only way to accurately calculate the solutions. Since the solitary waves seen in (2.1) have similar soliton shapes to solutions of integrable systems, we shall follow the physics convention and use the word "soliton" interchangeably with solitary wave.

We are seeking physical localized solutions of (2.1) which correspond to the solutions that decay (exponentially) to zero at infinity. We can see from the linearization of (2.2) at infinity that exponentially decaying solutions in both harmonics forming the soliton exist if the effective propagation constant $\kappa > \max(-1, -\gamma/4)$ and the GVD ratio coefficient $\delta \geq 0$. This last constraint corresponds to anomalous or zero dispersion in the SH. So we compute on a "large" domain and apply zero Dirichlet boundary conditions at the far-field edge of the domain. For the case when the GVD is normal at the SH ($\delta < 0$), both Beckwitt et al. [2] and Towers, Malomed, and Wise [33] showed analytically there should be a nonvanishing, oscillating tails in the SH v as $\tau \to \infty$.

3. Numerical approach. Our numerical method is based on setting up the BVP of (2.2) and using numerical continuation software, e.g., AUTO [12], to explore the parameter space of solitons. In particular, AUTO software implements the robust technique of pseudo-arclength continuation (see Keller [16]) for systems of nonlinear ODEs by solving the BVP using orthogonal collocation; see [1]. In order to use such software we need to discretize (2.2) to form an ODE system and apply appropriate boundary conditions. The numerical continuation software will then solve the BVP ODE system as an equation parameter is varied. To use such software one must provide an equations file defining the right-hand side of a first-order ODE system as well as specifying in which equation parameters one wants to continue. The AUTO manual and a tutorial on AUTO can be found in [11].

Two problems need to be tackled for this approach to work. The first is to efficiently discretize the problem in either the ρ or τ coordinate. A straight Fourier or finite-difference discretization scheme, though possible, has been found to require a large number of modes, and hence a huge amount of computing power is required to solve the resulting ODE system (both [33] and [24] cite the use of supercomputers). The reason for the inefficiency is due to interpolation points being equispaced, so a , , number of these points is required to interpolate the solution near the edge of

the computational domain where the soliton is zero. A more efficient approach is to reduce the number of interpolation points near the edge of the domain and move them to the center of the stationary soliton, where they can increase the accuracy of the method. To do this we use an adapted mesh to the localized soliton problem. Effectively, we map the spatial coordinate to a new scaled space and equidistribute. A similar approach to homoclinic orbits in 1 dimension was proposed by Liu, Moore, and Russell [17]. Another property of these solitons is that they are smooth and infinitely differentiable. For these types of problems, spectral methods provide [illegible] (unlike finite difference which provides only algebraic accuracy) and so only tens of modes should be required to accurately interpolate the soliton. The second problem is to find good solution data at certain fixed D, δ, γ parameters for the numerical continuation code to get started.

We shall tackle both of these problems in the next two subsections. In the third subsection we describe how we can adapt our numerical method to compute the stability of the solitons using the VK criteria and computation of the linear spectrum.

3.1. Numerical continuation. Using an adapted mesh to solve soliton-bearing equations is not a new idea. Boyd [3] talks about rational Chebyshev functions and other possible spectral functions for soliton equations including Hermite, Whittaker cardinal, and Fourier domain truncation functions. The biggest advantage with using Fourier and Chebyshev bases is that their functions and their corresponding collocation points have simple representations as well as the existence of a fast transform between the physical and spectral domains. Boyd [4] shows how the Chebyshev polynomials are just Fourier cosine functions in disguise and how a mapping can be used to adapt the mesh to the localized problem while still maintaining spectral accuracy.

There are three approaches one can take to implement the rational Chebyshev method: the discrete Fourier transform (DFT), the fast Fourier transform (FFT), and collocation differentiation matrices [34]. We have implemented all three methods and have found that they all produce the same results at roughly the same speed for truncations on the order of tens of modes. However, we have found that the collocation differentiation matrices approach is the most simple to program, and hence is the method of choice in practice. Henceforth, we describe only that method applied to (2.2). Note that the approach is quite general and is easily extended to other systems.

For the problem at hand, we choose to discretize in the τ coordinate since we wish to explore the $\delta = 0$ region and discretizing in ρ would make the problem singular at $\delta = 0$. To discretize in τ we expand u and v, such that

$$\begin{aligned} u_i &= u(\rho, \tau_i = f(x_i)), \\ v_i &= v(\rho, \tau_i = f(x_i)), \quad i = 0, \ldots, N, \end{aligned} \tag{3.1}$$

where $x \in [-1, 1]$ is the computational Chebyshev coordinate, N is the number of interpolation points, and f is a function that maps $[-1, 1]$ to the infinite domain $[-\infty, \infty]$. There two standard choices for such a mapping [4, 7] are the algebraic map

$$f(x) = L\frac{x}{\sqrt{1 - x^2}}, \tag{3.2}$$

and the exponential map

$$f(x) = L\tanh^{-1}x. \tag{3.3}$$

The constant L sets the length scale of the map and it can vary anywhere between 0 and $+\infty$, with the collocation points being more "bunched" near the origin for small

L. In general, if the solitons decay exponentially, then the exponential map will be the most efficient but it is also the most sensitive to changes in the mapping constant L; see Boyd [4]. Some experimentation of various length scales and number of collocation points is required to find a reasonable mapping. We shall stick with the exponential map in this paper since we shall mostly be exploring a parameter region where soliton tails decay exponentially.

Differentiation in τ is carried out using the chain rule for Chebyshev differentiation matrices [34] so that

$$(u_{\tau\tau})_i = \left[\left(\frac{\partial^2 f^{-1}(\tau_i)}{\partial \tau^2} D_N^{(1)} + \left[\frac{\partial f^{-1}(\tau_i)}{\partial \tau} \right]^2 D_N^{(2)} \right) u(\rho, x_i), \quad i = 0, \ldots, N \right],$$

$$\equiv A_N u(\rho, x_i), \quad i = 0, \ldots, N, \tag{3.4}$$

where $f^{-1}(\tau)$ is the inverse of the infinite mapping (3.3), $A_N = \partial^2 f^{-1}(\tau_i)/\partial \tau^2 D_N^{(1)} + [\partial f^{-1}(\tau_i)/\partial \tau]^2 D_N^{(2)}$ is an $(N+1) \times (N+1)$ matrix carrying out the differentiation in τ composed by the Chebyshev differentiation matrices $D_N^{(1)}$, where

$$(D_N)_{00} = \frac{2N^2+1}{6}, \qquad (D_N)_{NN} = -\frac{2N^2+1}{6},$$

$$(D_N)_{jj} = \frac{-x_j}{2(1-x_j^2)}, \qquad j = 1, \ldots, N-1,$$

$$(D_N)_{ij} = \frac{c_i}{c_j} \frac{(-1)^{i+j}}{(x_i - x_j)}, \qquad i \neq j, \ i, j = 0, \ldots, N,$$

and

$$c_i = \begin{cases} 2 & \text{when } i = 0 \text{ or } N, \\ 1 & \text{otherwise,} \end{cases}$$

and $D_N^{(2)} = D_N^{(1)} \circ D_N^{(1)}$. Finally, retaining a system of first-order ODEs in the ρ coordinate we have the BVP formulation

$$\begin{aligned} (u_\rho)_i &= w_i, \\ (w_\rho)_i &= -\frac{(d-1)}{\rho} w_i - (u_{\tau\tau})_i + (1+\kappa) u_i - (u_i v_i), \\ (v_\rho)_i &= y_i, \\ (y_\rho)_i &= -\frac{(d-1)}{\rho} y_i - \delta (v_{\tau\tau})_i + (\gamma + 4\kappa) v_i - \frac{1}{2}(u_i)^2, \\ \dot{\rho} &= 1. \end{aligned} \tag{3.5}$$

The numerical experiments in [23, 24, 25, 33] suggest that stable soliton solutions are even in both ρ and τ, respecting the reflectional symmetry of the equations $\rho \to -\rho$ and $\tau \to -\tau$. So we compute on the first quadrant and we rewrite the differentiation matrix A_N taking advantage of this symmetry.

We apply Neumann boundary conditions on u and v (i.e., $w_i = y_i = 0$) at both ends of the domain since we wish to preserve symmetry at the origin and any linear

boundary conditions at the far-field end of the domain are known to exponentially converge to the solution if we take a large enough truncation of the domain, say P, such that u and v decay to zero numerically; see [13]. So for the BVP equations (3.5), we imposed the boundary conditions

$$\begin{aligned} (w)_i &= 0, \\ (y)_i &= 0 \end{aligned} \tag{3.6}$$

for $i = 0, \ldots, N$ at $\rho = 0$ and $\rho = P$.

3.2. Initial solution. The problem (3.5) and boundary conditions (3.6) are set up well for continuation of one-parameter families of solutions. First we need a starting solution at one particular parameter value.

To acquire good starting data we assume that $\delta = 1$, $\kappa = 0$ at which parameter values (2.2) are radially symmetric. These symmetric solutions can be obtained from the reduced equations

$$\begin{aligned} u_{rr} + \frac{d}{r}u_r - u + uv &= 0, \\ v_{rr} + \frac{d}{r}v_r - \gamma v + \frac{1}{2}u^2 &= 0, \end{aligned} \tag{3.7}$$

where $r = \sqrt{\rho^2 + \tau^2}$ is the radial coordinate. Taking $d = 0$ and $\gamma = 1$, (3.7) have an exact soliton solution [5], of the form

$$\begin{aligned} u(r) &= \frac{3\sqrt{2}}{2}\mathrm{sech}^2\left(\frac{r}{2}\right), \\ v(r) &= \frac{3}{2}\mathrm{sech}^2\left(\frac{r}{2}\right). \end{aligned} \tag{3.8}$$

We can set up a simple continuation problem of solutions to (3.7), up to $d = 1$; see Champneys, Kuznetsov, and Sandstede [8]. Once we have our new solution for $d = 1$ we need to appropriately separate the data for the full continuation problem. Data for u and v are straightforward and the data for the first derivatives w and y are found by applying the chain rule to yield

$$w = u_\rho = \frac{\rho}{r}\frac{du(r)}{dr},$$

$$y = v_\rho = \frac{\rho}{r}\frac{dv(r)}{dr},$$

where $du(r)/dr$ is the numerical continuation, computed derivative. Finally, the data need to be interpolated to the Chebyshev grid in τ.

3.3. Computation of stability. Since we can now solve the BVP (3.5) to find the solitons, it would be useful to know whether the solitons found are stable or not. A standard approach to check the stability of stationary solitons is to use the VK criterion; see Skryabin [30]. This VK criterion is a nonrigorous, necessary condition stating that solitons in Hamiltonian systems may be stable if $\partial_\kappa Q < 0$, where Q is the energy of the system and κ is the effective propagation constant. The energy for problem (2.2) is defined in (2+1) dimensions as the integral $Q = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(|u|^2 + 4|v|^2)d\tau d\rho$ and in (3+1) dimensions, $Q = 2\pi\int_{-\infty}^{\infty}\int_{0}^{\infty}(|u|^2 + 4|v|^2)\rho d\tau d\rho$.

Integration with respect to τ is carried out by changing to Chebyshev coordinates to obtain the integral (in (2+1) dimensions)

$$Q = \int_{-\infty}^{\infty}\int_{-1}^{1} (|u(\rho,\tau)|^2 + 4|v(\rho,\tau)|^2) \left[\frac{\partial f^{-1}(\tau)}{\partial \tau}\right]^{-1} dx d\rho. \tag{3.9}$$

Noting that the internal integral is of the form $\int_{-1}^{1} p(x)dx$ and can be computed efficiently via a Clenshaw–Curtis quadrature [34], we can append the BVP (3.5) and boundary conditions (3.6), with an integral condition

$$Q - q = 0, \tag{3.10}$$

where Q is the energy of the system and q is a "free" continuation parameter. This free parameter is introduced into the BVP and takes the value of Q as the system is solved for a set of equation parameters. The parameter q does not fix to a value of Q and is allowed to freely vary. Using parameter continuation in both κ and q, we can find where $\partial Q_\kappa = 0$. This particular point presents a fold in the solution branch, where we think of q as the bifurcation parameter and κ as part of the solution. Standard path-following techniques exist to follow fold points as another parameter, either γ or δ, is varied (see [29]) and hence trace out the VK stability criterion in two physical parameters.

Since the VK criterion is found as the compatibility condition arising in the leading (zero) order asymptotic expansion of the linear eigenvalue problem governing the stability of the soliton (see [30]), the stability boundary defines only a necessary region for stability. Hence, we need to check its validity. In order to do this, we have implemented the above Chebyshev discretization scheme in both the ρ and τ coordinates, with a fourth-order Runge–Kutta scheme (see [4, 7, 34] for temporal discretizations) to simulate the full z-dependent equations (2.1) as well as calculating the linear spectrum directly. The data from the numerical continuation software are interpolated to a new mesh for use as an initial condition. For $\delta < 0$ the end points of u and v in the τ direction are set to zero to stop Runge interpolation errors. All solutions shown have been interpolated onto a finer equispaced grid. It should be noted that the Runge–Kutta method is suitable only for solutions which do not have large changes in the z-derivatives since the PDEs are numerically stiff.

In order to calculate the linear spectrum of the PDEs (2.1) about the stationary soliton, we perturb the stationary soliton such that

$$\begin{aligned} u(\rho,\tau,z) &= u_s(\rho,\tau,z) + e^{\lambda z}(\tilde{u}_r(\rho,\tau) + i\tilde{u}_i(\rho,\tau)),\\ v(\rho,\tau,z) &= v_s(\rho,\tau,z) + e^{\lambda z}(\tilde{v}_r(\rho,\tau) + i\tilde{v}_i(\rho,\tau)), \end{aligned} \tag{3.11}$$

where $u_s = u(\rho,\tau)e^{\kappa z i}, v_s = v(\rho,\tau)e^{2\kappa z i}$ is the stationary soliton and $|\tilde{u}|, |\tilde{v}| \ll 1$ are small perturbations in the FH and SH, respectively. Substituting the ansatz (3.11) in the PDEs (2.1) and retaining terms of first order in $\tilde{u}$ and $\tilde{v}$, we obtain the linear eigenvalue problem; see Buryak [5]. To carry out the two-dimensional differentiation we have used Kronecker products with either Chebyshev or Fourier collocation matrices; see [34] for application of the Kronecker product to the two-dimensional Laplace equation. Taking advantage of reflectional symmetry in (2.1) we can reduce the size of the eigenvalue problem and hence make the problem tractable without resorting to special matrix linear algebra techniques.

4. Results.

4.1. Implementation details. We have used the numerical continuation package AUTO [12] in our computations; see [10, 22, 35] for other possible continuation software.

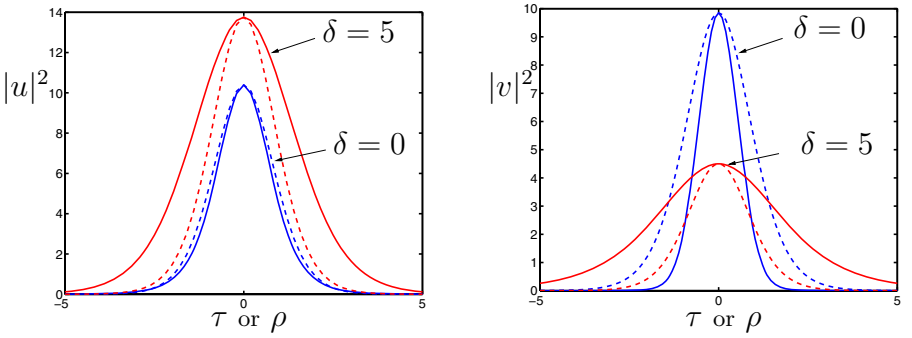
We have found that for a mesh of 11 points and $L = 6$ in the τ coordinate and $NTST = 10$, $NCOL = 4$ in AUTO, in the ρ coordinate, we have an error 10^{-4} in the maximum amplitude of u and v for $\delta = 1, \gamma = 1, \kappa = 0$ compared to AUTO computed ODEs (3.7) with $NTST = 50$ and $NCOL = 4$. For a mesh of 21 points ($L = 6$) the error decreases to 10^{-6}. The stability boundaries computed on the two meshes differ quantitatively at the third decimal place, though qualitatively they do not change; see Figures 4.2(a) and 4.2(c). Standard AUTO tolerances are used, see [12]. On a Pentium 4, 2 GHz microprocessor, computation of the stability boundaries, as in Figure 4.2(a), takes roughly 30 minutes depending on the size of the branch being computed. For numerical and interpolation reasons the infinite domain is truncated to $[-50, 50]$.

Since we are using small grids on the order of 11×11 modes, we have implemented the Runge–Kutta scheme simulating the full z-dependent equations (2.1) and the linear stability analysis in MATLAB. We have found that using an 11×11 grid the computation of the eigenvalues takes less than a minute on a 2 GHz PC and going up to a 21×21 grid takes less than five minutes. Taking advantage of symmetry is important here to minimize the size of the linear eigenvalue problem in MATLAB since the matrix grows like $4N^2$. Typical mesh sizes are for the Fourier method 20×20 points over $[0, 15]^2$ and the Chebyshev method 11×11 points with $L = 6$. Again the AUTO computed solution is interpolated to the new mesh. Any other mesh sizes will be stated where used.

4.2. Continuation and stability boundaries for $\delta \geq 0$ in (2+1) dimensions. Embedding the above continuation scheme into AUTO, we have explored the parameter region of the stationary solitons solutions to (2.1), as well as produced VK stability boundaries for various mismatch parameters.

For $\delta \neq 1$ the solitons are known to be "elliptic" in shape; see [24], where ellipticity is defined as the ratio of the soliton's temporal width to the transverse spatial width. In Figure 4.1, we see cross-sections of the computed light bullets for $\delta = 0$ and $\delta = 5$, along the spatial (ρ) and temporal (τ) coordinate. The continuous lines show the cross-section of the soliton along the temporal coordinate with the dashed lines showing the cross-section along the spatial coordinate. From Figure 4.1, for $\delta = 5$, the dashed line occurs below the solid line, in both harmonics, showing expansion along the temporal coordinate relative to the transverse coordinate, whereas for $\delta = 0$ we observe the opposite: the dashed line occurs above the solid line showing contraction along the temporal coordinate.

The VK stability plots δ versus κ are shown in Figure 4.2. Consider first Figure 4.2(a). The region of stable solitons can be found in the top right-hand corner of the figure with the unstable region directly below. The stability boundary described by the VK criterion is shown as a line separating the stable and unstable regions. Beneath the unstable region we have drawn the line showing the existence boundary of the solitons governed by the bound $\kappa > \max\{-\gamma/4, -1\}$ such that the tails in both harmonics decay exponentially. The region for the possible existence of quasi solitons is shown as $\delta < 0$. A "dip" in the stability boundary can be observed for mismatch parameters, $\gamma < 4$, close to $\delta = 0$ in (2+1) dimensions. Qualitatively similar behavior has been found by Malomed et al. [23] using a Gaussian approximation, albeit with



(a) (b)

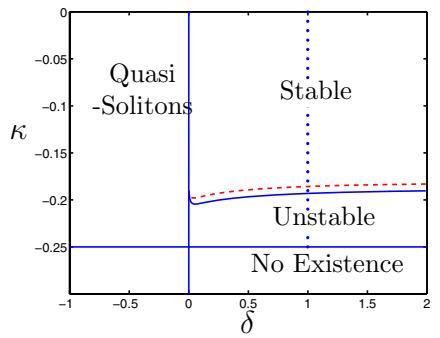
FIG. 4.1. *Spatial and temporal sections of the light bullets along* $\rho = 0$ *(solid line) and* $\tau = 0$ *(dashed line), respectively, for* $\gamma = 1$, $\kappa = 0$ *in* $(2+1)$ *dimensions and* $\delta = 0$ *and* $\delta = 5$. *The FH is shown in* (a) *and the SH is shown is* (b).

gross quantitative differences. As γ is increased towards 4, the stability boundary is found to uniformly approach the line $\kappa = -1$ ($\delta > 0$), such that for $\gamma \geq 4$ all solitons are found to be stable; see Figure 4.2(e), numerically confirming and extending the result found by Pelinovsky, Buryak, and Kivshar [28] and Steblina et al. [32]. For all γ between 0 and 4 the stability boundary is found to have similar qualitative shapes (results not depicted).

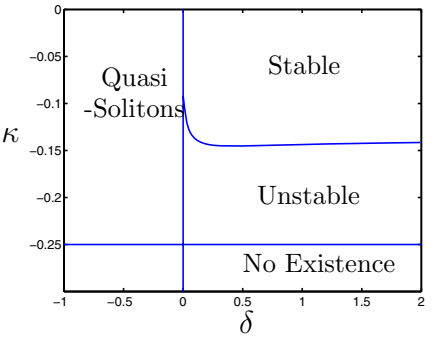
We can test the accuracy of the VK criteria by carrying out the linear stability analysis about the AUTO computed solitons using our Chebyshev method. In Figure 4.2(a) we have drawn a dotted line taking a cross-section through the VK stability boundary at $\gamma = 1$ and $\delta = 1$. In Figure 4.3 we depict the maximum computed positive eigenvalue for $\gamma = 1$ and $\delta = 1$ in (2+1) dimensions for a range of κ values. The VK stability boundary for these parameters occurs at $\kappa \approx -0.19$, shown in Figure 4.3 as a vertical dashed line. Using both the Chebyshev method and the Fourier method with a coarse mesh, described in section 3.3, we see that the VK criterion gives an overestimate for the stability region of the light bullets. Close to the stability boundary, both the Chebyshev method and the Fourier method give similar results. Under higher resolution, the Chebyshev method shows the solitons losing stability closer to the VK boundary. These results suggest (but do not prove) that in fact the VK criterion provides a sharp stability boundary.

Figure 4.4(a) shows an evolution plot of the intensity of the SH of a stable stationary soliton along the line $\rho = 0$. Just past the stability boundary at $\delta = 1$ and $\kappa = -0.2$ we see from Figure 4.4(d) that the evolution of the stationary soliton, at this point, eventually becomes unstable after a long distance.

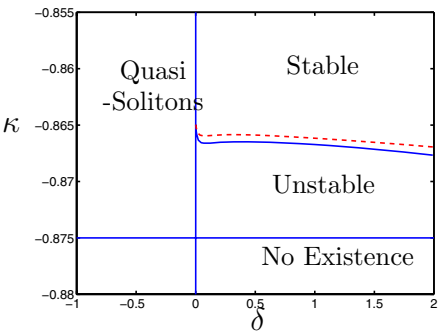
4.3. The region $\delta < 0$ in (2+1) dimensions. For $\delta < 0$, Towers, Malomed, and Wise [33] showed that no true solitons exist in this parameter region. Instead one should be able to find solutions that do not decay at infinity but have oscillatory tails; these solutions are known as quasi solitons [18]. Using our numerical method we find (for small negative δ values), solitons with oscillating tails; see Figure 4.5 with the corresponding evolution plot shown in Figure 4.4(b) showing the quasi soliton eventually becoming unstable. We have found, numerically, that one need not go



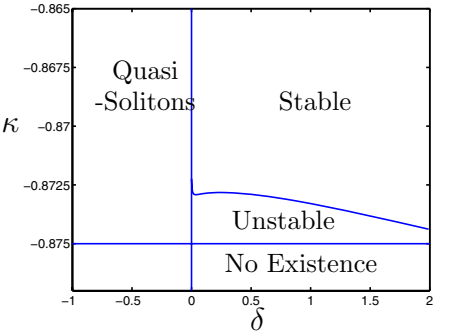
(a) $\gamma = 1, D = 2$



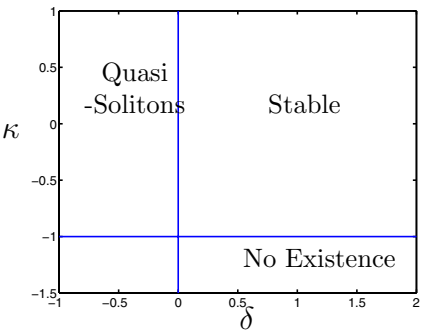
(b) $\gamma = 1, D = 3$



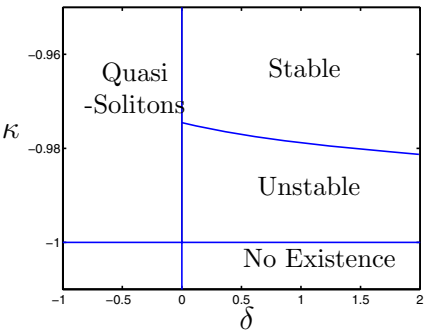
(c) $\gamma = 3.5, D = 2$



(d) $\gamma = 3.5, D = 3$



(e) $\gamma = 5, D = 2$



(f) $\gamma = 5, D = 3$

FIG. 4.2. *Existence and stability boundaries for soliton solutions. The left-hand plots are in* $(2+1)$ *dimensions* $(d = 1)$ *and the right-hand in* $(3+1)$ *dimensions* $(d = 2)$*. The vertical dotted line in* (a) *represents the parameter path taken in Figure* 4.3*. In* (a) *and* (c) *the stability boundaries computed with* 11 *Chebyshev modes (solid line) and* 21 *Chebyshev modes (dashed line),* $L = 6$*, are shown. Possible existence region for quasi solitons is noted in each figure.*

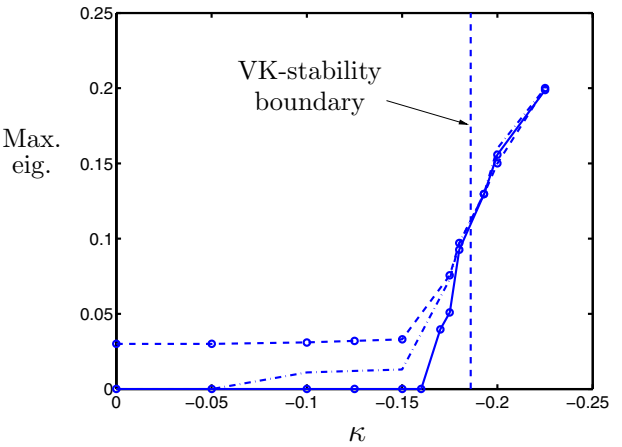


FIG. 4.3. *Maximum eigenvalue computed from the linearization about the stationary soliton for* $\gamma = 1$ *and* $\delta = 1$. *The VK stability boundary is shown as a vertical dashed line at* $\kappa = -0.186$ *(calculated on a mesh of* 21 *points). The dashed line with circles corresponds to the linear spectrum computed with a Fourier method with a* 20×20 *grid over a* $[0, 15]^2$ *domain. The dashed-dotted line corresponds to the linear spectrum being computed with the Chebyshev method with an* 11×11 *grid,* $L = 6$, *and the solid line corresponds to a* 21×21 *grid with* $L = 13$.

far into the quasi-soliton ($\delta < 0$) region to hit an existence boundary; e.g., in (2+1) dimensions, $\kappa = -0.15$ and $\gamma = 1$, the threshold for existence of such quasi solitons appears to be $0 > \delta \geq -0.042$. This may explain why Malomed et al. [23] found numerically stable quasi solitons for small negative δ, but no stable solitons far out into this region: they had stepped outside the existence boundary. As $\delta \to -0.042$ the fundamental harmonic tends to zero and the oscillations in the SH grow.

If either γ or κ is increased the existence region appears to increase in size with some of these quasi solitons found to be numerically stable; see Figure 4.6. Computation of quasi solitons using the Chebyshev basis on the infinite interval is known to poorly approximate solitons with oscillatory tails regardless of the number of modes used; see the discussion in Chapter 9 of Boyd [3]. The reason for the poor approximation of the oscillating tails is due to the Chebyshev method implicitly imposing zero boundary conditions at the ends of the domain, hence "cutting off" the oscillating tails. However, the numerical solution is known to provide an acceptable approximation of the core of the soliton, but questions of stability depend crucially on details of the tails and so cannot be answered with any certainty. As solitons become unstable in their tails this numerical method may falsely impose (in)stability where none may exist. This "cutting off" of the tails becomes more pronounced as either γ or κ is increased. In [26] it was shown that for small equations in (1+1) dimensions quasi solitons are always unstable albeit with a small unstable eigenvalue. We conjecture that this property is also true for quasi solitons in higher dimensions. Therefore, any numerical results on existence and stability of quasi solitons using a Chebyshev discretization method would be at best speculative. Hence, in Figure 4.3 for the region $\delta < 0$, any possibility of existence is labeled "quasi solitons."

Several amendments to the Chebyshev method on the infinite domain have been suggested (see Boyd [3]) but the computation of quasi solitons remains a challenging problem and has been solved only in (1+1) dimensions; see Champneys, Vanden-Broek, and Lord [9]. On should also stress that quasi solitons have questionable

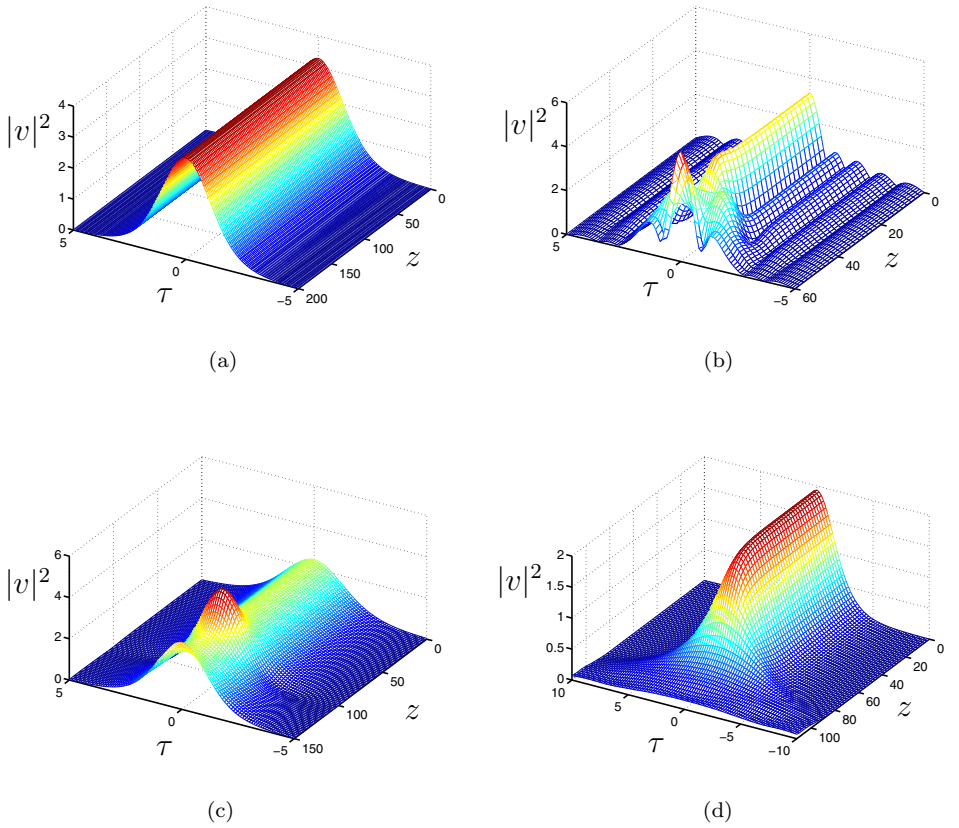
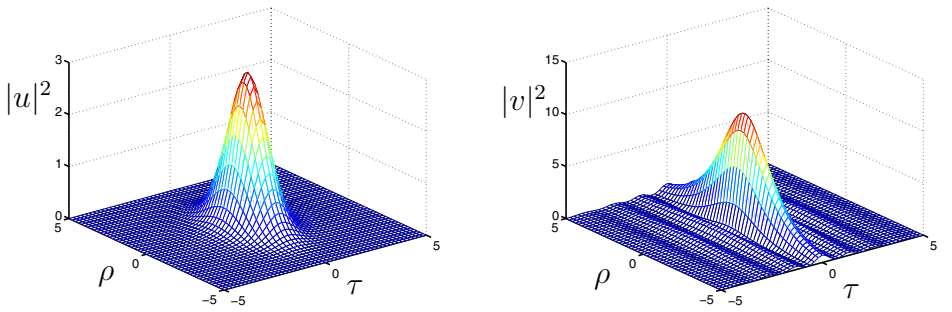


FIG. 4.4. *Evolution plots of* $|v(z,0,\tau)|^2$ *starting from initial data computed from numerical continuation for* $\gamma = 1$ *and* (a) *stable soliton* $\delta = 1$ *and* $\kappa = -0.15$, (b) *unstable quasi soliton* $\delta = -0.04$ *and* $\kappa = -0.15$, (c) *unstable soliton on stability boundary* $\delta = 1$ *and* $\kappa = -0.2$, (d) *unstable soliton* $\delta = 1$ *and* $\kappa = -0.225$

physical validity since they require sources and sinks of energy at infinity. At best their existence represents parameter regimes where finite-time, almost localized structures may be expected to be observed in an experiment.

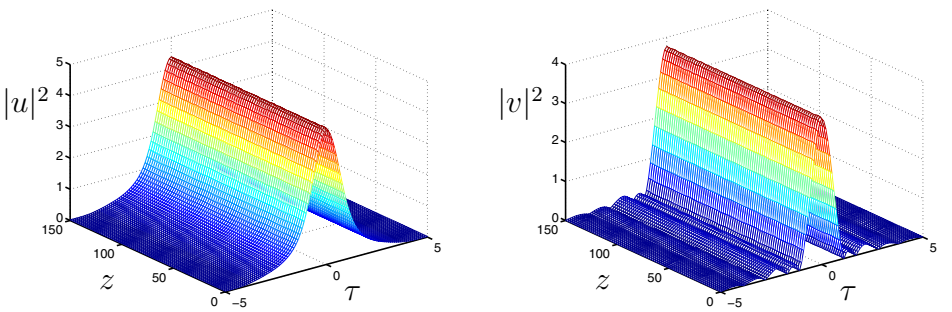
4.4. Stability boundaries in (3+1) dimensions. If we assume radial symmetry in (3+1) dimensions, we can reduce the equations to an effective (2+1) dimension with the addition of a $(1/\rho)d/d\rho$ term. This allows us to use our method and compute VK stability boundaries in (3+1) dimensions. (We shall not attempt to verify the VK criteria in (3+1) dimensions since results are already known; see Mihalache et al. [24].) The results here are to show the full three-parameter shape of the stability boundary.

In (2+1) dimensions it is known that solitons cannot be unstable for $\gamma \geq 4$, but in (3+1) dimensions it is possible to find unstable regions for this parameter space; see Figure 4.2(f). This unstable region decreases in size until $\gamma = 4$ where all solitons are found to be stable. As we pass through $\gamma = 4$ we see the existence of a new unstable region. This region is always larger than the one found in (2+1) dimensions for equivalent parameters; see Figures 4.2(a) and 4.2(b).



(a) (b)

FIG. 4.5. *Quasi soliton for* $\gamma = 1$, $\kappa = -0.15$ *and* $\delta = -0.04$. *Note the oscillations in the second harmonic* v.



(a) (b)

FIG. 4.6. *Evolution of the* (a) *fundamental and* (b) *second harmonic for* $\delta = -0.12, \gamma = 2, \kappa = 0$ *in* $(2+1)$ *dimensions. This soliton is outside the stability boundary found by Towers, Malomed, and Wise* [33].

5. Conclusion. In this paper we have set out an efficient numerical method for finding stationary soliton solutions for the type-I SHG equations in (2+1) and (3+1) dimensions. The method described uses an adapted mesh applied to the localized soliton problem and Chebyshev differentiation matrices to carry out either the spatial or temporal differentiation. This adapted mesh method allows us to reduce the size of the discretized problem enabling the use of numerical continuation software. Using the VK criterion we can trace out stability boundaries for various equation parameters in (2+1) and (3+1) dimensions. We have also found, numerically, quasi solitons when the dispersion at the SH is normal. Known results for the ellipticity of the light bullets for various dispersion coefficients at the SH and validity of the VK criteria are confirmed, and extended to three-parameter plots in δ, κ, γ.

We have used our discretization scheme to test the validity of the VK stability criteria in (2+1) dimensions via numerical simulation of the full z-dependent equations,

as well as direct computation of the linear spectrum about the stationary soliton. Confirmation of the VK stability criteria in (3+1) dimensions can be found in [24].

Our results in (2+1) dimensions confirm the findings of Mihalache et al. [24] and Skryabin and Firth [31], yet in addition we have been able to provide a full two-parameter plot of the stability region and indeed to show how this region shrinks when we vary a third parameter.

Our numerical method can be extended to a wide range of soliton-bearing equations including dissipative systems, though for dissipative systems the VK criterion is no longer valid and so stability boundaries would have to be calculated another way. Group velocity mismatch between the FH and SH could also be accommodated with our numerical method by moving into a traveling frame of reference (see Mihalache et al. [25]), but in this case the questions describing the soliton shape would be complex. For continuation purposes one must deal with the extra symmetry generated by the complex phase shift since the Jacobian of the symmetry will be singular. To form a well-posed BVP one could either explicitly decouple the complex phase (see Luce [21]) or use phase minimization techniques; see Munoz-Aimaraz et al. [27]. Future work will concentrate on the type-II interaction, SHG equations (see [5]) where far less is known about the structure of the solutions in (2+1) and (3+1) dimensions.

Acknowledgments. The authors would like to thank John P. Boyd, Boris A. Malomed and Dumitru Mihalache for their useful remarks. The authors would also like to thank Richard R. Kerswell for initially suggesting the numerical discretisation.

REFERENCES

[1] U. Ascher and R. Russell, eds., *Numerical Boundary Value ODEs*, Birkhäuser, Boston, 1984.

[2] K. Beckwitt, Y.-F. Chen, F. Wise, and B. Malomed, *Temporal solitons in quadratic nonlinear media with opposite group-velocity dispersions at the fundamental and second harmonics*, Phys. Rev. E, 68 (2003), 057601.

[3] J. P. Boyd, *Weakly Nonlocal Solitary Waves and Beyond-All-Orders Asymptotics: Generalised Solitons and Hyperasymptotic Perturbation Theory*, Kluwer Academic Publishers, Dordrecht, 1998.

[4] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, 2nd ed., Dover, New York, 2001.

[5] A. V. Buryak, *Solitons Due to Quadratic Nonlinearities*, Ph.D. thesis, Australia National University, Canberra, 1996.

[6] A. V. Buryak, P. D. Trapani, D. V. Skryabin, and S. Trillo, *Optical solitons due to quadratic nonlinearities: from basic physics to futuristic applications*, Phys. Reports, 370 (2002), pp. 63–235.

[7] C. Canuto, M. Hussaini, A. Quarteroni, and T. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.

[8] A. Champneys, Y. Kuznetsov, and B. Sandstede, *A numerical toolbox for homoclinic bifurcation analysis*, Internat. J. Bifurc. Chaos Appl. Sci. Engrg., 6 (1996), pp. 867–887.

[9] A. Champneys, J.-M. Vanden-Broek, and G. Lord, *Do true elevation solitary waves exist? A numerical investigation*, J. Fluid Mech., 454 (2002), pp. 403–417.

[10] A. Dhooge, W. Govaerts, and Y. Kuznetsov, *Matcont: A Matlab package for numerical bifurcation analysis of odes*, ACM Trans. Math. Software, 29 (2003), pp. 141–164.

[11] E. Doedel, H. B. Keller, and J. P. Kernevez, *Numerical analysis and control of bifurcation problems* (I) *Bifurcation in finite dimensions*, Internat. J. Bifurc. Chaos Appl. Sci. Engrg., 1 (1991), pp. 493–520.

[12] E. Doedel, H. B. Keller, and J. P. Kernevez, *Numerical analysis and control of bifurcation problems* (II) *Bifurcation in infinite dimensions*, Internat. J. Bifurc. Chaos Appl. Sci. Engrg., 1 (1991), pp. 745–772.

[13] M. J. Friedman and E. J. Doedel, *Numerical computation and continuation of invariant manifolds connecting fixed points*, SIAM J. Numer. Anal., 28 (1991), pp. 789–808.

[14] H. He, M. Werner, and P. Drummond, *Simultaneous solitary-wave solutions in a nonlinear parametric waveguide*, Phys. Rev. E, 54 (1996), pp. 896–911.
[15] A. Kanashov and A. Rubenchik, *On diffraction and dispersion effect on three wave interaction*, Phys. D, 4 (1981), pp. 122–134.
[16] H. Keller, *Lectures on Numerical Methods in Bifurcation Problems*, Tata Institute of Fundamental Research, Lectures on Mathematics and Physics 79, Tata Institute, Bombay, Springer-Verlag, Berlin, 1987.
[17] L. Liu, G. Moore, and R. D. Russell, *Computation and continuation of homoclinic and heteroclinic orbits with arclength parameterization*, SIAM J. Sci. Comput., 18 (1997), pp. 69–93.
[18] X. Liu, K. Beckwitt, and F. Wise, *Noncollinear generation of optical spatiotemporal solitons and application to ultrafast digital logic*, Phys. Rev. E, 61 (2000), R4722.
[19] X. Liu, K. Beckwitt, and F. Wise, *Two-dimensional optical spatiotemporal solitons in quadratic media*, Phys. Rev. E, 62 (2000), pp. 1328–1340.
[20] X. Liu, L. Qian, and F. Wise, *Generation of optical spatiotemporal solitons*, Phys. Rev. Lett., 82 (1999), pp. 4631–4634.
[21] B. Luce, *Homoclinic explosions in the complex Ginzburg-Landau equation*, Phys. D, 84 (1995), pp. 553–581.
[22] K. Lust, D. Roose, A. Spence, and A. R. Champneys, *An adaptive Newton–Picard algorithm with subspace iteration for computing periodic solutions*, SIAM J. Sci. Comput., 19 (1998), pp. 1188–1209.
[23] B. A. Malomed, P. Drummond, H. He, A. Bertson, D. Anderson, and M. Lisak, *Spatiotemporal solitons in multidimensional optical media with a quadratic nonlinearity*, Phys. Rev. E, 56 (1997), pp. 4725–4735.
[24] D. Mihalache, D. Mazil, J. Dorring, and L. Torner, *Elliptical light bullets*, Optics Communications, 159 (1999), pp. 129–138.
[25] D. Mihalache, D. Mazilu, L.-C. Crasovan, L. Torner, B. A. Malomed, and F. Lederer, *Three-dimensional walking spatiotemporal solitons in quadratic media*, Phys. Rev. E, 62 (2000), pp. 7340–7347.
[26] M. J. Pearce, *Embedded Solitons in a Three-Wave System*, Ph.D. thesis, Engineering Mathematics, University of Bristol, Bristol, UK, 2003.
[27] F. Munoz-Almaraz, E. Freire, J. Galán, E. Doedel, and A. Vanderbauwhede, *Continuation of periodic orbits in conservative and Hamiltonian systems*, Phys. D, 181 (2003), pp. 1–38.
[28] D. Pelinovsky, A. Buryak, and Y. Kivshar, *Instability of solitons governed by quadratic nonlinearities*, Phys. Rev. Lett., 75 (1995), pp. 591–595.
[29] R. Seydel, *Nonlinear computation*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 7 (1997), pp. 2105–2126.
[30] D. V. Skryabin, *Stability of multi-parameter solitons: Asymptotic approach*, Phys. D, 139 (2000), pp. 186–193.
[31] D. V. Skryabin and W. J. Firth, *Generation and stability of optical bullets in quadratic nonlinear media*, Optics Communications, 148 (1998), pp. 79–84.
[32] V. Steblina, Y. Kivshar, M. Lisak, and B. Malomed, *Self-guided beams in a diffractive $\chi^{(2)}$ medium - variational approach*, Optics Communications, 118 (1995), pp. 345–352.
[33] I. Towers, B. Malomed, and F. Wise, *Light bullets in quadratic media with normal dispersion at the second harmonic*, Phys. Rev. Lett., 90 (2003), 123902.
[34] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.
[35] F. Ulrike and J. Wolfgang, *Candys/qa—A software system for the qualitative analysis of nonlinear dynamical systems*, Internat. J. Bifur. Chaos Appl. Sci. Engrg., 2 (1992), pp. 773–794.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 774–792
© 2005 Society for Industrial and Applied Mathematics

COMBINING FAST MULTIPOLE TECHNIQUES AND AN APPROXIMATE INVERSE PRECONDITIONER FOR LARGE ELECTROMAGNETISM CALCULATIONS*

B. CARPENTIERI†, I. S. DUFF‡, L. GIRAUD§, AND G. SYLVAND¶

Abstract. The boundary element method has become a popular tool for the solution of Maxwell's equations in electromagnetism. From a linear algebra point of view, this leads to the solution of large dense complex linear systems, where the unknowns are associated with the edges of the mesh defined on the surface of the illuminated object. In this paper, we address the iterative solution of these linear systems via preconditioned Krylov solvers. Our primary focus is on the design of an efficient parallelizable preconditioner. In that respect, we consider an approximate inverse method based on the Frobenius-norm minimization. The preconditioner is constructed from a sparse approximation of the dense coefficient matrix, and the patterns both for the preconditioner and for the coefficient matrix are computed a priori using geometric information from the mesh. We describe how such a preconditioner can be naturally implemented in a parallel code that implements the multipole technique for the matrix-vector product calculation. We investigate the numerical scalability of our preconditioner on realistic industrial test problems and show that it exhibits some limitations on very large problems of size close to one million unknowns. To improve its robustness on those large problems we propose an embedded iterative scheme that combines nested GMRES solvers with different fast multipole computations. We show through extensive numerical experiments that this new scheme is extremely robust at affordable memory and CPU costs for the solution of very large and challenging problems.

Key words. sparse approximate inverse preconditioner, large dense complex systems, Maxwell equations, boundary elements, fast multipole technique, frequency domain, electric field integral equations, parallel computing, large calculations

AMS subject classifications. 65F10, 65R20, 65N38, 78A55, 78M15

DOI. 10.1137/040603917

1. Introduction.

The analysis of wave propagation phenomena is gaining an increasing interest in recent years in the simulation of many challenging industrial processes, including the prediction of the Radar Cross Section (RCS) of arbitrarily shaped 3D objects such as aircraft, the study of electromagnetic compatibility of electrical devices with their environment, the design of antennae and absorbing materials, and many others. All these simulations are very demanding in terms of computer resources and require fast and efficient numerical methods to compute an approximate solution of Maxwell's equations. Using the equivalence principle, Maxwell's equations

*Received by the editors February 6, 2004; accepted for publication (in revised form) May 6, 2005; published electronically November 15, 2005.
http://www.siam.org/journals/sisc/27-3/60391.html

†CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France (carpenti@cerfacs.fr). Current address: Karl-Franzens-University Graz, Institute of Mathematics and Scientific Computing, Heinrichstraße 36, A-8010 Graz, Austria (bruno.carpentieri@uni-graz.at).

‡CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France (duff@cerfacs.fr), and RAL, Oxfordshire, England (i.s.duff@rl.ac.uk).

§CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France (giraud@cerfacs.fr). Current address: ENSEEIHT, 2 rue Charles Camichel, 31 171 Toulouse, France (luc.giraud@enseeiht.fr).

¶CERMICS-INRIA, 2004 route des lucioles BP 93, 06902 Sophia-Antipolis Cedex, France. Current address: EADS CCR, Centre de Toulouse, Centreda 1, 4 Avenue Didier Daurat, 31700 Blagnac, France (Guillaume.Sylvand@eads.net). The work of this author was supported by EADS, Corporate Research Centre, Toulouse.

can be recast in the form of four integral equations that relate the electric and magnetic fields to the equivalent electric and magnetic currents on the surface of the object.

Among integral formulations, the electric field integral equation (EFIE) is the most general for electromagnetic scattering problems, as it can handle fairly general geometries, and thus is widely used in industrial simulations. The EFIE provides a first-kind integral equation, which is well known to be ill-conditioned and gives rise to linear systems that are challenging to solve by iterative methods. The discretization is performed on the surface of the object and gives rise to a linear system

$$Ax = b, \tag{1.1}$$

where the matrix A is an $n \times n$, dense, complex, symmetric, non-Hermitian matrix. For many years, direct solution methods have been the methods of choice for solving systems (1.1) because they are reliable and predictable in terms of both accuracy and computing costs. However, for the solution of large-scale problems, direct methods are infeasible even on large parallel platforms because they require the unaffordable storage of n^2 single or double precision complex entries of the coefficient matrix and $\mathcal{O}(n^3)$ floating-point operations to compute the factorization. The use of preconditioned Krylov solvers can be an alternative to direct solution methods, provided we have fast matrix-free matrix-vector products and robust preconditioners. Research efforts have recently concentrated on fast methods for performing matrix-vector products with $\mathcal{O}(n \log(n))$ computational complexity, including strategies for parallel distributed memory implementations. These methods, generally referred to as [illegible], were introduced originally in the context of the study of particle simulations and can be used effectively in boundary element applications.

In this paper, we focus on the design of a parallelizable preconditioner to be used in conjunction with a parallel distributed fast multipole technique that implements the matrix-vector calculation. We consider an approximate inverse preconditioner based on Frobenius-norm minimization with a pattern prescribed in advance. In section 2, we describe the implementation of the preconditioner within an out-of-core parallel code that implements the fast multipole method (FMM) for the matrix-vector product. We investigate the numerical scalability of the preconditioner on a set of industrial problems of increasing size and show that it becomes less effective when the problem size becomes very large. To overcome this weakness, we propose in section 3 an embedded iterative scheme based on the GMRES method that aims at improving the robustness of the preconditioner on large applications. We illustrate the numerical efficiency and the cost effectiveness of the proposed scheme on systems of up to one million unknowns arising from challenging problems from the electromagnetism community.

2. Preconditioning boundary integral equations. For large meshes with many surface details, the density of the discretization mesh is nonuniform, and the matrix generated by the method of moments can become ill-conditioned. The convergence of Krylov methods depends to a large extent on the eigenvalue distribution of the coefficient matrix. In Figure 2.1, we plot the eigenvalue distribution in the complex plane of the matrix associated with a satellite geometry. This distribution is representative of the general trend; it can be seen that the eigenvalues of the system are very scattered; many of them have a large negative real part and no clustering appears. Such a distribution is not at all favorable for the rapid convergence of Krylov methods. One goal of preconditioning is to improve this distribution by grouping the

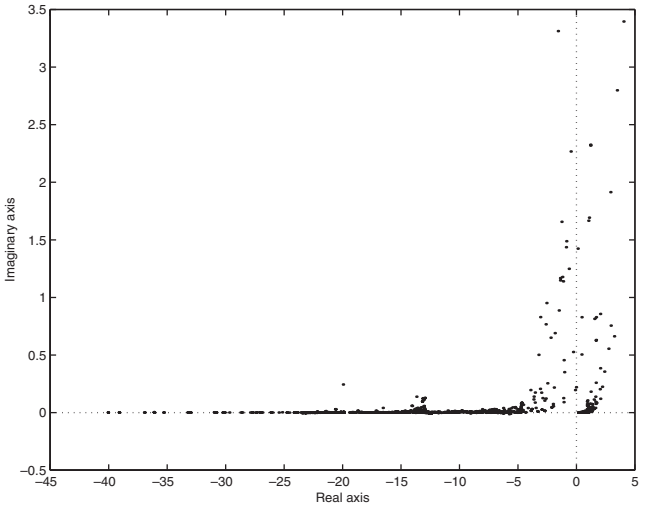


FIG. 2.1. *Eigenvalue distribution in the complex plane of the coefficient matrix for a scattering problem from a satellite that is representative of the general trend.*

eigenvalues into a few small clusters.

The design of robust preconditioners for boundary integral equations can be challenging. Simple preconditioners like the diagonal of A, diagonal blocks, or a band can be effective only when the coefficient matrix has some degree of diagonal dominance arising from the integral formulation [30]. Block diagonal preconditioners are generally more robust than their pointwise counterparts but may require matrix permutations or renumbering of the grid points to cluster the large entries close to the diagonal. Incomplete factorizations have been successfully used on nonsymmetric dense systems in [28] and hybrid integral formulations in [24], but for the EFIE the triangular factors computed by the factorization are often very ill-conditioned due to the indefiniteness of A. This makes the triangular solves highly unstable and the preconditioner useless. In [5], we used a small diagonal shift applied to A before computing the factorization with the intention of moving the eigenvalues along the imaginary axis. This might avoid a possible eigenvalue cluster close to zero and can help the computation of more stable factors in some cases. However, this shift is not easy to tune and there is no way to predict its effect.

Approximate inverse methods are generally less prone to instabilities on indefinite systems, and several preconditioners of this type have been proposed in electromagnetism (see for instance [1, 3, 4, 6, 23, 27, 35]). Owing to the rapid decay of the discrete Green's function, the location of the large entries in the inverse matrix exhibits some structure. In addition, only a few of its entries have relatively large magnitude. This means that a very sparse matrix is likely to retain the most relevant contributions to the exact inverse. This remarkable property can be effectively exploited in the design of robust approximate inverses as preconditioners for electromagnetism applications.

2.1. Frobenius-norm minimization preconditioner. In this section, we describe an approximate inverse preconditioner based on Frobenius-norm minimization. The original idea, due to Benson and Frederickson [2, 15], is to compute the sparse approximate inverse as the matrix M which minimizes $\|I - MA\|_F$ (or $\|I - AM\|_F$ for right preconditioning) subject to certain sparsity constraints. The Frobenius norm

is usually chosen since it allows for the decoupling of the minimization problems into n independent linear least-squares problems, one for each column of M, when preconditioning from the right (or row of M, when preconditioning from the left). The independence of these least-squares problems follows immediately from the identity

$$\|I - AM\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{\bullet j}\|_2^2, \tag{2.1}$$

where e_j is the jth canonical unit vector, $m_{\bullet j}$ is the column vector representing the jth column of M, and n the dimension of the square matrices. In the case of left preconditioning, because A is symmetric, the analogous relation

$$\|I - MA\|_F^2 = \|I - AM^T\|_F^2 = \sum_{j=1}^{n} \|e_j - Am_{j\bullet}\|_2^2 \tag{2.2}$$

holds, where $m_{j\bullet}$ is the column vector representing the jth row of M. Clearly, there is considerable scope for parallelism in this approach. The main issue is the selection of the sparsity pattern of M, that is, the set of indices

$$S = \{(i,j) \subseteq [1,n]^2 \text{ such that } m_{ij} = 0\}. \tag{2.3}$$

The idea is to keep M reasonably sparse while trying to capture the "large" entries of the inverse, which are expected to contribute the most to the quality of the preconditioner. Two different approaches can be followed for this purpose: an adaptive technique that dynamically tries to identify the best structure for M, and a static technique, where the pattern of M is prescribed a priori based on some heuristics. Adaptive methods usually start with a simple initial guess, like a diagonal matrix, and then improve the pattern until a criterion of the form $\|Am_{\bullet j} - e_j\|_2 < \varepsilon$ (for each j) is satisfied for a given $\varepsilon > 0$ or until a maximum number of nonzeros in $m_{\bullet j}$ is reached. If the norm is larger than ε and the number of nonzeros used is less than a fixed maximum, the pattern is enlarged according to some heuristics and the jth column of the approximate inverse is recomputed. The process is repeated until the required accuracy or storage limit is met (see [7, 21]).

Adaptive strategies can solve fairly general and hard problems but tend to be very expensive. The use of effective static pattern selection strategies can greatly reduce the amount of work in terms of CPU time and can substantially improve the overall setup process. When the coefficient matrix has a special structure or special properties, efforts have been made to find a pattern that can retain the entries of A^{-1} having large modulus [10, 11, 12, 32]. If A is row diagonally dominant, then the entries in the inverse decay columnwise and vice versa [32]. On boundary integral equations the discrete Green's function decays rapidly far from the diagonal, and the inverse of A may have a very similar structure to that of A. The discrete Green's function can be considered as a row or as a column of the exact inverse depicted on the physical computational grid. In this case a good pattern for the preconditioner can be computed in advance using graph information from $\tilde{A}$, a sparse approximation of the coefficient matrix constructed by dropping all the entries lower than a prescribed global threshold [1, 4, 22]. When fast methods are used for the matrix-vector products, all the entries of A are not available and the pattern can be formed by exploiting the near-field part of the matrix that is explicitly computed and available in the FMM [23].

Since we work in an integral equation context, relevant information for the construction of the pattern of M can be extracted from the mesh. When the object

geometries are smooth, only the neighboring edges (in the mesh topology sense) in the mesh can have a strong interaction with each other, while faraway connections are generally much weaker. Thus an effective pattern for the jth column of the approximate inverse can be computed by selecting in the mesh edge j and its qth level nearest neighbors. Three levels generally provide a good pattern for constructing an effective sparse approximate inverse. Using more levels increases the computational cost but does not improve the quality of the preconditioner substantially [4]. When the object geometries are not smooth or have disconnected parts, faraway edges in the mesh can have a strong interaction and be strongly coupled in the inverse matrix. In this case, a more robust pattern for the preconditioner can be computed using geometrical information, that is, selecting for each edge all those edges within a sufficiently large geometric neighborhood. In [4] we compared pattern selection strategies based on both algebraic and mesh information on a large set of problems and found that those exploiting geometric information are the most effective in capturing the large entries of the inverse.

In order to preserve sparsity, $\mathcal{O}(1)$ nonzero locations are computed in the pattern of the column $m_{\bullet j}$ of M. This makes each QR least-squares solution for each column of M from (2.1) cost $\mathcal{O}(n)$, and the overall construction of M costs approximately $\mathcal{O}(n^2)$ arithmetic operations. This cost can be significantly reduced if the preconditioner is computed using as input a sparse approximation $\tilde{A}$ of the dense coefficient matrix A. On general problems, this approach can cause a severe deterioration of the quality of the preconditioner. In an integral equation context, it is likely to be more effective because the boundary element method generally introduces a very localized strong coupling among the edges in the underlying mesh. It means that a very sparse matrix can still retain the most relevant contributions from the singular integrals that give rise to dense matrices. If the sparsity pattern S of M is known in advance, the nonzero structure for the jth column of M is automatically determined and defined as

$$J = \{i \in [1, n] \text{ such that } (i, j) \in S\}.$$

The least-squares solution involves only the columns of $\tilde{A}$ indexed by J; we indicate this subset by $\tilde{A}(:, J)$. When $\tilde{A}$ is sparse, many rows in $\tilde{A}(:, J)$ are usually null, not affecting the solution of the least-squares problems (2.1). Thus if I is the set of indices corresponding to the nonzero rows in $\tilde{A}(:, J)$, and if we define $\hat{A} = \tilde{A}(I, J)$, $\hat{m}_j = m_j(J)$, and $\hat{e}_j = e_j(J)$, the actual "reduced" least-squares problems are

$$\min \|\hat{e}_j - \hat{A}\hat{m}_j\|_2, \qquad j = 1, \ldots, n. \tag{2.4}$$

Usually problems (2.4) have much smaller size than problems (2.1) and can be efficiently solved by a dense QR factorization. In [1] the same nonzero sparsity pattern is selected for both $\tilde{A}$ and M; in that case, especially when the pattern is very sparse, the computed preconditioner may be poor on some geometries. Selecting more entries in $\tilde{A}$ than in M can provide a more robust preconditioner, and the additional cost in terms of CPU time is negligible because of the complexity of the QR factorization [4]. Increasing the number of rows q_I, that is, the number of entries of $\tilde{A}$, affects the CPU time far less than increasing the density of the preconditioner, that is, the number of columns p_J in the least-squares problems. This is because each least-squares solution costs $\mathcal{O}(q_I p_J^2)$.

2.2. Implementation of the preconditioner in the FMM context. The FMM, introduced by Greengard and Rokhlin in [20], provides an algorithm for computing approximate matrix-vector products for electromagnetic scattering problems.

The method is fast in the sense that the computation of one matrix-vector product costs $\mathcal{O}(n \log n)$ arithmetic operations instead of the usual $\mathcal{O}(n^2)$ operations. It requires storage only of the near-field part of the matrix, that is, of $\mathcal{O}(n \log n)$ instead of $\mathcal{O}(n^2)$ entries. These properties mean that using iterative solvers for the solution of large problems becomes feasible.

The basic idea of the algorithm is to compute interactions among degrees of freedom in the mesh at different levels of accuracy depending on their physical distance. Single and multilevel variants of the FMM exist. The 3D object is first entirely enclosed in a large cube that is subdivided into eight cubes. Each cube is recursively divided until the size is small compared with the wavelength. In the hierarchical multilevel algorithm, the box-wise partitioning of the obstacle is carried out until the size of the smallest box is generally half of a wavelength, and tree-structured data is used at all levels. In particular, only nonempty cubes are indexed and recorded in the data structure. The resulting tree is called the [illegible] (see Figure 2.2) and its leaves are referred to as [illegible]. The oct-tree provides a hierarchical representation of the computational domain partitioned by boxes: each cube has up to eight children and one parent in the oct-tree, except for the largest cube which encloses the whole domain. Obviously, the leaf boxes have no children. Multipole coefficients are computed for all cubes starting from the lowest level of the oct-tree, that is, from the leaf boxes, and then recursively for each parent cube by summing together multipole coefficients of its children. For each observation cube, an interaction list is defined which consists of those cubes that are not neighbors of the cube itself but whose parent is a neighbor of the cube's parent (see Figure 2.3 for a 2D representation). The interactions of degrees of freedom within neighboring boxes are computed exactly, while the interactions between cubes that are in the interaction list are computed using the FMM. All the other interactions are computed hierarchically on a coarser level by traversing the oct-tree. Both the computational cost and the memory requirement of the algorithm are of order $\mathcal{O}(n \log n)$. Further information on the algorithmic steps and recent theoretical investigations of the FMM can be found in [8, 9, 31]; also see [17, 19, 36] for discussions on parallel implementation issues.

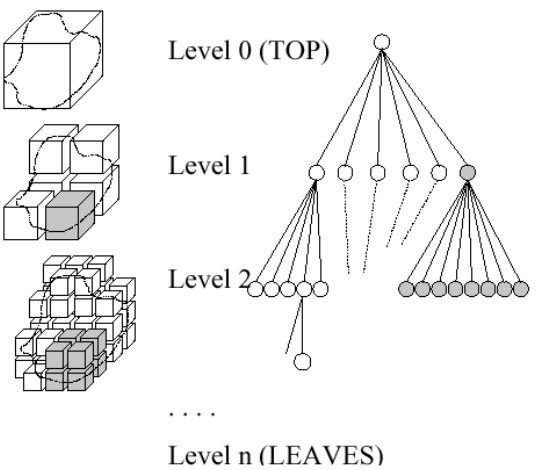


FIG. 2.2. *The oct-tree in the FMM algorithm. The maximum number of children is eight. The actual number corresponds to the subset that intersects the object.*

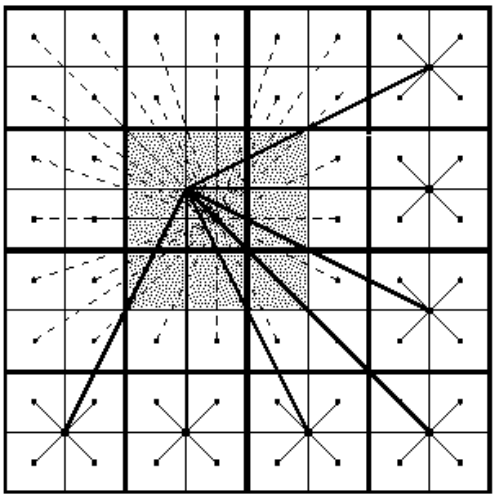


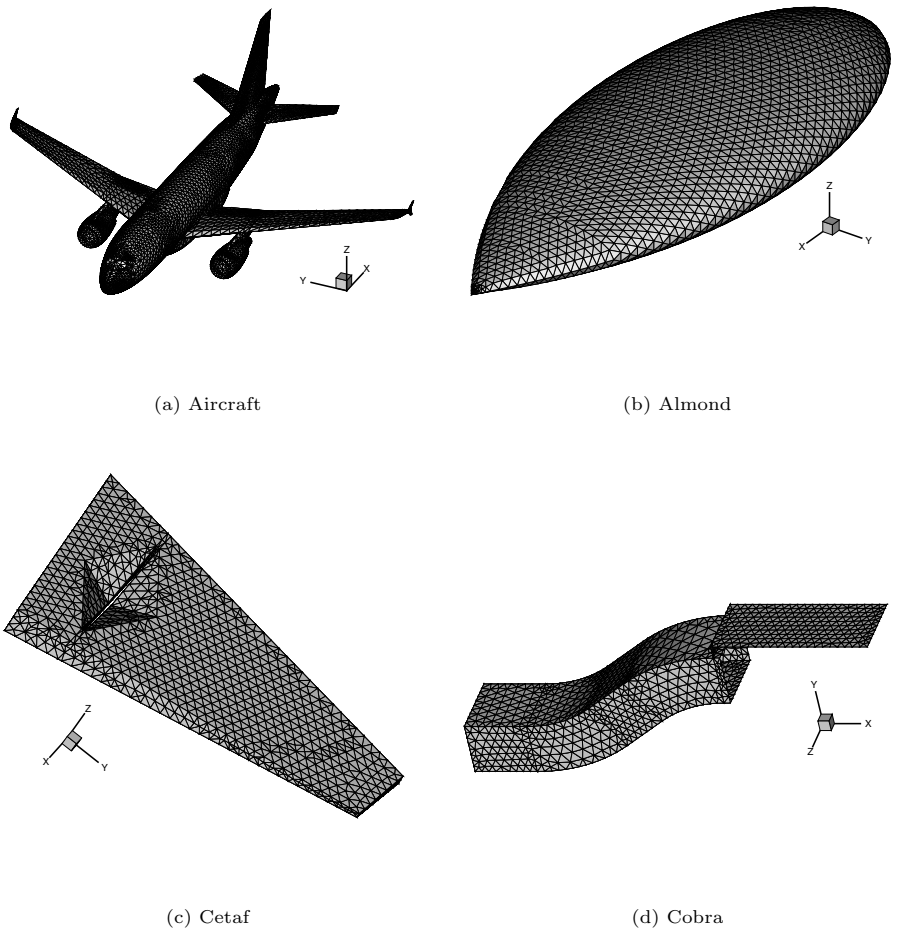
FIG. 2.3. *Interactions in the multilevel FMM. The interactions for the gray boxes are computed directly. We denote by dashed lines the interaction list for the observation box that consists of those cubes that are not neighbors of the cube itself but whose parent is a neighbor of the cube's parent. The interactions of the cubes in the list are computed using the FMM. All the other interactions are computed hierarchically on a coarser level, denoted by solid lines.*

The box-wise decomposition of the domain required by the FMM naturally leads to an a priori pattern selection strategy for M and $\tilde{A}$ using geometric information, that is on the spatial distribution of its degrees of freedom. We will adopt the following criterion: the nonzero structure of the column of the preconditioner associated with a given edge is defined by retaining all the edges within its leaf box and those in one level of neighboring boxes, and the structure for the sparse approximation of the dense coefficient matrix is defined by retaining the entries associated with edges included in the given leaf box as well as those belonging to the two levels of neighbors. The approximate inverse has a sparse block structure; each block is dense and is associated with one leaf box. Indeed the least-squares problems corresponding to edges within the same box are identical because they are defined using the same nonzero structure and the same set of entries of A. It means that we need only compute one QR factorization per leaf box. In our implementation we use two different oct-trees, and thus two different partitionings, to assemble the approximate inverse and for the approximate multipole coefficient matrix. The size of the smallest boxes in the partitioning associated with the preconditioner is a user-defined parameter that can be tuned to control the number of nonzeros computed per column, that is, the density of the preconditioner. According to our criterion, the larger the size of the leaf boxes, the larger the geometric neighborhood that determines the sparsity structure of the columns of the preconditioner. Parallelism can be exploited by assigning disjoint subsets of leaf boxes to different processors and performing the least-squares solutions independently on each processor. We refer to [31] for a complete description of the parallel code that we use.

2.3. Numerical scalability of the preconditioner. In this section, we study the numerical scalability of the Frobenius-norm minimization preconditioner. The optimal behavior would be to get frequency independent numerical behavior that

would result in convergence independent of the problem size. We show results on the following geometries:

- an industrial civil aircraft from a European company (see Figure 2.4(a)), a real-life model problem in an industrial context;
- an almond (see Figure 2.4(b)) that is also a typical test case in electromagnetic simulations;
- a Cetaf (see Figure 2.4(c)), a typical test case in electromagnetic simulations;
- a Cobra (see Figure 2.4(d)), a complex geometry that represents an air intake.



(a) Aircraft (b) Almond

(c) Cetaf (d) Cobra

FIG. 2.4. *Mesh associated with test examples.*

In all the numerical experiments, the surface of the object is always discretized using 10 points per wavelength; larger discretizations are obtained by increasing the frequency of the illuminating wave. In all the experiments, we consider a right preconditioned GMRES method [26] and the threshold for the stopping criterion is set to 10^{-3} on the normwise backward error $\frac{||r||}{||b||}$, where r denotes the residual and b the right-hand side of the linear system. This tolerance is accurate for engineering

purposes, as it enables the correct construction of the radar cross section of the object. The initial guess is the zero vector. All the runs have been performed in single precision on eight processors of a Compaq Alpha server. The Compaq Alpha server is a cluster of symmetric multiprocessors. Each node consists of four DEC Alpha processors (EV 6, 1.3 GFlops peak) that share 512 MB of memory. On that computer, the temporary disk space that can be used by the out-of-core solver is around 189 GB. Among all the possible right-hand sides for each geometry, we have selected those that are the most difficult to solve in order to better illustrate the robustness and the efficiency of our preconditioner.

In Table 1, we show the numerical behavior of the preconditioner observed on the four geometries when the size of the problems is increased. In this table "d" means day, "h" hour, and "m" minute. We see that the numerical behavior of the preconditioner does not scale well with the size of the problems, expecially GMRES(120) on the Cetaf, and there is no convergence on the largest aircraft problems. For GMRES(∞) the increase in the iteration count is less significant, even though on the Cetaf and the aircraft convergence cannot be obtained because we exceed either the memory limits of our computer or the time limit allocated to a single run. From a timing point of view, we see that the solution time of full GMRES is strongly affected by the orthogonalization involved in the Arnoldi procedure. On the Cetaf problem discretized with 531900 points, the number of iterations of GMRES(120) is twice as large with

TABLE 1

Number of matrix-vector products and elapsed time required to converge on the four problems on eight processors of the Compaq machine, except those marked with $^{(k)}$*, that were run on k processors. Tolerance for the iterative solution was* 10^{-3}*. Acronyms: N.A.* $\equiv$ *not available. M.L.E.* $\equiv$ *memory limits exceeded.*

Aircraft						
Size	Density FROB	Time FROB	GMRES(∞)		GMRES(120)	
			Iter	Time	Iter	Time
94704	0.28	11m	746	2h 9m	1956	3h 13m
213084	0.13	31m	973	7h 19m	+2000	7h 56m
591900	0.09	1h 30m	1461	16h 42m$^{(64)}$	+2000	1d 57m
1160124	0.02	3h 24m	M.L.E.$^{(64)}$	N.A.	+2000	> 4d
Almond						
Size	Density FROB	Time FROB	GMRES(∞)		GMRES(120)	
			Iter	Time	Iter	Time
104793	0.19	6m	234	20m	253	17m
419172	0.05	21m	413	2h 44m	571	2h 26m
943137	0.02	49m	454	3h 35m$^{(32)}$	589	5h 55m
Cetaf						
Size	Density FROB	Time FROB	GMRES(∞)		GMRES(120)	
			Iter	Time	Iter	Time
86256	0.18	4m	656	1h 25m	1546	1h 44m
134775	0.11	6m	618	1h 45m	1125	1h 55m
264156	0.06	13m	710	9h	1373	4h 46m
531900	0.03	20m	844	1d 18m	1717	14h 8m
1056636	0.01	37m	+750	+9h$^{(32)}$	+2000	> 1d
Cobra						
Size	Density FROB	Time FROB	GMRES(∞)		GMRES(120)	
			Iter	Time	Iter	Time
60695	0.24	2m	369	26m	516	23m
179460	0.09	7m	353	1h 11h	406	1h 2m

respect to full GMRES, but GMRES(120) is about twice as cheap. Provided we get convergence, the use of a large restart often reduces the solution time even though it significantly deteriorates the convergence. On the Cetaf geometry, the solution time for the GMRES method increases superlinearly for small and medium problems, but nearly quadratically for large problems. On the largest test case, discretized with one million unknowns, unrestarted GMRES does not converge after 750 iterations requiring more than nine hours of computation on 32 processors. The aircraft is very difficult to solve because the mesh has many surface details and the discretization matrices become ill-conditioned. On small and medium problems, the number of GMRES iterations increases with the problem size, and the solution time increases superlinearly. On the largest test case, discretized with one million unknowns, full GMRES exceeds the memory limit on 64 processors. In this case, the use of large restarts (120 in this table) does not enable convergence within 2000 iterations except on a small mesh of size 94704.

We see in Table 1 that our strategy for adjusting the leaf box dimension for increasing frequencies causes the density of the sparse approximate inverse to decrease for increasing problem size. The number of unknowns per box remains constant, but the number of boxes increases, leading to a decrease in density and to poorer preconditioners. In Table 2, we investigate the influence of the density on the quality of the preconditioner on the Cobra using GMRES with a large restart (120). We adopt the same criterion described in section 2 to define the sparsity patterns, but we increase the size of the leaf boxes in the oct-tree associated with the preconditioner. The best trade-off between cost and performance is obtained for a radius of around 0.12 wavelengths, the default value set in the code. If the preconditioner is used to solve systems with the same coefficient matrix and multiple right-hand sides, it might be worth computing more nonzeros if we have sufficient disk space, because the construction cost can be quickly amortized. However, significantly enlarging the density of the preconditioner is not feasible on the largest problems because we would exceed the memory and disk capacity of our computer. As mentioned already, our main focus is on linear systems arising from EFIE. The advantages of this formulation are numerous; in particular, it does not require any hypothesis on the geometry of the objects. We should nevertheless mention that for closed geometries the CFIE (combined field integral equation) can also be used. The linear systems arising from the CFIE formulation are much easier to solve. For instance the solution of the problem associated with the aircraft with 213084 degrees of freedom requires only 129 iterations of unpreconditioned full GMRES, and 22 iterations of preconditioned full GMRES. Furthermore, preconditioned full GMRES converges in 24 iterations on the aircraft with more than a million degrees of freedom. Because the linear systems arising from the CFIE are not challenging from a linear algebra point of view, we do not consider them further in this paper.

Finally, in Table 3, we show the parallel scalability of the implementation of the FMM code [31]. We solve problems of increasing size on a larger number of processors, keeping the number of unknowns per processor constant. It can be seen that the construction of the preconditioner scales perfectly. This is a benefit from keeping the size of the leaf box constant. Its use requires some communication but it still scales reasonably well. The scalability of the matrix-vector product is also satisfactory, as the increase in the elapsed time is due not only to the amount of data exchanges but also to the $\log(n)$ effect of its computational complexity.

TABLE 2
Number of matrix-vector products and elapsed time to build the preconditioner and to solve the problem using GMRES(120) on a Cobra problem of size 179460, varying the parameters controlling the density of the preconditioner. The runs were performed on eight processors of the Compaq machine.

Radius	Density	# mat-vec in GMRES(120)	Construction time	Solution time	Overall time
0.08	0.039	430	218	3562	3780
0.10	0.062	383	294	3008	3302
0.12	0.091	406	434	2954	3388
0.14	0.120	380	640	2873	3513
0.18	0.204	351	1863	3122	4985
0.24	0.358	280	4947	2583	7531

TABLE 3
Tests on the parallel scalability of the code with respect to the construction and application of the preconditioner and to the matrix-vector product operation on problems of increasing size. The test example is the aircraft.

Problem size	Nb procs	Construction time (sec)	Elapsed time precond (sec)	Elapsed time mat-vec (sec)
112908	8	513	0.39	1.77
161472	12	488	0.40	1.95
221952	16	497	0.43	2.15
288300	20	520	0.45	2.28
342732	24	523	0.47	3.10
393132	28	514	0.47	3.30
451632	32	509	0.48	2.80
674028	48	504	0.54	3.70
900912	64	514	0.60	3.80

3. Improving the preconditioner robustness using embedded iterations. The numerical results shown in the previous section indicate that the Frobenius-norm minimization preconditioner tends to become less effective when the problem size increases, especially on difficult problems. By its nature the sparse approximate inverse is inherently local because each degree of freedom is coupled to only a very few neighbors. Because the exact inverse is dense the compact support used to define the preconditioner may not allow an exchange of global information and on large problems the lack of a global approximation may have a severe impact on the convergence. In our implementation, the overall number of computed nonzeros decreases for increasing values of the frequency. When the preconditioner becomes very sparse, information related to the far-field is completely lost. In this case, some suitable mechanism has to be introduced to recover global information for the numerical behavior of the discrete Green's function.

In this section, we describe an embedded iterative scheme, combined with multipole techniques, that is designed to meet the goals of robustness, scalability, and parallelism of the iterative solver. The basic idea is to carry out a few steps of an inner Krylov method for the preconditioning operation. The overall algorithm results in the inner-outer scheme depicted in Figure 3.1. The outer solver must be able to work with variable preconditioners. Amongst various possibilities, we mention FGMRES [25] and GMRES$\star$ [33, p. 91]; this latter reduces to GMRESR [34] when the inner solver is GMRES. The efficiency of the proposed algorithm relies on two main factors: the inner solver has to be preconditioned so that the residual in the inner iterations can be significantly reduced in a few steps, and the matrix-vector

products within the inner and the outer solvers can be carried out with a different accuracy. The motivation that naturally leads us to consider inner-outer schemes is to try to balance the locality of the preconditioner with the use of the multipole approach. Experiments conducted in [18] with inner-outer schemes combined with multipole techniques on the potential equation were not entirely successful. In that case, no preconditioner was used in the inner solver. The desirable feature of using different accuracies for the matrix-vector products is enabled by the use of the FMM. In our scheme, a highly accurate FMM is used within the outer solver, as it governs the final accuracy of the computed solution. A less accurate FMM is used within the inner solver, as it is a preconditioner for the outer scheme; in this respect the inner iterations attempt only to give a rough approximation of the solution and consequently do not require an accurate matrix-vector calculation. In fact, we solve a nearby system for the preconditioning operation that enables us to save considerable computational effort during the iterative process, as the less accurate matrix-vector calculation requires about half of the computing time of the accurate one. In Table 4, we show the average elapsed time in seconds observed on eight processors for a matrix-vector product using the FMM with different accuracy levels.

Outer solver $\longrightarrow$ FGMRES, GMRES$\star$

Do k $= 1, 2, \ldots$
- Matrix-vector product: FMM with high accuracy
- Preconditioning: **Inner solver** (GMRES, TFQMR, $\ldots$)

  Do i $= 1, 2, \ldots$
  - Matrix-vector product: FMM with low accuracy
  - Preconditioning: M_{Frob}

  End Do

End Do

FIG. 3.1. *Inner-outer solution schemes in the FMM context. Sketch of the algorithm.*

TABLE 4

Average elapsed time observed on eight processors in seconds for a matrix-vector product using the FMM with different levels of accuracy.

Geometry	Inner FMM	Outer FMM
Aircraft 94704	3.5	4.8
Aircraft 213084	6.9	11.8
Aircraft 591900	17.4	30.2
Aircraft 1160124	34.5	66.5
Almond 104793	1.8	3.0
Almond 419172	6.2	11.2
Almond 943137	14.3	25.5
Cetaf 86256	1.9	3.7
Cetaf 134775	3.2	5.2
Cetaf 264159	5.6	11.4
Cetaf 539100	11.5	19.3
Cetaf 1056636	20.6	38.2
Cobra 60695	1.1	2.0
Cobra 179460	3.2	5.6

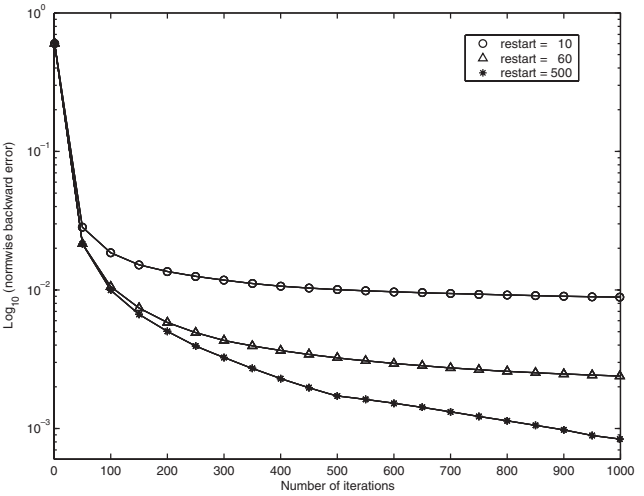


FIG. 3.2. *Convergence history of GMRES for different values of restarts on an aircraft discretized with* 94704 *points.*

3.1. Numerical results. In this section, we conduct experiments using the FGMRES method as the outer solver with an inner GMRES iteration preconditioned with the Frobenius-norm minimization method described in section 2. It is possible to vary the accuracy of the inner solver as the solution converges. However, in this paper, we present preconditioning results for a constant number of inner GMRES iterations. For the inner scheme we do not restart and perform only a prescribed number of full GMRES iterations or equivalently one step of restarted GMRES with the same prescribed restart. For the GMRES and FGMRES methods, we consider the implementations described in [14] and [13], respectively. The convergence history of GMRES depicted in Figure 3.2 for different values of the restart gives us some clues as to the numerical behavior of the proposed scheme. The residual of GMRES tends to decrease very rapidly in the first few iterations independently of the restarts, then decreases much more slowly, and finally tends to stagnate to a value that depends on the restart; the larger the restart, the lower the stagnation value. It suggests that a few steps in the inner solver could be very effective for obtaining a significant reduction of the initial residual. A different convergence behavior was observed when using other Krylov methods as an inner solver, for example, the TFQMR solver [3]. In this case, the residual at the beginning of the convergence is nearly constant or decreases very slowly. The use of TFQMR as an inner solver is ineffective.

In Table 5, we describe the results of experiments on an aircraft with 213084 degrees of freedom using different combinations of restarts for the inner and outer solvers. If the number of inner steps is too small, the preconditioning operation is poor and the convergence slows down, while too large restarts of GMRES tend to increase the overall computational cost but do not cause further reduction of the normwise backward error at the beginning of convergence. The choice of the restart for the outer solver depends to a large extent on the available memory of the target machine and the difficulty of the problem at hand, an issue that is also related to the illuminating direction of the incident wave that defines the right-hand side. Among the various possibilities, we select FGMRES(30) and GMRES(60) on the aircraft and

TABLE 5

Global elapsed time and total number of matrix-vector products required to converge on an aircraft with 213084 *points varying the inner restart parameter. The runs have been performed on eight processors of the Compaq machine.*

Restart FGMRES	Restart GMRES	# total inner mat-vec	# total outer mat-vec	Times
30	40	1960	51	4h 58m
30	50	1900	40	4h 53m
30	60	1920	34	4h 55m
30	70	2030	30	5h 13m
30	80	2240	29	5h 50m

the Cetaf problem, and FGMRES(15) and GMRES(30) on the Cobra and the Almond. These combinations are a good trade-off.

In Table 6 we show the results of experiments on the same geometries that we considered in section 2. We show the number of inner and outer matrix-vector products and the elapsed time needed to achieve convergence using a tolerance of 10^{-3} on eight processors of the Compaq machine. For comparison purposes, we show again in this table the results obtained with the restarted GMRES method. The comparison between the two solvers made in the tables is fair because GMRES has exactly the same storage requirements as the combination FGMRES/GMRES. In fact, for the same restart value, the storage requirements for the FGMRES algorithm are twice that for the standard GMRES algorithm, because it also stores the preconditioned vectors of the Krylov basis. It can be seen that the combination FGMRES/GMRES remarkably enhances the robustness of the preconditioner especially on large problems. The increase in the number of outer iterations is fairly modest except on the largest aircraft test cases. Nevertheless, the scheme is the only one that enables us to get the solution of this challenging problem since classical restarted GMRES does not converge and full GMRES exceeds the memory of our computer. Similarly, on the Cetaf discretized with one million points, the embedded scheme enables us to get convergence in 22 outer iterations, whereas GMRES(120) does not converge in 2000 iterations. The savings in time is also noticeable. The gain ranges from two to four depending on the geometry and tends to become larger when the problem size increases. On the Cobra and the Almond test cases, the embedded solver reduces not only the solution time but also the memory used, as FGMRES(15)/GMRES(30) is faster than GMRES(120) (see Table 1).

Similarly to what has been observed with GMRES, using a full FGMRES in the outer loop enables us to reduce the number of outer iterations but often results in a longer solution time. In Table 7 we show results obtained with full FGMRES and compare this with the full GMRES method. It can also be seen that the savings in time due to the use of the inner-outer scheme with respect to GMRES is also strongly related to the reduction in the number of dot product calculations since there are far fewer basis vectors to be orthogonalized. This gain is particularly visible on the aircraft test example with 213084 unknowns. The time spent in the FMM is longer for the FGMRES/GMRES solver, about 3 hours and 47 minutes (i.e., $33\times11.8+1920\times6.9$; see Tables 7 and 4), and it is about 3 hours and 11 minutes for GMRES. In that example, the embedded scheme is about 2 hours faster than regular GMRES. In this latter case, we have to orthogonalize up to 973 basis vectors while in the inner-outer scheme we do not have to orthogonalize a basis larger than 60. This is another illustration that the orthogonalization procedure is extremly time consuming.

TABLE 6

Number of matrix-vector products and elapsed time required to converge on eight processors of the Compaq machine. The tests were run on eight processors of the Compaq machine, except those marked with $^{(k)}$, which were run on k processors.

Aircraft						
Size	Density FROB	Time FROB	GMRES(120)		FGMRES(30)/GMRES(60)	
			Iter	Time	Iter	Time
94704	0.28	11m	1956	3h 13m	27+1560	2h 14m
213084	0.13	31m	+2000	7h 56m	34+1920	5h
591900	0.09	1h 30m	+2000	1d 57m	57+3300	1d 9h 45m
1160124	0.02	3h 24m	+2000	> 4d	51+2940	16h 41m$^{(64)}$
Almond						
Size	Density FROB	Time FROB	GMRES(60)		FGMRES(15)/GMRES(30)	
			Iter	Time	Iter	Time
104793	0.19	6m	302	19m	11+300	14m
419172	0.05	21m	+2000	> 10h	20+540	1h 24m
943137	0.02	49m	1633	6h 36m$^{(16)}$	22+600	3h 32m
Cetaf						
Size	Density FROB	Time FROB	GMRES(120)		FGMRES(30)/GMRES(60)	
			Iter	Time	Iter	Time
86256	0.18	4m	1546	1h 43m	17+ 960	55m
134775	0.11	6m	1125	1h 55m	15+ 840	1h 19m
264156	0.06	13m	1373	4h 46m	17+ 960	2h 22m
531900	0.03	20m	1717	14h 8m	19+1080	6h
1056636	0.01	37m	+2000	> 1d	22+1260	14h
Cobra						
Size	Density FROB	Time FROB	GMRES(60)		FGMRES(15)/GMRES(30)	
			Iter	Time	Iter	Time
60695	0.24	2m	708	29m	24+660	18m
179460	0.09	7m	433	48m	20+540	42m

TABLE 7

Number of matrix-vector products and elapsed time required to converge on eight processors of the Compaq machine. The tests were run on eight processors of the Compaq machine, except those marked with $^{(k)}$, that were run on k processors.

Aircraft				
Size	GMRES(∞)		FGMRES(∞,60)	
	Iter	Time	Iter	Time
94704	746	2h 9m	23+1320	2h 30m
213084	973	7h 19m	30+1740	6h 11m
591900	1461	16h 42m$^{(64)}$	43+2520	12h$^{(32)}$
1160124	M.L.E.$^{(64)}$	> 1d	43+2520	14 h 28m$^{(64)}$
Cobra				
Size	GMRES(∞)		FGMRES(∞,60)	
	Iter	Time	Iter	Time
60695	369	26m	21+600	17m
179460	353	1h 11m	18+510	38m

In Figure 3.3 we show the typical convergence history of the FGMRES/GMRES and GMRES solvers on a large Cetaf problem discretized with 264156 points. We depict the convergence curve as a function of the elapsed time. It can be seen that the embedded solver clearly outperforms the single GMRES scheme. The FGMRES/GMRES solver continues to succeed in reducing the residual norm while GMRES converges slowly. Similar behavior is also reported for simpler model examples in [29].

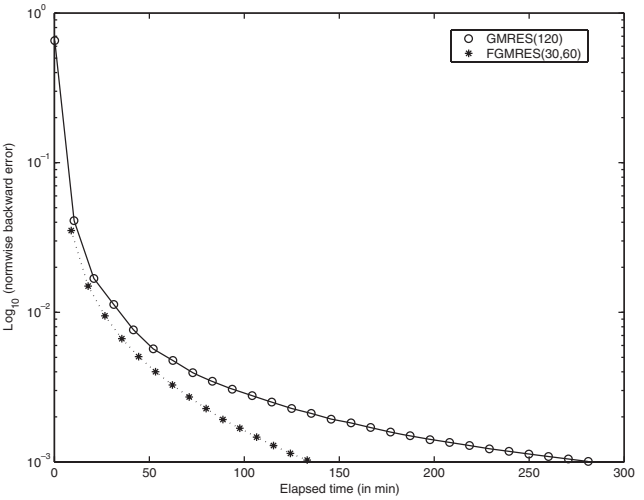


FIG. 3.3. *Convergence history as a function of elapsed time of GMRES versus FGMRES on the Cetaf problem discretized with* 264156 *points.*

For all the numerical experiments reported so far, we consider a discretization using ten points per wavelength. In that context, we vary the size of the linear system by varying the frequency of the illuminating wave and we consider objects with a fixed dimension expressed in meters. Consequently, each linear system corresponds to the simulation of a different physical phenomenon. In contrast, in Table 8 we solve the same physical phenomenon. We vary the size of the linear systems by varying the number of points per wavelength, keeping the frequency of the illuminating wave constant as well as the dimension of the object. In that table, λ denotes the wavelength and h the average mesh size. Because the leaf boxes used to build the preconditioner have a size that is a fraction of λ, the preconditioner becomes denser as the problem size increases. This observation partially explains the fact that the number of iterations of GMRES(60) slightly decreases when the the size of the linear system increases. This is an opposite behavior to what we observed in Table 6, where the density of the preconditioner decreases when the size of the problem is increased. We also observe that the behavior of FGMRES(15)/GMRES(30) is not affected by the mesh refinement.

4. Conclusions. In this paper we have described an effective and inherently parallel approximate inverse preconditioner, based on a Frobenius-norm minimization preconditioner, that can easily be implemented in an FMM code. We have studied the numerical scalability of our method for the solution of very large and dense linear systems of equations arising from real-life appplications in electromagnetism. The locality of the preconditioner on large problems can be significantly improved by embedding it within inner-outer solvers. We have described an embedded iterative scheme based on the GMRES method, implemented in a multipole context with different levels of accuracy for the matrix-vector products in the inner and outer loops. We have shown that the combination FGMRES/GMRES can effectively enhance the robustness of the preconditioner and reduce significantly the computational cost and the storage requirements for the solution of large problems. One could apply this idea recursively and embed several FGMRES schemes with decreasing FMM accuracy down to the

TABLE 8

Number of matrix-vector products and elapsed time required to converge on eight processors of the Compaq machine when the discretization is refined. The tests were run on eight processors of the Compaq machine, except those marked with $^{(k)}$, that were run on k processors.

Almond - length = 2.5 m - frequency 2.6 GHz					
h	Size of linear system	GMRES(60)		FGMRES(15,30)	
		Iter	Time	Iter	Time
$\lambda/6$	38724	345	15m	12+330	11m
$\lambda/10$	104793	302	19m	11+300	14m
$\lambda/15$	240294	281	1h 21m	10+270	1h 06m
$\lambda/20$	426750	271	2h 36m$^{(16)}$	10+270	2h 45m$^{(16)}$

lowest accuracy in the innermost GMRES. However, in our work, we consider only a two-level scheme. Most of the experiments shown in this paper require a huge amount of computation and storage, and they often reach the limits of our target machine in terms of memory and disk storage. To give an idea of how large these simulations are, the solution of systems with one million unknowns using a direct method would require 8 Tbytes of storage and 37 years of computation on one processor of the target computer (assuming the computation runs at peak performance). Such simulations are nowadays feasible thanks to the use of iterative methods and can be integrated in the design processes where the bottleneck moves from the simulation to the pre- and postprocessing of the results, as the tools are not yet available to easily manipulate meshes with millions of degrees of freedom. Finally we mention that the techniques described in this paper have recently been applied succesfully to large simulations in computational acoustics and extended to the computation of monostatic radar cross sections [16].

Acknowledgments. We would like to thank Guillaume Alléon from EADS-CCR for providing us with some test examples considered in this paper. We would also like to acknowledge the computing support provided by the CINES (Centre Informatique National de l'Enseignement Supérieur) and the CEA (Commissariat à l'Énergie Atomique) that enabled us to access their computing facilities.

REFERENCES

[1] G. ALLÉON, M. BENZI, AND L. GIRAUD, *Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics*, Numer. Algorithms, 16 (1997), pp. 1–15.

[2] M. W. BENSON, *Iterative Solution of Large Scale Linear Systems*, Master's thesis, Lakehead University, Thunder Bay, Canada, 1973.

[3] B. CARPENTIERI, *Sparse Preconditioners for Dense Linear Systems from Electromagnetic Applications*, Ph.D. thesis, CERFACS, Toulouse, France, 2002.

[4] B. CARPENTIERI, I. S. DUFF, AND L. GIRAUD, *Sparse pattern selection strategies for robust Frobenius-norm minimization preconditioners in electromagnetism*, Numer. Linear Algebra Appl., 7 (2000), pp. 667–685.

[5] B. CARPENTIERI, I. S. DUFF, L. GIRAUD, AND M. MAGOLU MONGA MADE, *Sparse symmetric preconditioners for dense linear systems in electromagnetism*, Numer. Linear Anal. Appl., 11 (2004), pp. 753–771.

[6] K. CHEN, *An analysis of sparse approximate inverse preconditioners for boundary integral equations*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1058–1078.

[7] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.
[8] E. Darve, *The fast multipole method* I: *Error analysis and asymptotic complexity*, SIAM J. Numer. Anal., 38 (2000), pp. 98–128.
[9] E. Darve, *The fast multipole method: Numerical implementation*, J. Comput. Phys., 160 (2000), pp. 195–240.
[10] C. de Boor, *Dichotomies for band matrices*, SIAM J. Numer. Anal., 17 (1980), pp. 894–907.
[11] S. Demko, *Inverses of band matrices and local convergence of spline projections*, SIAM J. Numer. Anal., 14 (1977), pp. 616–619.
[12] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Math. Comput., 43 (1984), pp. 491–499.
[13] V. Frayssé, L. Giraud, and S. Gratton, *A Set of Flexible-GMRES Routines for Real and Complex Arithmetics*, Technical Report TR/PA/98/20, CERFACS, Toulouse, France, 1998. Public domain implementation available at http://www.cerfacs.fr/algor/Softs/FGMRES.
[14] V. Frayssé, L. Giraud, S. Gratton, and J. Langou, *A Set of GMRES Routines for Real and Complex Arithmetics on High Performance Computers*, Technical Report TR/PA/03/3, CERFACS, Toulouse, France, 2003. Public domain implementation available at http://www.cerfacs.fr/algor/Softs/GMRES.
[15] P. O. Frederickson, *Fast Approximate Inversion of Large Sparse Linear Systems*, Math. Report 7, Lakehead University, Thunder Bay, Canada, 1975.
[16] L. Giraud, J. Langou, and G. Sylvand, *On the Parallel Solution of Large Industrial Wave Propagation Problems*, Technical Report TR/PA/04/52, CERFACS, Toulouse, France, 2004. Preliminary version of the paper to appear in J. Comput. Acoust.
[17] A. Grama, V. Kumar, and A. Sameh, *Parallel matrix-vector product using approximate hierarchical methods*, in Proceedings of the 1995 ACM/IEEE Supercomputing Conference, December 3–8, 1995, San Diego Convention Center, San Diego, CA, Sidney Karin, ed., ACM, New York, IEEE Computer Society Press, Piscataway, NJ, 1995. CD-ROM.
[18] A. Grama, V. Kumar, and A. Sameh, *Parallel hierarchical solvers and preconditioners for boundary element methods*, SIAM J. Sci. Comput., 20 (1998), pp. 337–358.
[19] L. Greengard and W. Gropp, *A parallel version of the fast multipole method*, Comput. Math. Appl., 20 (1990), pp. 63–71.
[20] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.
[21] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
[22] L. Yu. Kolotilina, *Explicit preconditioning of systems of linear algebraic equations with dense matrices*, J. Sov. Math., 43 (1988), pp. 2566–2573.
[23] J. Lee, J. Zhang, and C.-C. Lu, *Sparse Inverse Preconditioning of Multilevel Fast Multipole Algorithm for Hybrid Integral Equations in Electromagnetics*, Technical Report 363-02, Department of Computer Science, University of Kentucky, Lexington, KY, 2002.
[24] J. Lee, J. Zhang, and C.-C. Lu, *Incomplete LU preconditioning for large scale dense complex linear systems from electromagnetic wave scattering problems*, J. Comput. Phys., 185 (2003), pp. 158–175.
[25] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
[26] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[27] A. R. Samant, E. Michielssen, and P. Saylor, *Approximate Inverse Based Preconditioners for* 2*D Dense Matrix Problems*, Technical Report CCEM-11-96, University of Illinois, Urbana-Champaign, 1996.
[28] K. Sertel and J. L. Volakis, *Incomplete LU preconditioner for FMM implementation*, Microwave Opt. Tech. Lett., 26 (2000), pp. 265–267.
[29] V. Simoncini and D. B. Szyld, *Flexible inner-outer Krylov subspace methods*, SIAM J. Numer. Anal., 40 (2003), pp. 2219–2239.
[30] J. Song, C.-C. Lu, and W. C. Chew, *Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects*, IEEE Trans. Antennas and Propagation, 45 (1997), pp. 1488–1493.
[31] G. Sylvand, *La Méthode Multipôle Rapide en Electromagnétisme: Performances, Parallélisation, Applications*, Ph.D. thesis, Ecole Nationale des Ponts et Chaussées, Marne-la-Vallée, France, 2002.

[32] W.-P. Tang, *Schwartz Splitting and Template Operators*, Ph.D. thesis, Computer Science Department, Stanford University, Stanford, CA, 1987.
[33] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, UK, 2003.
[34] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[35] S. A. Vavasis, *Preconditioning for boundary integral equations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 905–925.
[36] F. Zhao and S. L. Johnsson, *The parallel multipole method on the connection machine*, SIAM J. Sci. Stat. Comput., 12 (1991), pp. 1420–1437.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 793–808
© 2005 Society for Industrial and Applied Mathematics

MULTIDIMENSIONAL EXPONENTIAL TIMESTEPPING WITH BOUNDARY TEST*

KALVIS M. JANSONS† AND G. D. LYTHE‡

Abstract. Exponential timestepping algorithms are efficient for exit-time problems because a boundary test can be performed at the end of each timestep, giving high-order convergence in numerical evaluation of mean exit times. Successive time increments are independent random variables with an exponential distribution. We show how to perform exact timestepping for Brownian motion in more than one dimension and consider hitting times of curved surfaces. Taking as examples the exit times from a circle, from a ball, from an ellipse, and from the region bounded by the two lines of a hyperbola, we report the results of numerical experiments performed with the algorithms developed in this work.

Key words. stochastic calculus, numerical methods, multidimensional diffusion, exit time, curved boundaries

AMS subject classifications. 60-08, 65C30

DOI. 10.1137/040612865

1. Introduction.

One numerical realization of a stochastic process is a series of values generated successively. Under standard algorithms [1], the values correspond to times separated by a fixed, or at least deterministic, duration Δt. Under exponential timestepping [2], the times corresponding to numerical values are instead separated by independent realizations of a random variable $\boldsymbol{\delta t}$, where

$$\mathcal{P}[\boldsymbol{\delta t} > t] = \exp(-\lambda t) \tag{1.1}$$

and λ is a constant. Thus $\langle \boldsymbol{\delta t} \rangle = \lambda^{-1}$, where angled brackets denote mean over realizations. Random variables are set in boldface.

The increment of Brownian motion in one dimension with diffusivity D over a fixed time interval Δt is a Gaussian random variable with mean zero and variance $2D\Delta t$. Thus exact numerical updates require realizations of Gaussian random variables. If the time interval is not fixed but distributed according to (1.1), the increment of Brownian motion has a symmetric exponential distribution [2]. Exact updates under exponential timestepping therefore require realizations of random variables with an exponential distribution, which are easily generated as $-\log \mathbf{u}$, where $\mathbf{u}$ is distributed uniformly on $(0, 1)$. In this work, we construct the exact update for Brownian motion in n dimensions by separating the increment into a modulus and an angle.

The particular numerical problem we focus on in this paper is the measurement of exit or passage times, where the quantity of interest is the first time that a stochastic process attains a given value or exits a region [3, 4, 5, 6, 7, 8]. A numerical scheme for such a task has three components. The first component is an algorithm for updating the stochastic process so that realizations can be generated. The second is a criterion for deciding when the path hits the boundary, bringing the realization to an end. The

*Received by the editors August 4, 2004; accepted for publication (in revised form) May 17, 2005; published electronically November 15, 2005.
http://www.siam.org/journals/sisc/27-3/61286.html
†Department of Mathematics, University College London, Gower Street, London WC1E 6BT, England (multidim@randomideas.org, http://randomideas.org).
‡Department of Applied Mathematics, University of Leeds, Leeds LS2 9JT, England (expmultidim @randomideas.org, http://randomideas.org).

third is a relationship between the mean number of numerical timesteps per realization and the mean exit time.

The simplest criterion for bringing a realization to an end is that the process be outside the region at the end of a timestep. Numerical schemes using this criterion overestimate the mean exit time because of the strictly positive probability that an excursion is made outside the region during a timestep that starts and ends inside the region [2, 9, 10, 11, 12, 13, 14, 15]. To counteract this systematic error, a boundary test can be performed after each timestep, in which with probability p_i the realization is brought to an end, where p_i is an estimate for the conditional probability, given the starting and ending positions of a timestep, that an excursion out of the region has been made during the timestep [2, 10, 11, 12, 13, 14, 15]. Such tests have been shown to reduce the systematic error in mean exit times from being proportional to $\Delta t^{1/2}$ to being proportional to Δt when Δt is fixed [13]. Under exponential timestepping, the corresponding improvement is from an error proportional to $\langle \boldsymbol{\delta t} \rangle^{1/2}$, i.e., to the square root of the mean duration of a timestep, to an error proportional to $\langle \boldsymbol{\delta t} \rangle$ [2]. Alternatively, under the walk-on-spheres method, a path is constructed as a sequence of jumps from ball centers to ball surfaces and an absorption layer is defined close to the boundary [16, 17]. Realizations are brought to an end when they enter the layer.

The advantage of exponential timestepping is that expressions for the density of increments and for the conditional hitting probabilities required to devise boundary tests have a simple form. They are the Laplace transform of the corresponding quantities under fixed-duration timesteps and can be derived intuitively using the equivalence between the end of an exponential timestep and a mark on the path made by an independent Poisson point process with constant rate. In this work, we devise boundary tests under exponential timestepping in more than one dimension. Flat boundaries are equivalent to the one-dimensional case and thus can be dealt with exactly. Approximate boundary tests for smooth boundaries are derived and are valid when the radius of curvature is much larger than typical displacements in one timestep.

Armed with a criterion for bringing realizations to a stop, we generate large numbers of them and estimate the mean exit time from the mean number of timesteps per realization. In the case of timesteps of fixed duration Δt, the elapsed time until the [illegible] of the timestep at which the boundary was deemed hit is known exactly. The time [illegible] at which the boundary is hit is not known and must be estimated to obtain a mean exit time accurate to better than Δt [18]. In the case of exponential timestepping, the duration of any one timestep is not generated or known. Despite this apparent complication, evaluation of the mean exit time is straightforward under exponential timestepping because an update can be thought of as a jump to the next mark on the Brownian path made by an independent Poisson point process with rate λ (Figure 1.1). If N is the mean number of marks [illegible] a random time $\mathbf{H}$ that is independent of the marking process, then

$$\langle \mathbf{H} \rangle = \lambda N, \tag{1.2}$$

whatever the distribution of $\mathbf{H}$. Suppose there are M realizations and the boundary is deemed to have been hit during the n_jth timestep of realization j. The numerical estimate of the mean exit time is $M^{-1} \sum_{j=1}^{M} (n_j - 1)$.

Section 2 is a summary of basic results in one dimension and multiple dimensions. In section 3, we consider hitting times of curved boundaries. Exponential timestepping methods in n dimensions are described in section 4. Section 5 contains the results

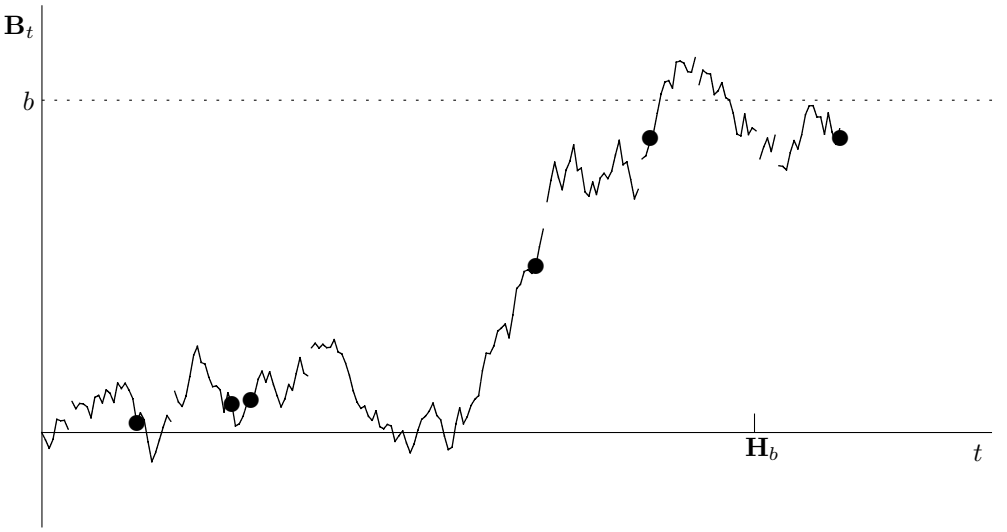


FIG. 1.1. *One realization is depicted. Exponential timesteps are jumps from one black point to the next. The values of the process at these times, but not the times themselves, are generated. The aim is to estimate the mean of* $\mathbf{H}_b$ *from many realizations.*

of numerical experiments with the new algorithms described here and with the Euler algorithm.

2. Basic results and notation. An n-dimensional Brownian motion $\mathbf{B}$ with diffusivity D has Cartesian components that are independent Brownian motions with diffusivity D. $\mathbf{B}_0 = 0$ and the probability density function of $\mathbf{B}_t$ for any fixed $t > 0$ is

$$p^{(n)}(x,t) = (4\pi Dt)^{-n/2} \exp\left(-|x|^2/4Dt\right), \tag{2.1}$$

where $x = (x_1, x_2, \ldots, x_n)$ and $|x|^2 = \sum_{i=1}^n x_i^2$. Consider an exponential timestep, started at 0, with mean duration λ^{-1}. The density of $\mathbf{B}_{\boldsymbol{\delta t}}$ is [23]

$$\begin{aligned} p_\nu^{(n)}(x) &= \int_0^\infty \lambda \exp(-\lambda t) p^{(n)}(x,t)\, \mathrm{d}t \\ &= \frac{\nu^n}{2\pi} \left(2\pi\nu|x|\right)^{1-n/2} K_{n/2-1}\left(\nu|x|\right), \end{aligned} \tag{2.2}$$

where K_m is the modified Bessel function of the second kind of order m and

$$\nu = (\lambda/D)^{1/2}. \tag{2.3}$$

In one dimension, let $\mathbf{H}_b = \inf\{t > 0 : \mathbf{B}_t = b\}$ and let $u(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{P}[\mathbf{H}_b < t]$. Then [3, 7, 2]

$$u(t) = |b| \left(4\pi Dt^3\right)^{-1/2} \exp(-b^2/4Dt). \tag{2.4}$$

The probability that $\mathbf{H}_b < \boldsymbol{\delta t}$ is given by

$$\mathcal{P}[\mathbf{H}_b < \boldsymbol{\delta t}] = \mathrm{e}^{-\nu|b|}. \tag{2.5}$$

If the value at the end of the timestep is known, we have the following conditional probability:

$$\mathcal{P}[\mathbf{H}_b < \boldsymbol{\delta t}\,|\mathbf{B}_{\boldsymbol{\delta t}} = z\,] = \mathrm{e}^{-2\nu|b-\tilde{z}|}, \tag{2.6}$$

where $\tilde{z}$ is the closer of 0 and z to b. The one-dimensional results (2.5) and (2.6) hold for flat surfaces in any number of dimensions. For curved surfaces it remains true that the probability of hitting a surface during a timestep is significant when the starting or ending value is within a layer of width ν^{-1} of the surface.

2.1. Hitting circles and spheres. Exact results are also known for exit times from n-dimensional balls. Let

$$\mathbf{h}_R^{(n)}(r) = \inf\{t > 0 : |\mathbf{B}_t| = R \,|\, |\mathbf{B}_0| = r\,\}, \tag{2.7}$$

the first time $t > 0$ that $|\mathbf{B}_t| = R$, starting from $|\mathbf{B}_0| = r$. Then [8, 19]

$$\mathcal{P}[\mathbf{h}_R^{(n)}(r) < \boldsymbol{\delta t}] = \begin{cases} \left(\dfrac{r}{R}\right)^{1-n/2} \dfrac{I_{n/2-1}(\nu r)}{I_{n/2-1}(\nu R)}, & r < R, \\ \left(\dfrac{r}{R}\right)^{1-n/2} \dfrac{K_{n/2-1}(\nu r)}{K_{n/2-1}(\nu R)}, & r \geq R, \end{cases} \tag{2.8}$$

where I_m is the modified Bessel function of the second kind of order m.

Now consider the density of increments of the modulus of Brownian motion in n dimensions. Define

$$b^{(n)}(r, r_0) = \frac{\mathrm{d}}{\mathrm{d}r}\mathcal{P}[|\mathbf{B}_{\boldsymbol{\delta t}}| < r \,|\, |\mathbf{B}_0| = r_0\,]. \tag{2.9}$$

Then [19]

$$b^{(n)}(r, r_0) = \begin{cases} \nu^2 r^{n/2} r_0^{-n/2+1} I_{n/2-1}(\nu r_0) K_{n/2-1}(\nu r), & r_0 < r, \\ \nu^2 r^{n/2} r_0^{-n/2+1} K_{n/2-1}(\nu r_0) I_{n/2-1}(\nu r), & r_0 \geq r. \end{cases} \tag{2.10}$$

An advantage of the exponential duration of a timestep is that many conditional probabilities factorize. In particular, using (2.8) and (2.10) gives

$$\begin{aligned} \mathcal{P}[\mathbf{h}_R^{(n)}(r_0) < \boldsymbol{\delta t}\,|\,|\mathbf{B}_{\boldsymbol{\delta t}}| = r_1\,] &= \mathcal{P}[\mathbf{h}_R^{(n)}(r_0) < \boldsymbol{\delta t}] \left(b^{(n)}(r_1, r_0)\right)^{-1} b^{(n)}(r_1, R) \\ &= \begin{cases} \dfrac{I_{n/2-1}(\nu(r_0 \vee r_1))}{I_{n/2-1}(\nu R)} \dfrac{K_{n/2-1}(\nu R)}{K_{n/2-1}(\nu(r_0 \vee r_1))}, & (r_0 \vee r_1) < R, \\ \dfrac{I_{n/2-1}(\nu R)}{I_{n/2-1}(\nu(r_0 \wedge r_1))} \dfrac{K_{n/2-1}(\nu(r_0 \wedge r_1))}{K_{n/2-1}(\nu R)}, & (r_1 \wedge r_0) \geq R, \end{cases} \end{aligned} \tag{2.11}$$

where $(a \vee b) = \max(a, b)$ and $(a \wedge b) = \min(a, b)$.

It is illuminating to compare these results to (2.5) and (2.6). Taking the limit of (2.8) as $\nu R \to \infty$ with $S = \nu(R - r)$ fixed gives

$$\mathcal{P}[\mathbf{h}_R^{(n)}(r) < \boldsymbol{\delta t}] = \mathrm{e}^{-|S|}\left(1 - \frac{1}{2}(n-1)(\nu R)^{-1} S + \mathcal{O}((\nu R)^{-2})\right). \tag{2.12}$$

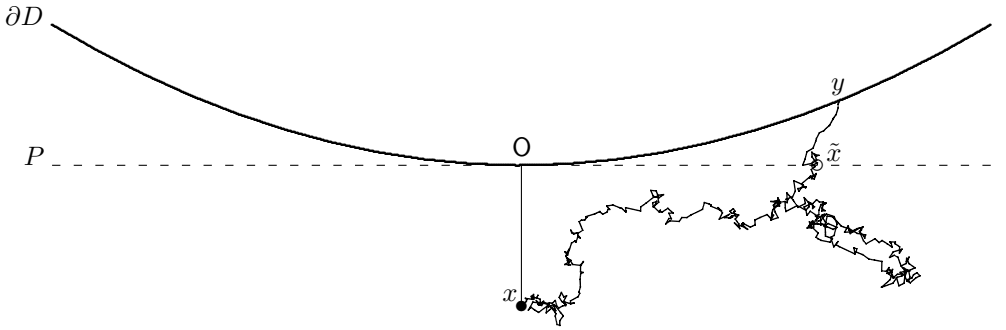


FIG. 3.1. *The initial position x is marked with a filled circle. Before hitting the curved surface ∂D, the path of the stochastic process hits the surface P, tangent to the point on ∂D closest to x. The distance from x to O is denoted by s.*

In (2.11), let r be the closer of r_0 and r_1 to R. In the same limit,

(2.13)
$$\mathcal{P}[\mathbf{h}_R^{(n)}(r_0) < \boldsymbol{\delta t} \,|\, |\mathbf{B}_{\boldsymbol{\delta t}}| = r_1\,] = \exp(-2|S|)\left(1 - \frac{n^2-4n+3}{(2\nu R)^2}S + \mathcal{O}((\nu R)^3)\right).$$

3. Curved boundaries. Consider a timestep started at a point x in a concave domain in $\mathbb{R}^n$ with boundary ∂D (Figure 3.1). We place O, the origin of coordinates in $\mathbb{R}^n$, at the point on ∂D nearest to x; define the x_n direction to be along the line $x\mathsf{O}$; and let s denote the distance from x to O. Let P be the $(n-1)$-dimensional surface tangent to ∂D at O. Let $\mathbf{H}_P^{(n)}(x)$ be the first hitting time of P,

$$\mathbf{H}_P^{(n)}(x) = \inf\{t > 0 : \mathbf{B}_t \in P \,|\mathbf{B}_0 = x\}. \tag{3.1}$$

We denote the value of $\mathbf{B}$ at the time $\mathbf{H}_P^{(n)}(x)$ by $\tilde{x}$ and write $\tilde{x} = (\tilde{x}^\perp, 0)$ where $\tilde{x}^\perp = (\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{n-1})$.

Independent of n, we have

$$\mathcal{P}[\mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t}] = \mathrm{e}^{-\nu s}. \tag{3.2}$$

Consider the subset of paths for which $\mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t}$. For an event Θ, let

$$\mathcal{S}[\Theta] = \mathcal{P}[\Theta \,|\, \mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t}] = \frac{\mathcal{P}[\Theta, \mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t}]}{\mathcal{P}[\mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t}]}. \tag{3.3}$$

Let $q_P(x^\perp)$ be the density of arrival positions on the surface P, i.e.,

$$q_P(x^\perp) = \frac{\partial}{\partial x_1}\frac{\partial}{\partial x_2}\cdots\frac{\partial}{\partial x_{n-1}}\mathcal{S}[\tilde{x}_1 < x_1, \tilde{x}_2 < x_2, \ldots, \tilde{x}_{n-1} < x_{n-1}]. \tag{3.4}$$

The density of arrival times on P, conditioned on being earlier than $\boldsymbol{\delta t}$, is

$$u_\nu(t) = \frac{\mathrm{d}}{\mathrm{d}t}\mathcal{S}[\mathbf{H}_P^{(n)}(x) < t]. \tag{3.5}$$

Using (2.4), we find that

$$u_\nu(t) = s\exp(\nu s)\left(2\pi t^3\right)^{-1/2}\exp\left(-s^2/2t - \nu^2 t/2\right). \tag{3.6}$$

An expression for $q_P(x^\perp)$ is obtained by integrating $u_\nu(t)$ over the density of an $(n-1)$-dimensional Brownian motion. Explicitly,

$$q_P(x^\perp) = \int_0^\infty u_\nu(t)(2\pi t)^{-(n-1)/2}\exp\left(\frac{-|x^\perp|^2}{2t}\right)\mathrm{d}t \tag{3.7}$$

$$= 2s\exp(\nu s)\left(\frac{\nu}{2\pi}\right)^{n/2}\left(s^2+|x^\perp|^2\right)^{-n/4}K_{n/2}\left(\nu(s^2+|x^\perp|^2)^{1/2}\right). \tag{3.8}$$

Note that, for $1 \le i \le n-1$,

$$\int x_i^2 q_P(x^\perp)\,\mathrm{d}x^\perp = \int_{-\infty}^{\infty} x^2(s^2+x^2)^{-1/4}K_1(\nu(s^2+x^2)^{1/2})\mathrm{d}x = \frac{s}{\nu}. \tag{3.9}$$

Timesteps during which the boundary is hit typically have $\nu s = \mathcal{O}(1)$. Thus the typical spread of hitting points on P is such that $\tilde{x}_i = \mathcal{O}(\nu^{-1})$.

Now we consider the probability of hitting the curved boundary ∂D. Similarly to (3.1), let

$$\mathbf{H}_{\partial D}^{(n)}(x) = \inf\{t>0 : \mathbf{B}_t \in \partial D\,|\mathbf{B}_0 = x\}. \tag{3.10}$$

For the geometry chosen for Figure 3.1, $\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}] < \mathcal{P}[\mathbf{H}_{P}^{(n)}(x) < \boldsymbol{\delta t}]$. We invoke the following "memoryless" property of exponential timesteps: [illegible] ∂D [illegible] $\tilde{x} \in P$ [illegible] ∂D [illegible] $\tilde{x}$. Thus

$$\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}] = \mathrm{e}^{-\nu s}\int q_P(\tilde{x}^\perp)\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(\tilde{x}) < \boldsymbol{\delta t}]\,\mathrm{d}\tilde{x}^\perp. \tag{3.11}$$

When the boundary is flat, the integral in (3.11) is unity and we regain (3.2). As $\nu \to \infty$ with νs fixed, a smooth surface appears almost flat on the scale of typical displacements in one timestep and there is a small correction to (3.2).

Given a point y on ∂D, let $y = (y^\perp, y_n)$, where $y^\perp = (y_1, y_2, \ldots, y_{n-1})$. The directions of the axes are chosen so that the surface ∂D is parametrized as $y_n = y_n(y^\perp)$, where $y_n(y^\perp)$ has the Taylor series [21]

$$y_n(y^\perp) = \frac{1}{2}\left(\kappa_1 y_1^2 + \kappa_2 y_2^2 + \cdots + \kappa_{n-1}y_{n-1}^2\right) + \cdots. \tag{3.12}$$

Let $\kappa = \max_i \kappa_i$. The statement that ν is sufficiently large so that the maximum radius of curvature of ∂D is smaller than typical displacements in one timestep is equivalent to taking $\kappa/\nu \ll 1$.

The probability (3.2) is the lowest-order estimate of $\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}]$, the probability of hitting a curved boundary ∂D before the end of a timestep started at x. At next order, $\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(\tilde{x}) < \boldsymbol{\delta t}]$ in (3.11) is approximated by the probability of hitting the tangent plane through the point on ∂D closest to $\tilde{x}$ [22]. That is,

$$\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(\tilde{x}) < \boldsymbol{\delta t}] \to \exp(-\nu h(x^\perp)), \tag{3.13}$$

where $h(\tilde{x}^{\perp})$ is the distance from $\tilde{x} = (\tilde{x}^{\perp}, 0)$ to ∂D. Taking $\nu s = \mathcal{O}(1)$ and using (3.9), we can expand the exponential as follows:

$$\exp\left(-\nu h(x^{\perp})\right) = 1 - \frac{1}{2}\nu\left(\kappa_1 x_1^2 + \kappa_2 x_2^2 + \cdots + \kappa_{n-1} x_{n-1}^2\right) + \mathcal{O}\left(\left(\frac{\kappa}{\nu}\right)^2\right). \tag{3.14}$$

Thus, as $\kappa/\nu \to 0$ with νs fixed,

$$\mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}] = \mathrm{e}^{-\nu s}\left(1 - \frac{1}{2}s(\kappa_1 + \kappa_2 + \cdots + \kappa_{n-1}) + \mathcal{O}\left(\left(\frac{\kappa}{\nu}\right)^2\right)\right). \tag{3.15}$$

A similar argument can be used to show that (3.15) holds when $\kappa_i < 0$.

For the boundary test, we need [illegible], given the end point of a timestep. Let

$$c(x, x') = \mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t} | \mathbf{B}_{\boldsymbol{\delta t}} = x']. \tag{3.16}$$

Because of the exponential distribution of $\boldsymbol{\delta t}$, any one path can be considered as starting afresh at the boundary and the conditional probability can be written as the product

$$c(x, x') = \mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}]\left(p_{\nu}^{(n)}(x' - x)\right)^{-1} p_{\nu}^{*}(x, x'), \tag{3.17}$$

where $p_{\nu}^{(n)}(x)$ is given by (2.2),

$$p_{\nu}^{*}(x, x') = \int p_{\nu}^{(n)}(x' - y) q_{\partial D}(y^{\perp}) \mathrm{d}y^{\perp}, \tag{3.18}$$

and $q_{\partial D}(y^{\perp})$ is the density of arrival positions on ∂D, defined in the same way as (3.4). In one dimension, $p_{\nu}^{*}(x, x')$ is the density at x' after an exponential timestep started at the boundary and is independent of x. In more than one dimension, $p_{\nu}^{*}(x, x')$ is the density at x' resulting from the ensemble of realizations with initial conditions distributed according to the density of arrival positions on ∂D, which depends on x. Analysis of the conditional hitting probability (3.17) is the basis for boundary tests of arbitrary accuracy.

4. Exponential timestepping. In this section, we describe the update algorithm and alternative boundary test criteria that we have implemented for exponential timestepping in n dimensions. The Cartesian components of the increment of an n-dimensional Brownian motion over a time interval of fixed length are independent Gaussian random variables. Under exponential timestepping, the Cartesian coordinates of the increment are no longer independent. However, the fact that the distance traveled is independent of the direction traveled permits a simple algorithm to be developed.

4.1. Multidimensional increments. The density of the distance traveled after an exponentially distributed time is obtained from (2.9) by letting $r_0 \to 0$, which gives

$$b^{(n)}(r, 0) = 2^{1-n/2}\nu(\nu r)^{n/2} K_{n/2-1}(\nu r)/\Gamma(n/2). \tag{4.1}$$

To carry out an exact update for Brownian motion in n dimensions, we generate an increment whose modulus is $\nu^{-1}\mathbf{a}^{(n)}$, where

$$\frac{\mathrm{d}}{\mathrm{d}x}\mathcal{P}[\mathbf{a}^{(n)} < x] = 2^{1-n/2} x^{n/2} K_{n/2-1}(x)/\Gamma\left(\frac{n}{2}\right). \tag{4.2}$$

We construct realizations of the random variable $\mathbf{a}^{(n)}$ as

$$\mathbf{a}^{(n)} = \left(\mathbf{h}_1^2 + \mathbf{h}_2^2\right)^{1/2} \left(\mathbf{g}_1^2 + \mathbf{g}_2^2 + \cdots + \mathbf{g}_n^2\right)^{1/2}, \tag{4.3}$$

where $\mathbf{h}_1, \mathbf{h}_2$, and $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_n$ are independent Gaussian random variables with zero mean and unit variance [24]. The factor $(\mathbf{h}_1^2 + \mathbf{h}_2^2)^{1/2}$ is conveniently generated as $(-2\log \mathbf{u})^{1/2}$, where $\mathbf{u}$ is distributed uniformly on $(0, 1)$ [25].

The direction of the increment is independent of the modulus and uniformly distributed on the surface of the unit n-dimensional ball. Let $\mathbf{v}_1$ and $\mathbf{v}_2$ be independent and uniformly distributed between -1 and 1. For all pairs such that $\mathbf{v}_1^2 + \mathbf{v}_2^2 < 1$, let $\mathbf{x} = (\mathbf{v}_1^2 - \mathbf{v}_2^2)/(\mathbf{v}_1^2 + \mathbf{v}_2^2)$ and $\mathbf{y} = 2\mathbf{v}_1\mathbf{v}_2/(\mathbf{v}_1^2 + \mathbf{v}_2^2)$. Then $(\mathbf{x}, \mathbf{y})$ is uniformly distributed on the unit circle [26]. A uniform distribution on the unit sphere is produced by choosing $\mathbf{z}$ uniformly in $(-1, 1)$; then $\mathbf{x}$ and $\mathbf{y}$ uniformly on a circle with radius $\left(1 - \mathbf{z}^2\right)^{1/2}$. In dimension $n > 3$, two possible methods are as follows. First, a vector whose n components are independent mean-zero Gaussian random variables with equal variance has density that depends only on its magnitude. Normalizing to unit magnitude produces a uniform distribution on the n-dimensional sphere. Alternatively, for n not much above 3, one may sample from a uniform distribution in the n-cube that contains the unit ball, reject those outside the ball, then renormalize to unit magnitude.

4.2. Boundary tests. A boundary test is performed after each update of the process to take account of the probability that the boundary has been hit during the timestep. The boundary is deemed to have been hit the first time that

$$C_{\mathrm{num}}(s, s') > \mathbf{v}, \tag{4.4}$$

where $C_{\mathrm{num}}(s, s')$ is the approximation of (4.7) used in the numerical method and $\mathbf{v}$ is drawn, independently at each timestep, from a uniform distribution on $(0, 1)$.

The full information available for the boundary test is the value of the process at the start of the timestep, x, and the value at the end, x' (Figure 4.1). However, the conditional probability,

$$\mathcal{P}[\mathbf{H}_P^{(n)}(x) < \boldsymbol{\delta t} | \mathbf{B}_{\boldsymbol{\delta t}} = x'] = \exp\left(-2\nu(s \wedge s')\right), \tag{4.5}$$

is exact for the flat surface P, even though it makes no use of the information contained in the directions parallel to P. It can be obtained from (3.17) by integrating over $x^{\perp}$ and $(x')^{\perp}$. We call a boundary test using (4.5) the [illegible]. For numerical timesteps of fixed length Δt, the flat boundary approximation has been called a "half-space" approximation in [15]. The corresponding [illegible] is

$$C_{\mathrm{Euler}}(s, s') = \exp\left(2ss'\Delta t^{-1}\right). \tag{4.6}$$

We seek corrections to (4.5) by analyzing the conditional hitting probability in the form

$$\bar{c}(s, s') = \mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t} | x_n' = -s']. \tag{4.7}$$

This is a simplification of (3.16) because the conditional hitting probability is analyzed as a function of $s \in \mathbb{R}$ and $s' \in \mathbb{R}$, rather than as a function of $x \in \mathbb{R}^n$ and $x' \in \mathbb{R}^n$. We also need only the properties of the surface at its point nearest x.

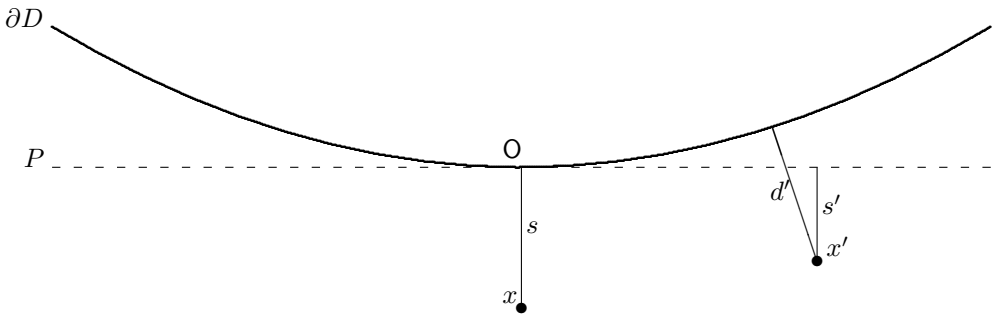


FIG. 4.1. *The position at the start of a timestep is x and the position at the end of the timestep is x'. The distance from x to ∂D is d and the distance from x' to ∂D is d'. The distance from x to P is $s = d$ and the distance from x' to P is $s' \neq d'$.*

Assuming that the surface ∂D can be parametrized as in (3.12), (4.7) is

$$\begin{aligned}(4.8)\quad \bar{c}(s,s') &= \mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}] \left(p_\nu^{(1)}(s-s')\right)^{-1} \int p_\nu^{(1)}(y_n(y^\perp)+s') q_{\partial D}(y^\perp)\mathrm{d}y^\perp \\ &= \mathcal{P}[\mathbf{H}_{\partial D}^{(n)}(x) < \boldsymbol{\delta t}] \exp\left(\nu|s'-s| - \nu s'\right) \int \exp\left(-\nu y_n(y^\perp)\right) q_{\partial D}(y^\perp)\mathrm{d}y^\perp.\end{aligned}$$

Note that, if all κ_i in (3.12) are zero, then $y_n(y^\perp) = 0$ and (4.8) reduces to (4.5). The first factor in (4.8) was evaluated in (3.15). Now, similarly to (3.14),

$$(4.9)\qquad \exp\left(-\nu y_n(x^\perp)\right) = 1 - \frac{1}{2}\nu\left(\kappa_1 y_1^2 + \kappa_2 y_2^2 + \cdots + \kappa_{n-1} y_{n-1}^2\right) + \mathcal{O}\left(\left(\frac{\kappa}{\nu}\right)^2\right),$$

where $y_i = \mathcal{O}(\nu^{-1})$. With (3.12) we find

$$\begin{aligned}\int \exp\left(-\nu y_n(x^\perp)\right) q_{\partial D}(x^\perp)\mathrm{d}x^\perp &= 1 - \frac{1}{2}s(\kappa_1 + \kappa_2 + \cdots + \kappa_{n-1}) \\ &\quad + \mathcal{O}\left(\left(\frac{\kappa}{\nu}\right)^2\right). \qquad (4.10)\end{aligned}$$

Using (3.15) and (4.10) in (4.8) we find the [illegible], with errors proportional to $(\kappa/\nu)^2$:

$$(4.11)\qquad C_{\mathrm{cc}}(s,s') = \exp(-2\nu((s \wedge s') \vee 0))(1 - s(\kappa_1 + \kappa_2 + \cdots + \kappa_{n-1})).$$

We also implement a [illegible] boundary test, invariant under interchange of the start and end points of the timestep,

$$(4.12)\qquad C_{\mathrm{sym}} = \exp(-2\nu(d \wedge d')).$$

An extra complication arises when measuring the exit time from a nonconcave domain. In that case ∂D may be reached before P. If x' is between ∂D and P, then the boundary is deemed hit; the conditional hitting probability (4.4) needed for the boundary test is that restricted to those paths with x' inside the domain. Consider the conditional hitting probability as a function of s' for a given s. As $s' \to \infty$, we regain

(4.11). On the other hand, as $s' \to 0$, the curvature is not seen by the subset of paths that remain inside the domain, and so the limit is (4.5). In our numerical experiments with exit from domains such that $\kappa_i < 0$ for some i, we use the [illegible] given by

$$C_{\rm cv}(s,s') = \exp(-2\nu(s\wedge s'))\,(1-(s\wedge s')(\kappa_1+\kappa_2+\cdots+\kappa_{n-1})). \tag{4.13}$$

5. Numerical experiments. We restrict ourselves to Brownian motion, where the update is exact under both Euler and exponential timestepping, but a systematic error in the numerical mean exit time may arise from the numerical decision as to when the boundary is deemed to have been hit. Simple use of a boundary test gives an error in the mean exit time proportional to λ^{-1}, the mean duration of a timestep [2]. In this work we are able to obtain high-order convergence of the mean exit time under exponential timestepping.

We begin with exits from circles and spheres, where the problem can be transformed into a one-dimensional problem (although the transformed process is no longer driftless). In the case of an exit from a circle of radius R, the exact conditional hitting probability involves special functions that can be expanded in powers of $(\nu R)^{-1}$ in a form convenient for numerical implementation. The more terms that are included in the expansion, the higher is the order of the numerical method. In the case of exit from a sphere, the special functions reduce to a simple form that can be used to produce numerical estimates of the mean exit time [illegible]

We also study exit from the interior of an ellipse and from the region bounded by the two lines of a hyperbola. These examples illustrate the two cases of locally convex and locally concave domains with smooth boundaries that could be found in applications, and have the advantage that exact analytical results are available for comparison with numerics.

Numerical results are plotted as a function of the parameter λ^{-1}, the mean duration of a timestep. The same horizontal axis is used to display results obtained using the Euler method, where the duration of a timestep, Δt, is fixed and equal to λ^{-1}. We choose $D = \frac{1}{2}$ in all numerical experiments, which typically consist of 10^{10} realizations.

The programs used in the numerical experiments reported in this section are available at http://randomideas.org.

5.1. Exits from circles and spheres. We consider the problem of the numerical estimation of the mean exit time from an n-dimensional ball. The mean exit time with initial condition $x \in \mathbb{R}^n$ is denoted by $\langle \mathbf{h}_R^{(n)}(x)\rangle$ and given by

$$\left\langle \mathbf{h}_R^{(n)}(x)\right\rangle = \frac{R^2-|x|^2}{n}. \tag{5.1}$$

Figure 5.1 was obtained using the naive method of evaluating a mean exit time; i.e., each realization is continued without a boundary test until the first time that a value is generated at a distance greater than R from the origin. The difference between the mean time recorded and (5.1) is proportional to $\langle \boldsymbol{\delta t}\rangle^{1/2}$.

In Figure 5.2, we show the error in the mean exit time from a circle using various boundary tests and timestepping methods. The results labeled "Euler" are obtained using the Euler update and the boundary test (4.6). The results labeled "simple" and "corrected" are obtained using exponential timestepping and boundary tests based on (2.13). When only the leading term in (2.13) is used, we obtain errors proportional

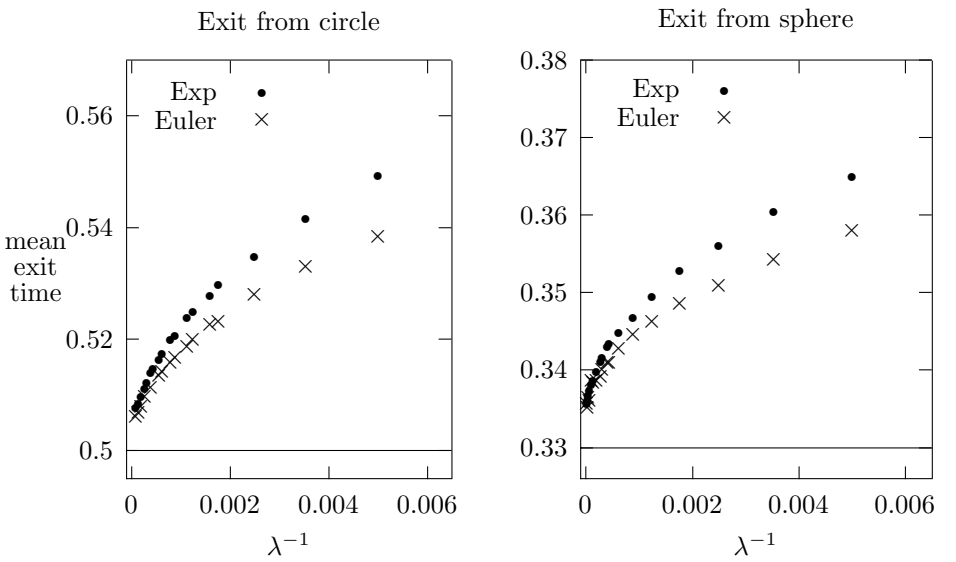


FIG. 5.1. *Measuring the mean exit time without a boundary test. Exit from a circle (left) and sphere (right) with unit radius, starting from the origin. Although the updates of the process are exact, the mean exit time is overestimated by an amount proportional to the square root of the mean duration of a timestep. The horizontal axis is the mean duration of a timestep in the case of exponential timestepping and the (fixed) length Δt of a timestep in the case of the Euler method. Statistical errors are smaller than the plotted symbols.*

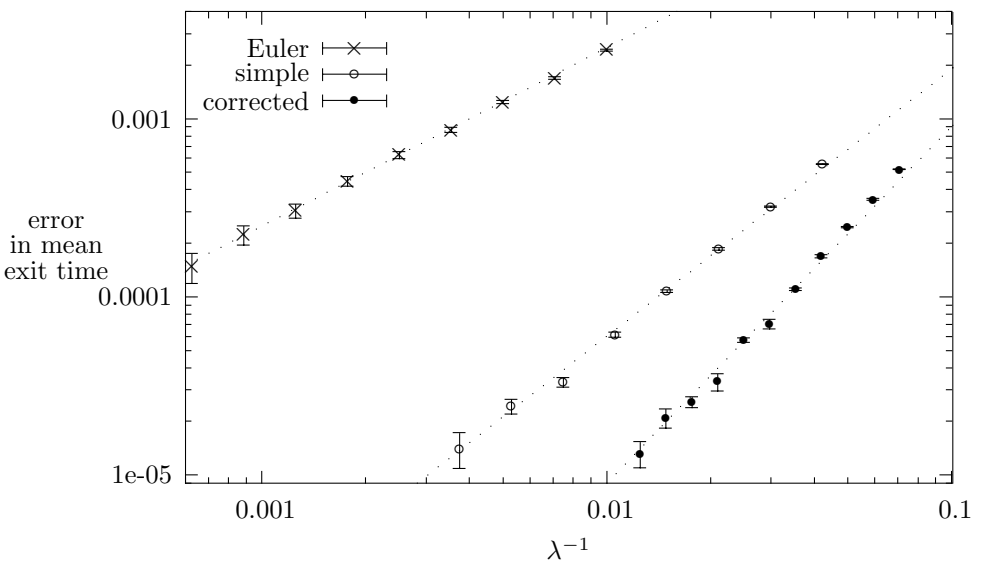


FIG. 5.2. *Error in numerical mean exit time, using boundary correction (logarithmic scales). Exit from a circle with radius $R = 1$, starting from the origin. Statistical errors are indicated by error bars. The dotted lines are error $= 0.25\langle \boldsymbol{\delta t} \rangle$, error $= 0.06\langle \boldsymbol{\delta t} \rangle^{3/2}$, and error $= 0.10\langle \boldsymbol{\delta t} \rangle^{2}$.*

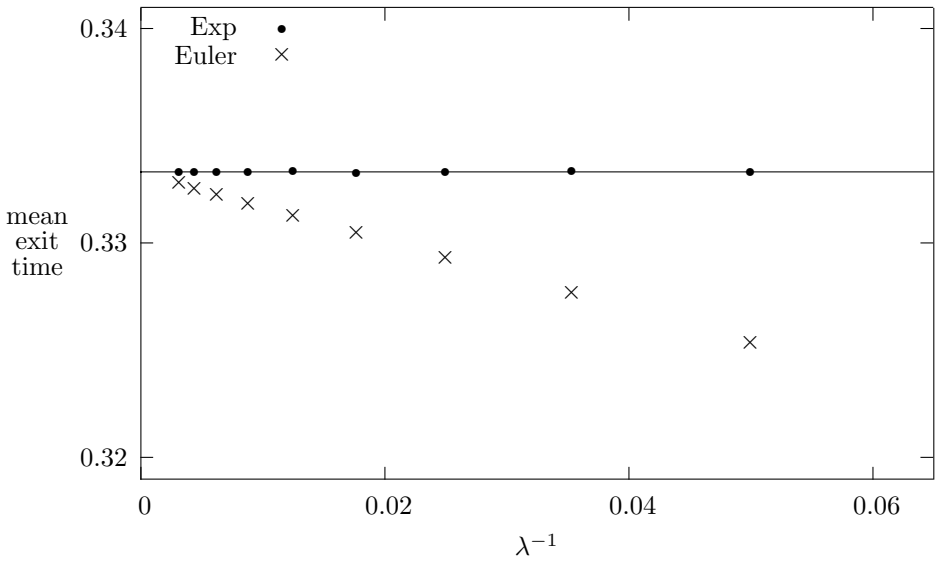


FIG. 5.3. *Error in numerical mean exit time, using boundary correction. Exit from a sphere with radius $R = 1$, starting from the origin. The solid line is the exact result* (5.1)*. The systematic error is zero for the exponential method with boundary test. Statistical errors are smaller than the plotted symbols.*

to $\langle \boldsymbol{\delta t} \rangle^{3/2}$. When the next term in (2.13) is included, we find errors proportional to $\langle \boldsymbol{\delta t} \rangle^{2}$. The size of the statistical errors is independent of λ, but on a logarithmic scale they appear large when the systematic error is small.

When $n = 3$, we find the convenient exact form

$$\mathcal{P}[\mathbf{h}_R^{(3)}(r_0) < \boldsymbol{\delta t} \,|\, |\mathbf{B}_{\boldsymbol{\delta t}}| = r_1] = \mathrm{e}^{-\nu|R-r|} \frac{\sinh \nu(r \vee R)}{\sinh \nu(r \wedge R)}, \tag{5.2}$$

where r is the closer of r_0 and r_1 to R. The mean exit time from a sphere is obtained [illegible] λ under exponential timestepping. In Figure 5.3 the mean exit time is plotted against λ^{-1}. Results obtained using exponential timestepping with (5.2) are shown as small filled circles and results obtained using the Euler method with (4.6) as crosses.

5.2. Exit from an ellipse. As an example of a convex domain, we consider the interior of an ellipse, i.e., the set of points $(x_1, x_2) \in \mathbb{R}^2$ satisfying

$$a x_1^2 + b x_2^2 < 1, \tag{5.3}$$

where a and b are positive constants. Let $\mathbf{H}_{a,b}(x_1, x_2)$ be the first time that a Brownian motion, started at (x_1, x_2), exits the interior of the ellipse. Then $\langle \mathbf{H}_{a,b}(x_1, x_2) \rangle = T(x_1, x_2)$, where

$$\frac{1}{2}\left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right) T(x_1, x_2) = -1, \qquad T(x_1, x_2) = 0 \quad \text{when} \quad a x_1^2 + b x_2^2 = 1. \tag{5.4}$$

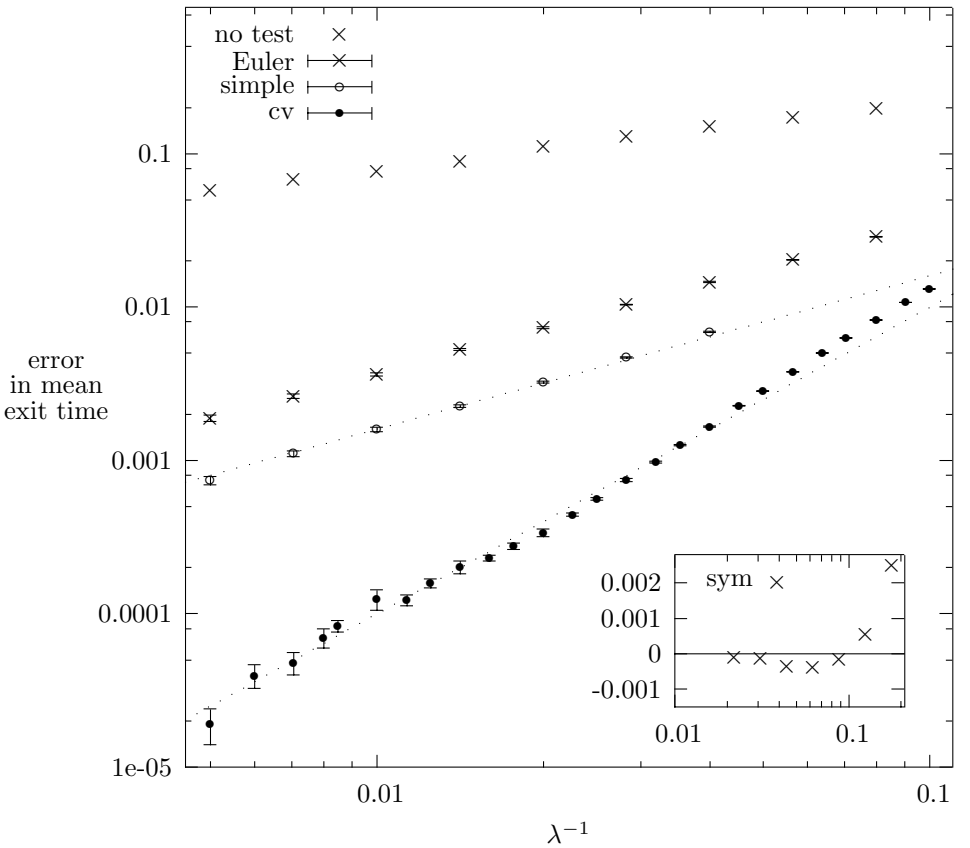


FIG. 5.4. *Error in numerical mean exit time (logarithmic scales). Exit from an ellipse, described by* (5.3) *with* $a = 1/4$ *and* $b = 1$. *The crosses are results using the Euler method, with no boundary test (crosses) and with the boundary test* (4.6) *(crosses with error bars). The results labeled "simple" were obtained using the boundary test,* (4.5); *those labeled "cv" were obtained using the curvature-corrected convex boundary test,* (4.13). *The dotted lines are error* $= 0.16\langle \boldsymbol{\delta t} \rangle^1$ *and error* $= 1.5\langle \boldsymbol{\delta t} \rangle^2$. *The inset displays the difference between the numerical and exact mean exit times using the symmetric boundary test* (4.12).

The solution of (5.4) is

$$T(x_1, x_2) = \left(1 - ax_1^2 - bx_2^2\right)/(a+b). \tag{5.5}$$

In our numerical experiments, the update is the same as that used for studying exits from circles. The boundary is deemed hit if the end point of a timestep is outside the ellipse or if the boundary test is satisfied. To evaluate the conditional probabilities for the boundary test, the closest point on the ellipse to the starting point of each timestep is found by Newton iteration.

In Figure 5.4, we display results obtained using five different methods: the standard Euler method with no boundary test and with the boundary test (4.6), and exponential timestepping with the boundary tests (4.5), (4.13), and (4.12). With (4.5) we find a systematic error in the mean exit time proportional to $\langle \boldsymbol{\delta t} \rangle^1$; with (4.13) we find a systematic error proportional to $\langle \boldsymbol{\delta t} \rangle^2$. Using (4.12) produces an error that passes through 0 at a finite value of λ (inset).

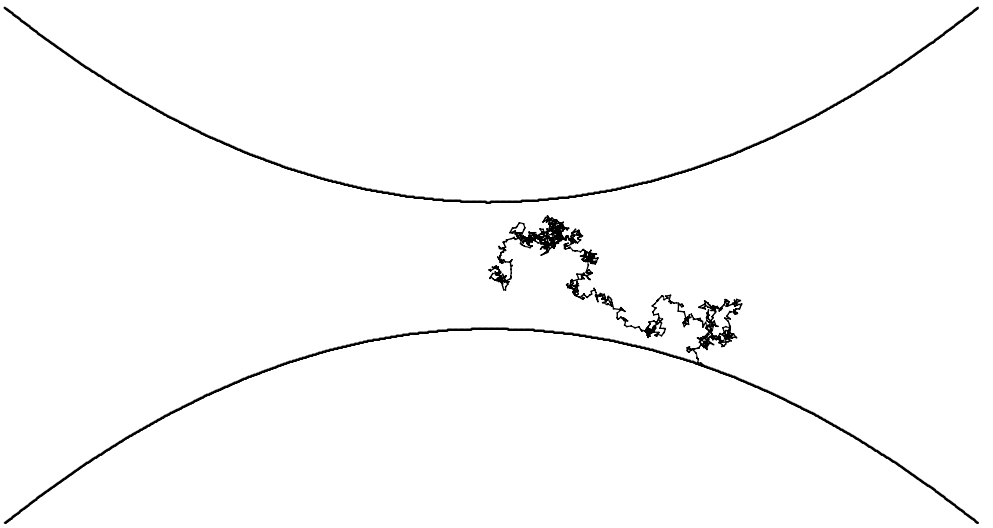


FIG. 5.5. *We measure the mean exit time from concave regions bounded by the two lines of a hyperbola. The figure shows one realization.*

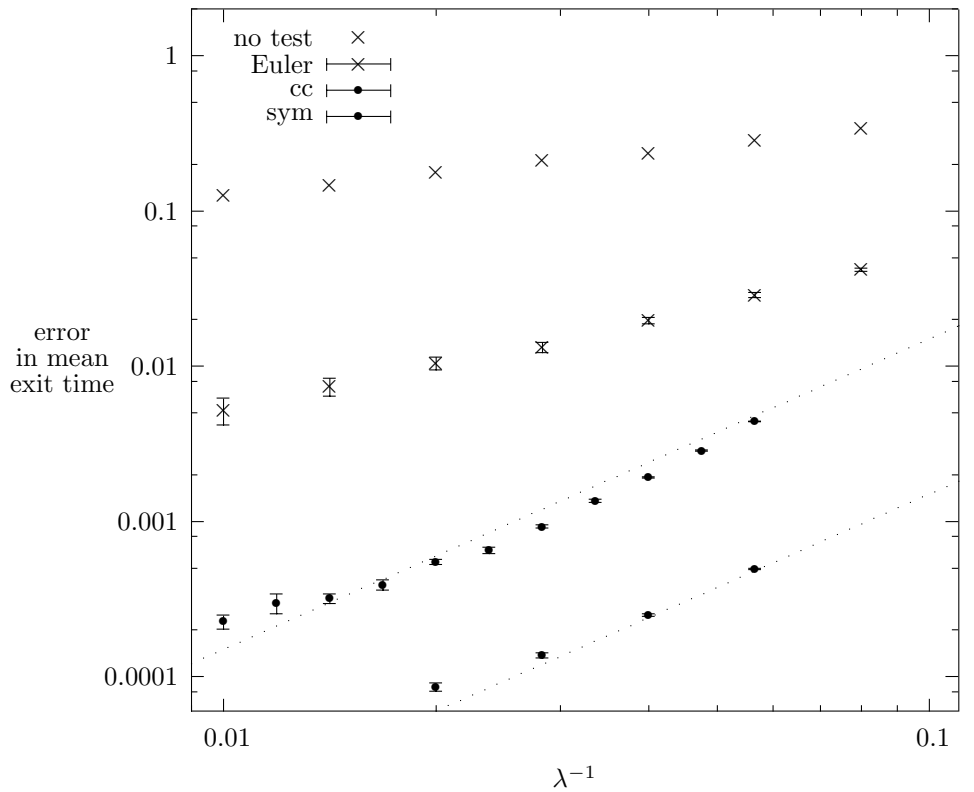


FIG. 5.6. *Error in numerical mean exit time, using boundary correction (logarithmic scales). Exit from a two-dimensional concave region. The hyperbola is described by* (5.3) *with* $a = -1/16$ *and* $b = 1$. *The dotted lines are error* $= 1.5\langle \boldsymbol{\delta t} \rangle^2$ *and error* $= 0.15\langle \boldsymbol{\delta t} \rangle^2$.

5.3. Exit from a region bounded by the two lines of a hyperbola. Consider a hyperbola, defined as the set of points $(x_1, x_2) \in \mathbb{R}^2$ satisfying

$$ax_1^2 + bx_2^2 = 1, \tag{5.6}$$

where $a < 0$ and $b > 0$. The mean exit time from the concave region between the two lines of the hyperbola, depicted in Figure 5.5, is also given by (5.5).

The numerical runs whose results are displayed in Figure 5.6 were performed using the symmetric boundary test (4.12) and the curvature-corrected boundary test (4.11). Statistical errors are large because many paths make long excursions before reaching a boundary. Both boundary tests used produce a systematic error proportional to $\langle \boldsymbol{\delta t} \rangle^2$. The symmetric boundary test is the most efficient for this concave region; the phenomenon of change of sign of the error at a finite value of λ, seen in the ellipsoidal region studied in section 5.2, is not found.

6. Conclusion. An exact numerical procedure has been described for the update of Brownian motion in n dimensions under exponential timestepping. The modulus of an increment is generated as a combination of simple random variables and the angle is uniformly distributed on the surface of the n-dimensional ball.

To evaluate a mean exit time, numerous numerical realizations are generated, each proceeding by updating the process, then performing a test for the possibility of a boundary having been reached in the time interval. A realization is brought to an end when the boundary is deemed hit. Under exponential timestepping, numerical orders of convergence greater than 1 are attainable because of the simple relationship that holds between the mean number of timesteps per realization and the mean exit time. Numerical evidence is presented for convergence proportional to the square of the mean duration of a timestep. An analytical demonstration that this is indeed the case will be the subject of future research.

Taking as examples the exit times from circles, spheres, ellipses, and regions bounded by the two lines of a hyperbola, we report the results of numerical experiments performed with the algorithms developed in this work. The mean exit time from a circle of radius R can be evaluated to arbitrary accuracy by expanding the conditional hitting probability in powers of $(\nu R)^{-1}$. The mean exit time from a sphere is evaluated without systematic error.

The simple boundary test for general smooth surfaces under exponential timestepping uses the minimum of the distance from the starting point and the ending point to the tangent plane through the point on the boundary closest to the starting point. We were able to obtain second-order convergence, i.e., an error proportional to the square of the mean duration of a timestep, using a boundary test that includes a term proportional to κ/ν, where κ is the curvature of the boundary measured at the point on the boundary closest to the starting point. Still higher orders of convergence are attainable if boundary tests including further terms in (3.12) can be derived.

We have considered concave and convex smooth boundaries. Analysis of hitting times of hyperbolae may prove useful in devising algorithms to deal with regions with corners because a suitable choice of parameters gives a geometry approximating that found near a corner with an acute angle. Exponential timestepping has been implemented for processes other than Brownian motion in one dimension [14]. The best way to generalize to multiple dimensions may be to use an approximation by mixing two processes with constant drifts at each timestep. Other areas for future development are algorithms capable of dealing with several close boundaries simultaneously, and boundary tests in three-dimensional regions that take as their starting point the simple exact solution for the conditional hitting probability of a sphere.

REFERENCES

[1] P. E. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer, Berlin, 1992.

[2] K. M. Jansons and G. D. Lythe, *Efficient numerical solution of stochastic differential equations using exponential timestepping*, J. Statist. Phys., 100 (2000), pp. 1097–1109.

[3] K. Itô and H. P. McKean, Jr., *Diffusion Processes and Their Sample Paths*, Springer, Berlin, 1974.

[4] Z. Schuss, *Theory and Applications of Stochastic Differential Equations*, Wiley, New York, 1980.

[5] S. Karlin and H. M. Taylor, *A Second Course in Stochastic Processes*, Academic Press, Orlando, FL, 1981.

[6] C. W. Gardiner, *Handbook of Stochastic Methods*, Springer, Berlin, 1985.

[7] I. Karatzas and S. E. Shreve, *Brownian Motion and Stochastic Calculus*, Springer, New York, 1988.

[8] S. Redner, *A Guide to First-Passage Processes*, Cambridge University Press, Cambridge, UK, 2001.

[9] M. Beccaria, G. Curci, and A. Viceré, *Numerical solutions of first-exit-time problems*, Phys. Rev. E, 48 (1993), pp. 1539–1546.

[10] P. Baldi, *Exact asymptotics for the probability of exit from a domain and applications to simulation*, Ann. Probab., 23 (1995), pp. 1644–1670.

[11] G. N. Milstein and M. V. Tretyakov, *Simulation of a space-time bounded diffusion*, Ann. Appl. Probab., 9 (1999), pp. 732–779.

[12] R. Mannella, *Boundaries and optimal stopping in a stochastic differential equation*, Phys. Lett. A, 254 (1999), pp. 257–262.

[13] E. Gobet, *Weak approximation of killed diffusion using Euler schemes*, Stochastic Process. Appl., 87 (2000), pp. 167–197.

[14] K. M. Jansons and G. D. Lythe, *Exponential timestepping with boundary test for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 1809–1822.

[15] E. Gobet, *Euler schemes and half-space approximation for the simulation of diffusion in a domain*, ESAIM Probab. Stat., 5 (2001), pp. 261–297.

[16] M. E. Muller, *Some continuous Monte Carlo methods for the Dirichlet problem*, Ann. Math. Statist., 27 (1956), pp. 569–589.

[17] C.-O. Hwang, M. Mascagni, and J. A. Given, *A Feynman-Kac path-integral implementation for Poisson's equation using an h-conditioned Green's function*, Math. Comput. Simulation, 62 (2003) pp. 347–355.

[18] F. M. Buchmann, *Simulation of stopped diffusions*, J. Comput. Phys., 202 (2005), pp. 446–462.

[19] A. N. Borodin and P. Salminen, *Handbook of Brownian Motion—Facts and Formulae*, Birkhäuser, Basel, 1996.

[20] K. M. Jansons and G. D. Lythe, *Stochastic calculus: Application to dynamic bifurcations and threshold crossings*, J. Statist. Phys., 90 (1998), pp. 227–251.

[21] L. A. Santaló, *Integral Geometry and Geometric Probability*, Encyclopedia Math. Appl. 1, Addison-Wesley, Reading, MA, 1976.

[22] C. G. Phillips and K. M. Jansons, *The short-time transient of diffusion outside a conducting body*, Proc. Roy. Soc. London Ser. A, 428 (1990), pp. 431–449.

[23] F. Knight, *Essentials of Brownian Motion and Diffusion*, AMS, Providence, RI, 1981.

[24] K. S. Miller, *Multidimensional Gaussian Distributions*, Wiley, New York, 1964.

[25] B. D. Ripley, *Stochastic Simulation*, Wiley, New York, 1987.

[26] D. E. Knuth, *Seminumerical Algorithms*, 3rd Ed., Addison-Wesley, Reading, MA, 1998.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 809–830
© 2005 Society for Industrial and Applied Mathematics

IMPLICIT SCHEMES FOR THE FOKKER–PLANCK–LANDAU EQUATION*

MOHAMMED LEMOU† AND LUC MIEUSSENS†

Abstract. We propose time implicit schemes to solve the homogeneous Fokker–Planck–Landau equation in both the isotropic and 3-dimensional geometries. These schemes have properties of conservation and entropy. Moreover, they allow for large time steps, making them faster than the usual explicit schemes. To solve the involved linear systems, we prove that the use of Krylov-like solvers preserves the conservation properties. We show in particular that the conjugate gradient method can be used. Numerical tests are performed for the isotropic case and demonstrate an important gain in terms of CPU time, with the same accuracy as explicit schemes. This work is a first step to the development of fast implicit schemes to solve a class of inhomogeneous kinetic equations.

Key words. kinetic equation, Fokker–Planck–Landau equation, implicit schemes, conservative schemes, Krylov methods

AMS subject classifications. 82C40, 82D10, 82C80, 65M06, 65Y20, 65F10

DOI. 10.1137/040609422

1. Introduction.

The Fokker–Planck–Landau (FPL) is a kinetic collisional model used to describe a system of particles in plasma physics (see [17] for instance). The particles are described through a distribution function $f(t,x,v)$ depending on time t, particle position $x \in \mathbb{R}^d$, and their velocity $v \in \mathbb{R}^d$ $(d = 2, 3)$. In this paper we are concerned with the homogeneous case where $f(t,x,v)$ does not depend on x. The model is written in the so-called Landau form

$$\partial_t f(t,v) = Q(f)(v) = \nabla \cdot \int_{\mathbb{R}^d} \Phi(v - v_*) \left(f(v_*) \nabla f(v) - f(v) \nabla f(v_*) \right) dv_*, \tag{1}$$

where $\Phi(w)$ is the following $d \times d$ matrix:

$$\Phi(w) = C|w|^{\gamma+2} S(w) = C|w|^{\gamma+2} \left(I_d - \frac{w \otimes w}{|w|^2} \right).$$

In this expression, C is a positive constant and γ is a parameter leading to the standard classification in hard potentials ($\gamma > 0$), Maxwellian potential ($\gamma = 0$), and soft potentials ($\gamma < 0$). This last case includes the most physically interesting case, the Coulombian case ($\gamma = -3$). The $d \times d$ matrix $S(w)$ is simply the orthogonal projection onto the plane orthogonal to w. For all $w \neq 0$, $\Phi(w)$ is a positive matrix whose null-space is

$$\operatorname{Ker} \Phi(w) = \mathbb{R} w.$$

Throughout this paper, when there is no confusion, the values of any function f under the integral signs will be denoted by f for $f(v)$ and by f_* for $f(v_*)$. Additionally,

*Received by the editors June 4, 2004; accepted for publication (in revised form) March 16, 2005; published electronically November 22, 2005. This research was supported by the European network HYKE (EC contract HPRN-CT-2002-00282).

http://www.siam.org/journals/sisc/27-3/60942.html

†MIP (UMR CNRS 5640), UFR MIG, Université Paul Sabatier, 118 Route de Narbonne, 31062 Toulouse Cedex, France (lemou@mip.ups-tlse.fr, mieussens@mip.ups-tlse.fr).

it is also useful to write the FPL collision operator in the so-called "Log form":

$$Q(f)(v) = \nabla \cdot \int_{\mathbb{R}^d} \Phi(v - v_*) f f_* \left(\nabla \log f - \nabla \log f_*\right) \, dv_*. \tag{2}$$

This collision operator $Q(f)$ satisfies the following weak formulation:

$$\int_{\mathbb{R}^d} Q(f)\phi dv = -\frac{1}{2} \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} \Phi(v - v_*) f f_* \left(\nabla \log f - \nabla \log f_*\right) \cdot \left(\nabla \phi - \nabla \phi_*\right) dv dv_* \tag{3}$$

for all distribution functions f and all test functions ϕ. From this formulation, one can immediately derive the following conservation and entropy properties:

(i) The only functions $\phi(v)$ such that

$$\int_{\mathbb{R}^d} Q(f)\phi \, dv = 0 \qquad \forall f$$

are linear combinations of $1, v, |v|^2$. In particular, total mass, momentum, and energy are conserved.

(ii) The entropy dissipation inequality

$$\int_{\mathbb{R}^d} Q(f) \log(f) \, dv \leq 0 \qquad \forall f > 0.$$

This also gives the well known H-theorem, saying that the functional $H(f) = \int_{\mathbb{R}^d} f \log(f) \, dv$ is a time nonincreasing function. Furthermore, this inequality becomes an equality if and only if f is a Maxwellian

$$f_{eq}(v) = \frac{\rho}{(2\pi T)^{\frac{d}{2}}} \exp\left(-\frac{|v - u|^2}{2T}\right), \tag{4}$$

where ρ, u, and T are velocity independent parameters. This is formally equivalent to saying that f is an equilibrium function, that is, $Q(f) = 0$.

Recently, numerous works have been concerned with constructing discretizations of the collision operator that obey the above physical properties of conservation and entropy. The first scheme in this direction was established in [8]. Unfortunately, it turned out to be very expensive in terms of CPU time. Later, various fast algorithms have been constructed to reduce this cost (see [6, 15, 4, 1]). Numerical experiments usually show that the exact preservation of properties (i) and (ii) is crucial to design efficient numerical schemes. In particular, a small error on conservation properties (i) may result in a substantial error on the equilibrium state and may generate some instabilities. On the other hand, nonconservative schemes have to be sufficiently accurate to reduce this error. This task can be achieved by refining the velocity grid (which is expensive) or by using spectral schemes as in [18, 19].

In all these works, the time discretization problem has not been completely solved. Indeed the schemes used are explicit in time, such as the usual Euler explicit scheme

$$f^{n+1} = f^n + \Delta t Q(f^n) \tag{5}$$

or higher order versions (explicit Runge–Kutta methods). It is known that such schemes induce a strong parabolic CFL condition of the form $\Delta t \leq C \Delta v^2$, Δt and Δv being the time and velocity steps. This condition is due to the diffusive nature

of the FPL operator and has been rigorously established in [4] for the isotropic case (that is, where the distribution function only depends on the modulus of the velocity). Therefore, to reach a given simulation time, a large number of iterations n_{it} is required. For instance, in the isotropic case, we have $n_{it} = N^2$, N being the dimension of the approximation space. In that case, even with fast evaluations of the collision operator in $O(N)$ or $O(N \log N)$ (as proposed in the previous works), these explicit schemes require a total simulation cost of the order of $n_{it}O(N) = O(N^3)$ or $O(N^3 \log N)$. However, according to some recent works, it is possible to use explicit schemes with slightly larger time steps. For instance, in [10] a high order explicit scheme with a large stability interval has been used. But the gain obtained with this method remains relatively small: the time step can be taken 5 to 10 times as large as that of the Euler method (5), and the total CPU time is divided by a factor of 4 only.

Consequently, it is attractive to use time implicit schemes, since it is known that they can be free of restrictive time step conditions. This consists of replacing $Q(f^n)$ in (5) by an approximation that also depends on f^{n+1}. Then the problem is how to construct such implicit schemes in order to satisfy the following two requirements:

- the properties (i) and (ii) must be preserved;
- the total computational cost must be smaller than that of the explicit scheme with almost the same accuracy.

Many works in plasma physics area use implicit schemes to solve the FPL equation (for instance, see [12, 13] and the references therein). However, the problem of exact preservation of (i) and (ii) is generally not addressed. Moreover, the total complexity of these algorithms is not optimal. We note that, beyond the fact that implicit schemes usually induce an additional computational cost (nonsparse matrix inversion), they may also affect the properties of conservation and entropy. In [9] for instance, Epperlein has proposed an implicit scheme which is conservative, whereas its total numerical complexity is comparable to the usual explicit resolution. This has been clearly shown in [5]. A recent work by Chacón et al. [7] uses a fast linear solver to reduce the cost of their implicit scheme. However, their approach does not exactly respect the conservation and entropy properties. In fact, whereas they claim that their scheme preserves the energy, they also point out that the solvers and the velocity boundary conditions that they used in practice affect the conservation properties.

In this paper we develop a strategy leading to exact conservative implicit schemes with a reduced computational cost. One of our schemes also satisfies the entropy property. Iterative Krylov solvers are used to efficiently solve the linear systems generated by the implicit schemes. This strategy is also proved to be conservative, even if the linear systems are only solved approximately, and an important gain in terms of computational cost is obtained. All the schemes developed in this work concern both the 2- and 3-dimensional cases as well as the isotropic FPL equation.

We point out that our method is based on the Landau form (1) or the Log form (2) of the FPL equation. Hence it can be used with various potentials and easily be extended to quantum and relativistic cases. This does not seem to be possible if one uses the so-called Rosenbluth form of the classical FPL equation, as done in [7]. Note also that our schemes can easily be used in the resolution of inhomogeneous problems via standard splitting techniques.

The outline of this paper is as follows. In the next section we focus on the time discretization only and present the different implicit schemes. In section 3, the velocity variable is also discretized, leading to completely discrete implicit schemes. All these schemes require the resolution of large and nonsparse linear systems. This is

addressed in section 4 where fast linear solvers are proposed. In section 5, we discuss the numerical complexity of our algorithms and present various numerical tests in the isotropic case. The validation of the present strategy for the 3-dimensional case needs further investigations to solve the linear systems. Therefore the numerical tests for the 3-dimensional geometries are deferred to a forthcoming paper. We finally mention that a part of this work has been summarized in a previous note [16].

2. Implicit schemes. In this section, we focus on the time discretization only. Suitable discretizations in the velocity variable will be developed in section 3. Below, we present different strategies to construct linear time implicit schemes that have properties of conservation and entropy.

2.1. Contracted implicit scheme. We first note that the FPL operator (1) can be rewritten in the following diffusive form:

$$Q(f) = \nabla \cdot (D(f)\nabla f + F(f)f), \tag{6}$$

where $D(f) = \int_{\mathbb{R}^d} \Phi(v - v_*) f(v_*)\, dv_*$, and $F(f) = \int_{\mathbb{R}^d} \Phi(v - v_*)\nabla f(v_*)\, dv_*$. This shows in particular that the CFL condition resulting from the use of time explicit schemes is due to the diffusive term $\nabla \cdot (D(f)\nabla f)$ in (6). Therefore the first idea is to make ∇f implicit in this last expression. On the other hand, the conservation and entropy properties are a consequence of the symmetry property (between v and v_*) of the collision operator. To preserve this symmetry, the friction coefficient $F(f)$ has to be implicit too. This leads to the following contracted implicit scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = q^c(f^n, f^{n+1}), \quad \text{with} \tag{7}$$

$$q^c(f, g) = \nabla \cdot \int_{\mathbb{R}^d} \Phi(v - v_*)\, (f_*\nabla g - f\nabla g_*)\, dv_*. \tag{8}$$

Note that this operator q^c is not the linearization of Q around f. The linearized operator around f is in fact

$$\begin{aligned} q(f, g) &= \nabla \cdot \int_{\mathbb{R}^d} \Phi(v - v_*)\, (f_*\nabla g - f\nabla g_* + g_*\nabla f - g\nabla f_*)\, dv_* \\ &= q^c(f, g) + q^c(g, f). \end{aligned} \tag{9}$$

Therefore, the contracted scheme (7) is not the usual linearized implicit scheme as used by Epperlein in [9]. The operator q^c is contracted in the sense that we only keep the terms that are necessary to ensure the symmetry between v and v_* in the linearized operator q.

PROPOSITION 2.1.

(i) [illegible] q^c [illegible]

$$\int_{\mathbb{R}^d} q^c(f, g)\phi dv = -\frac{1}{2}\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \Phi(v - v_*)\, (f_*\nabla g - f\nabla g_*) \cdot (\nabla\phi - \nabla\phi_*)\, dv dv_* \tag{10}$$

(ii) [illegible] ϕ [illegible] (7, 8) [illegible]

$$\forall n, \quad \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^n\, dv = \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^0\, dv.$$

(iii) [illegible]. The weak form is classically obtained using integration by parts and the permutation of the variables v and v_*. The conservation property is a direct consequence of the weak form: $\nabla\phi - \nabla\phi_*$ is zero for $\phi = 1$ and v and is in $\operatorname{Ker}\Phi(v - v_*)$ for $\phi = \frac{1}{2}|v|^2$. Finally, a simple Taylor expansion and the consistency relation $q^c(f, f) = Q(f)$ give the time order of the contracted scheme. ∎

2.2. A θ-scheme. The present approach first consists of a time integration of (1) using the standard θ-scheme

$$\frac{f^{n+1} - f^n}{\Delta t} = (1 - \theta)Q(f^n) + \theta Q(f^{n+1}). \tag{11}$$

Then we linearize $Q(f^{n+1})$ around f^n:

$$Q(f^{n+1}) = Q(f^n) + DQ(f^n)(f^{n+1} - f^n), \tag{12}$$

where $DQ(f)(g) = q(f, g)$ is given by formula (9). With this linearization, the θ-scheme (11) turns to

$$\frac{f^{n+1} - f^n}{\Delta t} = \theta q(f^n, f^{n+1}) + \left(\frac{1}{2} - \theta\right) q(f^n, f^n) \tag{13}$$

for any $\theta \in \mathbb{R}$.

If we introduce the classical symmetrized bilinear operator $q_s(f, g) = \frac{1}{2}q(f, g)$, then the previous scheme can be written as a convex combination:

$$\frac{f^{n+1} - f^n}{\Delta t} = 2\theta q_s(f^n, f^{n+1}) + (1 - 2\theta)q_s(f^n, f^n).$$

Note that the contracted scheme (7) is not a θ-scheme and that the linearized implicit scheme used by Epperlein in [9] is obtained for $\theta = 1$. Now we give the main properties of this scheme.

PROPOSITION 2.2.

(i) [illegible] θ [illegible] (13) [illegible]

$$\forall n, \quad \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^n \, dv = \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^0 \, dv.$$

(ii) [illegible] θ [illegible] $\theta = \frac{1}{2}$ [illegible]. The same arguments as for Proposition 2.1 give the results. ∎

However, we are not able to prove any entropy property for both the contracted and the θ-scheme, except in the isotropic case for the contracted scheme (see [16]). Therefore, we propose another strategy that uses the Log form (2) of the FPL collision operator. This leads to conservative and entropic schemes that are detailed in the next section.

2.3. Log implicit schemes. The first step is to make implicit only the log terms in the Log form (2) of the collision operator. This gives the following nonlinear implicit scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = q^{\log}(f^n, f^{n+1}) = \nabla \cdot \int_{\mathbb{R}^d} \Phi(v - v_*) f^n f_*^n \left(\nabla \log f^{n+1} - \nabla \log f_*^{n+1}\right) dv_*. \tag{14}$$

PROPOSITION 2.3.

(i) $q^{\log}$

$$\int_{\mathbb{R}^d} q^{\log}(f,g)\phi dv = -\frac{1}{2}\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \Phi(v-v_*) f f_* \left(\nabla \log g - \nabla \log g_*\right)\cdot\left(\nabla\phi - \nabla\phi_*\right) dv dv_* \tag{15}$$

(ii) ϕ (14)

$$\forall n, \quad \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^n\, dv = \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^0\, dv.$$

(iii) $q^{\log}$

$$\int_{\mathbb{R}^d} q^{\log}(f,g)\log g\, dv \le 0.$$

(iv) $H_n = \int_{\mathbb{R}^d} f^n \log f^n\, dv$

. Property (i) is again obtained by classical arguments as in the proof of Proposition 2.1. Properties (ii) and (iii) are direct consequences of (i). To prove (iv), we first compute the entropy variation:

$$H(f^{n+1}) - H(f^n) = \int_{\mathbb{R}^d} (f^{n+1} - f^n)\log f^{n+1}\, dv + \int_{\mathbb{R}^d} f^n \log\frac{f^{n+1}}{f^n}\, dv.$$

Using (14) and (iii) shows that the first integral is nonpositive. Moreover, it is classical to use a convexity inequality to prove that the second integral is also nonpositive. This completes the proof of (iv). ∎

However, $q^{\log}(f^n, f^{n+1})$ is nonlinear with respect to f^{n+1} which makes it difficult to use. In the second step, we propose the following linearization around f^n. We write

$$\log f^{n+1} \approx \log f^n + \frac{f^{n+1} - f^n}{f^n},$$

which is inserted in (14) to obtain the following log-linear implicit scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = Q(f^n) + q^l(f^n, f^{n+1}), \tag{16}$$

with

$$q^l(f,g) = \nabla\cdot\int_{\mathbb{R}^d} \Phi(v-v_*) f f_* \left(\nabla\left(\frac{g}{f}\right) - \nabla\left(\frac{g}{f}\right)_*\right) dv_*. \tag{17}$$

Note that $q^l(f,g) = q^l(f, g-f)$ which clearly shows that the scheme is consistent.

Proposition 2.4.

(i) q^l

$$\int_{\mathbb{R}^d} q^l(f,g)\phi dv = -\frac{1}{2}\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \Phi(v-v_*) f f_* \left(\nabla\left(\frac{g}{f}\right) - \nabla\left(\frac{g}{f}\right)_*\right)\cdot\left(\nabla\phi - \nabla\phi_*\right) dv dv_*. \tag{18}$$

(ii) [illegible] (16, 17) [illegible]

$$\forall n, \quad \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^n \, dv = \int_{\mathbb{R}^d} \left(1, v, \frac{1}{2}|v|^2\right)^T f^0 \, dv.$$

(iii) [illegible] (16, 17) [illegible]

$$\int_{\mathbb{R}^d} (Q(f) + q^l(f,g)) \left(\log f + \frac{g}{f}\right) dv \le 0.$$

(iv) [illegible] $H_n = \int_{\mathbb{R}^d} f^n \log f^n \, dv$ [illegible]

$$\inf_{n \in \mathbb{N}, v \in \mathbb{R}^d} \left(\frac{f^{n+1}}{f^n}\right) \ge \frac{1}{2}. \tag{19}$$

(v) [illegible] f [illegible] $g \mapsto q^l(f,g)$ [illegible]

$$\langle q^l(f,g), h\rangle_{\frac{1}{f}} := \int_{\mathbb{R}^d} q^l(f,g) h \, \frac{dv}{f} = \langle q^l(f,h), g\rangle_{\frac{1}{f}}, \qquad \langle q^l(f,g), g\rangle_{\frac{1}{f}} \le 0$$

[illegible] g [illegible] h

[illegible]. Again, we use an integration by parts and the symmetry between v and v_* in (17) to obtain (i). Using (3) and (i), we obtain the following weak formulation:

$$\int_{\mathbb{R}^d} (Q(f) + q^l(f,g))\phi \, dv = \int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \Phi(v - v_*) f f_* \left[\nabla \left(\log f + \frac{g}{f}\right) - \nabla \left(\log f + \frac{g}{f}\right)_*\right] [\nabla\phi - \nabla\phi_*] \, dv dv_*.$$

This gives (ii) with $\phi(v) = 1, v, \frac{1}{2}|v|^2$ and (iii) with $\phi(v) = \log f + \frac{g}{f}$. Now we prove (iv): first a simple calculation shows that the entropy variation is

$$H(f^{n+1}) - H(f^n) = \int_{\mathbb{R}^d} (f^{n+1} - f^n) \left(\log f^n + \frac{f^{n+1}}{f^n}\right) dv + \int_{\mathbb{R}^d} B(f^n, f^{n+1}) \, dv, \tag{20}$$

where

$$B(f^n, f^{n+1}) = f^n \left(\frac{f^{n+1}}{f^n} \log \frac{f^{n+1}}{f^n} - \left(\frac{f^{n+1}}{f^n}\right)^2 + \frac{f^{n+1}}{f^n}\right).$$

From relation (16), the first integral of (20) is

$$\int_{\mathbb{R}^d} (f^{n+1} - f^n) \left(\log f^n + \frac{f^{n+1}}{f^n}\right) dv = \Delta t \int_{\mathbb{R}^d} (Q(f^n) + q^l(f^n, f^{n+1})) \left(\log f^n + \frac{f^{n+1}}{f^n}\right) dv,$$

which is nonpositive thanks to (iii).

To study the second integral of (20), we set $\theta(X) = X\log X - X^2 + X$ for $X \in \mathbb{R}^+$, which is concave on $[\frac{1}{2}, +\infty)$. Therefore, if $\frac{f^{n+1}}{f^n} \geq \frac{1}{2}$ we can use the Jensen inequality to get

$$\int_{\mathbb{R}^d} B(f^n, f^{n+1})\,dv = \int_{\mathbb{R}^d} f^n \theta\left(\frac{f^{n+1}}{f^n}\right) dv \leq \theta\left(\frac{\int_{\mathbb{R}^d} f^{n+1}\,dv}{\int_{\mathbb{R}^d} f^n\,dv}\right) \int_{\mathbb{R}^d} f^n\,dv = \theta(1)\int_{\mathbb{R}^d} f^n\,dv = 0.$$

Consequently, the two integrals of the RHS of (20) are nonpositive under assumption (19) of the proposition, which completes the proof of (iv).

For property (v), we use weak formulation (18) with $\phi = \frac{h}{f}$:

$$\langle q^l(f,g), h\rangle_{\frac{1}{f}} = -\frac{1}{2}\int_{\mathbb{R}^d}\int_{\mathbb{R}^d} \Phi(v-v_*) f f_* \left(\nabla\left(\frac{g}{f}\right) - \nabla\left(\frac{g}{f}\right)_*\right)\cdot\left(\nabla\left(\frac{h}{f}\right) - \nabla\left(\frac{h}{f}\right)_*\right) dv dv_*,$$

which is clearly a symmetric formula with respect to g and h. Moreover, taking $h = g$ in the previous relation gives the last inequality of (v). ∎

Remark 2.1. The condition (19) of the proposition is reasonable, since in practice the ratio $\frac{f^{n+1}}{f^n}$ should be close to 1. However, to get a weaker condition, one could consider the following modified scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = Q(f^n) + A q^l(f^n, f^{n+1}),$$

where A is a free positive parameter. This scheme is still consistent and condition (19) becomes

$$\inf_{n\in\mathbb{N}, v\in\mathbb{R}^d}\left(\frac{f^{n+1}}{f^n}\right) \geq \frac{1}{2A}.$$

The last property of the proposition is of practical importance since in that case, the conjugate gradient (CG) method could be used in the inversion process of the linear system (see details in section 4). However, the weight $\frac{1}{f^n}$, which is used to construct an inner product in the CG algorithm, should be positive at any time. Unfortunately, we cannot prove that this property is satisfied. We point out that the two first schemes (contracted and θ-schemes) do not have the self-adjointness property. Therefore, we propose modified versions of the θ and log-linear schemes in which the linear operators are self-adjoint at any time (in particular, the positivity of the weight is guaranteed at any time).

2.4. Equilibrium linearized implicit schemes. We write implicit terms $q(f^n, f^{n+1})$ and $q^l(f^n, f^{n+1})$ of schemes (13) and (16), (17) under the following perturbation form: $q(f^n, f^{n+1} - f^n) + q(f^n, f^n)$ and $q^l(f^n, f^{n+1} - f^n)$. The idea is to replace the first argument f^n of the perturbative terms $q(f^n, f^{n+1} - f^n)$ and $q^l(f^n, f^{n+1} - f^n)$ by its associate Maxwellian equilibrium f_{eq} of the form (4) that has the same mass, momentum, and energy as f^n:

$$q(f^n, f^{n+1} - f^n) \approx q(f_{eq}, f^{n+1} - f^n) \quad \text{and} \quad q^l(f^n, f^{n+1} - f^n) \approx q^l(f_{eq}, f^{n+1} - f^n).$$

TABLE 1
Properties of the different implicit schemes.

	Time order	Conservation	Entropy	H-theorem	Linear operator
contracted scheme (7-8)	1	yes	?	?	?
θ-scheme (11)	2 if $\theta = \frac{1}{2}$, 1 if not	yes	?	?	?
log-linear scheme (16-17)	1	yes	yes	yes	nonpositive self-adjoint
eq. θ-scheme (21)	1	yes	?	?	nonpositive self-adjoint
eq. log-linear scheme (22)	1	yes	?	?	nonpositive self-adjoint

This leads to the following schemes:

- Equilibrium θ-scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = Q(f^n) + \theta q(f_{eq}, f^{n+1} - f^n). \tag{21}$$

  Because f_{eq} is a Maxwellian, it is well known that the linear operator $g \mapsto q(f_{eq}, g)$ is nonpositive self-adjoint for the weight $\frac{1}{f_{eq}} > 0$. Moreover, this scheme is conservative.

- Equilibrium log linear scheme:

$$\frac{f^{n+1} - f^n}{\Delta t} = Q(f^n) + q^l(f_{eq}, f^{n+1} - f^n). \tag{22}$$

  Because f_{eq} is positive, from property (v) of Proposition 2.4, the linear operator $g \mapsto q^l(f_{eq}, g)$ is nonpositive self-adjoint for the weight $\frac{1}{f_{eq}}$. Moreover, this scheme is conservative.

We mention that the use of equilibrium approximation has already been exploited in other contexts. We refer, for instance, to [11] where equilibrium is used to truncate the so-called Wild sums, and also to the standard BGK model of the Boltzmann equation [2].

REMARK 2.2. Schemes (21), (22) are not obtained by a linearization of $Q(f^{n+1})$ near the equilibrium f_{eq}.

2.5. Summary. We summarize in Table 1 the properties of the previous implicit linear schemes: time order, conservation of mass and energy, the entropy dissipation property of the approximated collision operator, the decreasing of the entropy functional (H-theorem), and the properties of the linear operators $g \mapsto q(f, g), q^c(f, g), q^l(f, g), q(f_{eq}, g)$, and $q^l(f_{eq}, g)$.

2.6. A remark about an unconditionally positive implicit scheme. In this section, we introduce an alternative idea to make linear the nonlinear implicit scheme (14). To that purpose, we modify the approximation of the time derivative in the following way:

$$\partial_t f(n\Delta t) = (f \partial_t \log f)(n\Delta t) \approx f^n \frac{\log f^{n+1} - \log f^n}{\Delta t}.$$

According to this approximation, the nonlinear log scheme (14) is modified into

$$f^n \frac{\log f^{n+1} - \log f^n}{\Delta t} = q^{\log}(f^n, f^{n+1}). \tag{23}$$

This is still nonlinear in f^{n+1}, but it is linear in $F^{n+1} = \log f^{n+1}$ and can be easily solved. Then $f^{n+1} = \exp(F^{n+1})$ is unconditionally positive.

However, it is shown in the following proposition that this scheme cannot preserve the mass $\int f^n\,dv$.

PROPOSITION 2.5.

(i) [illegible] (23) [illegible] $f_0 > 0$ [illegible] $f_n > 0$ [illegible] n

(ii) [illegible] f^n [illegible]

$$\int f^{n+1}\,dv > \int f^n\,dv.$$

[illegible] n_0 [illegible] f^{n_0} [illegible] $n \geq n_0$ [illegible] $\int f^n\,dv$ [illegible]

[illegible]. By construction, the proof of (i) is straightforward. Now we prove (ii): first, integrating (23) with respect to v gives the following relation:

$$\int f^n \log\left(\frac{f^{n+1}}{f^n}\right) dv = 0. \tag{24}$$

Now using the Jensen inequality yields

$$\int f^n \log\left(\frac{f^{n+1}}{f^n}\right) dv \leq \log\left(\frac{\int f^{n+1}\,dv}{\int f^n\,dv}\right) \int f^n\,dv.$$

Since f^n is positive, the previous inequality gives

$$\log\left(\frac{\int f^{n+1}\,dv}{\int f^n\,dv}\right) \geq 0$$

which is equivalent to $\int f^{n+1}\,dv \geq \int f^n\,dv$.

To prove that this last inequality is strict, we assume the contrary, that is $\int f^{n+1}\,dv = \int f^n\,dv$, and use the Czisar–Kullback inequality to obtain

$$\int f^n \log\left(\frac{f^{n+1}}{f^n}\right) dv \leq -\frac{1}{4(\int f^n\,dv)^2} \|f^{n+1} - f^n\|_{L^1}^2.$$

Using (24) then gives $f^{n+1} = f^n$. Then we replace f^{n+1} by f^n in scheme (23) and this gives $q^{\log}(f^n, f^n) = Q(f^n) = 0$. Thus f^n is a Maxwellian, which is impossible due to the assumption of the proposition. Consequently, $\int f^{n+1}\,dv > \int f^n\,dv$. ∎

Despite its simplicity and its positivity property, this scheme does not provide attractive numerical results. This is probably due to the fact that it does not satisfy the conservation properties.

3. Implicit schemes and velocity discretization of the FPL equation: the isotropic case. The previous schemes can naturally be discretized in the velocity variable using standard conservative and entropic discretizations [8, 6, 15, 4, 1]. In this section, we illustrate this assertion with a simple velocity discretization of the FPL equation in the isotropic case.

The isotropic FPL model is (1) in which the distribution function f only depends on time t and on the particle kinetic energy $\varepsilon = |v|^2$. In this case, the FPL equation is written as

$$\partial_t f(t,\varepsilon) = Q(f) = \frac{1}{\sqrt{\varepsilon}} \frac{\partial}{\partial \varepsilon} \int_0^{+\infty} K(\varepsilon, \varepsilon_*) \left(f(\varepsilon_*)\partial_\varepsilon f(\varepsilon) - f(\varepsilon)\partial_\varepsilon f(\varepsilon_*)\right) d\varepsilon_*, \tag{25}$$

with $K(\varepsilon,\varepsilon_*)=\frac{16}{3}\pi\inf(\varepsilon^{3/2},\varepsilon_*^{3/2})$ for Coulombian interactions, and $K(\varepsilon,\varepsilon_*)=\frac{16}{3}\pi\varepsilon^{3/2}\varepsilon_*^{3/2}$ for Maxwellian interactions.

For any distribution function f, the collision operator $Q(f)$ satisfies the following weak formulation:

(26)
$$\int_0^{+\infty} Q(f)\phi(\varepsilon)\sqrt{\varepsilon}\,d\varepsilon = -\frac{1}{2}\int_0^{+\infty}\int_0^{+\infty} K(\varepsilon,\varepsilon_*)\left(f_*\partial_\varepsilon f - f\partial_\varepsilon f_*\right)\left(\partial_\varepsilon\phi-\partial_\varepsilon\phi_*\right)\,d\varepsilon d\varepsilon_*,$$

ϕ being any test function. From this formulation, one can immediately derive the following conservation and entropy properties:

$$\int_0^{+\infty}(1,\varepsilon)^T Q(f)\sqrt{\varepsilon}\,d\varepsilon = 0, \qquad \int_0^{+\infty} Q(f)\log(f)\sqrt{\varepsilon}\,d\varepsilon \le 0.$$

3.1. A conservative and entropic discretization of the isotropic FPL equation. We briefly recall here the discretization used by Berezin, Khudic, and Pekker [3] (also studied by Buet and Cordier [5]). The energy domain is replaced by a regular energy grid of step $\Delta\varepsilon$ and of nodes $\varepsilon_i=(i-1)\Delta\varepsilon$, with $i=1$ to N. The case of an irregular discretization is also considered at the end of this section. Here, the length of the grid is $e=(N-1)\Delta\varepsilon$. Any function f of ε is approximated on the grid by values $(f_i)_{i=1}^N$ supposed to be approximations of $(f(\varepsilon_i))_{i=1}^N$.

Integrals on $\mathbb{R}^+$ with respect to the measure $\sqrt{\varepsilon}d\varepsilon$ are approximated by the following weighted trapezoidal quadrature formula:

$$\int_0^{+\infty}\phi(\varepsilon)\sqrt{\varepsilon}\,d\varepsilon \approx \sum_{i=1}^N \phi(\varepsilon_i)\omega_i,$$

where $\omega_1=\frac{1}{2}\int_{\varepsilon_1}^{\varepsilon_2}\sqrt{\varepsilon}\,d\varepsilon$, $\omega_i=\int_{\varepsilon_{i-1}}^{\varepsilon_{i+1}}\sqrt{\varepsilon}\,d\varepsilon$ for $i=2$ to $N-1$, and $\omega_{N-1}=\frac{1}{2}\int_{\varepsilon_{N-1}}^{\varepsilon_N}\sqrt{\varepsilon}\,d\varepsilon$.

A simple discretization of (25) is

$$\partial_t f = Q(f),$$

where $Q(f)$ now is an N-vector of components

$$Q_i(f) = -\frac{1}{\omega_i}(D^*\mathcal{F})_i, \quad i=1:N. \tag{27}$$

The operator D^* is defined by $(D^*f)_i=f_{i-1}-f_i$, and the vector $\mathcal{F}$ is defined by

$$\mathcal{F}_i = \sum_{j=1}^{N-1} K_{ij}\big(f_j(Df)_i - f_i(Df)_j\big), \quad i=1:N-1, \tag{28}$$

and $\mathcal{F}_0=\mathcal{F}_N=0$. The finite difference operator D is the formal adjoint of D^* defined by $(Df)_i=f_{i+1}-f_i$.

This discretization is constructed so that the discrete collision operator satisfies the following weak formulation:

$$\sum_{i=1}^N Q_i(f)\phi_i\omega_i = -\frac{1}{2}\sum_{i=1}^{N-1}\sum_{j=1}^{N-1} K_{ij}\big(f_j(Df)_i - f_i(Df)_j\big)\big((D\phi)_i-(D\phi)_j\big).$$

This implies that conservation and entropy properties are preserved (see the proof in [5]). Namely, we have

$$\sum_{i=1}^{N} Q_i(f)(1, \sqrt{\varepsilon_i})^T \omega_i = 0, \quad \sum_{i=1}^{N} Q_i(f)(\log f_i)\, \omega_i \leq 0.$$

Another discretization is deduced from the log form of (25) similarly to (2) (see [3]). In that case the discrete collision operator has the same form as (27) but the vector $\mathcal{F}$ now is

$$\mathcal{F}_i = \sum_{j=1}^{N-1} K_{ij} f_i f_j \big((D \log f)_i - (D \log f)_j \big), \quad i = 1 : N-1,$$

and $\mathcal{F}_0 = \mathcal{F}_N = 0$. This scheme has the same properties as the nonlog scheme.

The main drawback of a regular discretization is the fact that the resolution is not accurate near $\varepsilon = 0$, while there are too many points for large ε. Therefore, it is interesting to consider an irregular discretization with the energy step $\Delta\varepsilon_i = \varepsilon_{i+1} - \varepsilon_i$ as in [4]. Following the same procedure as for the regular case, we obtain this discrete collision operator

$$Q_i(f) = \frac{1}{\omega_i}(\mathcal{F}_i - \mathcal{F}_{i-1}) \qquad i = 1 : N, \tag{29}$$

with

$$\mathcal{F}_i = \sum_{j=1}^{N-1} K_{ij} \big(f_j (Df)_i - f_i (Df)_j \big) \Delta\varepsilon_j, \quad i = 1 : N-1,$$

and $\mathcal{F}_0 = \mathcal{F}_N = 0$. Here D is the following finite difference operator $(Df)_i = \frac{f_{i+1} - f_i}{\Delta\varepsilon_i}$, for $i = 1 : N-1$. This approximation is conservative, but the associated equilibrium states are only approximations of the Maxwellians, i.e.,

$$(f_{eq})_i = \beta \prod_{j=1}^{i} (1 + \alpha \Delta\varepsilon_j),$$

β and α being some constants. Maxwellian equilibrium states can be obtained by using the conservative and entropic approximation of [4], which is based on the log form.

Consequently, we have a discrete collision operator that possesses all desired properties (both in log and Landau forms). Our implicit schemes can now be derived exactly as in the continuous case, as is illustrated in the next section.

3.2. Implicit schemes. In this section, we give completely discretized implicit schemes (in both time and velocity). For the sake of simplicity we only present the case of a regular discretization. The case of an irregular grid can be treated in the same way.

The different implicit schemes are given by (7, 13, 16, 21, 22), where the discrete collision operators are the N-vectors defined as follows:

- contracted scheme: $q_i^c(f, g) = -\frac{1}{\omega_i}(D^* \mathcal{F}^c(f, g))_i$ for $i = 1 : N$, where the flux $\mathcal{F}^c$ is

$$\mathcal{F}_i^c(f, g) = \sum_{j=1}^{N-1} K_{ij} \big(f_j (Dg)_i - f_i (Dg)_j \big), \quad i = 1 : N-1.$$

- θ-scheme: $q_i(f,g) = q_i^c(f,g) + q_i^c(g,f)$, $i = 1 : N$.
- log-linear scheme: $q_i^l(f,g) = -\frac{1}{\omega_i}(D^*\mathcal{F}^l(f,g))_i$ for $i = 1 : N$, where the flux $\mathcal{F}^l$ is

$$\mathcal{F}_i^l(f,g) = \sum_{j=1}^{N-1} K_{ij} f_i f_j \left[\left(D\left(\frac{g}{f}\right)\right)_i - \left(D\left(\frac{g}{f}\right)\right)_j \right], \quad i = 1 : N-1.$$

For all these definitions, numerical fluxes are zero for $i = 0$ and $i = N$, and D and D^* are the finite difference operators defined in section 3.1. Following the same strategy as in the continuous case, we can write the discrete versions of equilibrium schemes (21) and (22). In that case, the discrete equilibrium is the discrete Maxwellian $f_{eq,i} = \exp(\alpha + \beta\varepsilon_i)$ whose coefficients α and β are computed so as f_{eq} has the same discrete mass and energy as f.

It is now an easy matter to prove that these discrete collision operators satisfy weak formulations similar to those satisfied by the continuous model. Hence the discrete analogous properties of Propositions 2.1, 2.2, and 2.4 are satisfied by all these discrete schemes.

4. Linear solvers. In this section, we present a strategy to solve the linear implicit schemes (7, 13, 16, 21, 22), the unknown being f^{n+1}. The present method obeys the conservation properties as soon as the approximation of the collision operator Q is conservative. Here we deal with both the FPL equation (1) with $d = 2$ or 3 and the isotropic case (25). In this section, these models only differ by the dimension of the integration domain: $d = 2$ or 3 for (1), and $d = 1$ for (25).

We first assume that the collision operator is discretized with N velocity points and write the schemes (7, 13, 16) in the following matrix-forms:

contracted scheme (7)	$L^c(f^n)f^{n+1} = f^n,$
θ-scheme (13)	$L_\theta(f^n)f^{n+1} = f^n + \Delta t(1-2\theta)Q(f^n),$
log-linear scheme (16)	$L^l(f^n)f^{n+1} = f^n + \Delta t Q(f^n),$

where $L^c(f)$, $L_\theta(f)$, and $L^l(f)$ are the $N \times N$-matrices defined by

$$L^c(f)g = g - \Delta t\, q^c(f,g), \tag{30}$$

$$L_\theta(f)g = g - \Delta t\, \theta q(f,g), \tag{31}$$

$$L^l(f)g = g - \Delta t\, q^l(f,g) \tag{32}$$

for every vector g in $\mathbb{R}^N$.

We can also consider the following equivalent (and more convenient) δ-form:

$$L^c(f^n)\delta f^n = \Delta t Q(f^n), \tag{33}$$

$$L_\theta(f^n)\delta f^n = \Delta t Q(f^n), \tag{34}$$

$$L^l(f^n)\delta f^n = \Delta t Q(f^n), \tag{35}$$

where $\delta f^n = f^{n+1} - f^n$. Equilibrium implicit schemes (21) and (22) are defined in δ-form by

$$L_\theta(f_{eq})\delta f^n = \Delta t Q(f^n), \tag{36}$$

$$L^l(f_{eq})\delta f^n = \Delta t Q(f^n). \tag{37}$$

Before going to the resolution of these linear systems, we give some of their important algebraic properties.

PROPOSITION 4.1.

- $\Delta t > 0$ $f > 0$ $L^l(f)$ $\frac{1}{f}$
- $\Delta t > 0$ $\theta \in [0,1]$ f_{eq} $L_\theta(f_{eq})$ $\frac{1}{f_{eq}}$
- $f > 0$ $L^c(f)$ $L_\theta(f)$ Δt

. We do not give the proof of this proposition, since it is a direct consequence of the discrete versions of the properties of q^c, q, and q^l given in Propositions 2.1, 2.2, and 2.4. □

These linear systems are nonsparse, with generally large dimension. For a numerical resolution, one can mainly investigate three classes of methods: direct (as LU), iterative nonstationary (as Krylov methods), iterative stationary (as relaxation methods).

Direct methods have been used by Epperlein [9] for the isotropic FPL equation with LU factorization. However, it is clear that such solvers are not usable in multidimensional cases. The complexity of the algorithms is $O(N^3)$, and the memory storage for the matrices is $O(N^2)$. Even in the one-dimensional case as an isotropic equation, the total complexity for a given simulation time is asymptotically the same as the simple explicit scheme (see section 5.1).

Consequently, it is clear that iterative methods must be considered. At this stage, we want to point out a crucial fact: since iterative methods generally construct an approximate solution to the linear system, Indeed, it is questionable to construct perfectly conservative schemes if that property is destroyed by the linear solvers.

To make this point more precise, we define the $(d+2) \times N$ matrix M that associates to every vector $f \in \mathbb{R}^N$ its corresponding $(d+2)$-vector of moments Mf , an approximation of $\int_{\mathbb{R}^d}(1, v, \frac{1}{2}|v|^2)^T f\, dv$). For instance, we can set $Mf = \sum_{i=1}^N (1, v_i, \frac{1}{2}|v_i|^2)^T f_i\, \Delta v^d$ for a regular discretization and $d = 2$ or 3. For the isotropic case, a possible definition is $Mf = \sum_{i=1}^N (1, \varepsilon_i)^T f_i \sqrt{\varepsilon_i} \Delta\varepsilon$. Since we assume the discrete collision operator Q to be conservative, the following relations are satisfied for every f and g in $\mathbb{R}^N$:

$$Mq^c(f,g) = 0, \qquad Mq(f,g) = 0, \quad \text{and} \quad Mq^l(f,g) = 0.$$

This implies the following relations for the matrices (30), (31), (32):

$$ML^c(f) = M, \qquad ML_\theta(f) = M, \quad \text{and} \quad ML^l(f) = M.$$

Multiplying by M the δ-forms (33)–(37) of our implicit schemes is another way to check the conservation property $Mf^{n+1} = Mf^n$.

Consequently, the implicit schemes written under the δ-forms (33)–(37) are of the same type as the following general linear system in $\mathbb{R}^N$:

$$Ax = b,$$

where the $N \times N$-matrix A and N-vector b satisfy

$$MA = M \text{ and } Mb = 0. \tag{38}$$

Then we consider the following problem: find an iterative solver such that if we start with an initial vector $x^{(0)}$ satisfying $Mx^{(0)} = 0$, then any iterate $x^{(k)}$ also satisfies $Mx^{(k)} = 0$. Such solvers will be called [illegible]. They guarantee that the implicit scheme is still conservative even if convergence to the exact solution of the linear system is not achieved.

In what follows, we prove that Krylov subspace methods are conservative iterative linear solvers. In our study, these methods can be set in the following general frame (see [20] for an introduction to Krylov solvers).

ALGORITHM 4.1.

1. [illegible] $x^{(0)}$ [illegible] $Mx^{(0)} = 0$ [illegible] $r^{(0)} = b - Ax^{(0)}$.
2. [illegible] $k = 1$ [illegible] K [illegible] $x^{(k)}$ [illegible] $x^{(0)} + \mathcal{K}_k$ [illegible]

$$\mathcal{K}_k = \{r^{(0)}, Ar^{(0)}, \dots, A^{k-1}r^{(0)}\}.$$

The different versions of Krylov methods arise from different choices of $x^{(k)}$ in $\mathcal{K}_k$. We now prove the following proposition.

PROPOSITION 4.2. [illegible] 4.1 [illegible] $Mx^{(k)} = 0$ [illegible] k [illegible]. Using conservation property on the initial vector $x^{(0)}$ and property (38) yields

$$Mr^{(0)} = M(b - Ax^{(0)}) = 0 - Mx^{(0)} = 0,$$

and therefore $MA^p r^{(0)} = MAA^{p-1}r^{(0)} = MA^{p-1}r^{(0)} = \cdots = Mr^{(0)} = 0$ for every $p \geq 1$. Consequently, we have $M\mathcal{K}_k = \{0\}$, and necessarily $x^{(k)} \in x^{(0)} + \mathcal{K}_k$ implies $Mx^{(k)} = 0$. □

It is remarkable that this conservation property holds even if the linear system is not exactly solved. Another advantage of these methods is well known: they are "matrix-free," i.e., the matrix A only appears in matrix-vector products Ay in the solver. Thus we do not need to form and store the matrix A. Moreover in our case, if the quantities $q^c(f,g)$, $q(f,g)$, and $q^l(f,g)$ can be computed in $O(N)$ operations, then this is also possible for products Ay in Krylov solvers, since $A = L^c(f^n), L_\theta(f^n)$, or $L^l(f^n)$ which are given by (30), (31), (32).

Finally, we would emphasize the fact that all iterative solvers are not necessarily conservative. For instance, consider the general class of stationary methods, such as Jacobi, Gauss–Seidel, or relaxation methods. These methods consist of the splitting matrix A into $A = E - F$, and can be written as the following algorithm.

ALGORITHM 4.2.

1. [illegible] $x^{(0)}$ [illegible] $Mx^{(0)} = 0$ [illegible] $r^{(0)} = b - Ax^{(0)}$.
2. [illegible] $k = 0$ [illegible] K [illegible] $Ex^{(k+1)} = b + Fx^{(k)}$

In this case, we only have $MEx^{(k+1)} = 0 + MFx^{(k)} = M(E-A)x^{(k)} = MEx^{(k)} - Mx^{(k)}$. Even if we assume $Mx^{(k)} = 0$, there is no reason to have $Mx^{(k+1)} = 0$. For instance, with Jacobi method, matrix E is the diagonal part of A, and we do not have any information about the value of ME.

Among the above Krylov methods, we mainly use the GMRES and the CG methods. The CG method is used to solve linear systems whose matrices are positive definite self-adjoint. As stated in Proposition 4.1, this is the case for the log-linear and equilibrium log-linear schemes (16), (22) and the equilibrium θ-scheme (21). This is a real advantage, since the CG method is one of the most efficient Krylov solvers (in terms of CPU cost and memory storage). For the other schemes, we simply use the GMRES method.

TABLE 2
Complexity of explicit and linear implicit schemes for $d = 1, 2, 3$.

	$d = 1$	$d = 2$	$d = 3$
explicit	$O(N^3)$	$O(N^2)$	$O(N^{5/3})$
implicit	$KO(N)$	$KO(N)$	$KO(N)$

5. Complexity and numerical tests.

5.1. Complexity of the algorithms. In this section, we assume that for an arbitrary number of dimensions $d \geq 1$, the number of operations for computing $q^c(f,g), q(f,g)$, and $q^l(f,g)$ is $O(N)$. This is true for $d = 1$ in the case of isotropic equation with Coulomb or Maxwell potential (see [5]). For $d \geq 2$, cost reductions to $O(N \log N)$ can be obtained through rapid algorithms as multipole methods [15] and spectral methods [18]. The cost $O(N)$ could be reached by an extension of isotropic wavelet methods [1] to the multidimensional case. We also assume that the velocity domain is discretized with a step Δv or $\Delta\varepsilon = \frac{1}{n}$ in each direction and a total number of points of $N = n^d$. For a problem with a time scale τ, we want to compare the complexity of the usual Euler explicit scheme (5) and our linear implicit schemes for a given simulation time T.

For the explicit scheme, one iteration requires only one evaluation of $Q(f)$, which costs $O(N)$. Assume that the CFL condition imposes a time step $\Delta t = O(\frac{1}{n^2})$ (this seems to be true from numerical experiments and has been proved in the isotropic case [4]). Then the number of iterations is $\frac{T}{\Delta t} = O(n^2)$. Consequently, the total complexity is $O(N^{1+2/d})$.

For linear implicit schemes with Krylov solvers, the cost of one Krylov iteration is $O(N)$ (since only evaluations of $q^c(f,g)$, $q(f,g)$, and $q^l(f,g)$ are needed). We assume that K iterations of the Krylov solver are necessary to get a correct approximation of the time iterate. Thus the cost of one time iteration is $KO(N)$. Finally, if the time step can be set to τ, the number of time iterations is $\frac{T}{\tau}$, and therefore, the total complexity is $KO(N)$.

In Table 2, we give a comparison of these complexities for $d = 1, 2, 3$. We can see that for $d = 1$, there is an important gain even if $K = N$. For $d = 2$, there is still a gain if $K \ll N$. If $K = O(N)$, the explicit and implicit schemes have the same asymptotic cost $O(N^2)$. For $d = 3$, we have a significant gain only if $K \ll N^{2/3}$. Therefore, the efficiency of the implicit schemes decreases as the dimension increases. Consequently, the reduction of the number K of Krylov iterations becomes necessary in multidimensional cases. This could be done by using adapted preconditioning techniques. As noted by [7], such techniques can render K virtually independent of N. In that case, our implicit schemes would always be advantageous as compared to the explicit scheme. This is the subject of a future work.

5.2. Numerical tests. In order to check the properties of the implicit schemes that we introduced in this work, we present various numerical tests with both the Maxwellian and Coulombian potentials.

Maxwellian potential: comparison with an exact solution. First, in the case of Maxwellian potential, it is proved in [14] that

$$f(t,\varepsilon) = \frac{\rho}{(2\pi T)^{3/2}} \exp\left(-\frac{\varepsilon}{2T}\right)\left(1 + \frac{11}{120}\left(\left(\frac{\varepsilon}{T}\right)^2 - 10\frac{\varepsilon}{T} + 15\right)\exp(-8\rho t)\right)$$

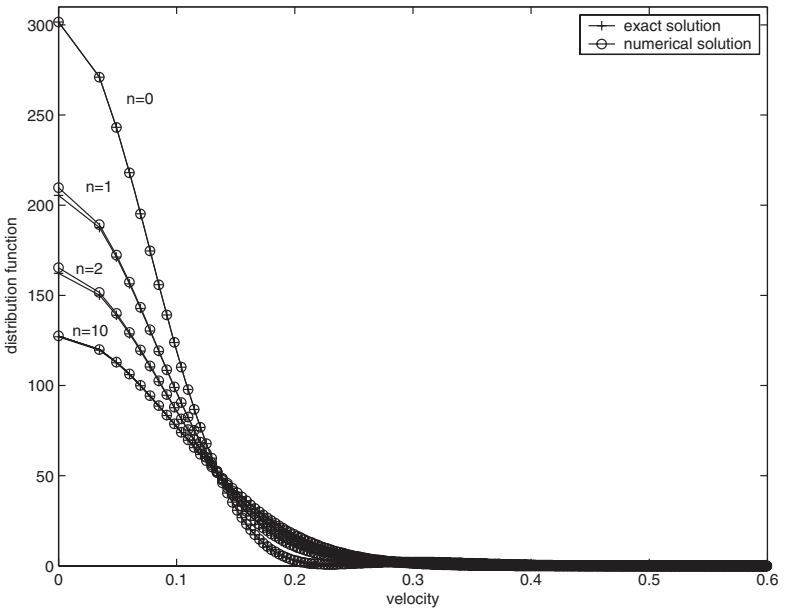


FIG. 1. *Exact solution and implicit scheme for Maxwellian potential: distribution function at different time steps.*

is an exact solution of (25). Then, in Figures 1 and 2, we compare the numerical solution obtained with the second-order θ-scheme (13) to this exact solution. The energy domain is $[0, 2]$ discretized with 500 points. The parameters of the exact solution are $\rho = 2$ and $T = 0.01$. The time step used with the implicit scheme is about 300 times the time step required by an explicit computation. In Figure 1, we plot f as a function of $v = \sqrt{\varepsilon}$ at different time steps. We observe that the numerical solution is very close to the exact one, and the equilibrium is reached in only ten iterations. This corresponds to a final physical time equal to $t_{\max} = 10$. This shows that the dynamics described by the exact FPL equation can efficiently be simulated with a much larger time step than those of usual explicit schemes. This is also clear in Figure 2 where the kinetic entropy is plotted. The slight difference that can be observed in the stationary regime is due to the velocity discretization itself. In fact, the discrete Maxwellian is different from the exact one.

Coulombian potential: regular energy discretization. The second test case uses Coulombian potential with the so-called Rosenbluth initial data:

$$f^0(\varepsilon) = 0.01 \exp(-10((\sqrt{\varepsilon} - 0.3)/0.3)^2).$$

In Figure 3, we plot the kinetic entropy obtained by the explicit scheme (5) with time step Δt_{exp}, which is the largest step ensuring the stability of the scheme. This entropy is compared with that obtained by the contracted implicit scheme (7). For this last scheme, we take a time step $\Delta t_{imp} = 50\Delta t_{exp}$. We use a regular energy discretization of $[0, 2]$ with $N = 100$ points. Tolerance for CG algorithm is 10^{-6}. The dynamics is well described by the implicit scheme even if the time step is much larger. However, the gain in terms of CPU time is not obtained unless the number of energy points is sufficiently large. Indeed, the implicit scheme is really advantageous for $N \geq 500$ only. This is clearly shown in Figure 4 where the CPU time versus N is plotted for

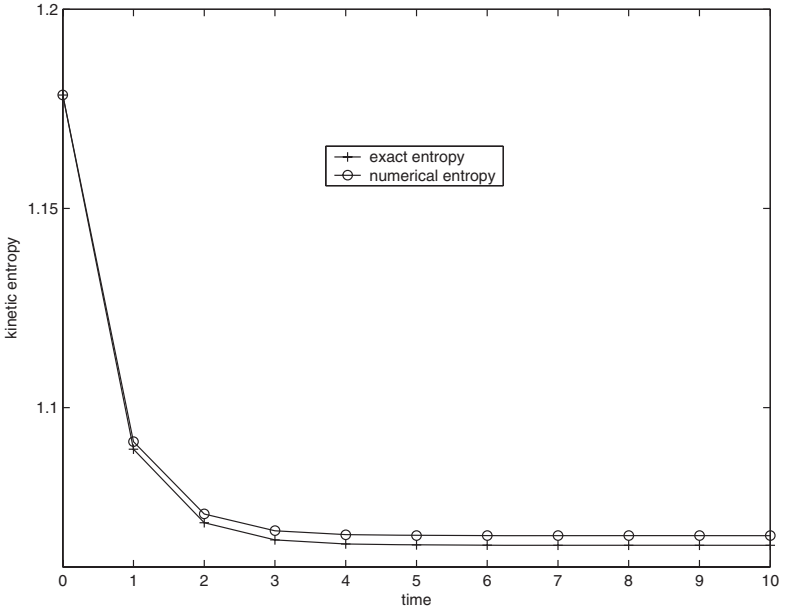


FIG. 2. *Exact solution and implicit scheme for Maxwellian potential: entropy.*

explicit and implicit schemes. According to what we explained in section 5.1, the CPU time of the explicit scheme behaves as $O(N^3)$. A contrario, the implicit scheme only requires $O(KN)$ operations, where K is the number of iterations in the Krylov procedure. It is known that $K \leq N$, and we observe that on this test case, K is much smaller than N. The test in Figure 4 confirms that the numerical complexity of the implicit scheme behaves like $O(N^2)$. Indeed, for $N \geq 500$, the slope of the curve is 1.95 for the implicit scheme whereas it is 2.8 for the explicit scheme.

Coulombian potential: irregular energy discretization. The use of irregular discretizations could be necessary when one is only interested in the dynamic of a localized region in the velocity space for instance or, in general, when using adaptive meshes. In such cases, one can expect a more important gain when using implicit schemes, since the time step of the explicit scheme is constrained by the smallest step size of the irregular mesh.

To illustrate this fact, we perform the previous Rosenbluth test case using an irregular energy discretization where the discrete energy points are $\varepsilon_i = 2(\frac{i}{N})^3$ for $i = 1$ to N. Here we use the equilibrium θ-scheme (11) with $\theta = 1$ and plot on Figure 5 the CPU time versus N for explicit and implicit schemes. We observe that for $N = 50$ points, the implicit scheme is 1.5 times faster than the explicit scheme. The CPU time is further reduced for larger N: it is divided by a factor 6 for $N = 100$ points and by a factor 33 for $N = 200$.

To go further in this direction, we mention that in most cases, the evaluation of the collision operator is very expensive (the present isotropic case is very particular). Example are 3-dimensional geometries with classical or relativistic particles. In these cases, the arguments of the kernel K cannot be separated in a simple way, and the use of fast evaluation algorithms is not as efficient as in the isotropic case. Consequently, in such situations it is attractive to reduce the number of time iterations by using implicit

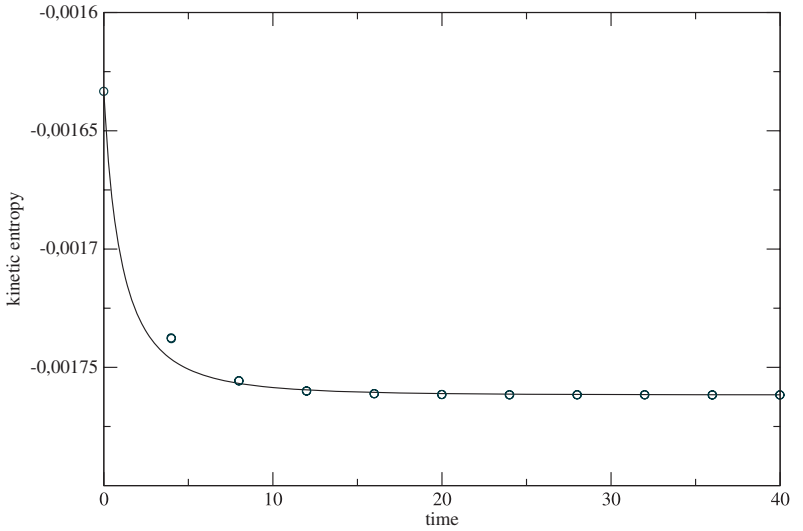


FIG. 3. *Kinetic entropy for explicit* (−) *and contracted implicit scheme* (7) (o) *for Coulombian potential. Case of a regular grid.*

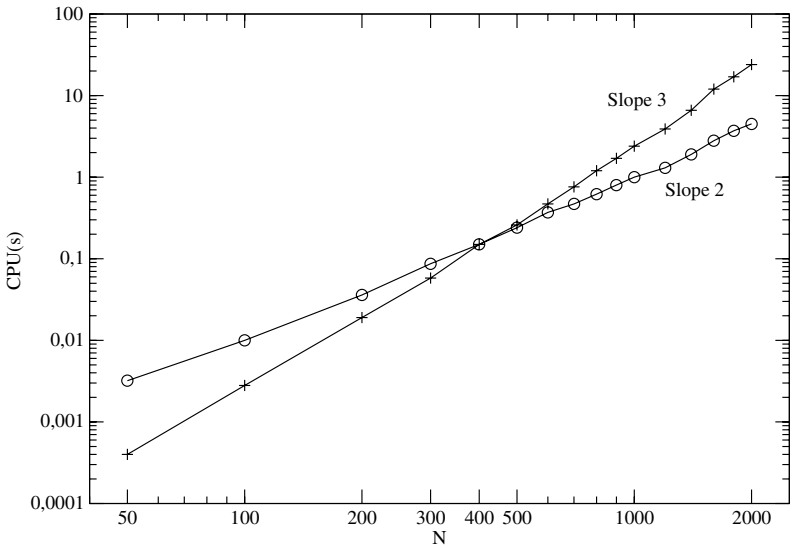


FIG. 4. *CPU cost of explicit* (+) *and contracted implicit scheme* (7) (o) *versus the number N of energy points in a logarithmic scale* (*Coulombian potential*). *Tolerance for CG is* 10^{-6}, *time step in implicit scheme is* 3. *Regular energy grid is used.*

schemes. To illustrate this fact, we consider the isotropic case in which we do not use any fast algorithms: we directly compute the collision operator with formula like (27), (28), which costs $O(N^2)$ operations. Of course, this is just used as a prototype of the previously mentioned realistic situations. In the case of the previous irregular grid with the same initial data, we obtain that for $N = 50$ points, the same implicit scheme is 2 times faster than the explicit scheme. The CPU time is further reduced

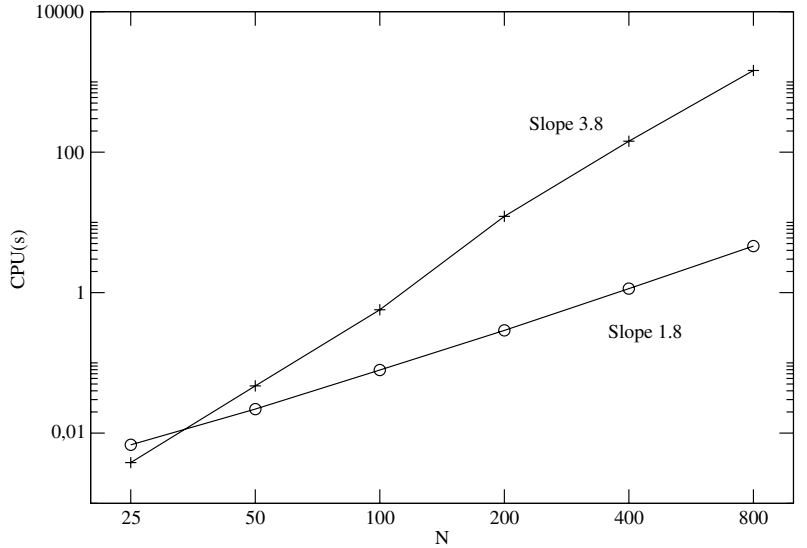


FIG. 5. *CPU cost of explicit* (+) *and equilibrium* θ*-scheme* (11) *with* $\theta = 1$ (o) *versus the number* N *of energy points in a logarithmic scale* (*Coulombian potential*). *Tolerance for CG is* 10^{-6}, *time step in implicit scheme is* 3. *Irregular energy grid is used.*

for $N = 100$ since it is divided by a factor 10.

Comparison of the different implicit schemes. For the same initial data, we compare the different implicit schemes studied in this work to the explicit one: contracted (7), θ-scheme (13), log-linear implicit scheme (16), equilibrium θ-scheme (21), equilibrium log-linear scheme (22). In Figure 6 we plot the fourth order moment $\int f(\varepsilon)\varepsilon^2\sqrt{\varepsilon}\,d\varepsilon$ as a function of time. We only compare first order (in time) schemes with $\Delta t_{imp} = 50\Delta t_{exp}$ and $N = 100$ on a regular grid. We observe that the equilibrium θ-scheme with $\theta = 1$ is the most accurate in this case. Despite its good mathematical properties, the log-linear scheme is not sufficiently accurate. This is probably due to the linearization of the log function.

In Figure 7, we compare our second order (in time) θ-scheme (with $\theta = 0.5$) to a second order explicit scheme (Runge–Kutta scheme). We use the same parameters as in Figure 6 and can see that the two results are in a very good agreement.

Conservation properties. Finally, we point out that all the schemes proposed in this paper are perfectly conservative, up to the machine error. For instance, on the second test case the variation amplitude of density and energy during the time evolution is about 10^{-16}, as for the explicit scheme. This confirms that the approximate resolution of the linear systems does not affect the conservation properties.

6. Conclusion. We have constructed various linear implicit schemes to solve the homogeneous FPL equation. It has been shown that these schemes possess important properties of conservation and entropy. Moreover, numerical tests in the isotropic case have shown a significant gain in terms of CPU time with the same accuracy when compared with usual explicit schemes.

As we pointed out in the introduction, our strategy applies to multidimensional collision operators, but still some details must be investigated in that case. Indeed, the involved linear systems are much larger, and suitable preconditioners are required. We

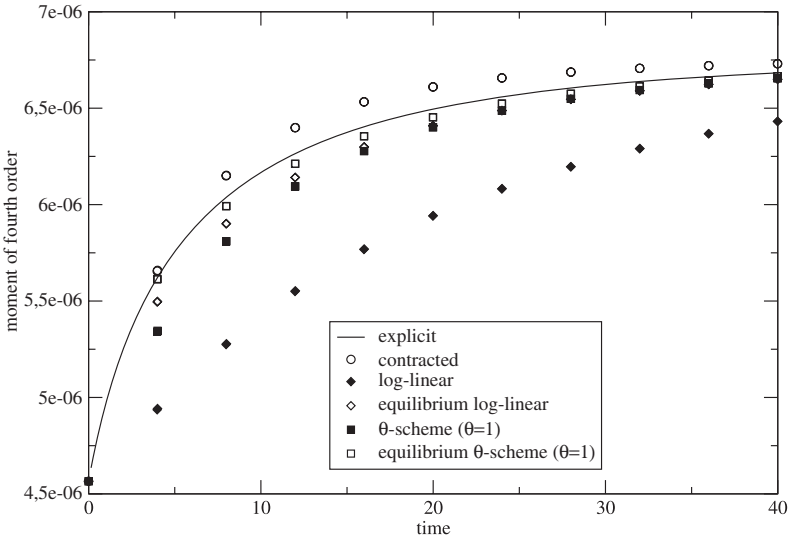


FIG. 6. *Fourth order moment for different implicit schemes.*

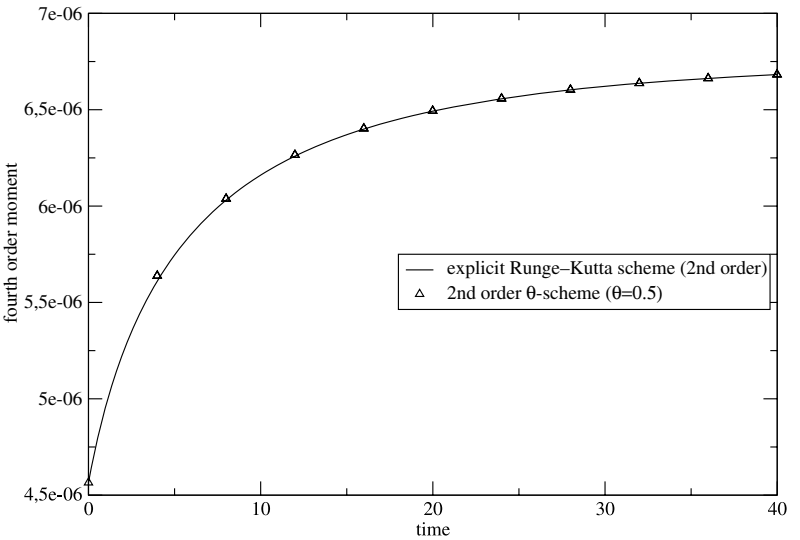


FIG. 7. *Fourth order moment for second order (in time) schemes.*

recall that in a Krylov method with a good preconditioner, the number of iterations is independent of N. Thus the complexity of our implicit schemes could be reduced to $O(N)$ only. Moreover, the rapid matrix-vector product that has been used in the isotropic case cannot be applied to the multidimensional cases. Therefore, to get a fast implicit solver in several dimensions, other acceleration techniques are required. We believe that this can be done using multipole or wavelet techniques [15, 1].

Note that for the inhomogeneous (space dependent) case, our implicit schemes should be more efficient. Indeed, in many inhomogeneous situations the transition phase is rapid and does not need to be accurately described. In such cases, our

implicit schemes allow one to directly reach the hydrodynamic behavior with a few time steps.

Finally, the extension of our strategy to other collision operators (including the relativistic and quantum effects for instance) is the subject of a future work.

REFERENCES

[1] X. Antoine and M. Lemou, *Wavelet approximation of a collision operator in kinetic theory*, C. R. Math. Acad. Sci. Paris., Ser. I, 337 (2003), pp. 353–358.

[2] P. L. Bathnagar, E. P. Gross, and M. Krook, *A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems*, Phys. Rev., 94 (1954), pp. 511–525.

[3] Yu. A. Berezin, V. N. Khudick, and M. S. Pekker, *Conservative finite-difference schemes for the Fokker-Planck equation not violating the law of an increasing entropy*, J. Comput. Phys., 69 (1987), pp. 163–174.

[4] C. Buet and S. Cordier, *Conservative and entropy decaying numerical scheme for the isotropic Fokker-Planck-Landau equation*, J. Comput. Phys., 145 (1998), pp. 228–245.

[5] C. Buet and S. Cordier, *Numerical analysis of the isotropic Fokker-Planck-Landau equation*, J. Comput. Phys., 179 (2002), pp. 43–67.

[6] C. Buet, S. Cordier, P. Degond, and M. Lemou, *Fast algorithms for numerical, conservative, and entropy approximations of the Fokker-Planck-Landau equation*, J. Comput. Phys., 133 (1997), pp. 310–322.

[7] L. Chacón, D. C. Barnes, D. A. Knoll, and G. H. Miley, *An implicit energy-conservative 2D Fokker-Planck algorithm. II. Jacobian-free Newton-Krylov solver*, J. Comput. Phys., 157 (2000), pp. 654–682.

[8] P. Degond and B. Lucquin-Desreux, *An entropy scheme for the Fokker-Planck collision operator of plasma kinetic theory*, Numer. Math., 68 (1994), pp. 239–262.

[9] E. M. Epperlein, *Implicit and conservative difference scheme for the Fokker-Planck equation*, J. Comput. Phys., 112 (1994), pp. 291–297.

[10] F. Filbet and L. Pareschi, *A numerical method for the accurate solution of the Fokker-Planck-Landau equation in the nonhomogeneous case*, J. Comput. Phys., 179 (2002), pp. 1–26.

[11] E. Gabetta, L. Pareschi, and G. Toscani, *Relaxation schemes for nonlinear kinetic equations*, SIAM J. Numer. Anal., 34 (1997), pp. 2168–2194.

[12] A. Girard, C. Lécot, and K. Serebrennikov, *Numerical simulation of the plasma of an electron cyclotron resonance ion source*, J. Comput. Phys., 191 (2003), pp. 228–248.

[13] R. J. Kingham and A. R. Bell, *An implicit Vlasov-Fokker-Planck code to model non-local electron transport in 2-D with magnetic fields*, J. Comput. Phys., 194 (2004), pp. 1–34.

[14] M. Lemou, *Exact solutions of the Fokker-Planck equation*, C. R. Acad. Sci. Paris Sér. 1, 319 (1994), pp. 579–583.

[15] M. Lemou, *Multipole expansions for the Fokker-Planck-Landau operator*, Numer. Math., 78 (1998), pp. 597–618.

[16] M. Lemou and L. Mieussens, *Fast implicit schemes for the Fokker-Planck-Landau equation*, C. R. Acad. Sci. Paris, Sér. I, 338 (2004), pp. 809–814.

[17] E. M. Lifchitz and L. P. Petaevski, *Kinetic Theory*, MIR, Moscow, 1979.

[18] L. Pareschi and B. Perthame, *A Fourier spectral method for homogeneous Boltzmann equations*, Transport Theory Statist Phys., 25 (1996), pp. 369–382.

[19] L. Pareschi, G. Russo, and G. Toscani, *Fast spectral methods for the Fokker-Planck-Landau collision operator*, J. Comput. Phys., 165 (2000), pp. 216–236.

[20] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 831–849
© 2005 Society for Industrial and Applied Mathematics

A MULTIGRID METHOD FOR A FOURTH-ORDER DIFFUSION EQUATION WITH APPLICATION TO IMAGE PROCESSING*

STEFAN HENN†

Abstract. In this paper we are concerned with the efficient solution of the elliptic system $(\alpha L + I)u = f$, where L is the biharmonic operator supplemented by higher-order boundary conditions. This is a core task in image processing arising from the minimization of rotation invariant energy functionals.

For the purpose of numerical solution we approximate the partial differential equation by finite differences. In order to obtain efficient and robust solution methods we discuss how the system can be treated efficiently by multigrid methods. Finally, experimental results are reported that demonstrate the effectiveness of the multigrid approach.

Key words. multigrid methods, nonlinear diffusion equation, biharmonic equation, finite differences, image processing

AMS subject classifications. 65F10, 65M55, 68U10, 73V15

DOI. 10.1137/040611124

1. Introduction.

In recent years there has been an increasing number of papers devoted to the application of partial differential equations (PDEs) in image processing, such as image restoration, segmentation, optical flow computation, and image registration. A good survey of these applications is given in [2, 8, 26] and the references therein. The behavior of image processing problems is often governed by parabolic initial value problems, frequently arising from the minimization of some energy functional. A principle task of scientific computing is the efficient numerical solution of these equations.

In this paper we are concerned with the solution of fourth-order diffusion equations given by the PDE:

$$\left.\begin{array}{rl} \frac{\partial u_i(x,y,t)}{\partial t} + \epsilon(t)\Delta^2 u_i(x,y,t) = d(u(x,y,t)) & \text{on } \Omega \times (0,T) \\ u_i(x,y,0) = z_i(x,y) & \text{on } \Omega \end{array}\right\} \tag{1.1}$$

supplemented by the boundary conditions

$$\left.\begin{array}{l} B_1[u_i(x,y,t)] = \frac{\partial}{\partial n}\Delta u_i(x,y,t) + (1-\sigma)K[u_i(x,y,t)] = 0 \\ B_2[u_i(x,y,t)] = \sigma\Delta u_i(x,y,t) + (1-\sigma)\frac{\partial^2 u_i(x,y,t)}{\partial n^2} = 0 \end{array}\right\} \text{on } \partial\Omega \times (0,T) \tag{1.2}$$

with the outward normal direction $n = (n_x, n_y)$, the tangential direction s, and the operator

$$K[u_i(x,y,t)] := \frac{\partial}{\partial s}\left[\frac{\partial^2 u_i(x,y,t)}{\partial x \partial y}(n_x^2 - n_y^2) + \left(\frac{\partial^2 u_i(x,y,t)}{\partial y^2} - \frac{\partial^2 u_i(x,y,t)}{\partial x^2}\right) n_x n_y\right]$$

for $i = 1, \ldots, m$. Here, the initial condition $z(x,y) = (z_1(x,y), \ldots, z_m(x,y))^t$ represents vector-valued input data, which can be, for instance, a single image ($m = 1$),

*Received by the editors July 6, 2004; accepted for publication (in revised form) March 17, 2005; published electronically November 22, 2005.

http://www.siam.org/journals/sisc/27-3/61112.html

†Mathematisches Institut, Heinrich-Heine Universität Düsseldorf, Universitätsstraße 1, D-40225 Düsseldorf, Germany (henn@am.uni-duesseldorf.de).

an initial motion estimation ($m = 2$), or an initial displacement field ($m = 2$). The behavior of the input data is governed by the so-called data term d. For instance, in the case of image denoising, the data term is given by

$$d(u(x, y, t)) = -2(u(x, y, t) - z(x, y)),$$

where u is to be recovered from the noisy data z; see, e.g., [6, 7, 22, 27]. In the case of two-dimensional mono-modal image registration the data term is given by $d(u(x, y, t)) =$

$$-\nabla T(x - u_1(x, y, t), y - u_2(x, y, t)) \cdot \big(T(x - u_1(x, y, t), y - u_2(x, y, t)) - R(x, y)\big),$$

where $u(x, y, t) = (u_1(x, y, t),\, u_2(x, y, t))^t$ is a displacement vector that should transform one image T, called the template, into another image R, called the reference; see, e.g., see [19]. Although the focus of our research is on image registration problems, for ease of presentation we assume in the following that $z(x, y)$ and $u(x, y)$ are single two-dimensional functions. The solution $u(x, y, t)$ of (1.1) represents the family of scaled versions of $z(x, y)$. The parameter $\epsilon(t) > 0$ for all $t \in (0, T)$ can be regarded as a scale parameter, which captures information at multiple spatial scales of $u(x, y, t)$. Alternatively, we can regard $z(x, y)$ as a thin plate with free edges under the action of a force d during the time t. Here, ϵ and the Poisson coefficient $\sigma \in [0, 1)$ reflect the plate material properties. We choose σ so that the changes in the plate are maximal, i.e., $\sigma = 0$. The diffusion time t is an artificial evolution parameter and Ω is a convex and polygonal domain. In image processing problems Ω will simply be the unit square $[0, 1]^2 \subset \mathbb{R}^2$.

Fourth-order PDEs in digital image processing have also been used in [12] for image registration. Here, the so-called curvature approach uses homogeneous Neumann boundary conditions for u and Δu, i.e., $\frac{\partial u}{\partial n} = 0$ and $\frac{\partial \Delta u}{\partial n} = 0$. The main difference between the two approaches is that the proposed approach is invariant under planar rotation while the curvature approach is not.

Since in the curvature approach no coupling of the boundary conditions is presented, the biharmonic system can be fully decoupled into two Poisson equations. Thus in this case fast Poisson solvers such as fast Fourier transformation (FFT) (see [25]) or straightforward multigrid methods [23] can be used to compute a solution. Unfortunately, for the boundary conditions (1.2) the situation is more sophisticated since the PDEs are coupled via the boundary conditions.

The paper is organized as follows. In section 2, we introduce the fully discrete semi-implicit time discretization of (1.1). Then we discuss the scale-space decomposition of a solution as presented in the companion paper [20] for image registration, and afterwards we consider computational aspects for solving (1.1). Employing semi-implicit time discretization, we are facing in each time step a stationary problem given by an elliptic PDE. The spatial discretization of the stationary problem by finite differences on the unit square is introduced in section 3.

This paper mainly addresses the issue of developing a fast numerical method for solving (1.1). Therefore in section 4, we construct a multigrid method for the fast iterative solution of the resulting linear systems that arise from applying the finite difference method. Numerical examples verify optimal multigrid complexity, i.e., the amount of work is proportional to the number of unknowns. In the fifth section we present a real image registration example that demonstrates the principle and reliability of the biharmonic diffusion approach considered. The last section is devoted to some conclusions and final remarks.

2. Preliminaries. In this section, we first describe the discretization of (1.1) in time, describe the scale-space properties of the equation, and give some preliminary computational considerations.

2.1. Semi-implicit time discretization. In order to avoid the CFL condition for explicit time discretization methods, which couples the length of the time step and the spatial discretization, we opt for an implicit time discretization scheme.

A full implicit scheme for (1.1) usually leads us to solve a nonlinear equation. Fast solution methods for the resulting nonlinear PDE require nonlinear full approximation storage (FAS) multigrid methods (as described, e.g., in [19]) for the image registration problem. However, in this paper, we choose a semi-implicit time discretization scheme which leads us to solve a linear PDE in each time step.

This is at least due to three factors. First, it is often hard to get FAS to work on new problems. Second, the evaluation of the data term d is often needed within the FAS and is usually expensive, especially for image processing problems. Third, in contrast to explicit schemes, the limitation of τ for the implicit scheme is not a delicate problem as far as stability is concerned. But if the problem arises from the minimization of an energy functional, a limitation is required in order to maintain the energy diminishing property during the iteration. In this case suitable choice rules for τ are given in [20].

However, using an implicit scheme for the stiff biharmonic term, an explicit scheme for the remaining data term, and first-order discretization in time leads to the following semi-implicit time discretization scheme:

$$\left.\begin{aligned} \tfrac{u^{(k+1)}-u^{(k)}}{\tau} &= -\epsilon_k \Delta^2 u^{(k+1)}(x,y) + d(u^{(k)}(x,y)) \quad \text{for} \quad (x,y) \in \Omega, \\ B_1[u^{(k+1)}(x,y)] &= B_2[u^{(k+1)}(x,y)] = 0 \qquad\quad \text{for} \quad (x,y) \in \partial\Omega, \end{aligned}\right\}, \tag{2.1}$$

where $\tau > 0$ is the length of the time step.

Let $L = (\Delta^2, B_1, B_2)$; then $Lu = f$ may be interpreted physically as the equation of a plate with free edges subject to a load f. Using L, (2.1) can be written by the linear elliptic system

$$\big(I + \tau\epsilon_k L\big)u^{(k+1)}(x,y) = f(\tau, u^{(k)}(x,y)), \quad (x,y) \in \Omega, \tag{2.2}$$

with right-hand side

$$f(\tau, u^{(k)}(x,y)) = u^{(k)}(x,y) + \tau d(u^{(k)}(x,y)),$$

which depends on τ, the approximation of the previous time step $u^{(k)}$, and the data term d. In order to give a variational formulation of problem (2.2), we introduce the symmetric bilinear form

$$b(u,v) = \epsilon\tau l(u,v) + (u,v) = \big\langle (\epsilon\tau L + I)u, v\big\rangle, \tag{2.3}$$

with the inner product $(u,v) = \int_\Omega uv dx$ and associated norm $||u|| = (u,u)^{1/2}$ for the Lebesgue space of square integrable functions on Ω. In the following, we use the symbol B to refer to the linear bounded operator related to the bilinear form b, and $||\cdot||_B$ stands for the energy norm induced by $b(\cdot,\cdot)$. The bilinear form l

$$l(u,v) = \int_\Omega \Delta u \Delta v + (1-\sigma)\left\{2\frac{\partial^2 u}{\partial x \partial y}\frac{\partial^2 v}{\partial x \partial y} - \left(\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial x^2}\frac{\partial^2 u}{\partial y^2}\right)\right\} d\Omega$$

is for $\sigma \in [0,1)$ symmetric and positive semidefinite on $H^2(\Omega)$ and symmetric positive definite on

$$\left\{ v \in H^2(\Omega), \int_\Omega v dx = \int_\Omega x v dx = \int_\Omega y v dx = 0 \right\} \subset H^2(\Omega).$$

Since ϵ and τ are positive the bilinear form $b(\cdot,\cdot)$ is coercive and bounded on $H^2(\Omega)$. Consequently the Lax–Milgram theorem can be used to prove the existence and uniqueness of the variational equation

$$b(u,v) = \langle f, v \rangle \quad \text{for all} \quad v \in H^2(\Omega) \tag{2.4}$$

for any functional $f \in V'$, the dual space of $H^2(\Omega)$.

2.2. Multiscale decomposition of u. In order to analyze multiscale properties of a weak solution $u \in H^2(\Omega)$ of (2.4), we consider

$$u = \sum_{i=1}^{\infty} \lambda_i^{-1} \langle f, \phi_i \rangle \phi_i.$$

The orthonormal system $\Phi = \{\phi_i\}_{i=1}^{\infty}$ is given by the set of eigenvectors of B. For the countable set of eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$ of B, it holds that

$$1 = \lambda_1 = \lambda_2 = \lambda_3 < \lambda_4 \le \lambda_5 \le \cdots \le \lim_{j\to\infty} \lambda_j = \infty, \text{ with } \quad \lambda_j = 1 + \epsilon\tau\sigma_j, \tag{2.5}$$

with eigenvalues $0 = \sigma_1 = \sigma_2 = \sigma_3 < \sigma_4 \le \sigma_5 \le \cdots \le \lim_{j\to\infty} \sigma_j = \infty$ of L.

For the corresponding orthogonal system $[\phi_1^n, \ldots, \phi_n^n]$ of eigenfunctions of a n-dimensional subspace V_n of $H^2(\Omega)$ with corresponding positive eigenvalues $(1+\epsilon\tau\sigma_i)^{-1}$, the solution is given by

$$u_h = \sum_{i=1}^{n} \frac{1}{1+\epsilon\tau\sigma_i} \langle \phi_i^n, f_h \rangle \phi_i^n. \tag{2.6}$$

Note that this is the Galerkin approximation for the orthonormal system Φ. The ellipticity of the bilinear form b implies that the eigenvalue λ_i belonging to an eigenfunction ϕ_i of L depends only on the frequency of the eigenfunction and becomes larger the higher this frequency. Thus, the eigenspace (λ_i, ϕ_i) of the operator B can be regarded as a sequence of spatial scales; see [20]. Here, the coarsest scale is given by the affine-linear functions corresponding to the eigenvalues $\lambda_1 = \lambda_2 = \lambda_3 = 1$.

As a consequence of these considerations iteration (1.1) is embedded into a so-called scale-space framework, which efficiently treats different scales of u; see [20]. Here, the iteration starts with a large initial scale parameter, i.e., $\epsilon_0 \gg 0$. During the iteration the parameter is reduced by $\epsilon_k := \gamma^k \epsilon_0$ with some decay rate $\gamma \in (0,1)$, and the solution incorporates an increasing number of finer scaled functions.

2.3. Computational considerations. The kernel of the operator L is a linear subspace of $H^2(\Omega)$ given by

$$\ker(L) = \left\{ k \ : \ k = \sum_{i=1}^{3} s_i \phi_i, \quad \phi_i \in \{1, x, y\}, \ s_i \in \mathbb{R} \right\}.$$

Every element $u \in H^2(\Omega)$ can be uniquely decomposed into the sum $u = k + w$, where $k \in \ker(L)$ and $w \in \ker^{\perp}(L)$. Consequently, $H^2(\Omega)$ is the orthogonal sum of $\ker(L)$ and $\ker^{\perp}(L)$, i.e.,

$$H^2(\Omega) = \ker^{\perp}(L) \oplus \ker(L),$$

and numerical solution of (2.1) requires a different treatment reflecting the different character of the problem for these components. It is well known that the performance of multigrid solution methods for linear discrete variational problems crucially depends on the ellipticity of the underlying bilinear form. For ϵ large, the bilinear form $b(\cdot,\cdot)$ has a very small ellipticity constant. Nevertheless, if we archive an integrability condition good multigrid performance (independent of ϵ) can be obtained (see section 4). For $\epsilon \to \infty$ the solution is restricted to $\ker(L)$. Here the bilinear form agrees with the $L_2(\Omega)$ inner product. In this case the solution is given by $u = \sum_{j=1}^{3} s_j \phi_j$, where the s_j are given by the unique solution of the following linear 3×3 system:

$$\sum_{j=1}^{3} (\phi_j, \phi_i) s_j = (f, \phi_i), \ \phi_i \in \ker(L) \quad \text{for } i = 1, 2, 3.$$

On the other hand, for small ϵ the bilinear form $b(\cdot,\cdot)$ has a proper elliptic character on $\ker^{\perp}(L)$. Thus the use of standard multigrid schemes for the PDE (2.1) is possible.

3. The finite difference method for the biharmonic diffusion equation. This section concerns the problem of discretization and approximation of the PDE (2.1) on the unit square $\Omega = [0,1]^2$. We choose the finite difference method, which is widely used in image processing. This is a natural approach here, due to the fact that digital images are typically encoded by uniformly distributed picture elements (pixels).

3.1. Discretization. We use the regular grid

$$G_h = \big\{ x \in \mathbb{R}^2 : x = (ih_x, jh_y), \quad i, j = -\infty, \ldots, \infty \big\}$$

and get $\Omega_h = G_h \cap \Omega$ with $(N_x + 1) \times (N_y + 1)$ grid points

$$g^h_{i,j} = (x_i, y_j) = (ih_x, jh_y) \in \Omega, \ 0 \le i \le N_x, \ 0 \le j \le N_y.$$

The pixel width h_x and pixel height h_y are given by $(h_x, h_y) = (N_x^{-1}, N_y^{-1})$. In the following we assume that $h = hx = hy$ (resp., $N = N_x = N_y$) and denote by u_h a grid function defined by $u_h(i,j) = u(g^h_{i,j})$.

3.2. Operator splitting. Various approaches for the numerical solution of the biharmonic equation have been considered in the literature. For example, the well-known 13-point approximation with truncation error of order h^2 is given by

$$\begin{aligned}\frac{1}{h^2}\Big(&20u_h(i,j) - 8\big(u_h(i+1,j) + u_h(i-1,j) + u_h(i,j+1) + u_h(i,j-1)\big)\\ &+ 2\big(u_h(i+1,j+1) + u_h(i-1,j-1) + u_h(i-1,j+1) + u_h(i+1,j-1)\big)\\ &+ u_h(i+2,j) + u_h(i-2,j) + u_h(i,j+2) + u_h(i,j-2)\Big) = f_h(i,j),\end{aligned}$$

which needs to be modified at the grid points near the boundary; see [14]. Unfortunately, even conventional iterative methods such as Jacobi or Gauss–Seidel either

converge very slowly or diverge; also, the smoothing factor of the lexicographical Gauss–Seidel relaxation is not satisfactory and leads to poor multigrid efficiency [24]. Moreover, direct solvers for the resulting systems of linear equations can only be used for moderate grid size h [10].

The scalar biharmonic equation $\Delta\Delta u = f$ can be more efficiently solved as a system of two Poisson equations (see [9]):

$$\begin{pmatrix} \Delta & 0 \\ -I & \Delta \end{pmatrix} \begin{pmatrix} v(x,y) \\ u(x,y) \end{pmatrix} = \begin{pmatrix} f(x,y) \\ 0 \end{pmatrix}. \tag{3.1}$$

This approach has been used by many authors [1, 4, 5, 11, 15, 18]. In the case of simply supported edges, with boundary conditions $u(x,y) = v(x,y) = 0$, the biharmonic system is fully decoupled and can be solved efficiently with the same amount of work as solving two Poisson equations with Dirichlet boundary conditions.

Dirichlet boundary conditions for the biharmonic equation $u(x,y) = \frac{\partial u(x,y)}{\partial n} = 0$ lead to a system with undefined boundary conditions for the new variable $v(x,y)$. Consequently, the effectiveness of standard multigrid solvers near the boundary is much less than for interior points. Plenty of multigrid approaches for the biharmonic Dirichlet problem have been investigated to treat the slowness; see, e.g., [1, 4, 5, 18]. Due to the splitting (3.1), the elliptic system (2.1) can be written as

$$\Delta u(x,y) - v(x,y) = 0 \quad \text{and} \quad \alpha\Delta v(x,y) + u(x,y) = f(x,y)$$

for all $(x,y) \in \Omega$ and boundary conditions given as

$$\sigma v(x,y) + (1-\sigma)\frac{\partial^2 u(x,y)}{\partial n^2} = 0 \quad \text{and} \quad \frac{\partial v(x,y)}{\partial n} + (1-\sigma)K[u(x,y)] = 0$$

for all $(x,y) \in \partial\Omega$. By using $\Delta u = \frac{\partial^2 u}{\partial n^2} + \frac{\partial^2 u}{\partial s^2}$ and setting $\alpha := \tau\epsilon_k$ and $\sigma = 0$, the solution of (2.2) consists of solving the two Poisson-like equations

$$\begin{cases} \alpha\Delta v(x,y) + u(x,y) = f(x,y) & \text{for } (x,y) \in \Omega, \\ \frac{\partial v(x,y)}{\partial n} + K[u(x,y)] = 0 & \text{for } (x,y) \in \partial\Omega \end{cases} \tag{3.2}$$

and

$$\begin{cases} \Delta u(x,y) - v(x,y) = 0 & \text{for } (x,y) \in \Omega, \\ \frac{\partial^2 u(x,y)}{\partial n^2} = v(x,y) - \frac{\partial^2 u(x,y)}{\partial s^2} = 0 & \text{for } (x,y) \in \partial\Omega, \end{cases} \tag{3.3}$$

which are coupled via $u(x,y)$ and $v(x,y)$. We have to solve the following system:

$$\begin{pmatrix} L_v & I+K \\ -I & L_u \end{pmatrix} \begin{pmatrix} v(x,y) \\ u(x,y) \end{pmatrix} = \begin{pmatrix} f(x,y) \\ 0 \end{pmatrix}, \tag{3.4}$$

where I denotes the identity on Ω and

$$L_v v(x,y) = \begin{cases} \alpha\Delta v(x,y) & \text{on } \Omega, \\ \frac{\partial v(x,y)}{\partial n} & \text{on } \partial\Omega \end{cases} \quad \text{as well as} \quad L_u u(x,y) = \begin{cases} \Delta u(x,y) & \text{on } \Omega, \\ \frac{\partial^2 u(x,y)}{\partial s^2} & \text{on } \partial\Omega. \end{cases}$$

3.3. Approximation of the system. Let $D^+_{x,h}$, $D^+_{y,h}$ be the finite forward difference, $D^-_{x,h}$, $D^-_{y,h}$ be the finite backward difference, and $D^s_{x,h}$, $D^s_{y,h}$ be the finite symmetric discretization approximation at the grid points $g^h_{i,j}$ of $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$, respectively. The second derivative $\frac{\partial^2}{\partial x^2}$ is discretized as the operator $D^+_{x,h}D^-_{x,h}$ and Δ by standard five-point approximation denoted by $\Delta_h := D^+_{x,h}D^-_{x,h} + D^+_{y,h}D^-_{y,h}$. Then the discretized (3.1) is given by

$$\begin{aligned} \alpha\Delta_h v_h(x,y) + u_h(x,y) &= f_h(x,y), \\ \Delta_h u_h(x,y) - v_h(x,y) &= 0 \end{aligned} \tag{3.5}$$

on Ω.

3.4. Approximation of the boundary conditions. In order to apply the finite difference method at the boundary of (2.2), we first consider the solution u of the plate equation with free boundary conditions $Lu = f$ on the unit square. It is well known that u has the regularity $u \in H^2(\Omega) \cap H^3(\Omega)$ and satisfies $||u||_3 \le c||f||_{-1}$. In the case $f \in L_2(\Omega)$, a weak solution of

$$l(u,v) = (f,v) \quad \text{for all } v \in H^2(\Omega)$$

is also a strong solution on each $\Omega' \subset \Omega$ with a small distance of $\partial\Omega'$ from the corner points of Ω. We refer to [3, 13] and references therein for more discussions on well-posedness, regularities, and restrictions for biharmonic problems on polygonal domains.

3.4.1. Finite difference discretization of L_u. A suitable discretization at the boundary of (3.3) can be found by approximating the boundary $\partial\Omega$ near the corner points by sufficiently smooth functions; see Figure 1.

In the following we denote this modified domain by Ω'. In order to solve the PDE numerically by a finite difference discretization approach, we map the nearest point on $\partial\Omega'$ to the corresponding corner point of Ω_h. If the distance between these two

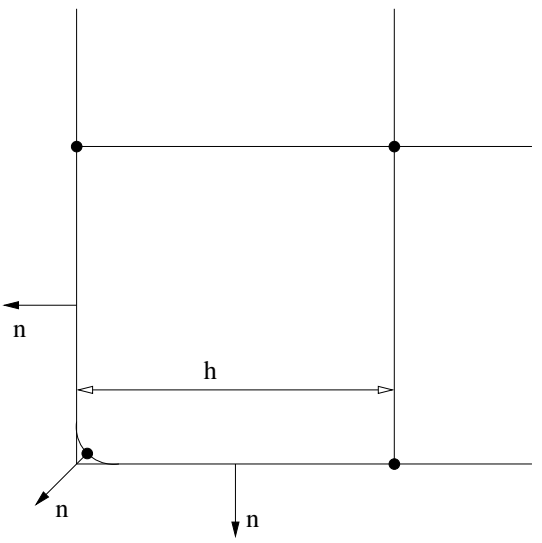


FIG. 1. *Discretization at the corner.*

points is small (in comparison to the mesh size h), the rates of convergence of the multigrid algorithm (presented in section 4) are conserved. This approach leads to the following normal directions in the corner points of Ω_h:

$$n = \frac{1}{\sqrt{2}} \begin{cases} (1,-1)^t & \text{at} \quad (N,0), \\ (-1,1)^t & \text{at} \quad (0,N) \end{cases} \quad \text{and} \quad n = \frac{1}{\sqrt{2}} \begin{cases} -(1,1)^t & \text{at} \quad (0,0), \\ (1,1)^t & \text{at} \quad (N,N). \end{cases} \tag{3.6}$$

Using this modification we get the following result.

PROPOSITION 3.1. [illegible] (3.3) [illegible]

$$v(x,y) = \begin{cases} \frac{\partial^2 u(x,y)}{\partial x^2} & \text{on} \quad \Gamma_S, \Gamma_N, \\ \frac{\partial^2 u(x,y)}{\partial y^2} & \text{on} \quad \Gamma_W, \Gamma_E \end{cases} \qquad v(x,y) = \begin{cases} 2\frac{\partial^2 u(x,y)}{\partial x \partial y} & \text{on} \quad (0,N),(N,0), \\ -2\frac{\partial^2 u(x,y)}{\partial x \partial y} & \text{on} \quad (0,0),(N,N), \end{cases}$$

[illegible] Γ_S Γ_N Γ_W [illegible] Γ_E [illegible]. By using

$$\frac{\partial^2 u(x,y)}{\partial s^2} = \left(n_y^2 \frac{\partial^2 u(x,y)}{\partial x^2} + n_x^2 \frac{\partial^2 u(x,y)}{\partial y^2} - 2 n_x n_y \frac{\partial^2 u(x,y)}{\partial x \partial y} \right)$$

the assumption for Γ_S, Γ_N, Γ_W, and Γ_E follows directly. For the corner points we get, because of (3.6),

$$\frac{\partial^2 u(x,y)}{\partial s^2} = \begin{cases} \frac{1}{2}\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} \right) - \frac{\partial^2 u(x,y)}{\partial x \partial y} & \text{on } (0,N),(N,0), \\ \frac{1}{2}\left(\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} \right) + \frac{\partial^2 u(x,y)}{\partial x \partial y} & \text{on } (0,0),(N,N). \end{cases}$$

Inserting this into the boundary condition $v(x,y) - \frac{\partial^2 u(x,y)}{\partial s^2} = 0$, we get the assumption, due to $\Delta u(x,y) = v(x,y)$. □

By this result we are now in the position to discretize L_u by finite differences.

PROPOSITION 3.2. [illegible] Ω [illegible] (3.3)

(a) [illegible] $D^+_{x,h} D^-_{x,h}$ [illegible] $\frac{\partial}{\partial x^2}$ [illegible] $D^+_{y,h} D^-_{y,h}$ [illegible] $\frac{\partial}{\partial y^2}$ [illegible]

$$v(x,y) = \frac{\partial^2 u}{\partial s^2}(x,y) \quad \text{on} \quad \partial\Omega$$

[illegible]

$$v_h(i,j) = \frac{1}{h^2}\big(u_h(i,j-1) - 2u_h(i,j) + u_h(i,j+1)\big) \tag{3.7}$$

[illegible] $(i,j) \in \Gamma_{W,O}$ [illegible]

$$v_h(i,j) = \frac{1}{h^2}\big(u_h(i-1,j) - 2u_h(i,j) + u_h(i+1,j)\big) \tag{3.8}$$

[illegible] $(i,j) \in \Gamma_{N,S}$

(b) [illegible]

$$\frac{\partial^2}{\partial x \partial y} : \frac{h^{-2}}{2} \begin{bmatrix} 0 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix} \tag{3.9}$$

[illegible] $\frac{\partial^2 u(x,y)}{\partial x \partial y}$ [illegible] $(0,N)$ [illegible] $(N,0)$ [illegible]

$$v_h(N,0) = \frac{1}{h^2}\big(u_h(N-1,0) + u_h(N,1) - u_h(N-1,1) - u_h(N,0)\big)$$

[illegible]

$$v_h(0,N) = \frac{1}{h^2}\big(u_h(0,N-1) + u_h(1,N) - u_h(1,N-1) - u_h(0,N)\big).$$

(c) [illegible]

$$\frac{\partial^2}{\partial x \partial y} : \frac{h^{-2}}{2}\begin{bmatrix} -1 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -1 \end{bmatrix} \tag{3.10}$$

[illegible] $\frac{\partial^2 u(x,y)}{\partial x \partial y}$ [illegible] $(0,0)$ [illegible] (N,N) [illegible]

$$v_h(0,0) = \frac{1}{h^2}\big(u_h(1,0) + u_h(0,1) - u_h(1,1) - u_h(0,0)\big)$$

[illegible]

$$v_h(N,N) = \frac{1}{h^2}\big(u_h(N-1,N) + u_h(N,N-1) - u_h(N-1,N-1) - u_h(N,N)\big).$$

[illegible]

(a) The proof of this part is straightforward.

(b) By using (3.9) to approximate $\frac{\partial^2 u(x,y)}{\partial x \partial y}$ on $(N,0)$, we get

$$v_h(N,0) - \frac{1}{h^2}\begin{bmatrix} 0 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 0 \end{bmatrix} u_h(N,0) = 0.$$

By adding equation $\frac{\partial^2 u(x,y)}{\partial n^2} = 0$ at $(N,1)$ and $(N-1,0)$, we get, by using $D^+_{x,h}D^-_{x,h}$ and $D^+_{y,h}D^-_{y,h}$ to approximate $\frac{\partial^2 u(x,y)}{\partial n^2}$,

$$v_h(N,1) - \frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 \\ 1 & 2 & -1 \\ 0 & -1 & 0 \end{bmatrix} u_h = 0.$$

By adding $\Delta_h u_h(N,0) - v_h(N,0) = 0$ to this equation, we get

$$v_h(N,0) - \frac{1}{h^2}\begin{bmatrix} -1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} u_h(N,0) = 0.$$

By the same approach we get the approximation for $(0,N)$.

(c) By using (3.10) to approximate u_{xy} we get the approximations on $(0,0)$ and (N,N), respectively. □

3.4.2. Finite difference discretization of L_v. The boundary value problem (3.2) can be seen as a Poisson equation with Neumann boundary conditions perturbed by the operator K. In order to discretize the equation, we follow the approach given in [17] with second-order approximations for the derivates in outward normal direction. The proof of the following result is straightforward.

PROPOSITION 3.3. [illegible] Ω [illegible] K [illegible]

$$K[u(x,y)] = \begin{cases} -\frac{\partial^3 u(x,y)}{\partial x^2 \partial y} & \text{[illegible]} \ \Gamma_S, \\ \frac{\partial^3 u(x,y)}{\partial x^2 \partial y} & \text{[illegible]} \ \Gamma_N \end{cases} \qquad K[u(x,y)] = \begin{cases} -\frac{\partial^3 u(x,y)}{\partial x \partial y^2} & \text{[illegible]} \ \Gamma_W, \\ \frac{\partial^3 u(x,y)}{\partial x \partial y^2} & \text{[illegible]} \ \Gamma_E. \end{cases}$$

Using Proposition 3.3 leads to the following discretization of K on the unit square.

PROPOSITION 3.4. [illegible] $D^+_{x,h}$ $D^-_{x,h}$ [illegible] $\frac{\partial}{\partial x}$ [illegible] $D^+_{y,h}D^-_{y,h}$ [illegible] $\frac{\partial^2}{\partial y^2}$ [illegible]

(a) [illegible] K [illegible] Γ_W

$$K_h^W := -D^+_{y,h}D^-_{y,h}D^+_{x,h} = \frac{1}{h^3}\begin{bmatrix} 0 & 1 & -1 \\ 0 & -2 & 2 \\ 0 & 1 & -1 \end{bmatrix}. \tag{3.11}$$

(b) [illegible] K [illegible] Γ_E

$$K_h^E := D^+_{y,h}D^-_{y,h}D^-_{x,h} = \frac{1}{h^3}\begin{bmatrix} -1 & 1 & 0 \\ 2 & -2 & 0 \\ -1 & 1 & 0 \end{bmatrix}. \tag{3.12}$$

[illegible] $D^+_{y,h}$ $D^-_{y,h}$ [illegible] $\frac{\partial}{\partial y}$ [illegible] $D^+_{x,h}D^-_{x,h}$ [illegible] $\frac{\partial^2}{\partial x^2}$ [illegible]

(c) [illegible] K [illegible] Γ_S

$$K_h^S := -D^+_{x,h}D^-_{x,h}D^+_{y,h} = \frac{1}{h^3}\begin{bmatrix} -1 & 2 & -1 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{3.13}$$

(d) [illegible] K [illegible] Γ_N

$$K_h^N := D^+_{x,h}D^-_{x,h}D^-_{y,h} = \frac{1}{h^3}\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ -1 & 2 & -1 \end{bmatrix}. \tag{3.14}$$

In order to discretize the derivatives of v normal to the boundary, a set of virtual v-points outside of Ω (so-called ghost points) are introduced parallel to the boundary and one mesh size from it, i.e., lines of points $v(-h, y)$, $v(1+h, y)$, $v(x, -h)$, and $v(x, 1+h)$. The virtual v-points are eliminated by imposing at boundary points the discrete versions of the differential equation and the boundary equations simultaneously. For example, we get the following discretization at the left boundary when we use the central difference D^s between the virtual points and the first interior v-points.

PROPOSITION 3.5. [illegible] K_h^W [illegible] (3.11) [illegible] [illegible] (3.2) Γ_W [illegible]

$$\frac{\alpha}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & -4 & 2 \\ 0 & 1 & 0 \end{bmatrix} v_h(0,j) = f_h(0,j) - \frac{2\alpha}{h}K_h^W[u(0,j)] - u(0,j)$$

[illegible] $(0, j)$ Γ_W

Similarly, we get finite difference discretizations for the corner points. For example, we get the following discretization at point $(0,0)$.

PROPOSITION 3.6. K_h^W K Γ_W K_h^S K Γ_S (3.2) $x=(0,0)$

(3.15)
$$\frac{2\alpha}{h^2}\big(v_h(1,0)+v_h(0,1)-2v_h(0,0)\big)=f_h(x)-\frac{2\alpha}{h}\big(K_h^S[u(x)]+K_h^W[u(x)]\big)-u(x).$$

(3.11) (3.13) K_h^S K_h^W $(0,0)$ (3.2):

$$\frac{\alpha}{h^2}\Big(4\big(v_h(0,1)+v_h(1,0)\big)-6v_h(0,0)\Big)$$
$$=f_h(0,0)-\frac{2\alpha}{h}\left(\frac{1}{h^3}\big(u(0,2)-2u(0,1)+2u(0,0)-2u(1,0)+u(2,0)\big)\right)+u(0,0).$$

This yields

$$K_h^S+K_h^W=\frac{1}{h^3}\left(\begin{bmatrix}-1&2&-1\\1&-2&1\\0&0&0\end{bmatrix}+\begin{bmatrix}0&1&-1\\0&-2&2\\0&1&-1\end{bmatrix}\right)=\frac{1}{h^3}\begin{bmatrix}-1&3&-2\\1&-4&3\\0&1&-1\end{bmatrix}.$$

Since $\Delta u-v=0$ at $\partial\Omega$, we have

$$\begin{aligned}(K_h^S+K_h^W)u_h&=(K_h^S+K_h^W)u_h+\frac{1}{h}(-\Delta_h u_h(0,0)+v_h(0,0))\\&\quad+\frac{1}{h}(\Delta_h u_h(0,1)-v_h(0,1))+\frac{1}{h}(\Delta_h u_h(1,0)-v_h(1,0))\\&=\frac{1}{h^3}\Big(u_h(0,2)-2u_h(0,1)+2u_h(0,0)-2u_h(1,0)+u_h(2,0)\Big)\\&\quad+\frac{1}{h}\Big(v_h(0,0)-v_h(0,1)-v_h(1,0)\Big).\end{aligned}$$

Substituting this into (3.15) leads to the result. □

3.5. The discretization matrix. The connection between the discrete operator L_h and its discretization matrix A_h is given by the following theorem. The proof is straightforward and omitted here.

THEOREM 3.1. 3.1–3.6 (3.4) $2\cdot(N+1)^2$

$$\begin{pmatrix}L_{v,h}&I_h+K_h\\-I_h&L_{u,h}\end{pmatrix}\begin{pmatrix}v_h\\u_h\end{pmatrix}=\begin{pmatrix}f_h\\0\end{pmatrix}. \tag{3.16}$$

L

$$L_h=\begin{pmatrix}L_{v,h}&K_h\\-I_h&L_{u,h}\end{pmatrix}.$$

Notice that the matrices $L_{v,h}$, K_h, and $L_{u,h}$ are neither symmetric nor diagonally dominant. It is easy to see that the kernel of L_h is given by

$$\ker(L_h) = \left\{ \alpha_1 \begin{pmatrix} 0 \\ \phi_1^h \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ \phi_2^h \end{pmatrix} + \alpha_3 \begin{pmatrix} 0 \\ \phi_3^h \end{pmatrix}, \quad \alpha_1, \alpha_2, \alpha_3 \in \mathbb{R} \right\}.$$

Here, we identify by ϕ_1^h, ϕ_x^h, ϕ_y^h the vectors (in rowwise ordering) corresponding to the approximation on Ω_h of the two-dimensional functions 1, x, and y.

4. Multigrid treatment of the biharmonic diffusion equation. The aim of this section is to discuss the multigrid treatment of the system (3.4). The computational problem of solving a linear system arising from the discretization of a differential operator is well studied. Good introductions to multigrid computational methods are given, for instance, in [16, 21, 24]. Here, however, there are complications since the resulting discretization matrix is not at all diagonally dominant and consequently conventional iterative methods such as Jacobi or Gauss–Seidel diverge.

4.1. The multigrid scheme. The basic multigrid schemes are the V-cycle and the W-cycle, obtained by solving the coarse grid problem by one (i.e., $\gamma = 1$ in Algorithm 1) or two (i.e., $\gamma = 2$ in Algorithm 1) recursive applications of the multigrid scheme. The so-called F-cycle is given for $\gamma = -1$ in Algorithm 1.

Algorithm 1.

```
function MGS(l, u_l, f_l, γ)
    if l = 0 then
        u_0 ← A_0^{-1} f_0
    else
        for g = 1, ..., |γ| do
            u_l ← S_l(u_l, f_l, ν_1)                          ▷ Presmoothing
            u_{l-1} ← 0
            if γ = −1 then
                u_{l-1} ← MGS(l − 1, u_{l-1}, I_l^{l-1}(f_l − A_l u_l), γ)
            end if
            MGS(l − 1, u_{l-1}, I_l^{l-1}(f_l − A_l u_l), |γ|)
            u_l ← S_l(u_l + I_{l-1}^l u_{l-1}, f_l, ν_2)      ▷ Postsmoothing
        end for
    end if
end function
```

In any multigrid cycle there are two types of intergrid communication. The operators

$$I_l^{l-1} : G_l \to G_{l-1} \quad \text{and} \quad I_{l-1}^l : G_{l-1} \to G_l$$

designate the canonical intergrid transfers, where G_l denotes a grid with mesh size $2^l h$. The fine-to-coarse transfer (restriction) produces an approximation of the fine-grid residual on a coarse-grid G_{l-1}. The coarse-to-fine transfer (prolongation) interpolates the coarse-grid correction to the next finer grid. In our multigrid implementation, we used full weighting for restriction and bilinear interpolation, i.e.,

$$I_{l-1}^l = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_l^{l-1}, \quad I_{l-1}^l = \frac{1}{8} \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix}_l^{l-1}, \quad I_{l-1}^l = \frac{1}{4} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}_l^{l-1}$$

for the inner, edge, and corner points. On all grids the discretization is done in the same way as on the finest grid.

4.2. Boundary relaxation. An important part of multigrid schemes for the solution of discretized elliptic boundary value problems is relaxation. Although the smoothing factors of the Gauss–Seidel relaxation with lexicographical order of the grid points for the biharmonic operator are identical to the factors for Poisson's equation (see [24]), it is well known that finite difference multigrid solvers for the biharmonic operator are very sensitive to the boundary conditions associated with the problem. For instance, the convergence rates for multigrid schemes, by applying a point relaxation for the discretization of the boundary as proposed in section 3, are much slower than predicted for the interior points; see Figure 2.

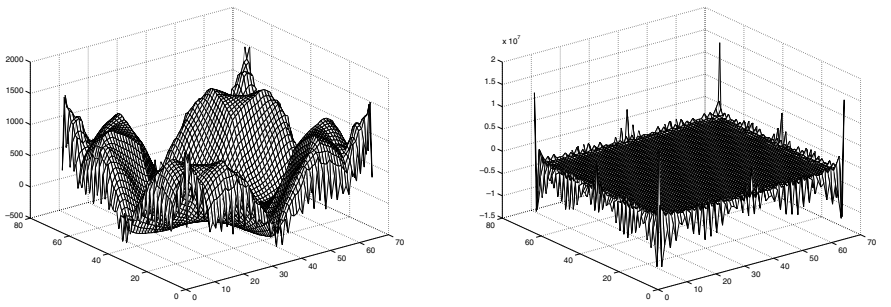


FIG. 2. *Defect after some multigrid $W(1,2)$-cycle by using Gauss–Seidel iteration without special boundary treatment.*

The reason is that the finite difference discretization is not diagonally dominant due to the equations at the boundary. Thus standard relaxations lack the smoothing property and must lose their efficiency. This is a fundamental observation for any local relaxation scheme for nondiagonally dominant operators. Consequently, a special treatment of the equations corresponding to the boundary points is necessary to attain full multigrid efficiency. It has been shown by Brandt in [4] for the biharmonic Dirichlet problem that the multigrid efficiency predicted by local mode analysis for the interior smoothing rates can be obtained by a special treatment of the boundary. Following this idea we use the following steps for a relaxation step S_l:

(1) For $\Gamma \in \{\Gamma_O, \Gamma_N, \Gamma_W, \Gamma_S\}$: A simultaneous relaxation of all boundary points $(i,j) \in \Gamma$ and the points near the boundary is performed. This means that all unknowns on Γ (including the corner points) and all points of the next q lines/columns (see Figure 3) are updated collectively.

(2) Simultaneous point relaxation is performed at interior points, meaning that new values $u^{new}(i,j)$ and $v^{new}(i,j)$ for $1 \le i \le N-1$, $1 \le j \le N-1$ are computed for each time the point (i,j) is relaxed. For instance, the Gauss–Seidel relaxation with lexicographical order of the grid points leads us to solve the 2×2 system

$$\begin{pmatrix} -4 & \alpha \\ 1 & -4 \end{pmatrix} \begin{pmatrix} u^{new}(i,j) \\ v^{new}(i,j) \end{pmatrix} = \begin{pmatrix} f(i,j) - \Sigma_u(i,j) \\ -\Sigma_v(i,j) \end{pmatrix}$$

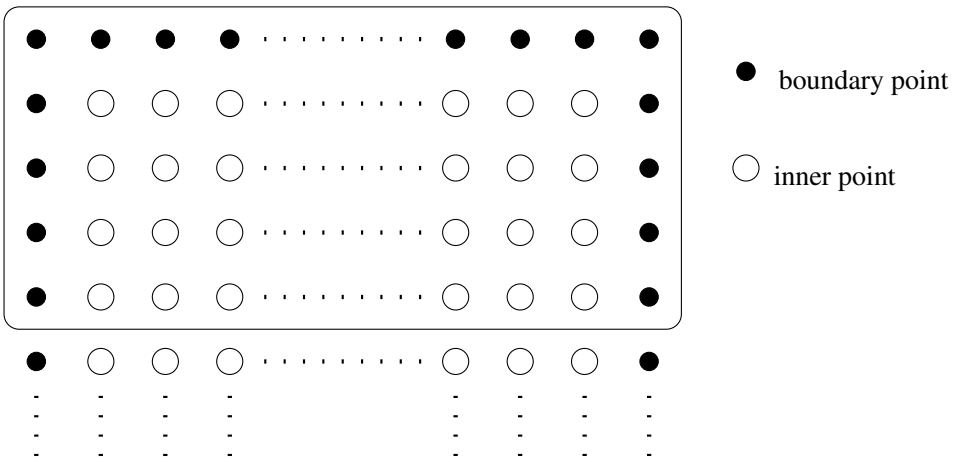


FIG. 3. *Collective line relaxation near upper boundary* Γ_N.

for all $1 \le i \le N-1$, $1 \le j \le N-1$, where

$$\Sigma_u(i,j) = u^{new}(i-1,j) + u^{new}(i,j-1) + u^{old}(i+1,j) + u^{old}(i,j+1)$$

and

$$\Sigma_v(i,j) = v^{new}(i-1,j) + v^{new}(i,j-1) + v^{old}(i+1,j) + v^{old}(i,j+1).$$

The treatment of the boundary points (1) leads to tridiagonal matrices, which can be solved effectively by an amount of work proportional to the number N of all grid points $\mathcal{O}(N)$.

4.3. Multigrid convergence. We apply the proposed multigrid method to the discretization presented in section 3 using the damped Jacobi relaxation with an underrelaxation parameter $\omega = 0.8$ to carry out the relaxation in the interior of Ω. For boundary treatment we solve five lines/columns collectively as outlined in the previous section. On the coarsest grid (with 5×5 grid points due to the boundary relaxation) we solve the problem exactly by using a direct Gaussian elimination scheme.

The asymptotic convergence rates are measured by the geometric average defect factor over k iterations, given by

$$\hat{q}^{(k)} := \sqrt[k]{\frac{||d_h^k||}{||d_h^0||}}.$$

We solve the linear system (3.4) for $\epsilon = \tau = 1$ (i.e., $\alpha = 1$) using a zero initial guess and a right-hand side f with random entries (chosen from a uniform distribution on the interval $[0, 100]$) for different mesh sizes h_l. In Table 4.1 the multigrid convergence of the $W(1,2)$-cycle (meaning 1 pre- and 2 postsmoothing steps) for different step sizes of the finest grid are presented.

Here, the convergence rates of our multigrid implementation are approximately 0.195 and independent of the grid size h_l. These rates can be improved by using a $W(3,0)$-cycle ($\hat{q}^{(10)} = 0.180$) or if the number of smoothing steps is increased; see

TABLE 4.1
The average defect reduction factor over 10 iterations $\hat{q}^{(10)}$ for a $W(1,2)$-cycle.

Cycle	$h = \frac{1}{32}$	$h = \frac{1}{64}$	$h = \frac{1}{128}$	$h = \frac{1}{256}$
$\hat{q}^{(10)}$	0.195	0.194	0.193	0.195

TABLE 4.2
W-cycle convergence rates $\hat{q}^{(10)}$ of our multigrid implementation for different choices of pre- and postsmoothing steps and grid size $h = \frac{1}{256}$.

Cycle	$W(1,0)$	$W(0,1)$	$W(2,0)$	$W(1,1)$	$W(3,0)$	$W(2,1)$	$W(4,0)$
$\hat{q}^{(10)}$	0.425	0.555	0.269	0.288	0.180	0.196	0.124

TABLE 4.3
V-cycle convergence rates $\hat{q}^{(10)}$ of our multigrid implementation for different choices of pre- and postsmoothing steps and grid size $h = \frac{1}{256}$.

Cycle	$V(1,0)$	$V(0,1)$	$V(2,0)$	$V(1,1)$	$V(3,0)$	$V(2,1)$	$V(4,0)$
$\hat{q}^{(10)}$	n.c.	n.c.	0.775	0.681	0.347	0.296	0.354

TABLE 4.4
F-cycle convergence rates $\hat{q}^{(10)}$ of our multigrid implementation for different choices of pre- and postsmoothing steps and grid size $h = \frac{1}{256}$.

Cycle	$F(1,0)$	$F(2,0)$	$F(1,1)$	$F(0,2)$	$F(3,0)$	$F(4,0)$
$\hat{q}^{(10)}$	0.607	0.330	0.303	0.314	0.186	0.129

Table 4.2. Moreover, it can be seen that the presented $W(\nu_1, \nu_2)$-cycle convergence factors are roughly the same as predicted by the local Fourier analysis (LFA); cf. [24, p. 305].

By using V-cycle we observe no convergence (n.c.) by using only one smoothing step and a convergence factor of the $V(1,1)$ of 0.681; see Table 4.3.

This behavior of the multigrid V-cycle has been also observed for the biharmonic Dirichlet problem; see [24]. This effect is caused by the boundary conditions, since in the case of simply supported edges (see section 3.2) we obtain good convergence factors for the same multigrid scheme in the interior of Ω.

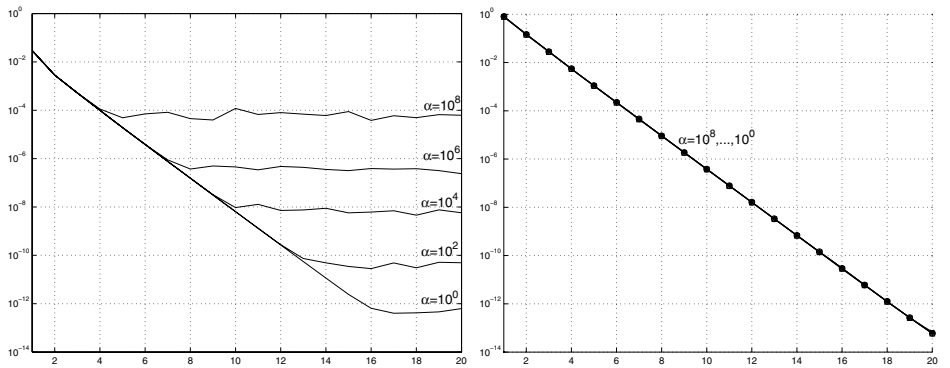
In Table 4.4 the convergence rates for F-cycles are presented. It turns out that the F-cycle is a good compromise between the V-cycle and the W-cycle, since the F-cycle has only slightly higher computational costs than the V-cycle but convergence properties similar to those of the W-cycle.

In Table 4.5 we present the L_2-norm of the defects ($d_{h_l}^{(k)} = f_{h_l} - A_{h_l} u_{h_l}^{(k)}$) for the numerical solution $u_{h_l}^{(k)}$ after the kth multigrid iteration with mesh size h_l on the finest grid. In order to illustrate the asymptotic convergence of the multigrid cycle, we reduce the L_2-norm of the defect to less than 10^{-08}. It can be seen that the multigrid solver converges to this accuracy after approximately eleven multigrid iterations—independently of the grid size h_l. This indicates optimal multigrid convergence.

Of course, in image processing applications it is usually not necessary to reduce the defect down to an accuracy of 10^{-8}. Here, we typically apply 1–3 multigrid iterations per time step.

TABLE 4.5
Defect history of $F(0,3)$ cycles.

	$h_l = \frac{1}{32}$	$h_l = \frac{1}{64}$	$h_l = \frac{1}{128}$	$h_l = \frac{1}{256}$
$k = 1$	$1.5771 \cdot 10^{-01}$	$1.4701 \cdot 10^{-01}$	$1.2050 \cdot 10^{-01}$	$8.8106 \cdot 10^{-02}$
$k = 2$	$1.6028 \cdot 10^{-02}$	$1.5453 \cdot 10^{-02}$	$1.3399 \cdot 10^{-02}$	$1.0239 \cdot 10^{-02}$
$k = 3$	$2.8981 \cdot 10^{-03}$	$2.7933 \cdot 10^{-03}$	$2.4353 \cdot 10^{-03}$	$2.0029 \cdot 10^{-03}$
$k = 4$	$5.4126 \cdot 10^{-04}$	$5.2086 \cdot 10^{-04}$	$4.5334 \cdot 10^{-04}$	$3.8455 \cdot 10^{-04}$
$k = 5$	$1.0361 \cdot 10^{-04}$	$9.8470 \cdot 10^{-05}$	$1.6223 \cdot 10^{-05}$	$7.4786 \cdot 10^{-05}$
$k = 6$	$2.0134 \cdot 10^{-05}$	$1.8822 \cdot 10^{-05}$	$2.8307 \cdot 10^{-05}$	$1.4669 \cdot 10^{-05}$
$k = 7$	$3.9536 \cdot 10^{-06}$	$3.6344 \cdot 10^{-06}$	$3.1180 \cdot 10^{-06}$	$2.9072 \cdot 10^{-06}$
$k = 8$	$7.8265 \cdot 10^{-07}$	$7.0857 \cdot 10^{-07}$	$6.0551 \cdot 10^{-07}$	$5.8230 \cdot 10^{-07}$
$k = 9$	$1.5600 \cdot 10^{-07}$	$1.3940 \cdot 10^{-07}$	$1.1877 \cdot 10^{-07}$	$1.1783 \cdot 10^{-07}$
$k = 10$	$3.1285 \cdot 10^{-08}$	$2.7659 \cdot 10^{-08}$	$2.3520 \cdot 10^{-08}$	$2.4069 \cdot 10^{-08}$
$k = 11$	$6.3091 \cdot 10^{-09}$	$5.5308 \cdot 10^{-09}$	$4.6984 \cdot 10^{-09}$	$4.9581 \cdot 10^{-09}$



(a) (4.1) violated (b) (4.1) fulfilled

FIG. 4. *Behavior of the defects (during the multigrid iteration) for different parameter α. The discrete L_2 norm of the defect is plotted in a logarithmic scale along the y-axis w.r.t. the kth multigrid iteration ($F(0,3)$-cycles) at the abscissa.*

In the following we want to discuss the influence of the parameter α on the convergence speed of the multigrid algorithm. In Figure 4(a) the L_2 norm of the defect is plotted in a logarithmic scale along the y-axis w.r.t. the kth multigrid iteration (x-axis) by using different parameters α. Here, we observe a rapid convergence of the proposed multigrid scheme during the first multigrid iteration steps. After a few steps the defect reduction stagnates. The reason is (see section 2.3) that the system for large α converges more and more to the singular system $Lu = f$, which could not be solved uniquely if the integrability condition

$$\langle f, \phi_l^* \rangle = 0 \quad \text{for all} \quad \phi_l^* \in \ker(L^T) \tag{4.1}$$

is violated. In this case the errors in $\ker^{\perp}(L)$ will be reduced efficiently in a few steps by the multigrid iteration, but the errors in $\ker(L)$ will not be reduced by the multigrid iteration. This error becomes greater the larger the parameter α.

There are many approaches to overcome this problem, e.g., solving an augmented system; see [16]. In order to solve the time-dependent problem (1.1), the elliptic system has to be solved many times with changing right-hand side f. In this case it is attractive to compute the kernel of L^T once (and to save the kernel for later use) with an arbitrarily high precision and to enforce the integrability condition for

each time step. If the integrability condition is satisfied, good multigrid convergence (independent of α) is observed; see Figure 4(b).

5. Image registration application. In order to demonstrate the principle and reliability of the proposed approach we present experimental results for mono-modal image registration; cf. [19]. The goal of image registration is to determine a displacement vector $u(x,y) = (u_1(x,y),\, u_2(x,y))^t$ (whose components are functions of the variables $x = (x,\, y)^t$) in such a way that $u(x,y)$ transforms the so-called template image T, into another image R, called the reference, so that

$$R(x,y) = T(x - u_1(x,y), y - u_2(x,y)). \tag{5.1}$$

This is a common criterion. It is used in the case where the images are recorded with the same imaging machinery. As described in [20], this leads us to solve the system

$$\frac{\partial u_i(x,y,t)}{\partial t} + \epsilon(t)\Delta^2 u_i(x,y,t) = -\nabla T(x - u_1(x,y,t), y - u_2(x,y,t)) \\ \big(T(x - u_1(x,y,t), y - u_2(x,y,t)) - R(x,y)\big)$$

on $\Omega \times (0,T)$,

$$B_1[u_i(x,y,t)] = B_2[u_i(x,y,t)] = 0 \quad \text{on } \partial\Omega \times (0,T),$$

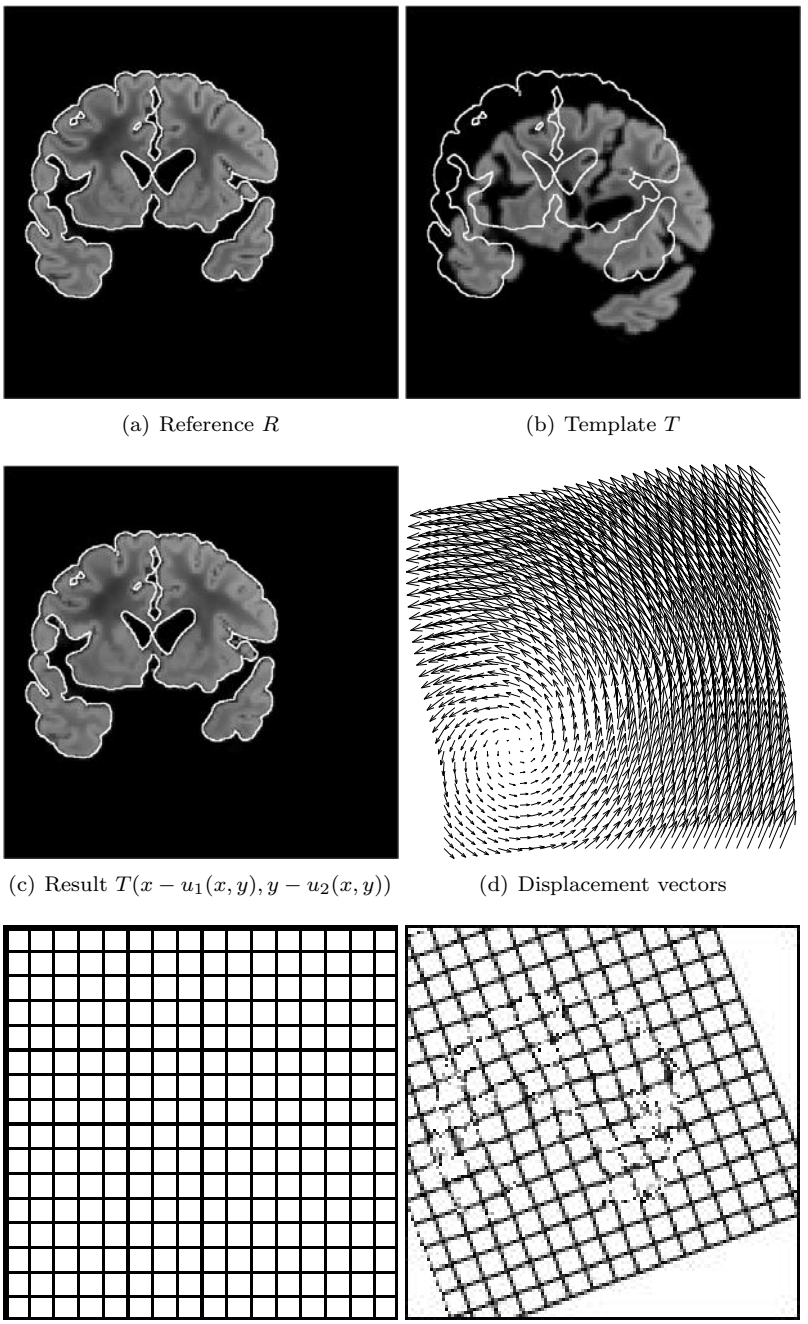
and $u_i(x,y,0) = z_i(x,y)$ on Ω for $i = 1,2$. Here, the biharmonic diffusion approach unifies existing image registration approaches, which are normally classified due to the underlying transformations into either affine-linear or nonlinear approaches.

Figure 5 shows a typical example of a brain data set with a standard brain from a human brain atlas (different individuals). Here, the MRI image in Figure 5(a) is assumed to be the deformed template T and should be matched onto the corresponding reference MRI image (Figure 5(b)). Figure 5(c) shows the calculated result $T(x - u(x,t))$. All images are displayed with superimposed contour of the reference image. For this example, we use $\epsilon(0) = 10^6$ and $\tau = 0.015$. It is suitable to start the image registration with $z_1(x,y) = z_2(x,y) = 0$, which corresponds to the identity map. We use 100 time steps. During the iteration we decrease ϵ and τ as described in [20]. For each time step we chose two multigrid $F(2,1)$-cycles to determine the displacements as described in section 4. For images consisting of 256×256 picture elements, our algorithm takes (with a nonoptimized C implementation) approximately one minute on a 2GHz PC with 2GB of RAM.

6. Conclusion and remarks. In this paper we have introduced an efficient multigrid scheme for solving the biharmonic diffusion equation arising in image processing problems. We use a semi-implicit time discretization. The resulting stationary problem is given by a biharmonic operator with higher-order boundary conditions. For spatial discretization, we use the finite difference method.

Operator splitting leads us to solve two Poisson-like equations, which are coupled via the boundary conditions. We have shown via practical examples that the asymptotic convergence behavior, predicted by local mode analysis, can be attained by performing collective relaxation sweeps of some lines near the boundaries. Here, the results for V-cycle schemes are less remarkable, but F- and W-cycles are able to restore these rates.

There are a number of potential extensions of our work. First, in this paper we have only shown two-dimensional results; the extension to three-dimensional problems is our aim for future research. Second, in order to speed up the computations,



(a) Reference R (b) Template T

(c) Result $T(x - u_1(x, y), y - u_2(x, y))$ (d) Displacement vectors

(e) Uniform grid (f) Deformed grid

FIG. 5. (a) *Template image* T. (b) *Reference image* R. (c) *Template deformed by the solution of biharmonic diffusion approach.* (d) *Plot of displacement vectors as arrows with components* (u_1, u_2) *at the points* (x, y). (e) *Uniform grid.* (f) *Uniform grid deformed by the solution.*

the multiscale nature of the biharmonic diffusion equation suggests a treatment on different spatial grids. These approaches are similar to the full multigrid scheme and are described in [19].

REFERENCES

[1] I. Altas, J. Dym, M. M. Gupta, and R. P. Manohar, *Multigrid solution of automatically generated high-order discretizations for the biharmonic equation*, SIAM J. Sci. Comput., 19 (1998), pp. 1575–1585.

[2] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing – Partial Differential Equations and the Calculus of Variations*, Springer-Verlag, New York, 2002.

[3] H. Blum and R. Rannacher, *On the boundary value problem of the biharmonic operator on domains with angular corners*, Math. Methods Appl. Sci., 2 (1980), pp. 556–581.

[4] A. Brandt, *Rigorous quantitative analysis of multigrid* I: *Constant coefficients two-level cycle with L_2-norm*, SIAM J. Numer. Anal., 31 (1994), pp. 1695–1730.

[5] S. Brenner, *An optimal-order nonconforming multigrid method for the biharmonic equation*, SIAM J. Numer. Anal., 26 (1989), pp. 1124–1138.

[6] A. Chambolle and P.-L. Lions, *Image recovery via total variation minimization and related problems*, Numer. Math., 76 (1997), pp. 167–188.

[7] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, SIAM J. Sci. Comput., 20 (1999), pp. 1964–1977.

[8] T. Chan, J. Shen, and L. Vese, *Variational PDE models in image processing*, Notices Amer. Math. Soc., 50 (2003), pp. 14–26.

[9] P. Ciarlet and P. Raviart, *A mixed method for the biharmonic equation*, in Mathematical Aspects of Finite Elements in Partial Differential Equations, C. de Boor, ed., Academic Press, New York, 1975, pp. 125–145.

[10] S. D. Conte and R. T. Dames, *On an alternating direction method for solving the plate problem with mixed boundary conditions*, J. ACM, 7 (1960), pp. 264–273.

[11] L. W. Ehrlich and M. M. Gupta, *Some difference schemes for the biharmonic equation*, SIAM J. Numer. Anal., 12 (1975), pp. 773–790.

[12] B. Fischer and J. Modersitzki, *Curvature based image registration*, J. Math. Imaging Vision, 18 (2003), pp. 81–85.

[13] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston, 1985.

[14] M. Gupta and R. Manohar, *Direct solution of the biharmonic equation using noncoupled approaches*, J. Comput. Phys., 33 (1979), pp. 236–248.

[15] M. M. Gupta, *Discretization error estimates for certain splitting procedures for solving first biharmonic boundary value problems*, SIAM J. Numer. Anal., 12 (1975), pp. 364–377.

[16] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin, 1985.

[17] W. Hackbusch, *Elliptic Differential Equations. Theory and Numerical Treatment*, Springer Ser. Comput. Math. 18, Springer-Verlag, Berlin, 1992.

[18] M. R. Hanisch, *Multigrid preconditioning for the biharmonic Dirichlet problem*, SIAM J. Numer. Anal., 30 (1993), pp. 184–214.

[19] S. Henn and K. Witsch, *Iterative multigrid regularization techniques for image matching*, SIAM J. Sci. Comput., 23 (2001), pp. 1077–1093.

[20] S. Henn and K. Witsch, *Image registration based on multiscale energy information*, Multiscale Model. Simul., 4 (2005), pp. 584–609.

[21] S. F. McCormick, ed., *Multigrid Methods*, Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987.

[22] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation-based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[23] K. Stüben and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, Springer-Verlag, Berlin, New York, 1982.

[24] U. Trottenberg, C. Osterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, CA, 2000.

[25] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, Frontiers Appl. Math. 10, SIAM, Philadelphia, 1992.

[26] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers Appl. Math., 23, SIAM, Philadelphia, 2002.

[27] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 850–872
© 2005 Society for Industrial and Applied Mathematics

APPROXIMATION OF HYPERBOLIC MODELS FOR CHEMOSENSITIVE MOVEMENT*

FRANCIS FILBET† AND CHI-WANG SHU‡

Abstract. Numerical methods with different orders of accuracy are proposed to approximate hyperbolic models for chemosensitive movements. On the one hand, first- and second-order well-balanced finite volume schemes are presented. This approach provides exact conservation of the steady state solutions. On the other hand, a high-order finite difference weighted essentially nonoscillatory (WENO) scheme is constructed and the well-balanced reconstruction is adapted to this scheme to exactly preserve steady states and to retain high-order accuracy. Numerical simulations are performed to verify accuracy and the well-balanced property of the proposed schemes and to observe the formation of networks in the hyperbolic models similar to those observed in the experiments.

Key words. chemotaxis, hyperbolic systems, finite volume methods, finite difference methods, weighted essentially nonoscillatory schemes, well-balanced schemes

AMS subject classifications. 35L60, 92C15, 65M06

DOI. 10.1137/040604054

1. Introduction.

Chemosensitive movement is a process by which cells change their direction when reacting to the presence of a chemical substance, approaching chemically favorable environments, and avoiding unfavorable environments. In the biological literature there is a distinction between chemotaxis and chemokinesis. Chemokinesis describes nondirected bias in the movement behavior, which indirectly leads to an oriented movement of the population, whereas chemotaxis denotes a directed orientation toward or away from a chemical stimulus. However, from a modeling viewpoint this distinction is not necessary and we will summarize all these orientation effects in a chemosensitive movement.

In a simple situation, where we consider only cells and one chemical substance (the chemo-attractant), a model for the space and time evolution of the density $n = n(t,x)$ of cells and the chemical concentration $c = c(t,x)$ at time t and position $x \in \Omega \subset \mathbb{R}^d$ has been introduced by Patlak [19] and Keller and Segel [13]. This model assumes that the propagation of cells is performed via a diffusion process and reads

$$\frac{\partial n}{\partial t} - \operatorname{div}\left(\nabla n - n\,\chi'(c)\,\nabla c\right) = 0, \tag{1}$$

where $\chi'(c) \geq 0$ denotes the chemotactic sensitivity and the chemical concentration c is given either by the following parabolic equation describing the diffusion of the chemo-attractant created by the cells themselves:

$$\frac{\partial c}{\partial t} - D_c\,\Delta c = n - c, \tag{2}$$

*Received by the editors February 10, 2004; accepted for publication (in revised form) April 27, 2005; published electronically November 22, 2005.
http://www.siam.org/journals/sisc/27-3/60405.html

†Mathématiques et Applications, Physique Mathématique d'Orléans, CNRS UMR 6628, Université d'Orléans, B.P. 6759, F–45067 Orléans cedex 2, France (filbet@labomath.univ-orleans.fr).

‡Division of Applied Mathematics, Brown University, Providence, RI 02912 (shu@dam.brown.edu). This author's research was supported by ARO grants DAAD19-00-1-0405 and W911NF-04-1-0291, NSF grant DMS-0207451, and AFOSR grant F49620-02-1-0113.

with $D_c \geq 0$ denoting the motility (ability of the cell to move), or in the angiogenesis case, simply by

$$\frac{\partial c}{\partial t} = -c\, n. \tag{3}$$

However, this approach is not always sufficiently precise to describe the evolution of bacteria movements. Indeed, experiments show that bacteria such as [illegible] move along straight lines, suddenly stop to choose a new direction, and then continue moving in the new direction. This phenomenon, called run and tumble, can be modeled by a stochastic process called the velocity-jump process [10, 25]. A kinetic transport model for describing this velocity-jump process has been introduced in [18]. In fact, parabolic chemotaxis equations, such as the Patlak–Keller–Segel (PKS) model (1)–(2), have been obtained as the diffusion limit of this kinetic model, thus allowing the determination of the motility D_c and the chemotactic sensitivity χ' [10, 17]. A more recent tendency has been to use hyperbolic equations, which respect finite propagation speeds and are based on the individual movement patterns of the cells; see for instance [3, 4, 11, 21].

In this paper, we propose accurate numerical methods to approximate hyperbolic chemotaxis models. Our approach relies on finite volume schemes and finite difference schemes with high-order weighted essentially nonoscillatory (WENO) reconstructions and allows us to exactly preserve a particular class of steady state solutions.

The outline of this paper is as follows. In section 2, we present a hyperbolic system previously proposed in [21] to describe the early stages of the formation of blood vessels. In section 3, we first propose a first-order well-balanced (i.e., preserving exactly certain steady state solutions) finite volume scheme and then extend it to second-order accuracy. In section 4, we design a high-order well-balanced finite difference WENO scheme. Finally, in section 5 we perform numerical simulations of the nonlinear hyperbolic models of chemotaxis to verify accuracy and the well-balanced property of the proposed finite volume and high-order WENO finite difference schemes and to observe the formation of networks in the hyperbolic models similar to those observed in the experiments [4, 5, 21].

2. A hyperbolic model for chemotaxis. In this paper, we consider numerical approximations of a hyperbolic model introduced in [5, 21], in which the cell density $n(t,x)$ and the population flux $n\,u(t,x)$ are solutions of the hyperbolic system

$$\begin{cases} \dfrac{\partial n}{\partial t} + \operatorname{div}(n\,u) = 0, \\ \dfrac{\partial (n\,u)}{\partial t} + \operatorname{div}(n\ u \otimes u + P(n)) = n\,\chi'(c)\,\nabla c - \sigma\, n\ u, \end{cases} \tag{4}$$

where $P(n)$ is the pressure and σ is the friction coefficient. The evolution of the chemical concentration $c(t,x)$ is still given by (2).

This model is used to describe vasculogenesis (early formation of blood vessels) [5, 21] and can be recovered from a hydrodynamic scaling of a kinetic model [4]. Finite time blowup was proved in [14] for spherically symmetric solutions to the hyperbolic system (4) coupled with a Poisson-type equation to c,

$$-\Delta c = n.$$

Indeed, the Euler–Poisson system is widely studied in plasma physics and astrophysics to model star evolution and it is more expected that shock-type structures arise here,

which could explain why the singularities are not pointwise blow-up typical to the solution of the parabolic model (1) but are more like line concentrations (high concentrations along a line or curve). Such singularities are indeed observed in the numerical simulations in section 5 (formation of networks) and seem to be compatible with the experiments on endothelial cells [5, 21]. Moreover, the hyperbolic model (2), (4) admits an entropy inequality when $\chi(c) = c$, $P(n) = n$, and $D_c = 1$,

$$\frac{\partial \eta}{\partial t} + \operatorname{div} G = -\sigma n u^2 - \left(\frac{\partial c}{\partial t}\right)^2 \leq 0,$$

where

$$\begin{aligned} \eta(t,x) &= \eta_h(n, n\,u) - nc + \frac{1}{2}\left(c^2 + \left(\frac{\partial c}{\partial x}\right)^2\right), \\ G(t,x) &= G_h(n, n\,u) - nuc - \frac{\partial c}{\partial t}\,\frac{\partial c}{\partial x}, \end{aligned} \tag{5}$$

and (η_h, G_h) corresponds to the entropy pair for the homogeneous problem without the source term,

$$\eta_h(n, n\,u) = n(\log\, n - 1) + \frac{1}{2}\,n\,u^2, \qquad G_h(n, n\,u) = n\,u\left(\log n + \frac{1}{2}\,u^2\right). \tag{6}$$

Finally, we conclude this section with some remarks on the connection between the hyperbolic chemotaxis model (2)–(4) and the parabolic chemotaxis system (1)–(2). We first notice that, when $P(n) = n\,Id$, it is possible to recover (1) as the strong friction limit of (4), i.e., when σ goes to infinity. Moreover, the hyperbolic system (4) admits several types of steady state solutions, but the steady state solutions with a zero population flux ($nu = 0$) are particularly important since they are the same as those of the PKS model (1)–(2). Indeed, such steady state solutions satisfy

$$n\,\chi'(c)\,\nabla c - \nabla n = 0, \tag{7}$$

which is the equation satisfied by stationary solutions to the diffusive model for chemotaxis [17, 18].

In what follows, we restrict our attention to the one-dimensional case with $P(n) = n$:

$$\begin{cases} \dfrac{\partial n}{\partial t} + \dfrac{\partial (n\,u)}{\partial x} = 0, \\ \dfrac{\partial (n\ u)}{\partial t} + \dfrac{\partial}{\partial x}\left(n\ u^2 + n\right) = n\,\chi'(c)\dfrac{\partial c}{\partial x} - \sigma\ n\ u. \end{cases} \tag{8}$$

For future reference, we denote by U the unknown, by $F(U)$ the flux function, and by $S(U)$ the source term, which for the nonlinear system (8) are given by

$$U = \begin{pmatrix} n \\ n\,u \end{pmatrix}, \qquad F(U) = \begin{pmatrix} n\,u \\ n\,u^2 + n \end{pmatrix}, \qquad S(U) = \begin{pmatrix} 0 \\ n\,\chi'(c)\frac{\partial c}{\partial x} - \sigma\ n\ u \end{pmatrix}. \tag{9}$$

3. Finite volume approximations. Let us denote by $(x_{i-1/2})_{i\in\mathbb{Z}}$ a mesh of $\mathbb{R}$ and $x_i = (x_{i-1/2} + x_{i+1/2})/2$. Finite volume schemes for hyperbolic systems use an upwinding of the fluxes and in the semidiscrete case they provide a discretized version of (8) under the form

$$\Delta x_i \frac{d}{dt} U_i(t) + F_{i+1/2} - F_{i-1/2} = \Delta x_i \, S_i \tag{10}$$

where Δx_i denotes the mesh size $\Delta x_i = x_{i+1/2} - x_{i-1/2}$, and the discrete unknown $U_i(t)$ is

$$U_i(t) = \begin{pmatrix} n_i(t) \\ n_i(t)\, u_i(t) \end{pmatrix},$$

which represents an approximation to the cell-average of U in the cell $(x_{i-1/2}, x_{i+1/2})$. $F_{i+1/2}$ is an approximation of the flux at the point $x_{i+1/2}$ and S_i is an approximation of the cell-average of the source term $S(U)$ in the cell $(x_{i-1/2}, x_{i+1/2})$.

3.1. A first-order well-balanced scheme. In a basic first-order accurate scheme, the fluxes are classically computed as

$$F_{i+1/2} = \mathcal{F}(U_i(t), U_{i+1}(t)), \tag{11}$$

where the numerical flux $\mathcal{F}(U_i, U_{i+1})$ is computed via an approximation of the Riemann problem. We assume that the numerical flux $\mathcal{F}$ satisfies the following assumptions:

1. It is consistent with the physical flux:

$$\mathcal{F}(U, U) = F(U). \tag{12}$$

2. It preserves the nonnegativity of the density $n_i(t)$ for the homogeneous problem, namely, (8) without the source terms. That is, the scheme

$$\Delta x_i \frac{d}{dt} U_i(t) + F_{i+1/2} - F_{i-1/2} = 0, \tag{13}$$

where the numerical flux $F_{i+1/2}$ is defined by (11), maintains $n_i(t) \geq 0$ if $n_i(0) \geq 0$.

3. The scheme (13) with the numerical flux $F_{i+1/2}$ defined by (11) satisfies a cell entropy inequality for the entropy pair (6) for the homogeneous problem. According to [2], this means that we can find a numerical entropy flux $\mathcal{G}_h$ such that

$$\begin{aligned} &G_h(U_{i+1}) + \nabla_U \eta_h(U_{i+1}) \left(\mathcal{F}(U_i, U_{i+1}) - F(U_{i+1})\right) \\ &\qquad \leq \mathcal{G}_h(U_i, U_{i+1}) \leq G_h(U_i) + \nabla_U \eta_h(U_i) \left(\mathcal{F}(U_i, U_{i+1}) - F(U_i)\right), \end{aligned} \tag{14}$$

where $\nabla_U \eta_h$ is the derivative of η_h with respect to $U = \binom{n}{nu}$.

Since the appearance of [7, 8], it has been known that cell-centered evaluations of the source term in a nonconservative hyperbolic system will generally not be able to preserve steady state solutions. In our case a particular class of steady state solutions, namely, the solution to (7), is given by

$$\log(n_i) - \chi(c_i) = \textit{constant}, \qquad u_i = 0 \qquad \forall\, i. \tag{15}$$

These steady state solutions play an important role in the modeling of chemosensitive movement because they also correspond to the steady state solutions of the parabolic equation (1), which describes the long time behavior of cells under the influence of the chemo-attractant c when the solutions do not blow up. Thus we would like to preserve these steady state solutions exactly at the discrete level.

Our approach follows the ideas of [1] for the shallow water equations (see also [6]). We propose a finite volume scheme according to (10) with flux functions

$$F_{i+1/2} = \mathcal{F}(U^-_{i+1/2}, U^+_{i+1/2}), \qquad U^\pm_{i+1/2} = \begin{pmatrix} n^\pm_{i+1/2} \\ n^\pm_{i+1/2}\, u^\pm_{i+1/2} \end{pmatrix}, \tag{16}$$

where the interface values $U^-_{i+1/2}$ and $U^+_{i+1/2}$ are locally reconstructed from U_i and U_{i+1}. The purpose of the reconstruction is not to increase the order of accuracy as usual, but to achieve a well-balanced property for the steady state solution (15). The source term is discretized by

$$S_i = \begin{pmatrix} 0 \\ \frac{1}{\Delta x_i}\left(n^-_{i+1/2} - n^+_{i-1/2}\right) \end{pmatrix} - \sigma \begin{pmatrix} 0 \\ n_i\, u_i \end{pmatrix}. \tag{17}$$

This ansatz is motivated by a balancing requirement. Indeed, when the population flux $n\,u$ is zero, the cell density n satisfies the balance of the momentum and the source term, i.e.,

$$\frac{\partial n}{\partial x} = n\,\chi'(c)\frac{\partial c}{\partial x}.$$

Integrating over $(x_{i-1/2}, x_{i+1/2})$, we obtain an approximation to the source term,

$$\frac{1}{\Delta x_i}\int_{x_{i-1/2}}^{x_{i+1/2}} n\,\chi'(c)\frac{\partial c}{\partial x}\,dx = \frac{1}{\Delta x_i}\left(n^-_{i+1/2} - n^+_{i-1/2}\right).$$

Therefore, when the velocity u is small, we are able to locally represent the cell-averaged source term as the discrete gradient of the momentum flux. This motivates the source term discretization (17). We will show later that this discretization is consistent with the source term in (8) also for solutions far away from these steady state solutions. It will be shown later that the steady state (15) is maintained exactly if, for such a state, we have

$$n^-_{i+1/2} = n^+_{i+1/2}, \qquad u^-_{i+1/2} = u^+_{i+1/2} = 0. \tag{18}$$

To achieve this, we choose

$$n^-_{i+1/2} = n_i\, e^{\chi_{i+1/2} - \chi(c_i)}, \qquad u^-_{i+1/2} = u_i \tag{19}$$

and

$$n^+_{i+1/2} = n_{i+1}\, e^{\chi_{i+1/2} - \chi(c_{i+1})}, \qquad u^+_{i+1/2} = u_{i+1}, \tag{20}$$

which lead to (18) for an arbitrary choice of $\chi_{i+1/2}$ when the steady state (15) is achieved. The evaluation of the interface value $\chi_{i+1/2}$ has to be such that the scheme is consistent and stable. We have chosen

$$\chi_{i+1/2} = \max\left(\chi(c_i), \chi(c_{i+1})\right) \tag{21}$$

in our numerical experiments. Another possible choice could be

$$\chi_{i+1/2} = \frac{1}{2}\left(\chi(c_i) + \chi(c_{i+1})\right).$$

To summarize, our first-order well-balanced finite volume scheme (10) is defined with the flux and reconstructions given by (16) and (19)–(20). The source term is given by (17), or equivalently, by

$$S_i = S^-_{i+1/2} + S^+_{i-1/2} - \sigma \begin{pmatrix} 0 \\ n_i u_i \end{pmatrix}, \tag{22}$$

with

$$S^-_{i+1/2} = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ n^-_{i+1/2} - n_i \end{pmatrix}, \qquad S^+_{i-1/2} = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ n_i - n^+_{i-1/2} \end{pmatrix}. \tag{23}$$

We may rewrite the scheme in a nonconservative form as in [2],

$$\Delta x_i \frac{d}{dt} U_i(t) + \mathcal{F}_l(U_i, U_{i+1}, c_i, c_{i+1}) - \mathcal{F}_r(U_{i-1}, U_i, c_{i-1}, c_i) = -\Delta x_i\, \sigma \begin{pmatrix} 0 \\ n_i u_i \end{pmatrix},$$

with the left and right numerical fluxes defined by

$$\begin{aligned} \mathcal{F}_l(U_i, U_{i+1}, c_i, c_{i+1}) &= F_{i+1/2} - \Delta x_i\, S^-_{i+1/2}, \\ \mathcal{F}_r(U_i, U_{i+1}, c_i, c_{i+1}) &= F_{i+1/2} + \Delta x_i\, S^+_{i+1/2}. \end{aligned} \tag{24}$$

Finally, the chemical concentration c is obtained by approximating (2) with a finite volume scheme,

$$\frac{dc_i}{dt} = \frac{D_c}{\Delta x_i}\left(\frac{c_{i+1} - c_i}{\Delta x_{i+1/2}} - \frac{c_i - c_{i-1}}{\Delta x_{i-1/2}}\right) + n_i - c_i, \tag{25}$$

where $\Delta x_{i+1/2} = x_{i+1} - x_i$.

We do not consider the influence of boundary conditions and will assume a periodic boundary condition. However, the numerical methods can be applied to general boundary conditions, as demonstrated in the numerical examples in section 5.

We have the following results for the scheme (10) defined above.

THEOREM 1. [illegible] (10) [illegible] (16), (17), (19), [illegible] (20)

(i) [illegible] $n_i(t)$.

(ii) [illegible] (15).

(iii) [illegible] (8).

(iv) [illegible] (5) [illegible] $\chi(c_i) = c_i$ [illegible] c_i [illegible] (25)

$$\Delta x_i \frac{\partial \eta_i}{\partial t} + G_{i+1/2} - G_{i-1/2} \le 0,$$

[illegible]

$$\eta_i = n_i\left(\log(n_i) - 1 + \frac{u_i^2}{2} - c_i\right) + \frac{1}{2}\left(c_i^2 + \frac{1}{\Delta x_i}\frac{(c_{i+1} - c_i)^2}{\Delta x_{i+1/2}}\right)$$

[illegible] $G_{i+1/2}$

. The ODE system (10) admits a unique smooth (in time) solution since the flux function is smooth enough. By this smoothness of the numerical solution in time, the flux $\mathcal{F}(U_l, U_r) = (\mathcal{F}_1, \mathcal{F}_2)(U_l, U_r)$ to the homogeneous problem (for example, the Lax–Friedrichs flux (41) to be used later) preserves the nonnegativity of n_i if and only if the first component of the flux $\mathcal{F}_1$ satisfies

$$\mathcal{F}_1((n_i = 0, u_i), (n_{i+1}, u_{i+1})) - \mathcal{F}_1((n_{i-1}, u_{i-1}), (n_i = 0, u_i)) \le 0 \quad \forall\, (u_j, n_j)_j. \tag{26}$$

Therefore, for our reconstruction (19)–(20), we need to check that, when $n_i = 0$,

$$\mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2}) \le 0 \quad \forall\, u_{i-1}, u_i, u_{i+1}, n_{i-1}, n_{i+1}.$$

But, when $n_i = 0$, the reconstruction (19)–(20) gives that $n^+_{i-1/2} = n^-_{i+1/2} = 0$; hence,

$$\begin{aligned}
&\mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2}) = \mathcal{F}_1((0, u_i), (n^+_{i+1/2}, u_{i+1})) \quad \forall\, (u_j, n^+_{j+1/2}, n^-_{j-1/2})_j,\\
&\mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2}) = \mathcal{F}_1((n^-_{i-1/2}, u_{i-1}), (0, u_i)) \quad \forall\, (u_j, n^+_{j+1/2}, n^-_{j-1/2})_j.
\end{aligned}$$

Then from (26), we conclude that the scheme preserves nonnegativity of the density n_i.

To prove (ii), we assume the steady state (15) holds and would like to show

$$F_{i+1/2} - F_{i-1/2} = \Delta x_i\, S_i. \tag{27}$$

By the reconstruction of $n^+_{i+1/2}$ and $n^-_{i+1/2}$ given in (19)–(20), when (15) holds, we have

$$n^-_{i+1/2} = n^+_{i+1/2} = A\, e^{\chi_{i+1/2}}, \qquad u^-_{i+1/2} = u^+_{i+1/2} = 0,$$

where $A = n_i\, e^{-\chi(c_i)} = n_{i+1}\, e^{-\chi(c_{i+1})}$. On the one hand, the source term is given by

$$\Delta x_i\, S_i = \begin{pmatrix} 0 \\ n^-_{i+1/2} - n^+_{i-1/2} \end{pmatrix} = A \begin{pmatrix} 0 \\ e^{\chi_{i+1/2}} - e^{\chi_{i-1/2}} \end{pmatrix}. \tag{28}$$

On the other hand, by the consistency of the flux $\mathcal{F}$, we have

$$\begin{aligned}
F_{i+1/2} &= \mathcal{F}((A\, e^{\chi_{i+1/2}}, 0), (A\, e^{\chi_{i+1/2}}, 0))\\
&= F((A\, e^{\chi_{i+1/2}}, 0))\\
&= \begin{pmatrix} 0 \\ A\, e^{\chi_{i+1/2}} \end{pmatrix}.
\end{aligned} \tag{29}$$

Combining (28) and (29), we have proven (27).

To prove statement (iii), we apply the criterion in [1] and need to check two properties related to the consistency with the exact flux F and the consistency with the source term. The consistency with the exact flux is

$$\mathcal{F}_l(U, U, c, c) = \mathcal{F}_r(U, U, c, c) = F(U).$$

It is obviously satisfied since in this case $\chi_{i+1/2} = \chi(c)$, and therefore $n^+_{i+1/2} = n^-_{i+1/2} = n$. For the consistency with the source term, the criterion becomes

$$\mathcal{F}_r(U_l, U_r, c_l, c_r) - \mathcal{F}_l(U_l, U_r, c_l, c_r) = \begin{pmatrix} 0 \\ n\, \chi'(c)\, (c_r - c_l) + o(c_r - c_l) \end{pmatrix}, \tag{30}$$

as $U_r, U_l \to U = (n, u)$ and $c_r, c_l \to c$. In our case,

$$\mathcal{F}_r(U_l, U_r, c_l, c_r) - \mathcal{F}_l(U_l, U_r, c_l, c_r) = \begin{pmatrix} 0 \\ n_l\, e^{\chi_0 - \chi(c_l)} - n_l \end{pmatrix} - \begin{pmatrix} 0 \\ n_r\, e^{\chi_0 - \chi(c_r)} - n_r \end{pmatrix},$$

where $\chi_0 = \max(\chi(c_l), \chi(c_r))$. Now assume that $\chi_0 = \chi(c_l)$; then

$$\mathcal{F}_r(U_l, U_r, c_l, c_r) - \mathcal{F}_l(U_l, U_r, c_l, c_r) = \begin{pmatrix} 0 \\ n_r\, e^{-\chi(c_r)} \left(e^{\chi(c_r)} - e^{\chi(c_l)}\right) \end{pmatrix}$$

and we have

$$n_r\, e^{-\chi(c_r)} \left(e^{\chi(c_r)} - e^{\chi(c_l)}\right) = n\, e^{-\chi(c)} \left(e^{\chi(c_r)} - e^{\chi(c_l)}\right) + o(c_r - c_l),$$

which is consistent with the source term (30) since

$$n\, e^{-\chi(c)}\, \nabla e^{\chi(c)} = n\, \chi'(c)\, \nabla c.$$

The same argument applies when $\chi_0 = \chi(c_r)$, and we conclude that (30) always holds, which proves (iii).

In order to prove (iv), we proceed as in the continuous case and split the time derivative of the entropy into two pieces assuming $\chi(c) = c$,

$$\frac{\partial \eta_i}{\partial t} = \left(\frac{\partial \eta_{h,i}}{\partial t} - c_i \frac{\partial n_i}{\partial t}\right) + \left(-n_i \frac{\partial c_i}{\partial t} + \frac{1}{2}\left(\frac{\partial c_i^2}{\partial t} + \frac{1}{\Delta x_i} \frac{\partial}{\partial t} \frac{(c_{i+1} - c_i)^2}{\Delta x_{i+1/2}}\right)\right)$$

and we show that

$$\Delta x_i \left(\frac{\partial \eta_{h,i}}{\partial t} - c_i \frac{\partial n_i}{\partial t}\right) + G^1_{i+1/2} - G^1_{i-1/2} \le 0 \tag{31}$$

and

$$\Delta x_i \left(-n_i \frac{\partial c_i}{\partial t} + \frac{1}{2}\left(\frac{\partial c_i^2}{\partial t} + \frac{1}{\Delta x_i} \frac{\partial}{\partial t} \frac{(c_{i+1} - c_i)^2}{\Delta x_{i+1/2}}\right)\right) + G^2_{i+1/2} - G^2_{i-1/2} \le 0. \tag{32}$$

To prove (31), we first take notice of the assumption that the original numerical flux $\mathcal{F}$ satisfies a semidiscrete entropy inequality; hence (14) holds. Then, we set

$$G^1_{i+1/2} = \mathcal{G}_h(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2})\, c_{i+1/2}$$

and compute

$$\begin{aligned} G^1_{i+1/2} - G^1_{i-1/2} = {} & \mathcal{G}_h(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{G}_h(U^-_{i-1/2}, U^+_{i-1/2}) \\ & - \mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2})\, c_{i+1/2} + \mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2})\, c_{i-1/2}. \end{aligned}$$

Using the two inequalities in (14), we get

$$\begin{aligned} G^1_{i+1/2} - G^1_{i-1/2} \le {} & G_h(U^-_{i+1/2}) + \nabla_U \eta_h(U^-_{i+1/2}) \left(\mathcal{F}(U^-_{i+1/2}, U^+_{i+1/2}) - F(U^-_{i+1/2})\right) \\ & - G_h(U^+_{i-1/2}) - \nabla_U \eta_h(U^+_{i-1/2}) \left(\mathcal{F}(U^-_{i-1/2}, U^+_{i-1/2}) - F(U^+_{i-1/2})\right) \\ & - \mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2})\, c_{i+1/2} + \mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2})\, c_{i-1/2}. \end{aligned}$$

This inequality can be rewritten as

$$\begin{aligned} G^1_{i+1/2} - G^1_{i-1/2} \leq & \left(\log(n^-_{i+1/2}) - \frac{1}{2}u_i^2 - c_{i+1/2}\right) \mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2}) \\ & - \left(\log(n^+_{i-1/2}) - \frac{1}{2}u_i^2 - c_{i-1/2}\right) \mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2}) \\ & + u_i \left(\mathcal{F}_2(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{F}_2(U^-_{i-1/2}, U^+_{i-1/2}) + n^+_{i-1/2} - n^-_{i+1/2}\right). \end{aligned}$$

Now, by the definition of $n^-_{i+1/2}$, $n^+_{i-1/2}$, and S_i, we obtain

$$\begin{aligned} G^1_{i+1/2} - G^1_{i-1/2} \leq & \left(\log(n_i) - c_i - \frac{1}{2}u_i^2\right) \left(\mathcal{F}_1(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{F}_1(U^-_{i-1/2}, U^+_{i-1/2})\right) \\ & + u_i \left(\mathcal{F}_2(U^-_{i+1/2}, U^+_{i+1/2}) - \mathcal{F}_2(U^-_{i-1/2}, U^+_{i-1/2}) \right. \\ & \left. + n^+_{i-1/2} - n^-_{i+1/2} + \sigma\, \Delta x_i n_i u_i\right) \\ \leq & -\Delta x_i\, \nabla_U \eta_h(U_i)\, \frac{dU_i}{dt} + \Delta x_i\, c_i\, \frac{dn_i}{dt}. \end{aligned}$$

We conclude that (31) holds.

To prove (32), we use the approximation (25) to the parabolic equation (2),

$$\begin{aligned} -n_i\, \frac{dc_i}{dt} &= -\left(\frac{dc_i}{dt} - \frac{1}{\Delta x_i}\left(\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}} - \frac{c_i - c_{i-1}}{\Delta x_{i-1/2}}\right) + c_i\right) \frac{dc_i}{dt}, \\ &= -\left(\frac{dc_i}{dt}\right)^2 - \frac{1}{2}\,\frac{dc_i^2}{dt} + \frac{1}{\Delta x_i}\left(\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}} - \frac{c_i - c_{i-1}}{\Delta x_{i-1/2}}\right) \frac{dc_i}{dt}. \end{aligned}$$

Using the equality

$$\begin{aligned} (c_{i+1}-c_i)\, \frac{d}{dt}\left(\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}}\right) = & -\left(\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}} - \frac{c_i - c_{i-1}}{\Delta x_{i-1/2}}\right) \frac{dc_i}{dt} \\ & + \left(\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}}\, \frac{dc_{i+1}}{dt} - \frac{c_i - c_{i-1}}{\Delta x_{i-1/2}}\, \frac{dc_i}{dt}\right), \end{aligned}$$

we get

$$\frac{1}{2}\left(\frac{d}{dt}\frac{(c_{i+1}-c_i)^2}{\Delta x_{i+1/2}} + \Delta x_i \frac{dc_i^2}{dt}\right) - \Delta x_i\, n_i\, \frac{dc_i}{dt} + G^2_{i+1/2} - G^2_{i-1/2} \leq 0,$$

with

$$G^2_{i+1/2} = -\frac{c_{i+1}-c_i}{\Delta x_{i+1/2}}\, \frac{dc_{i+1}}{dt}.$$

Finally, gathering (31) and (32), we prove the result with $G_{i+1/2} = G^1_{i+1/2} + G^2_{i+1/2}$. ∎

. . . . 1. When using the time-space fully discrete scheme

$$U_i^{k+1}-U_i^{k+1}+\frac{\Delta t}{\Delta x_i}\left(\mathcal{F}_l(U_i^k,U_{i+1}^k,c_i^k,c_{i+1}^k)-\mathcal{F}_r(U_{i-1}^k,U_i^k,c_{i-1}^k,c_i^k)\right)=-\Delta t\,\sigma\begin{pmatrix}0\\ n_i^k u_i^k\end{pmatrix},$$

the consistency and the well-balanced property are still valid. The stability is guaranteed under a CFL condition,

$$\sigma(U_i,U_{i+1})\Delta t\le \min(\Delta x_i,\Delta x_{i+1}),$$

where $\sigma(U_i,U_{i+1})\ge 0$ is a numerical speed at the interface (see [1, 2] for a detailed proof). Moreover, if we use a time explicit scheme to (25), we get a more restrictive condition on the time step

$$\Delta t\le \frac{\Delta x_i}{2}\,\min(\Delta x_{i-1/2},\Delta x_{i+1/2}).$$

3.2. Extension to a second-order approximation. Now we describe the extension to second-order accuracy. To this aim we reconstruct values at both sides of the interface. These new values are obtained in three steps: prediction of the gradients in each cell, linear extrapolation, and a limiting procedure to preserve nonnegativity. First, we compute two second-order accurate interface values $U_{i,r}$ and $U_{i+1,l}$ at the point $x_{i+1/2}$ from linear reconstructions in cells $(x_{i-1/2},x_{i+1/2})$ and $(x_{i+1/2},x_{i+3/2})$, respectively, following the three steps mentioned above. We remark that second-order accuracy is achievable only away from the vacuum $n_i=0$, since in the cell where $n_i=0$ the reconstructed linear function is flat to maintain the nonnegativity of n. Using the same reconstruction procedure, we also compute two interface values for the chemoattractant $e^{\chi(c)}$ at the point $x_{i+1/2}$, denoted by $(e^{\chi(c)})_{i,r}$ and $(e^{\chi(c)})_{i+1,l}$. Once these second-order reconstructed values are known, we further apply the reconstruction in the previous section to achieve well balancedness. This gives the second-order well-balanced scheme (10) with the flux defined by

$$F_{i+1/2}=\mathcal{F}(U^-_{i+1/2},U^+_{i+1/2})$$

and

$$U^-_{i+1/2}=\begin{pmatrix} n^-_{i+1/2}\\ n^-_{i+1/2}\,u_{i,r}\end{pmatrix},\qquad U^+_{i+1/2}=\begin{pmatrix} n^+_{i+1/2}\\ n^+_{i+1/2}\,u_{i+1,l}\end{pmatrix},\tag{33}$$

where $n^-_{i+1/2}$ and $n^+_{i+1/2}$ are given by

$$n^-_{i+1/2}=n_{i,r}\,\frac{(e^{\chi(c)})_{i+1/2}}{(e^{\chi(c)})_{i,r}},\qquad n^+_{i+1/2}=n_{i+1,l}\,\frac{(e^{\chi(c)})_{i+1/2}}{(e^{\chi(c)})_{i+1,l}},\tag{34}$$

with

$$(e^{\chi(c)})_{i+1/2}=\max\left((e^{\chi(c)})_{i,r},(e^{\chi(c)})_{i+1,l}\right).\tag{35}$$

The source term is computed as before at the interface,

$$S_i=S^-_{i+1/2}+S^c_i+S^+_{i-1/2},\tag{36}$$

with

$$S^-_{i+1/2} = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ n^-_{i+1/2} - n_{i,r} \end{pmatrix}, \qquad S^+_{i-1/2} = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ n_{i,l} - n^+_{i-1/2} \end{pmatrix} \tag{37}$$

and

$$S^c_i = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ \frac{1}{2}\left(\frac{n_{i,r}}{(e^{\chi(c)})_{i,r}} + \frac{n_{i,l}}{(e^{\chi(c)})_{i,l}}\right)\left((e^{\chi(c)})_{i,r} - (e^{\chi(c)})_{i,l}\right) \end{pmatrix} - \sigma\begin{pmatrix} 0 \\ n_i u_i \end{pmatrix}. \tag{38}$$

Using the definitions of the left and right numerical fluxes $\mathcal{F}_l$ and $\mathcal{F}_r$ in (24), a compact formulation of the scheme is

$$\Delta x_i \frac{d}{dt} U_i(t) + \mathcal{F}_l(U_{i,r}, U_{i+1,l}, c_{i,r}, c_{i+1,l}) - \mathcal{F}_r(U_{i-1,r}, U_{i,l}, c_{i-1,r}, c_{i,l}) = \Delta x_i\, S^c_i$$

and we prove the following result.

THEOREM 2. [illegible] $\mathcal{F}$ (11) [illegible] (12) [illegible] $n_i(t)$ [illegible] (13). [illegible] n [illegible] n [illegible] (33)–(38)

(i) [illegible] $n_i(t)$.

(ii) [illegible] (15).

(iii) [illegible] (8).

[illegible]. We first prove (i). Assume that $n_i = 0$, which means that it is a local minimum of n. Since the linear reconstruction uses slope limiters to preserve nonnegativity of n, the reconstructed linear function in this cell is flat and we have $n_{i,r} = n_{i,l} = n_i = 0$ and $n^-_{i+1/2} = n^+_{i-1/2} = 0$. Therefore, we can apply the strategy of the proof of Theorem 1 to show the scheme preserves nonnegativity of $n_i(t)$.

To show that the steady state (15) is exactly maintained, we have to prove (27) when $n_i = A\, e^{\chi(c_i)}$ and $u_i = 0$ for all i with a constant A. To this aim, we observe that since the reconstruction procedure for n is the same as that for $e^{\chi(c)}$, we have

$$\frac{n_{i,r}}{(e^{\chi(c)})_{i,r}} = \frac{n_{i,l}}{(e^{\chi(c)})_{i,l}} = A \quad \forall\, i. \tag{39}$$

Of course we also have $u_{i,r} = u_{i,l} = 0$. On the one hand, we have

$$n^-_{i+1/2} = n^+_{i+1/2} = A\,(e^{\chi(c)})_{i+1/2},$$

which gives, by the consistency of the numerical flux,

$$F_{i+1/2} = \mathcal{F}(U^-_{i+1/2}, U^+_{i+1/2}) = F((A\,(e^{\chi(c)})_{i+1/2}, 0)) = \begin{pmatrix} 0 \\ A\,(e^{\chi(c)})_{i+1/2} \end{pmatrix}. \tag{40}$$

On the other hand, from (36) and (39) we also get

$$S_i = \frac{1}{\Delta x_i}\begin{pmatrix} 0 \\ n^-_{i+1/2} - n^+_{i-1/2} \end{pmatrix} = \frac{A}{\Delta x_i}\begin{pmatrix} 0 \\ (e^{\chi(c)})_{i+1/2} - (e^{\chi(c)})_{i-1/2} \end{pmatrix}.$$

Together with the expression (40), we have proved that the steady state (15) is exactly maintained at the discrete level.

To prove consistency, we apply the same technique as that in the proof of Theorem 1, whereas the second-order accuracy is easily obtained by observing that the well-balanced scheme is only a second-order perturbation of the second-order reconstruction. Moreover, the source term S_i^c is a second-order approximation of the source term $n\,e^{-\chi(c)}\,\nabla(e^{\chi(c(x))})$. □

It seems to be difficult to extend this method to higher than second-order accurate finite volume schemes because the well-balanced requirement induces constraints on the approximation of the source term. Moreover, in dimensions higher than one, finite volume schemes higher than second-order accurate are much more expensive in terms of computational cost than finite difference schemes. Therefore, in the next section we apply the well-balanced reconstruction to a high-order Lax–Friedrichs flux splitting finite difference WENO scheme, which does not have such an inconvenience.

4. Finite difference approximation. In this section we first briefly review some of the basics of the high-order finite difference approximation and the WENO reconstruction; for more details, see [12, 22]. We then adapt the well-balanced reconstruction to high-order finite difference WENO schemes. In this section we assume the mesh is uniform with mesh size Δx.

4.1. High-order finite difference WENO approximation for hyperbolic systems. A conservative finite difference spatial discretization to a hyperbolic system such as (4) approximates the derivative by a kth-order accurate conservative difference

$$\left.\frac{\partial F(U)}{\partial x}\right|_{x=x_i} = \frac{1}{\Delta x}\left(\hat{F}_{i+1/2} - \hat{F}_{i-1/2}\right) + O(\Delta x^k),$$

where $\hat{F}_{i+1/2}$ is the numerical flux, which typically is a Lipschitz continuous function of several neighboring values of $U = (n, nu)$. For the details of the numerical reconstruction of the flux $\hat{F}_{i+1/2}$ from the point values of $F(U)$, we refer to [12, 22]. Notice that the finite difference methods have the same format for one or more space dimensions, which is a major advantage in terms of accuracy and computational cost in dimensions higher than one.

In many applications the numerical flux $\hat{F}_{i+1/2}$ is obtained by the Lax–Friedrichs flux splitting

$$F^{\pm}(U) = \frac{1}{2}(F(U) \pm \alpha\, U), \qquad \alpha = \max_{m,U} |\lambda_m(U)|, \tag{41}$$

where $(\lambda_m(U))_m$ are the eigenvalues of the Jacobian $F'(U)$ and the maximum is taken over the relevant range of U. We will adopt this flux splitting in our scheme. The WENO procedure is used to reconstruct the numerical flux functions for $F^+(U)$ and $F^-(U)$ separately via upwinding.

For schemes higher than second-order accurate, the procedure should be carried out in local characteristic fields to avoid spurious oscillations. Thus one would first find an average $U_{i+1/2}$ of U_i and U_{i+1} (e.g., the Roe average [20], which exists for many physical systems), and then compute the left and right eigenvectors of the Jacobian $F'(U_{i+1/2})$ and put them into the rows of $R^{-1}_{i+1/2}$ and the columns of $R_{i+1/2}$, respectively, such that

$$R^{-1}_{i+1/2}\, F'(U_{i+1/2})\, R_{i+1/2} = \Lambda_{i+1/2},$$

where $\Lambda_{i+1/2}$ is a diagonal matrix containing the real eigenvalues of $F'(U_{i+1/2})$. One then transforms all the quantities needed for evaluating the numerical flux $\hat{F}_{i+1/2}$ to the local characteristic fields by left multiplying them with $R^{-1}_{i+1/2}$, and then computes the numerical fluxes by the scalar procedure in each characteristic field. Finally, the flux in the original physical space is obtained by left multiplying by $R_{i+1/2}$ the numerical flux obtained in the local characteristic fields.

We refer to [12, 22] for more details on WENO schemes and conservative finite difference methods.

4.2. A well-balanced reconstruction. In this section, we describe the strategy for obtaining a kth-order well-balanced finite difference WENO scheme for arbitrary k. From the previous description, we have constructed a numerical flux using the characteristic Lax–Friedrichs flux splitting and obtained a kth-order approximation of the flux $\hat{F}_{i+1/2}$ via WENO reconstruction. Then, we have

$$\hat{F}_{i+1/2} = \hat{F}^{+}_{i+1/2} + \hat{F}^{-}_{i+1/2},$$

with

$$\hat{F}^{+}_{i+1/2} = R_{i+1/2}\left(R^{-1}_{i+1/2}\left(\,F(U) \,+\, \alpha\, U\,\right)\right)_{i,r}, \tag{42}$$

$$\hat{F}^{-}_{i+1/2} = R_{i+1/2}\left(R^{-1}_{i+1/2}\left(\,F(U) \,-\, \alpha\, U\,\right)\right)_{i+1,l}, \tag{43}$$

where $(\cdot)_{i,r}$ and $(\cdot)_{i+1,l}$ denote high-order WENO reconstructions at the cell boundary $x_{i+1/2}$ from the cells $(x_{i-1/2}, x_{i+1/2})$ and $(x_{i+1/2}, x_{i+3/2})$, respectively.

We now modify the numerical flux $\hat{F}^{\pm}_{i+1/2}$ to achieve the well-balanced property. The well-balanced finite difference scheme approximating (8) has the form

$$\Delta x \frac{d}{dt} U_i(t) + F_{i+1/2} - F_{i-1/2} = n_i\, e^{-\chi(c_i)} \begin{pmatrix} 0 \\ S_{i+1/2} - S_{i-1/2} \end{pmatrix} - \sigma \begin{pmatrix} 0 \\ n_i\, u_i \end{pmatrix}, \tag{44}$$

where $S_{i+1/2} - S_{i-1/2}$ is a conservative finite difference approximation to $\Delta x\, \partial_x e^{\chi(c)}$. Indeed, from the points values $S_i = e^{\chi(c_i)}$, we can reconstruct two kth-order accurate WENO or ENO [24] fluxes $S_{i+1/2}$, one from the interval $(x_{i-1/2}, x_{i+1/2})$ denoted by $S_{i,r}$ (the same reconstruction procedure used for computing $\hat{F}^{+}_{i+1/2}$), and the other from the interval $(x_{i+1/2}, x_{i+3/2})$ denoted by $S_{i+1,l}$ (the same reconstruction procedure used for computing $\hat{F}^{-}_{i+1/2}$). Finally, we choose, for instance,

$$S_{i+1/2} = \max\left(S_{i,r}, S_{i+1,l}\right). \tag{45}$$

Now, the numerical flux $F_{i+1/2}$ for the well-balanced scheme (44) is constructed from the following perturbation from the classical WENO flux $\hat{F}_{i+1/2}$:

$$F^{+}_{i+1/2} = \hat{F}^{+}_{i+1/2}\, \frac{S_{i+1/2}}{S_{i,r}}, \qquad F^{-}_{i+1/2} = \hat{F}^{-}_{i+1/2}\, \frac{S_{i+1/2}}{S_{i+1,l}}. \tag{46}$$

For this algorithm, we prove the following result.

THEOREM 3. [illegible] (8). [illegible] (44)–(46)

(i) [illegible] (15) [illegible] ε [illegible] $\varepsilon = 10^{-6}$ [12];

(ii) [illegible] (8).

[illegible]. We first prove (i). Assume that the steady state (15) is achieved,

$$n_i = A\, e^{\chi(c_i)}, \quad u_i = 0 \quad \forall\, i.$$

Then, for such a discrete solution $U_i = (n_i, u_i)$ the Lax–Friedrichs flux splitting is computed as

$$R^{-1}_{i+1/2} = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \qquad R_{i+1/2} = \begin{pmatrix} 1/2 & -1/2 \\ 1/2 & 1/2 \end{pmatrix},$$

which do not vary with i. This leads to

$$\Big(R^{-1}_{i+1/2}\,(\,F(U) + \alpha\, U\,)\Big)_{i,r} = \left(n \begin{pmatrix} 1+\alpha \\ 1-\alpha \end{pmatrix}\right)_{i,r},$$

$$\Big(R^{-1}_{i+1/2}\,(\,F(U) - \alpha\, U\,)\Big)_{i+1,l} = \left(n \begin{pmatrix} 1-\alpha \\ 1+\alpha \end{pmatrix}\right)_{i+1,l}.$$

Now we use the property of the ENO reconstruction [24] that for any constant λ,

$$(\lambda\, n)_{i,r} = \lambda\,(n)_{i,r}, \qquad (\lambda\, n)_{i+1,l} = \lambda\,(n)_{i+1,l}, \tag{47}$$

whereas for the WENO reconstruction,

$$(\lambda\, n)_{i,r} = \lambda\,(n)_{i,r} + O(\varepsilon), \qquad (\lambda\, n)_{i+1,l} = \lambda\,(n)_{i+1,l} + O(\varepsilon),$$

where ε is the small parameter used in the smooth indicator to avoid having the denominator become zero and typically takes the value $\varepsilon = 10^{-6}$ [12]. From now on we will use (47) in the analysis. Then, we get

$$\Big(R^{-1}_{i+1/2}\,(\,F(U) + \alpha\, U\,)\Big)_{i,r} = \begin{pmatrix} 1+\alpha \\ 1-\alpha \end{pmatrix} n_{i,r},$$

$$\Big(R^{-1}_{i+1/2}\,(\,F(U) - \alpha\, U\,)\Big)_{i+1,l} = \begin{pmatrix} 1-\alpha \\ 1+\alpha \end{pmatrix} n_{i+1,l}.$$

When the steady state (15) is achieved, the classical finite difference flux is given by

$$\hat{F}^{+}_{i+1/2} = n_{i,r} \begin{pmatrix} \alpha \\ 1 \end{pmatrix}, \qquad \hat{F}^{-}_{i+1/2} = n_{i+1,l} \begin{pmatrix} -\alpha \\ 1 \end{pmatrix}.$$

On the one hand, when the steady state (15) is reached, we know that

$$n_i = A\, e^{\chi(c_i)} \quad \forall\, i,$$

and since the reconstruction procedures for n and $e^{\chi(c)}$ are identical (using WENO or ENO reconstruction), we have

$$\hat{F}^{+}_{i+1/2} = A \begin{pmatrix} \alpha \\ 1 \end{pmatrix} S_{i,r}, \qquad \hat{F}^{-}_{i+1/2} = A \begin{pmatrix} -\alpha \\ 1 \end{pmatrix} S_{i+1,l}.$$

Then, by the definition of $F^{\pm}_{i+1/2}$ in (46), we get

$$F_{i+1/2} = A \begin{pmatrix} 0 \\ 1 \end{pmatrix} S_{i+1/2}.$$

On the other hand, by the construction of the source term (44),

$$S_i = n_i\, e^{-\chi(c_i)} \begin{pmatrix} 0 \\ 1 \end{pmatrix} \left(S_{i+1/2} - S_{i-1/2}\right).$$

When the steady state (15) is reached, we have $n_i\, e^{-\chi(c_i)} = A$ for all i; hence,

$$S_i = A \begin{pmatrix} 0 \\ 1 \end{pmatrix} (S_{i+1/2} - S_{i-1/2}),$$

which exactly satisfies the balance with the flux difference $F_{i+1/2} - F_{i-1/2}$.

Now let us prove (ii); i.e., the numerical scheme is a kth-order approximation to (8). The source term can be written in the form

$$n\,\chi'(c)\frac{\partial c}{\partial x} = n\, e^{-\chi(c)}\, \frac{\partial}{\partial x} e^{\chi(c)}$$

and clearly the right-hand side of (44) is a kth-order approximation to this source term when the flux $S_{i+1/2}$ is defined by (45), since both $S_{i,r}$ and $S_{i+1,l}$ are kth-order fluxes computed by WENO reconstruction. Also, it is obvious that $F^{\pm}_{i+1/2}$ is only a kth-order perturbation of the original kth-order accurate WENO flux $\hat{F}^{\pm}_{i+1/2}$, since the difference between $S_{i,r}$ and $S_{i+1,l}$ is at the level of truncation error for the kth-order WENO reconstruction. This demonstrates that the scheme (44) is kth-order accurate. ∎

5. Numerical simulations.

5.1. Accuracy test in one dimension. In this example, we consider the one-dimensional hyperbolic system (8) with $\chi'(c) = \frac{1}{1+c}$,

$$\begin{cases} \dfrac{\partial n}{\partial t} + \dfrac{\partial (nu)}{\partial x} = 0, \\ \dfrac{\partial (nu)}{\partial t} + \dfrac{\partial}{\partial x}\left(nu^2 + n\right) = \dfrac{n}{1+c}\,\dfrac{\partial c}{\partial x} - \sigma\, nu. \end{cases} \tag{48}$$

We take periodic boundary conditions, $\sigma = 1$, $t \in [0, T_{end}]$, with $T_{end} = 1$ and smooth initial data

$$n(0,x) = 1 \,+\, 0.2\,\cos(\pi\, x), \quad u(0,x) = 0, \quad x \in [-1,1],$$

whereas the chemo-attractant c is fixed to

$$c(x) = \exp(-16\, x^2), \quad x \in [-1,1].$$

This system does not admit an explicit solution, but to evaluate the error and the order of accuracy we first compute an accurate approximation using the fifth-order accurate classical WENO scheme with 1600 points and compare different numerical solutions with this reference solution. To show the advantage of the well-balanced

TABLE 1
Comparison of L^1 errors and numerical orders of accuracy for the classical WENO scheme and well-balanced WENO scheme with fourth-order Runge–Kutta time discretization.

Number of points	WENO		Well-balanced WENO	
	L^1 error	order	L^1 error	order
50	1.21E-04		7.90E-05	
100	5.86E-06	4.37	3.69E-06	4.42
200	4.00E-07	3.87	2.22E-07	4.05
400	2.00E-08	4.32	1.27E-08	4.13

TABLE 2
Comparison of L^1 errors and numerical orders of accuracy for a second-order finite volume scheme and well-balanced finite volume scheme with second-order Runge–Kutta time discretization.

Number of points	Finite volume		Well-balanced finite volume	
	L^1 error	order	L^1 error	order
50	1.13E-02		8.01E-03	
100	2.94E-03	1.94	2.30E-03	1.80
200	7.74E-04	1.92	6.01E-04	1.94
400	2.01E-04	1.94	1.58E-04	1.93

algorithm, we compare a fifth-order WENO finite difference scheme with and without the well-balanced reconstruction (presented in section 4) coupled with a fourth-order Runge–Kutta scheme for the time discretization.

In Table 1, we present the L^1 errors and numerical orders of accuracy obtained from both schemes. We observe that both methods achieve orders of accuracy above four and the well-balanced approach not only preserves the high-order accuracy, but also in this case allows us to reduce the error even if the solution is far from the equilibrium. We also give in Table 2 the L^1 errors and numerical orders of accuracy for the second-order finite volume monotone upstream-centered scheme for conservation laws (MUSCL) [16], namely, a linear reconstruction based on the usual minmod limiters,

$$\operatorname{minmod}(a,b) = \frac{sign(a) + sign(b)}{2} \min(|a|,|b|),$$

the Lax–Friedrichs monotone numerical flux, and the second-order TVD time discretization [23], with and without the well-balanced reconstruction presented in section 3. Second order is well achieved but the numerical errors are large compared to those obtained from the WENO scheme on the same mesh.

To test the steady state conservation, we consider the steady state solution

$$n_0(x) = (1 + c(x))/10, \quad u(0,x) = 0, \tag{49}$$

where c is given by

$$c_0(x) = \begin{cases} 1 & \text{if } |x| \le 1/2, \\ 0.125 & \text{else.} \end{cases} \tag{50}$$

We compute the numerical solution using the WENO scheme with and without the well-balanced reconstruction and observe that the density n and the population flux nu are exactly conserved up to the round-off error with the well-balanced reconstruction, whereas spurious oscillations are generated when a centered approximation of

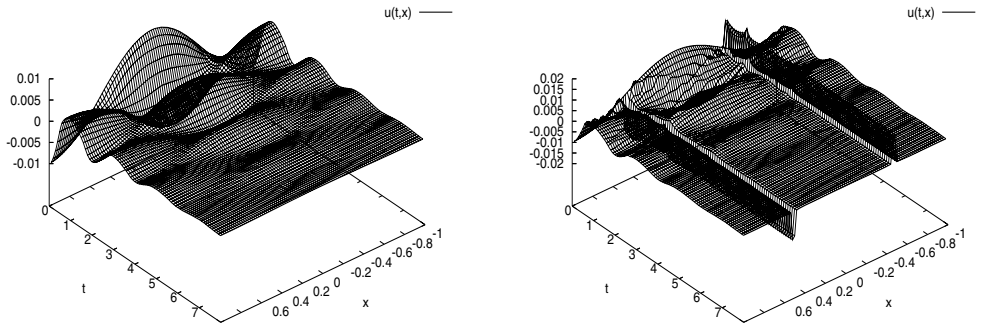


FIG. 1. *Accuracy test for well-balanced steady state resolution: time evolution of the population flux $n(t,x)\ u(t,x)$. Left: WENO scheme with the well-balanced reconstruction. Right: WENO schemes with a centered approximation of the source term.*

the source term is used. In Figure 1, we also present the numerical results obtained for a perturbation of the steady state: $n(t=0,x)$ and $c(t=0,x)$ are given by (49)–(50) and

$$u(t,x) = 0.01\,\cos(\pi x), \quad x \in [-1,1].$$

Then, some spurious oscillations appear in the population flux close to the discontinuities of the source term c when a centered approximation to the source term is used.

5.2. Convergence to the equilibrium in one dimension. Now we perform numerical simulations using the well-balanced fifth-order WENO finite difference scheme to observe the long time behavior of the solution to the hyperbolic model (48) with $\sigma = 0.25$ and $x \in [-6,6]$ with periodic boundary conditions, with the chemo-attractant c being the solution of the parabolic equation

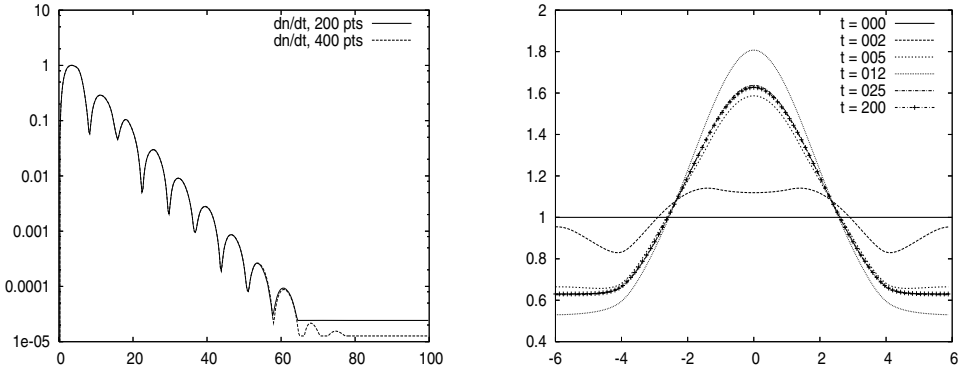
$$\begin{cases} \dfrac{\partial c}{\partial t} - \dfrac{1}{10}\dfrac{\partial^2 c}{\partial x^2} = \alpha\, n - c, \\ \\ c(t,-6) = c(t,6) = 0, \end{cases}$$

and α a given compactly supported function

$$\alpha(x) = \begin{cases} (16-x^2)/16 & \text{if } |x| \le 4, \\ 0 & \text{else.} \end{cases}$$

The initial density is chosen as a constant density $n = 1$, whereas the initial population flux $n\ u$ and chemo-attractant c are equal to zero.

To observe the convergence to an equilibrium state we plot in Figure 2 the evolution of the L^1 norm of the time derivative of the cell density $\partial_t n$. We also plot the evolution of the cell density on different time intervals and at the final time ($t = 200$) when the steady state is reached. Clearly, the curves at $t = 25$ and $t = 200$ are overlapping, indicating that the numerical solution has already converged to steady state by visual inspection at $t = 25$.



(a) (b)

FIG. 2. *Convergence to equilibrium.* (a) *Time evolution of the* L^1 *norm of the time derivative of the cell density* $\partial_t n(t,x)$. (b) *Time evolution of the cell density* $n(t,x)$ *for* $t = 0$, 2, 5, 12, 25, *and* 200.

5.3. Chemosensitive movement in two dimensions. In the previous two numerical examples, we have shown the efficiency of the finite difference WENO method for treating the convergence to steady states and the evolution of the cell density. Now we present numerical simulations obtained with the well-balanced fifth-order (in space) finite difference method to show that the hyperbolic models can give a good description of cell motion.

We consider the system in two space dimension,

$$\begin{cases} \dfrac{\partial n}{\partial t} + \operatorname{div}(nu) = 0, \\ \dfrac{\partial (nu)}{\partial t} + \operatorname{div}\left(nu \otimes u + n\right) = n\,\nabla c - \sigma\, nu, \end{cases}$$

where $\sigma = 0$ and the chemo-attractant c is fixed and concentrated at the origin (0,0),

$$c(x) = 400\, \exp(-x^2)/\pi, \quad x \in [0,2]^2.$$

We then consider two cell concentrations located far from the high concentration regions of the chemo-attractant. In Figure 3, we represent the evolution of cells, which move and stretch to the highest concentration region of the chemo-attractant. This simple test illustrates the attractive effect of the chemo-attractant gradient on the population flux nu and on the cell density n.

5.4. Formation of networks. As in [21, 5], we perform numerical simulations of the hyperbolic model (4) coupled with (2) on a square box of length $L = 20$ with periodic boundary conditions. For good accuracy, we use the well-balanced fifth-order WENO scheme described in the previous section to approximate the solution of (4) and a fourth-order finite difference scheme to approximate the solution of the parabolic equation (2),

$$\frac{dc_i}{dt} = D_c\, \frac{(-c_{i+2} + 16c_{i+1} - 30c_i + 16c_{i-1} - c_{i-2})}{12\Delta x^2} + n_i - c_i.$$

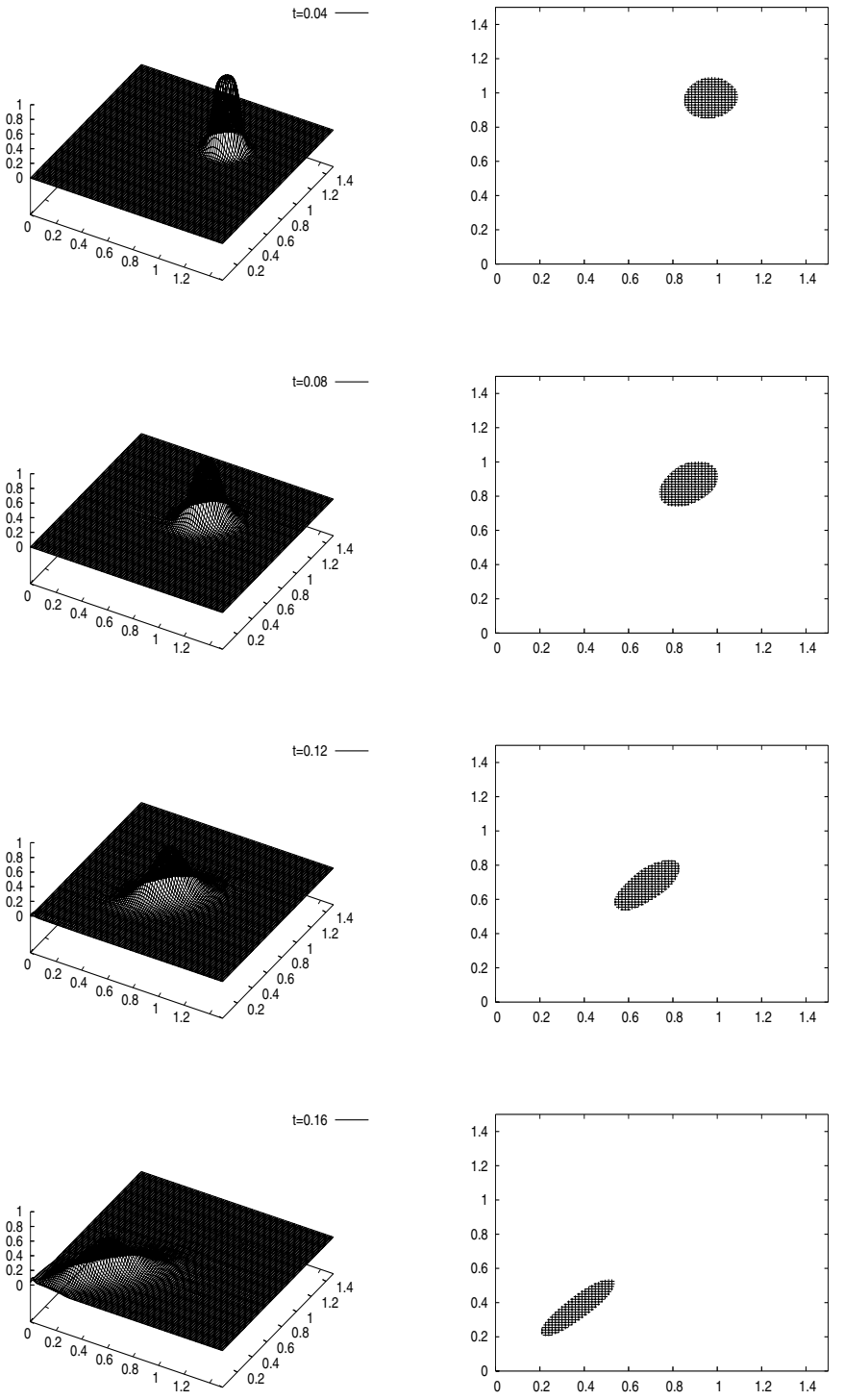
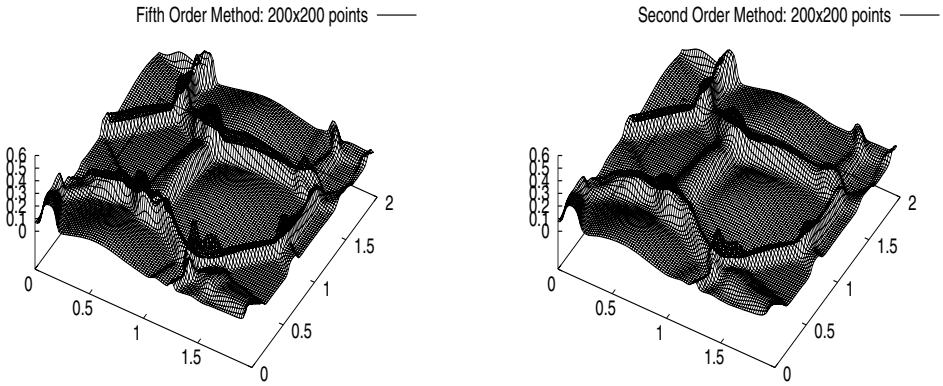


FIG. 3. *Chemosensitive movement: Time evolution of the cell density* $n(t,x)$ *at time* $t = 0.04$, 0.08, 0.12, *and* 0.16.



(a) (b)

FIG. 4. *Formation of network: comparison of numerical results between* (a) *the fifth-order WENO method and* (b) *the second-order finite volume scheme with* 200^2 *grid points.*

These results can be compared with those obtained in [4, 5] for the formation of networks: high cell concentrations are generated and linked together by line concentrations.

In (2) we take the diffusion rate $D_c = 1/10$ and also in (4) choose the pressure to be

$$P(n) = \beta\, n,$$

where $\beta = 1/4$ is the mean cell radius, $\chi'(c) = 1$, and the friction is $\sigma = 1/10$.

Now, the initial datum $n(0)$ is chosen by throwing cells randomly inside the box, with zero population flux $nu(0) = 0$ and zero concentration $c(0) = 0$ initially. More precisely, for $(x, y) \in (0, L)^2$, we choose

$$n(0, x, y) = \frac{1}{(2\,\pi\sigma)^2} \sum_{i=1}^{N} \exp\left(\frac{-\left((x - x_i)^2 + (y - y_i)^2\right)}{2\,\sigma^2} \right),$$

where $(x_i, y_i)_{1 \le i \le N}$ are uniformly distributed random variables in the box $(0, L)^2$, each Gaussian bump of width σ representing a single cell. Thus N denotes the total number of cells.

We first present a comparison between the fifth-order finite difference scheme and the second-order finite volume method to show the advantage of the high-order method. In Figure 4, we give the cell density on the square $[0, 2] \times [0, 2]$ at time $t = 0.31$ obtained with 200^2 points using the two methods. We can observe that the network is formed for both methods, but the numerical diffusion of the second-order finite volume scheme is more noticeable. We also present the cell density at time $t = 0.31$ and $x = -1.24$ for different meshes: 200^2 and 400^2 grid points for the fifth-order method, and 200^2, 400^2, and 800^2 points for the second-order method, in Figure 5. Clearly, the fifth-order WENO scheme gives results that converge faster and are more accurate than the finite volume scheme used in [4]. Even with 800^2 points, the numerical solution obtained with the second-order scheme still does not

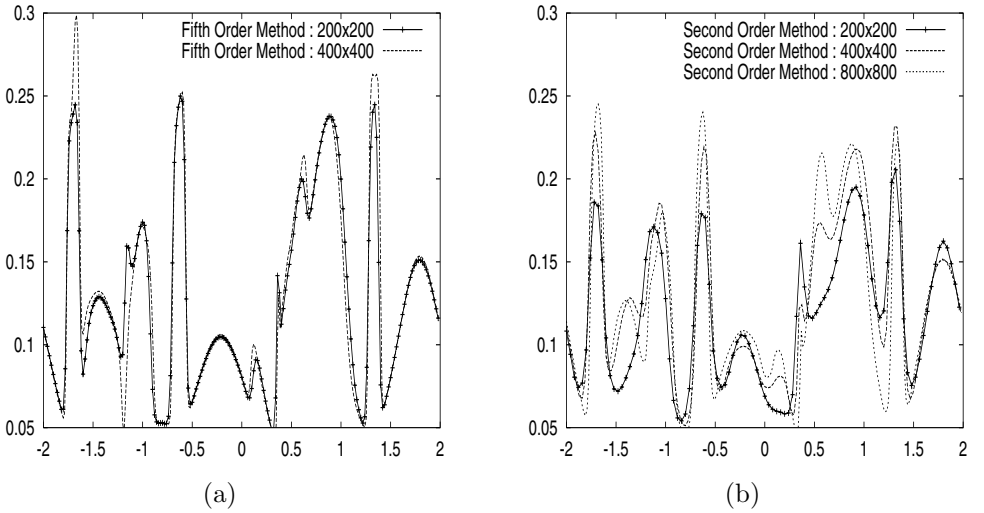


FIG. 5. *Formation of network: comparison of numerical results between* (a) *the fifth-order WENO method with* 200^2 *and* 400^2 *grid points and* (b) *the second-order finite volume scheme with* 200^2, 400^2, *and* 800^2 *grid points. Density cut at* $x = -1.24$.

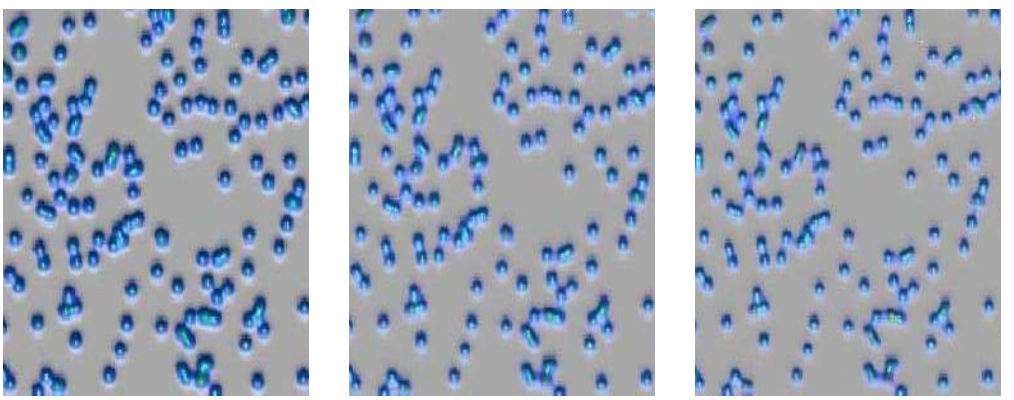


FIG. 6. *Formation of network: time evolution* ($t = 0.20$, 0.26, *and* 0.33) *of the cell density with* $N = 400$.

seem to be of the quality of the solution obtained with the fifth-order one using only 200^2 points, in terms of capturing the complicated features in the solution.

Finally, we present different numerical simulations when the initial total cell density N varies. Figures 6 and 7 represent a zoom of the cell density evolution for different initial concentrations ($N = 400$, 800, and 1200). On the one hand, in Figure 6 we observe the formation of high concentration regions but the number of cells is not large enough to generate a network. Indeed, a local density (representing several cells) does not interact with the surrounding densities; then local high concentrations are produced without any link to other cell densities. On the other hand, in Figure 7, cells are first uniformly located and then move to generate high concentration regions which are linked together by chords representing cells which are stretching. Finally, a network is formed as observed in the experiments [4, 21]. We cannot interpret the

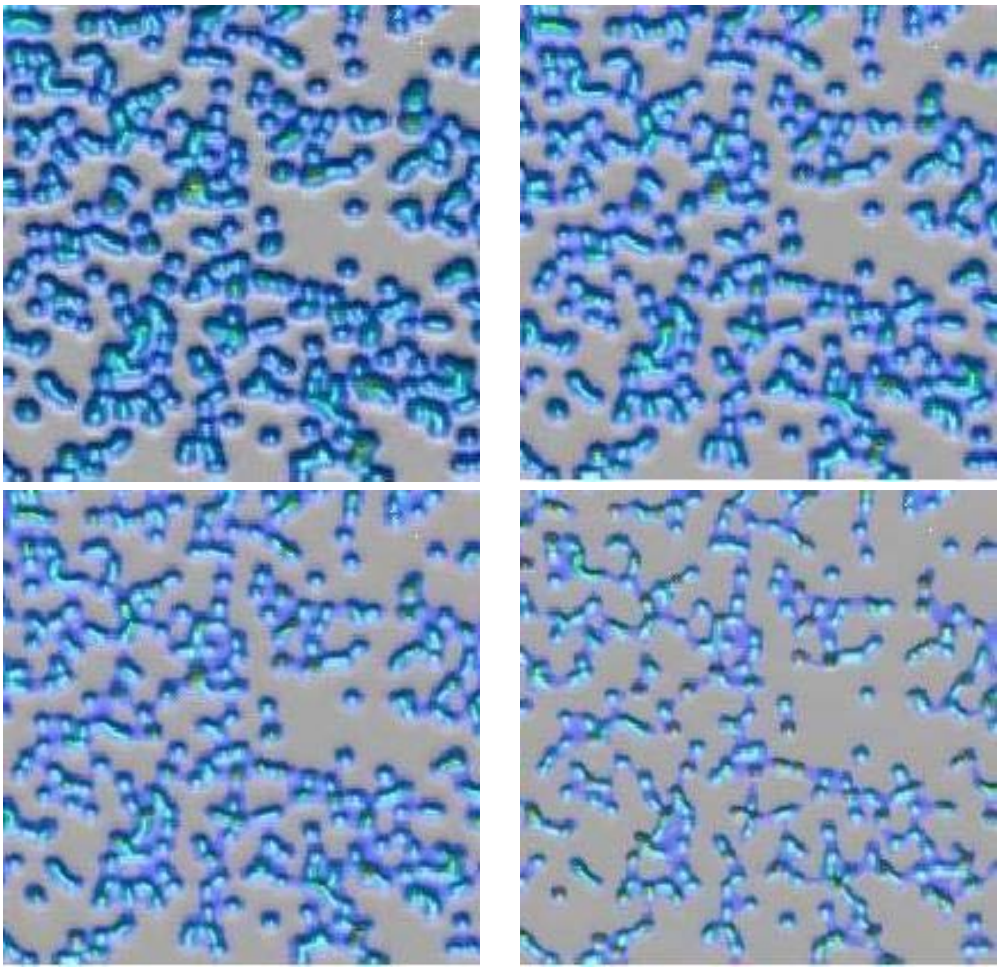


FIG. 7. *Formation of network: time evolution* ($t = 0.17$, 0.22, 0.26, *and* 0.42) *of the cell density with* $N = 800$.

results for longer time because the numerical solution blows up (cell concentrations become larger and larger), as is expected from theoretical results on other models (Euler–Poisson [14] or parabolic system PKS [9, 15]). But, the transient regime seems to be well described by the hyperbolic model. In this last example, we do not gain any advantage by using the well-balanced algorithm since the solution does not converge to a steady state, but even in this case the method works well.

REFERENCES

[1] E. AUDUSSE, F. BOUCHUT, M.-O. BRISTEAU, R. KLEIN, AND B. PERTHAME, *A fast and stable well-balanced scheme with hydrostatic reconstruction for shallow water flows*, SIAM J. Sci. Comput., 25 (2004), pp. 2050–2065.

[2] F. BOUCHUT, *Nonlinear Stability of Finite Volume Methods for Hyperbolic Conservation Laws and Well-balanced Schemes for Sources*, Birkhäuser, Basel, 2004.

[3] Y. DOLAK AND T. HILLEN, *Cattaneo models for chemotaxis, numerical solution and pattern formation*, J. Math. Biol., 46 (2003), pp. 461–478.

[4] F. Filbet, Ph. Laurençot, and B. Perthame, *Derivation of hyperbolic models for chemosensitive movement*, J. Math. Biol., 50 (2005), pp. 189–207.

[5] A. Gamba, D. Ambrosi, A. Coniglio, A. de Candia, S. Di Talia, E. Giraudo, G. Serini, L. Preziosi, and F. Bussolino, *Percolation, morphogenesis, and Burgers dynamics in blood vessels formation*, Phys. Rev. Lett., 90 (2003), 118101.

[6] L. Gosse, *Report on the Euler-Poisson Project*, CNR report, http://www-dimat.unipv.it/∼laurent/report.ps.gz (2000).

[7] L. Gosse and A.-Y. LeRoux, *A well-balanced scheme designed for inhomogeneous scalar conservation laws*, C. R. Acad. Sci. Paris Sér. I Math., 323 (1996), pp. 543–546 (in French).

[8] J. M. Greenberg and A. Y. Leroux, *A well-balanced scheme for the numerical processing of source terms in hyperbolic equations*, SIAM J. Numer. Anal., 33 (1996), pp. 1–16.

[9] M. A. Herrero, E. Medina, and J. J. L. Velázquez, *Finite-time aggregation into a single point in a reaction-diffusion system*, Nonlinearity, 10 (1997), pp. 1739–1754.

[10] T. Hillen and H. G. Othmer, *The diffusion limit of transport equations derived from velocity jump processes*, SIAM J. Appl Math., 61 (2000), pp. 751–775.

[11] T. Hillen, *Hyperbolic models for chemosensitive movement*, Math. Models Methods Appl. Sci., 12 (2002), pp. 1007–1034.

[12] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[13] E. F. Keller and L. A. Segel, *Traveling band of chemotactic bacteria: A theoretical analysis*, J. Theoret. Biol., 30 (1971), pp. 235–248.

[14] T. Makino and B. Perthame, *Sur les solutions à symétrie sphérique de l'équation d'Euler-Poisson pour l'évolution d'étoiles gazeuses*, Japan J. Appl. Math., 7 (1990), pp. 165–170.

[15] A. Marrocco, *Numerical simulation of chemotactic bacteria aggregation via mixed finite elements*, M2AN Math. Model. Numer. Anal., 37 (2003), pp. 617–630.

[16] S. Osher, *Convergence of generalized MUSCL schemes*, SIAM J. Numer. Anal., 22 (1985), pp. 947–961.

[17] H. G. Othmer and T. Hillen, *The diffusion limit of transport equations* II: *Chemotaxis equations*, SIAM J. Appl. Math., 62 (2002), pp. 1222–1250.

[18] H. G. Othmer, S. R. Dunbar, and W. Alt, *Models of dispersal in biological systems*, J. Math. Biol., 26 (1988), pp. 263–298.

[19] C. S. Patlak, *Random walk with persistence and external bias*, Bull. Math. Biol. Biophys., 15 (1953), pp. 311–338.

[20] P. Roe, *Approximate Riemann solvers, parameter vectors and difference schemes*, J. Comput. Phys., 27 (1978), pp. 1–31.

[21] G. Serini, D. Ambrosi, E. Giraudo, A. Gamba, L. Preziosi, and F. Bussolino, *Modeling the early stages of vascular network assembly*, EMBO J., 22 (2003), pp. 1771–1779.

[22] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, B. Cockburn, C. Johnson, C.-W. Shu, and E. Tadmor, eds., Lecture Notes in Math. 1697, Springer, New York, 1998, pp. 325–432.

[23] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[24] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, II. J. Comput. Phys., 83 (1989), pp. 32–78.

[25] A. Stevens, *The derivation of chemotaxis equations as limit dynamics of moderately interacting stochastic many-particle systems*, SIAM J. Appl. Math., 61 (2000), pp. 183–212.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 873–892
© 2005 Society for Industrial and Applied Mathematics

AN ALGEBRAIC SUBSTRUCTURING METHOD FOR LARGE-SCALE EIGENVALUE CALCULATION*

CHAO YANG†, WEIGUO GAO‡, ZHAOJUN BAI§, XIAOYE S. LI†, LIE-QUAN LEE¶, PARRY HUSBANDS†, AND ESMOND NG†

Abstract. This paper is concerned with solving large-scale eigenvalue problems by algebraic substructuring. Algebraic substructuring refers to the process of applying matrix reordering and partitioning algorithms to divide a large sparse matrix into smaller submatrices from which a subset of spectral components are extracted and combined to form approximate solutions to the original problem. Through an algebraic analysis, we identify critical conditions under which a simple version of algebraic substructuring works well. This particular version of substructuring is identical to the component mode synthesis (CMS) method (see [R. R. Craig and M. C. C. Bampton, *Coupling of substructures for dynamic analysis*, AIAA J., 6 (1968), pp. 1313–1319] and [W. C. Hurty, *Vibrations of structure systems by component-mode synthesis*, J. Engrg. Mech., 86 (1960), pp. 51–69]) when the matrix reordering is based on a geometric partitioning of the computational domain. We observe an interesting connection between the accuracy of an approximate eigenpair obtained through substructuring and the distribution of the components of eigenvectors of a canonical matrix pencil congruent to the original problem. A priori error bounds for the smallest eigenpair approximation are developed. This development leads to a simple heuristic for choosing spectral components (modes) from each substructure. The effectiveness of such a heuristic is demonstrated with numerical examples. We show that algebraic substructuring can be effectively used to solve a generalized eigenvalue problem arising from the finite element analysis of an accelerator structure.

Key words. algebraic substructuring, projection method, eigenvalue problem

AMS subject classifications. 65F15, 65G05

DOI. 10.1137/040613767

1. Introduction.

Substructuring is a commonly used technique for studying the static or dynamic properties of large engineering structures [11, 12, 23, 17]. The basic idea of substructuring is analogous to the concept of domain decomposition, which is widely used in the numerical solution of partial differential equations [28, 25]. By dividing a large structure model or computational domain into a few smaller components (substructures), one can often obtain an approximate solution to the original problem from a linear combination of solutions to similar problems defined on the

*Received by the editors August 23, 2004; accepted for publication (in revised form) May 12, 2005; published electronically November 22, 2005. This work was supported by the Director, Office of Science, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract DE-AC03-76SF00098. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/27-3/61376.html

†Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (CYang@lbl.gov, XSLi@lbl.gov, PJRHusbands@lbl.gov, EGNg@lbl.gov).

‡Department of Mathematics, Fudan University, Shanghai, People's Republic of China 200433 (WGGao@lbl.gov). The research of this author was carried out while he was visiting the Computational Research Division at Lawrence Berkeley National Laboratory.

§Department of Computer Science, University of California, Davis, CA 95616 (bai@cs.ucdavis.edu). The research of this author was supported in part by the National Science Foundation under grant 0220104.

¶Stanford Linear Accelerator Center, Menlo Park, CA 94025 (liequan@slac.stanford.edu). The research of this author was supported by U.S. Department of Energy contract DE-AC03-76SF00515.

substructures. Because solving problems on each substructure requires far less computational power than what would be required to solve the entire problem as a whole, substructuring can lead to a significant reduction in the computational time required to carry out a large-scale simulation and analysis.

The automated multilevel substructuring (AMLS) method [5, 6, 7, 18] is an extension of a substructuring method called [illegible] (CMS) [11, 17] originally developed in the 1960s. Recent studies have shown that AMLS can be used successfully in the vibration and acoustic analysis of large-scale finite element models of automobile bodies [18, 21]. The frequency response analysis performed in these studies requires computing several thousand eigenvalues and eigenvectors associated with a large-scale symmetric generalized eigenvalue problem. The timing results reported in [18, 21] indicate that AMLS is significantly faster than conventional Lanczos-based approaches [22, 16].

It is important to note that the accuracy achieved by a substructuring method such as AMLS is typically lower than that achieved by the standard Lanczos algorithm. However, in many applications, the level of accuracy required for an approximate solution to an algebraic problem is no more than what is provided by the finite element scheme used to discretize the original continuous problem. Thus, the use of substructuring is easily justified as long as the error associated with the substructuring approximation does not exceed that produced by the finite element discretization.

Asymptotic analysis is performed in [8, 9] to assess the level of accuracy attainable by the CMS method. The analysis is based on the standard finite element theory and properties of the partial differential equation governing the evolution of the structure. The recent work described in [7] provides a high-level mathematical description of the AMLS in a continuous variational setting. However, neither of these studies provides a satisfactory algebraic explanation on why substructuring works well in practice.

Our focus in this paper is on examining substructuring methods for solving large-scale eigenvalue problems from a purely algebraic point of view. We use the term [illegible] to refer to the process of applying matrix reordering and partitioning algorithms (such as the [illegible] algorithm [15]) to divide a large sparse matrix into smaller submatrices from which a subset of spectral components are extracted and combined to form an approximate solution to the original eigenvalue problem. Through algebraic manipulation, we identify the critical conditions under which a simple version of algebraic substructuring works well. This particular version of substructuring is identical to the CMS method when the matrix reordering is based on the geometric partitioning of the computational domain. We observe an interesting connection between the accuracy of an approximate eigenpair obtained through substructuring and the distribution of components of eigenvectors associated with a canonical matrix pencil congruent to the original problem. An error estimate for the approximation to the smallest eigenpair is developed. The estimate leads to a simple heuristic for choosing spectral components (modes) from each substructure. The effectiveness of such a heuristic is demonstrated with numerical examples. Our analysis is related to but different from the recent work by Bekas and Saad [4], who view algebraic substructuring as an approximation to the spectral Schur complement method [1, 2, 10].

Our interest in algebraic substructuring is motivated in part by an application arising from the simulation of the electromagnetic field associated with the next generation particle accelerator design [20]. We will show through a numerical example that algebraic substructuring can be used effectively to compute the cavity resonance frequencies and the electromagnetic field generated by a linear particle accelerator

model.

Our presentation is organized as follows. In section 2, we give a brief overview of the algorithmic ingredients of a simple algebraic substructuring method. The accuracy of the approximate eigenpairs is analyzed in section 3. Our analysis of algebraic substructuring is confirmed by numerical examples presented in section 4.

Throughout this paper, uppercase and lowercase Latin letters denote matrices and vectors, respectively, while lowercase Greek letters denote scalars. An $n \times n$ identity matrix will be denoted by I_n. The jth column of the identity matrix is denoted by e_j. The transpose of a matrix A is denoted by A^T. We use $\|x\|$ to denote the standard 2-norm of x and use $\|x\|_M$ to denote the M-norm defined by $\|x\|_M = \sqrt{x^T M x}$. We will use $\angle_M(x, y)$ to denote the M inner product induced acute angle (M-angle for short) between x and y. This angle can be computed from

$$\cos \angle_M(x, y) = \frac{x^T M y}{\|x\|_M \|y\|_M}.$$

Similarly, we use $\angle_M(x, \mathcal{S})$ to denote the M-angle between a vector x and a subspace $\mathcal{S}$. This angle can be computed from

$$\cos \angle_M(x, \mathcal{S}) = \frac{\|Q^T M x\|_2}{\|x\|_M}, \tag{1}$$

where Q is an M-orthonormal basis of the subspace $\mathcal{S}$; i.e., $\mathcal{S} = \text{span}\{S\}$ and $Q^T M Q = I$.

A matrix pencil (K, M) is said to be [illegible] if both K and M are symmetric and M is positive definite. A matrix pencil (K, M) is said to be [illegible] to another pencil (A, B) if there exists a nonsingular matrix P such that $A = P^T K P$ and $B = P^T M P$.

2. Algebraic substructuring. In this section, we briefly describe a single-level algebraic substructuring algorithm. Our description does not use any information regarding the geometry or the physical structure on which the original problem is defined.

We are concerned with solving the generalized algebraic eigenvalue problem

$$Kx = \lambda M x, \tag{2}$$

where K is symmetric and M is symmetric positive definite. We assume both K and M are sparse. They may or may not have the same sparsity pattern. Suppose the rows and columns of K and M have been permuted so that these matrices can be partitioned as

$$K = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_2 & n_3 \end{array} \\ \begin{pmatrix} K_{11} & & K_{13} \\ & K_{22} & K_{23} \\ K_{13}^T & K_{23}^T & K_{33} \end{pmatrix} \end{array} \quad \text{and} \quad M = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \end{array} \begin{array}{c} \begin{array}{ccc} n_1 & n_2 & n_3 \end{array} \\ \begin{pmatrix} M_{11} & & M_{13} \\ & M_{22} & M_{23} \\ M_{13}^T & M_{23}^T & M_{33} \end{pmatrix} \end{array}, \tag{3}$$

where the labels n_1, n_2, and n_3 are inserted into the top and left borders of the partitioned matrices to indicate the dimension of each submatrix block. The permutation can be accomplished by applying a matrix ordering and partitioning algorithm, such as the nested dissection algorithm [15], to the matrix $|K| + |M|$.

The pencils (K_{11}, M_{11}) and (K_{22}, M_{22}) now define two algebraic substructures that are connected by the third block rows and columns of K and M, which we will refer to collectively as the [illegible] block. We assume that n_3 is much smaller than n_1 and n_2.

A single-level algebraic substructuring algorithm proceeds by performing a block factorization

$$K = LDL^T, \tag{4}$$

where

$$L = \begin{pmatrix} I_{n_1} & & \\ & I_{n_2} & \\ K_{13}^T K_{11}^{-1} & K_{23}^T K_{22}^{-1} & I_{n_3} \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} K_{11} & & \\ & K_{22} & \\ & & \widehat{K}_{33} \end{pmatrix}.$$

The last diagonal block of D, often known as the [illegible], is defined by

$$\widehat{K}_{33} = K_{33} - K_{13}^T K_{11}^{-1} K_{13} - K_{23}^T K_{22}^{-1} K_{23}.$$

The inverse of the lower triangular factor L defines a congruence transformation that, when applied to the matrix pencil (K, M), yields a new matrix pencil $(\widehat{K}, \widehat{M})$:

$$\widehat{K} = L^{-1} K L^{-T} = D \quad \text{and} \quad \widehat{M} = L^{-1} M L^{-T} = \begin{pmatrix} M_{11} & & \widehat{M}_{13} \\ & M_{22} & \widehat{M}_{23} \\ \widehat{M}_{13}^T & \widehat{M}_{23}^T & \widehat{M}_{33} \end{pmatrix}. \tag{5}$$

The off-diagonal blocks of $\widehat{M}$ satisfy

$$\widehat{M}_{i3} = M_{i3} - M_{ii} K_{ii}^{-1} K_{i3} \quad \text{for} \quad i = 1, 2.$$

The last diagonal block of $\widehat{M}$ satisfies

$$\widehat{M}_{33} = M_{33} - \sum_{i=1}^{2} (K_{i3}^T K_{ii}^{-1} M_{i3} + M_{i3}^T K_{ii}^{-1} K_{i3} - K_{i3}^T K_{ii}^{-1} M_{ii} K_{ii}^{-1} K_{i3}).$$

The pencil $(\widehat{K}, \widehat{M})$ is often called the [illegible] form [11] in structural engineering. Note that the eigenvalues of $(\widehat{K}, \widehat{M})$ are identical to those of (K, M), and the corresponding eigenvectors $\widehat{x}$ are related to the eigenvectors of the original problem (2) through $\widehat{x} = L^T x$.

The substructuring algorithm constructs a subspace in the form of

$$S = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & k_2 & n_3 \end{array} \\ \begin{pmatrix} S_1 & & \\ & S_2 & \\ & & I_{n_3} \end{pmatrix} \end{array}, \tag{6}$$

where S_1 and S_2 consist of k_1 and k_2 selected eigenvectors of (K_{11}, M_{11}) and (K_{22}, M_{22}), respectively. These eigenvectors will be referred to as [illegible] in the discussion that follows. Note that k_1 and k_2 are typically much smaller than n_1 and n_2, respectively.

The approximation to the desired eigenvalues and eigenvectors of the pencil $(\widehat{K}, \widehat{M})$ is obtained by projecting the pencil $(\widehat{K}, \widehat{M})$ onto the subspace spanned by S; i.e., we seek θ and $q \in \mathbb{R}^{k_1+k_2+n_3}$ such that

$$(S^T \widehat{K} S)q = \theta(S^T \widehat{M} S)q. \tag{7}$$

It follows from the standard Rayleigh–Ritz theory [24, p. 213] that θ serves as an approximation to an eigenvalue of (K, M), and the vector formed by $z = L^{-T}Sq$ is the approximation to the corresponding eigenvector.

A summary of the single-level algebraic substructuring algorithm described in this section is provided below.

ALGORITHM. Single-level algebraic substructuring.

Input: A matrix pencil (K, M), where $K = K^T$ and $M = M^T > 0$;
Output: $\theta_j \in \mathbb{R}$ and $z_j \in \mathbb{R}^n$, $(j = 1, 2, \ldots, k)$ such that $Kz_j \approx \theta_j M z_j$.

1. Order K and M to be in the form of (3);
2. Perform block factorization $K = LDL^T$;
3. Compute a subset of eigenpairs of the substructures (K_{11}, M_{11}) and (K_{22}, M_{22}). The eigenvectors of each substructure form the columns of S_1 and S_2, respectively;
4. Project the matrix pencil (K, M) into subspace spanned by columns of $Z = L^{-T}S$, where S is defined by (6);
5. Compute k desired eigenpairs (θ_j, q_j) from $(Z^T K Z, Z^T M Z)$, and set $z_j = Zq_j$ $(j = 1, 2, \ldots, k)$; apply the inverse of the permutation used in step 1 to z_j to restore the original order of the solution.

A few remarks are in order.

- Note that the most expensive computational task associated with this algorithm is the block factorization $K = LDL^T$ and the congruence transformation of M required for projecting M into the subspace spanned by $Z = L^{-T}S$. These computational tasks must be carried out with care in order to reduce memory requirements and floating point operations. However, it is beyond the scope of this paper to discuss these important implementation issues.
- Since $k_1 \ll n_1$ and $k_2 \ll n_2$, step 3 of the algorithm can be carried out by using a shift-invert Lanczos algorithm to obtain a small number of desired eigenpairs from each substructure. The cost of this computation is generally small compared to the rest of the computation, especially when this algorithm is extended to a multilevel scheme.
- Similarly, because n_3 is typically much smaller than n_1 and n_2, the dimension of the projected problem (7) is significantly smaller than that of the original problem. Thus, the cost of solving (7) is also small compared to step 2 of the algorithm.
- Decisions must be made on how to select eigenvectors from each substructure. The selection should be made in such a way that the subspace spanned by the columns of Z retains a sufficient amount of spectral information from (K, M). The process of choosing appropriate eigenvectors from each substructure will be referred to as [illegible]. We will postpone the discussion on this key aspect of the algebraic substructuring algorithm until the next section.

The algebraic substructuring algorithm we presented above becomes identical to the CMS method [11, 17] when the permutation of the pencil in (3) is derived from a

geometric partitioning of the computational domain. The algorithm can be extended in two ways. First, the matrix reordering and partitioning scheme used to create the block structure of (3) can be applied recursively to (K_{11}, M_{11}) and (K_{22}, M_{22}), respectively, to produce a multilevel division of (K, M) into smaller submatrices. The reduced computational cost associated with finding selected eigenpairs from these even smaller submatrices further improves the efficiency of the algorithm. Second, one may replace I_{n_3} in (6) with a subset of eigenvectors of the interface pencil $(\widehat{K}_{33}, \widehat{M}_{33})$. This modification will further reduce the computational cost associated with solving the projected eigenvalue problem (7). A combination of these two extensions yields the AMLS algorithm presented in [18, 7]. However, we will limit the scope of our presentation to a single-level substructuring algorithm.

3. Accuracy and error estimation. Algebraic substructuring allows one to break a large-scale eigenvalue problem into a set of smaller subproblems that are easier to solve. The algorithm would be less attractive to use if one had to compute all eigenvalues and eigenvectors of each subproblem. Fortunately, such a calculation is not necessary, as we will show in this section. In practice, only a small subset of the eigenvectors of (K_{11}, M_{11}) and (K_{22}, M_{22}) is needed to assemble the projection subspace spanned by the columns of the matrix S in (6). To simplify the analysis, we will work with the matrix pencil $(\widehat{K}, \widehat{M})$, where $\widehat{K}$ and $\widehat{M}$ are defined as in (5). As we noted earlier, $(\widehat{K}, \widehat{M})$ and (K, M) have the same set of eigenvalues. If $\widehat{x}$ is an eigenvector of $(\widehat{K}, \widehat{M})$, then $x = L^{-T}\widehat{x}$ is an eigenvector of (K, M), where L is the transformation defined in (4).

Let $(\mu_j^{(i)}, v_j^{(i)})$ be the jth eigenpair of the ith subproblem, i.e.,

$$K_{ii} v_j^{(i)} = \mu_j^{(i)} M_{ii} v_j^{(i)},$$

where $v_j^{(i)}$ is M_{ii}-orthonormal, i.e.,

$$(v_j^{(i)})^T M_{ii} v_k^{(i)} = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise.} \end{cases}$$

To simplify our discussion, we assume that $\mu_j^{(i)}$ has been ordered such that

$$\mu_1^{(i)} \le \mu_2^{(i)} \le \cdots \le \mu_{n_i}^{(i)}. \tag{8}$$

Let us define $V_i = (v_1^{(i)} v_2^{(i)}, \ldots, v_{n_i}^{(i)})$, $V = \mathrm{diag}(V_1, V_2, I_{n_3})$ and $\Lambda_i = \mathrm{diag}(\mu_1^{(i)}, \mu_2^{(i)}, \ldots, \mu_{n_i}^{(i)})$. An eigenvector of $(\widehat{K}, \widehat{M})$, say $\widehat{x}$, can be expressed as a linear combination of columns of V. That is,

$$\widehat{x} = Vy = \begin{pmatrix} V_1 & & \\ & V_2 & \\ & & I_{n_3} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \tag{9}$$

where $y = (y_1^T, y_2^T, y_3^T)^T$ is an eigenvector of the generalized eigenvalue problem

$$\begin{pmatrix} \Lambda_1 & & \\ & \Lambda_2 & \\ & & \widehat{K}_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \lambda \begin{pmatrix} I_{n_1} & & G_{13} \\ & I_{n_2} & G_{23} \\ G_{13}^T & G_{13}^T & \widehat{M}_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}, \tag{10}$$

where $G_{13} = V_1^T \widehat{M}_{13}$ and $G_{23} = V_2^T \widehat{M}_{23}$. Note the matrix pencil defined by (10) can be obtained by applying V^T from the left to $\widehat{K}\widehat{x} = \lambda \widehat{M}\widehat{x}$ and expressing $\widehat{x}$ by $\widehat{x} = Vy$. This pencil is clearly congruent to the pencils $(\widehat{K}, \widehat{M})$ and (K, M). Thus they share the same set of eigenvalues. We will refer to (10) as a [illegible] of the generalized eigenvalue problem (2).

If $\widehat{x}$ can be well approximated by a linear combination of the columns of S, as suggested by the description of the algorithm in section 2, then the vectors y_1 and y_2 must contain only a few large entries. All other components of y_1 and y_2 are likely to be small and negligible. In this section, we seek to formalize this key concept by developing a priori error bounds for the approximations to the smallest eigenvalue of $(\widehat{K}, \widehat{M})$ and the corresponding eigenvector. As we will see below, these bounds can be expressed in terms of the small components of y_1 and y_2.

Suppose $\Lambda_i - \lambda I_{n_i}$ is nonsingular, for $i = 1, 2$. It follows from the first two block rows of the canonical eigenproblem (10) that

$$y_i = \lambda(\Lambda_i - \lambda I)^{-1} G_{i3} y_3. \tag{11}$$

Consequently, we can express the jth element of y_i by

$$e_j^T y_i = \frac{\lambda}{\mu_j^{(i)} - \lambda} g_j^{(i)} = \frac{1}{\mu_j^{(i)}/\lambda - 1} g_j^{(i)}, \tag{12}$$

where $g_j^{(i)} = e_j^T G_{i3} y_3$. It is easy to see from (12) that, when $|\mu_j^{(i)}/\lambda| \approx 1$, $|e_j^T y_i|$ will be relatively large, provided $|g_j^{(i)}|$ is bounded from below. On the other hand, if $\mu_j^{(i)}$ is far away from λ, and if $|g_j^{(i)}|$ is bounded from above, $|e_j^T y_i|$ will be relatively small. Thus, if λ is surrounded by a few eigenvalues of (K_{ii}, M_{ii}), and if S_i contains only the eigenvectors associated with these eigenvalues, one would expect to obtain an accurate approximation to λ by solving the projected problem (7).

To make the above statements more precise, we introduce some additional notation. Let us define

$$\rho_k(\omega) = \left| \frac{\lambda_k}{\omega - \lambda_k} \right|, \tag{13}$$

where λ_k is the kth eigenvalue of (K, M). If $|g_j^{(i)}| \in [\gamma_1, \gamma_2]$ for some modest-size constants $\gamma_1 < \gamma_2$, then $\rho_k(\mu_j^{(i)})$ provides a reliable measure for $|e_j^T y_i|$.

It is easy to verify that

$$\rho_k(\mu_{j+1}^{(i)}) \le \rho_k(\mu_j^{(i)}) \quad \text{for } \mu_j^{(i)} > \lambda_k$$

and

$$\rho_k(\mu_j^{(i)}) \le \rho_k(\mu_{j+1}^{(i)}) \quad \text{for } \mu_{j+1}^{(i)} < \lambda_k.$$

These inequalities suggest that $\rho_k(\mu_j^{(i)})$, and therefore $|e_j^T y_i|$, is relatively large when $\mu_j^{(i)}$ is sufficiently close to λ_k.

Let us now focus on the special case in which $k = 1$, i.e., the case associated with the smallest eigenvalue. Because (K_{ii}, M_{ii}) represents the restriction of the pencil $(\widehat{K}, \widehat{M})$ to a subspace, all of its eigenvalues satisfy

$$\lambda_1 \le \mu_j^{(i)} \le \lambda_n.$$

Consequently, the inequality

$$\rho_1(\mu_1^{(i)}) \geq \rho_1(\mu_2^{(i)}) \geq \cdots \geq \rho_1(\mu_{n_1}^{(i)}) \tag{14}$$

holds. Suppose $k_i < n_i$ is the smallest integer such that $\rho_1(\mu_{k_i+1}^{(i)}) \leq \tau$ for some $\tau \ll 1$; then we can assert, under the assumption

$$|g_j^{(i)}| \leq \gamma, \quad \text{for some small constant } \gamma,$$

that $|e_j^T y_i|$ is relatively small for all $j > k_i$. This assertion follows directly from (14) and the observation made in (12). Hence, if our goal is to seek an accurate approximation to λ_1 by projecting $(\widehat{K}, \widehat{M})$ into a subspace $\mathcal{S}$ spanned by the columns of

$$S = \begin{array}{c} \\ n_1 \\ n_2 \\ n_3 \end{array} \begin{array}{c} \begin{array}{ccc} k_1 & k_2 & n_3 \end{array} \\ \begin{pmatrix} S_1 & & \\ & S_2 & \\ & & I_{n_3} \end{pmatrix} \end{array}, \tag{15}$$

it is natural to set S_i to include only the leading k_i columns of V_i.

Given this choice of subspace, it remains to be shown how much accuracy one can expect from the approximate eigenvalue and eigenvector obtained by applying the Rayleigh–Ritz procedure to S. To simplify our discussion, let us assume that λ_1 is simple. Suppose θ_1 is the smallest eigenvalue of the projected problem

$$(S^T \widehat{K} S)q = \theta(S^T \widehat{M} S)q,$$

and q_1 is the corresponding eigenvector. We will now quantify the accuracy of the [illegible] (θ_1, u_1), where $u_1 = Sq_1$, by providing a priori error bounds for both $\theta_1 - \lambda_1$ and $\angle_{\widehat{M}}(\widehat{x}_1, u_1)$ in terms of small elements of y_1 and y_2. Note that $\angle_{\widehat{M}}(\widehat{x}_1, u_1)$ is the $\widehat{M}$-angle (between $\widehat{x}$ and u_1) defined in section 1.

To develop these error bounds, we use the following theorem, which is a generalization of a similar theorem associated with a standard symmetric eigenvalue problem [26, 27].

THEOREM 3.1. [illegible] $K, M \in \mathbb{R}^{n \times n}$ [illegible] M [illegible] (K, M) (λ_i, x_i) [illegible]

$$\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n.$$

[illegible] (θ_i, u_i) $i = 1, 2, \ldots, k$ [illegible] k [illegible] $\mathcal{S}$ [illegible]

$$\theta_1 \leq \theta_2 \leq \cdots \leq \theta_k,$$

[illegible]

$$\theta_1 - \lambda_1 \leq (\lambda_n - \lambda_1) \sin^2 \angle_M(x_1, \mathcal{S}), \tag{16}$$

$$\sin \angle_M(u_1, x_1) \leq \sqrt{\frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}} \sin \angle_M(x_1, \mathcal{S}), \tag{17}$$

[illegible] $\angle_M(u_1, x_1)$ [illegible] M [illegible] x_1 [illegible] u_1 [illegible] $\angle_M(x_1, \mathcal{S})$ [illegible] M [illegible] x_1 [illegible] $\mathcal{S}$

The proof of Theorem 3.1 is included in [30]. The theorem suggests that the accuracy of the desired Ritz pair is determined largely by the $\widehat{M}$-angle between the exact eigenvector $\widehat{x}_1$ and the subspace $\mathcal{S}$ from which the Ritz pair is extracted. We now focus on seeking a bound for $\sin\angle_{\widehat{M}}(\widehat{x}_1, \mathcal{S})$. The following theorem, which is a generalization of a similar theorem in [29, p. 250], provides a useful characterization for $\sin\angle_{\widehat{M}}(\widehat{x}_1, \mathcal{S})$.

THEOREM 3.2. [illegible] $\|x\|_M = 1$ [illegible] $\mathcal{S}$ [illegible]

$$\sin\angle_M(x, \mathcal{S}) = \min_{w \in \mathcal{S}} \|x - w\|_M.$$

Theorem 3.2 suggests that we can provide a bound for $\sin\angle_{\widehat{M}}(\widehat{x}_1, \mathcal{S})$ by measuring the distance between $\widehat{x}_1$ and a particular choice of a vector $w \in \mathcal{S}$ that is "close" to $\widehat{x}_1$ in the $\widehat{M}$-norm.

Our choice of such a vector $w \in \mathcal{S}$ is made as follows. We define $\hat{y}_i$ $(i = 1, 2)$ by

$$e_j^T \widehat{y}_i = \begin{cases} e_j^T y_i & \text{for } j \le k_i, \\ 0 & \text{for } k_i < j \le n_i, \end{cases} \tag{18}$$

where y_i satisfies

$$\begin{pmatrix} \Lambda_1 & & \\ & \Lambda_2 & \\ & & \widehat{K}_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \lambda_1 \begin{pmatrix} I_{n_1} & & G_{13} \\ & I_{n_2} & G_{23} \\ G_{13}^T & G_{23}^T & \widehat{M}_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}. \tag{19}$$

It is easy to verify that

$$w = \begin{pmatrix} V_1 & & \\ & V_2 & \\ & & I \end{pmatrix} \begin{pmatrix} \widehat{y}_1 \\ \widehat{y}_2 \\ y_3 \end{pmatrix} \in \mathcal{S} = \operatorname{span}\{S\}.$$

For this particular choice of w, we can easily show that

$$\widehat{x}_1 - w = \begin{pmatrix} V_1 & & \\ & V_2 & \\ & & I \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ 0 \end{pmatrix},$$

where $h_i = y_i - \widehat{y}_i$ for $i = 1, 2$. Consequently, we have

$$\|\widehat{x}_1 - w\|_{\widehat{M}}^2 = \begin{pmatrix} h_1^T & h_2^T & 0 \end{pmatrix} \begin{pmatrix} I & & G_{13} \\ & I & G_{23} \\ G_{13}^T & G_{23}^T & \widehat{M}_{33} \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \\ 0 \end{pmatrix} = h_1^T h_1 + h_2^T h_2.$$

Hence, we can now conclude that

$$\sin\angle_{\widehat{M}}(\widehat{x}_1, \mathcal{S}) = \min_{w \in \mathcal{S}} \|\widehat{x}_1 - w\|_{\widehat{M}} \le \sqrt{h_1^T h_1 + h_2^T h_2}. \tag{20}$$

Note that the vector w is essentially obtained from (9) by truncating components associated with the trailing $n_i - k_i$ elements of y_i. These elements are typically small, and they form the nonzero entries of h_i.

Combining (16) and (17) with (20), we obtain the following result.

THEOREM 3.3. [illegible] $\widehat{K}$ [illegible] $\widehat{M}$ [illegible] (5) [illegible] $(\lambda_i, \widehat{x}_i)$ $(i = 1, 2, \ldots, n)$ [illegible] $(\widehat{K}, \widehat{M})$ [illegible] $\lambda_1 < \lambda_2 \le \cdots \le \lambda_n$

(θ_i, u_i) $(i = 1, 2, \ldots, k)$ k $\mathcal{S}$ S (15) $\theta_1 \le \theta_2 \le \cdots \le \theta_k$

$$\theta_1 - \lambda_1 \le (\lambda_n - \lambda_1)(h_1^T h_1 + h_2^T h_2), \tag{21}$$

$$\sin \angle_{\widehat{M}}(u_1, \hat{x}_1) \le \sqrt{\frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}} \sqrt{h_1^T h_1 + h_2^T h_2}, \tag{22}$$

$h_i = y_i - \widehat{y}_i$ y_i $\widehat{y}_i$ $(i = 1, 2)$ (19) (18)

Theorem 3.3 indicates that the accuracy of (θ_1, u_1) is proportional to the size of $h_1^T h_1 + h_2^T h_2$, a quantity that provides a cumulative measure of the "truncated" components in (9). Similar a priori error estimates can be made for other Ritz pairs by utilizing a generalization of Theorems 4.5 and 4.6 in [26, pp. 135–136] which are developed for standard eigenvalue problems. However, to keep our presentation concise, we will not pursue this type of error estimate.

To end this section, we provide an estimate for $h_1^T h_1 + h_2^T h_2$ that is independent of the number of nonzero elements in h_1 and h_2. Note that the nonzero elements of h_i are those elements of y_i associated with

$$\rho_1(\mu_j^{(i)}) < \tau < 1.$$

If $|g_j^{(i)}| \le \gamma$ for some moderate-size constant γ, then it follows from (12) that each individual element of h_i is either zero or tiny. Moreover, since $\rho_1(\mu_j^{(i)})$ decreases rapidly as $\mu_j^{(i)}$ increases, we can establish a bound for $h_i^T h_i$ in terms of τ under some mild conditions.

To simplify our notation, we will drop the superscript of $\mu_j^{(i)}$ in the following. Suppose the eigenvalues μ_j of (K_{ii}, M_{ii}) are distinct and that

$$\min_{j \ge k_i}(\mu_{j+1} - \mu_j) \ge \delta$$

for some constant $\delta > 0$. Then it is easy to show that

$$\begin{aligned} h_i^T h_i &= \sum_{j=k_i+1}^{n_i} (e_j^T h_i)(e_j^T h_i) = \sum_{j=k_i+1}^{n_i} \rho_1^2(\mu_j)(e_j^T G_{i3} y_3)^2 \\ &\le \left[\sum_{j=k_i+1}^{n_i} \rho_1^2(\mu_j) \right] \gamma^2 \le \gamma^2 \left[\rho_1^2(\mu_{k_i+1}) + \frac{1}{\delta} \int_{\mu_{k_i+1}}^{\mu_{n_i}} \rho_1^2(\omega) d\omega \right] \\ &= \gamma^2 \left[\frac{\lambda_1}{\mu_{k_i+1} - \lambda_1} \rho_1(\mu_{k_i+1}) + \frac{\lambda_1}{\delta} \left(\frac{\lambda_1}{\mu_{k_i+1} - \lambda_1} - \frac{\lambda_1}{\mu_{n_i} - \lambda_1} \right) \right] \\ &\le \gamma^2 \lambda_1 \left[\frac{1}{\Delta_i} + \frac{1}{\delta} \right] \tau, \end{aligned} \tag{23}$$

where $\Delta_i = \mu_{k_i+1} - \lambda_1$. By combining (23) with inequalities (21)–(22) and setting $\Delta = \min\{\Delta_1, \Delta_2, \delta\}$. we obtain

$$\frac{\theta_1 - \lambda_1}{\lambda_1} \le (\lambda_n - \lambda_1)(\alpha\tau), \tag{24}$$

$$\sin \angle_{\widehat{M}}(\hat{x}_1, u_1) \le \sqrt{\lambda_1 \left(\frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1} \right)} \sqrt{\alpha\tau}, \tag{25}$$

where $\alpha = 4\gamma^2/\Delta$.

We should mention that the presence of multiple (or tightly clustered) eigenvalues of (K_{ii}, M_{ii}) does not alter the qualitative measure of the bounds established in (24) and (25). In that case, we can simply replace the definition of δ with the minimum distances between two adjacent eigenvalue clusters and multiply the bounds by the largest multiplicity of the eigenvalues of (K_{ii}, M_{ii}).

Our assumption that $|g_j^{(i)}|$ is bounded by a moderate constant γ can be justified by noting that

$$(y_1^T\ y_2^T\ y_3^T) \begin{pmatrix} I & & G_{13} \\ & I & G_{23} \\ G_{13}^T & G_{23}^T & \widehat{M}_{33} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = 1 \tag{26}$$

and by the fact that M is positive definite. Since

$$\begin{pmatrix} I & & G_{13} \\ & I & G_{23} \\ G_{13}^T & G_{23}^T & \widehat{M}_{33} \end{pmatrix} = \begin{pmatrix} I & & \\ & I & \\ G_{13}^T & & I \end{pmatrix} \begin{pmatrix} I & & \\ & I & G_{23} \\ & G_{23}^T & \widehat{M}_{33} - G_{13}^T G_{13} \end{pmatrix} \begin{pmatrix} I & & G_{13} \\ & I & \\ & & I \end{pmatrix},$$

it follows from (26) that

$$\|y_1 + G_{13} y_3\| \leq 1.$$

Hence, the jth component of $y_1 + G_{13} y_3$ must satisfy

$$|e_j^T y_1 + g_j^{(1)}| \leq 1. \tag{27}$$

It then follows from (27) and (12) that

$$|g_j^{(1)}| \leq 1 + |e_j^T y_1| = 1 + \rho_1(\mu_j^{(1)}) |g_j^{(1)}|.$$

Consequently,

$$|g_j^{(1)}| \leq \frac{1}{1 - \rho(\mu_j^{(1)})} \quad \text{when} \quad \rho(\mu_j^{(1)}) < 1.$$

Thus, if $\rho_1(\mu_j^{(1)}) \ll 1$, then $|g_j^{(1)}|$ is clearly bounded by a small constant. A similar argument can be used to show that $|g_j^{(2)}|$ is bounded by a small constant also.

We should also emphasize that (24) and (25) merely provide a qualitative estimate of the error in the Ritz pair (θ_1, u_1) in terms of the threshold τ that may be used as a heuristic in practice to determine which spectral components of a substructure should be included in the subspace $\mathcal{S}$ defined in (15). This result was verified independently in the recent work by Elssel and Voss [14]. They derived a bound on the relative accuracy of the approximate eigenvalue in a more general setting by making using of the minmax principle for a rational eigenvalue problem.

4. Numerical experiments. We present two numerical examples in this section to illustrate the effectiveness of the single-level algebraic substructuring algorithm presented in section 2. These examples also confirm the analysis carried out in section 3. All experiments are performed in MATLAB. The desired eigenpairs of all pencils are computed by using the MATLAB `eigs` function. For illustration, we computed more eigenvalues and eigenvectors of each subproblem than we actually need in the following experiments. In practice, one would only need to compute a selected number of eigenpairs of (K_{ii}, M_{ii}) incrementally.

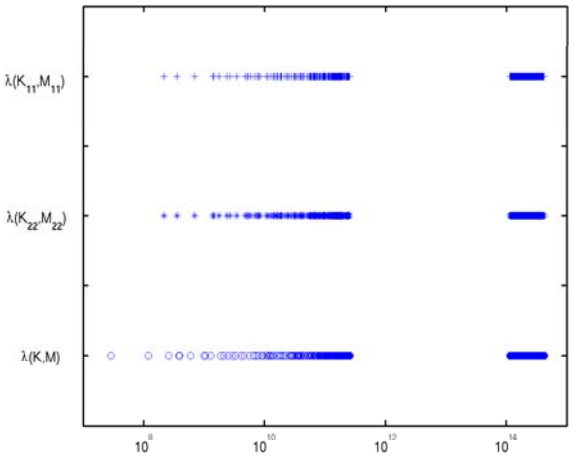


FIG. 1. *The spectra of the pencils* (K_{11}, M_{11}), (K_{22}, M_{22}), *and* (K, M) *associated with the structural dynamics example.*

4.1. Example 1. Structural dynamics. The matrices used in this example, BCSSTK09 and BCSSTM09, are part of the Harwell–Boeing sparse matrix collection [13]. These matrices originated from a dynamic analysis of a clamped plate. The dimensions of these matrices are $n = 1083$. We used METIS [19] to dissect the matrix into two main substructures coupled by a small separator (interface block). The two substructures of the reordered K are identical. The dimension of each substructure is $n_1 = n_2 = 513$. The separator contains only 57 rows and columns. The mass matrix M is diagonal in this example. Applying the same reordering to M does not change its structure.

The spectra of the original matrix pencil (K, M) and the substructure pencils (K_{ii}, M_{ii}) $(i = 1, 2)$ are shown in Figure 1. There is a large gap between the 361st and the 362nd eigenvalues of (K, M). Similar gaps are present in the spectra of (K_{ii}, M_{ii}). In this example, the eigenvalues of interest are the ones on the left side of the spectrum. Naturally, we would select the eigenvectors associated with the smallest eigenvalues of (K_{ii}, M_{ii}) to construct the subspace (6) required in step 5 of the single-level algebraic substructuring algorithm.

To determine how many eigenvectors of (K_{ii}, M_{ii}) we should include in the subspace represented by (6), we examine the ρ-factor defined in (13). It follows from the discussion in section 3 that one may develop a selection scheme by setting a threshold value τ for ρ_1; i.e., one can choose substructure modes that satisfy

$$\rho_1(\mu_j^{(i)}) > \tau$$

for some small τ. However, since the computation of ρ_1 requires the knowledge of the exact λ_1, which we do not have in advance, a more practical scheme is perhaps to compute an approximate ρ-factor by replacing λ_1 in (13) with an approximate eigenvalue σ.

We use $\sigma = \min(\mu_1^{(1)}, \mu_1^{(2)})/2$ in all of our experiments and define

$$\widehat{\rho}_1(\omega) = \left| \frac{\sigma}{\omega - \sigma} \right|. \tag{28}$$

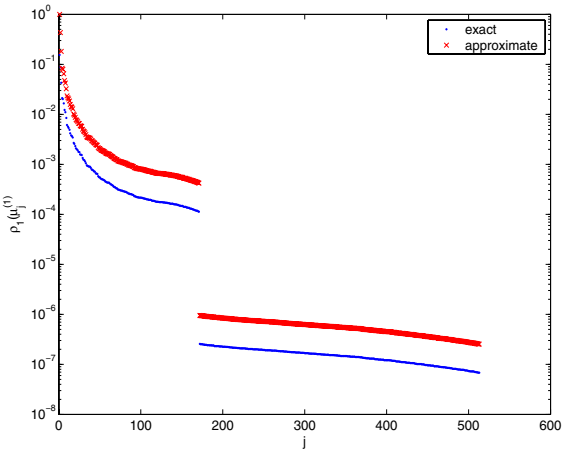


FIG. 2. *The exact (marked by* x*'s) and the approximate (marked by dots) ρ-factors associated with the first substructure of the structural dynamics problem. The exact ρ-factor is defined by* (13)*, and the approximate ρ-factor is defined by* (28)*.*

In Figure 2, we plot both $\rho_1(\mu_j^{(1)})$ and $\widehat{\rho}_1(\mu_j^{(1)})$. (The two substructures in this problem are identical.) The figure clearly shows that there is essentially no qualitative difference between $\rho_1(\mu_j^{(1)})$ and $\widehat{\rho}_1(\mu_j^{(1)})$. Both decrease rapidly as $\mu_j^{(1)}$ increases. There is a clear gap between $\rho_1(\mu_{171}^{(1)})$ and $\rho_1(\mu_{172}^{(1)})$. A similar gap is observed between $\widehat{\rho}_1(\mu_{171}^{(1)})$ and $\widehat{\rho}_1(\mu_{172}^{(1)})$. These gaps reflect the gaps observed in the spectrum of (K_{11}, M_{11}).

Several choices of τ values (listed in Table 1) have been made. The analysis performed in section 3 indicates that, the smaller the value of τ, the more accurate the smallest Ritz pair should be. This prediction is confirmed in Figure 3, where we plot the relative errors of the smallest 50 Ritz values extracted from three subspaces constructed by using these different choices of τ values. Notice that with the choice of $\tau = 10^{-4}$, which corresponds to selecting the leading 171 eigenvectors from each substructure to form the matrix S_i required in (15), θ_1 exhibits roughly 10 digits of accuracy.

TABLE 1

The effect of τ on the number of selected modes associated with the structural dynamics problem, the relative accuracy of the smallest Ritz value, and the relative error bound defined by (21)*.*

τ	k_i	$(\theta_1 - \lambda_1)/\lambda_1$	Relative error bound
10^{-2}	18	1.4×10^{-4}	3.4×10^{0}
10^{-3}	84	2.0×10^{-6}	6.4×10^{-3}
10^{-4}	171	1.2×10^{-12}	4.2×10^{-12}

Even though our error estimate presented in section 3 is targeted only at (θ_1, u_1), Figure 3 shows that the improvement in the accuracy of other Ritz values is also proportional to the decrease of τ.

In this example, the least upper bound for the elements of $g^{(i)}$ used in (12) is roughly $\gamma = 0.28$. Hence, $\rho_1(\mu_j^{(i)})$ provides a reliable upper bound for the magnitude of $e_j^T y_i$ $(i = 1, 2)$, where $(y_1^T, y_2^T, y_3^T)^T$ is the eigenvector associated with the smallest

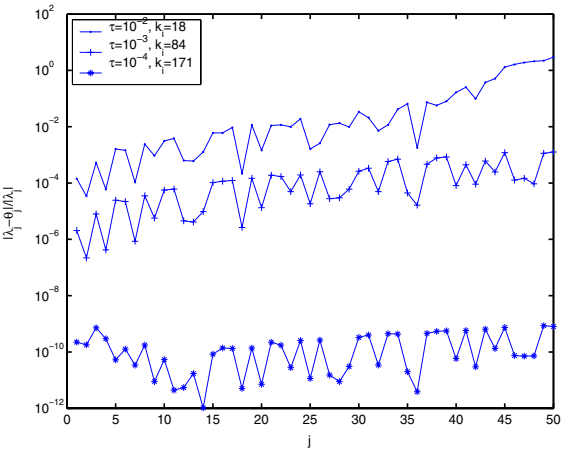


FIG. 3. *The relative error of the smallest* 50 *Ritz values extracted from three subspaces constructed by using different choices of the* ρ*-factor thresholds* (τ *values) for the structural dynamics problem.*

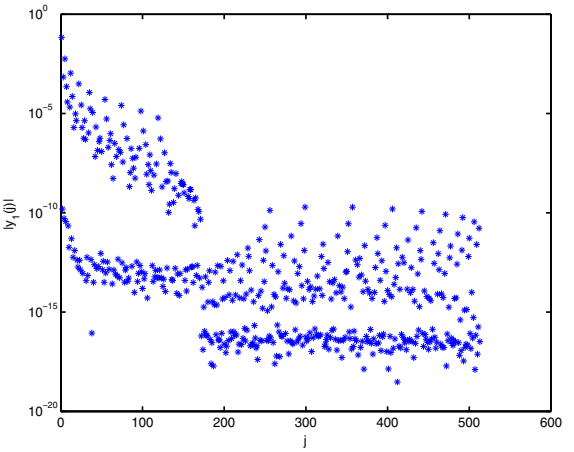


FIG. 4. *The magnitude of* $e_j^T y_1$*, where* $(y_1^T, y_2^T, y_3^T)^T$ *is the eigenvector corresponding to the smallest eigenvalue of the canonical problem* (10) *associated with the structural dynamics example.*

eigenvalue (λ_1) of the canonical eigenvalue problem (10).

Judging from the small magnitude of $\rho_1(\mu_j^{(i)})$ for $j > 171$, which is less than 10^{-6}, we predict the magnitude of $e_j^T y_i$, $i = 1, 2$, to be tiny for $j > 171$. This is indeed the case, as is demonstrated in Figure 4, where we plot $|e_j^T y_1|$ (The plot for y_2 is identical.) We observe that $|e_j^T y_1| < 2 \times 10^{-10}$ for all $j > k_1 = 171$. This observation, when used in conjunction with Theorem 3.3, provides a clear explanation for the high accuracy of θ_1 displayed in Figure 3.

Table 1 further illustrates the connections between the mode selection threshold τ, the number of modes selected from each substructure (k_i), the relative accuracy of θ_1, and the error estimates established in Theorem 3.3. Note that the relative error bound listed in the last column of Table 1, which is calculated directly from the right-hand side of (21), tends to be somewhat pessimistic. However, it does provide

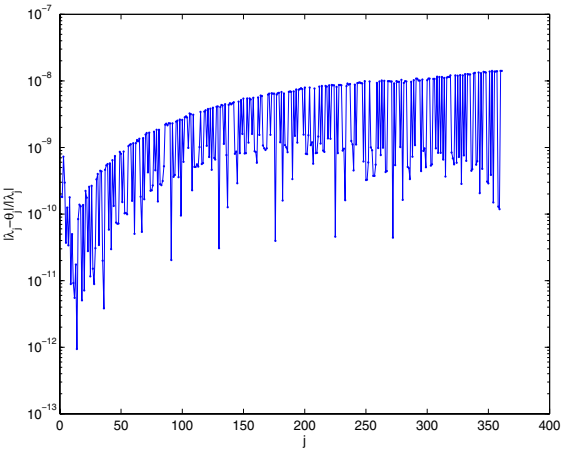


FIG. 5. *The relative error of the smallest* 361 *approximate eigenvalues associated with the structural dynamics problem.*

a qualitative estimate for the relative accuracy of θ_1.

It is interesting to see from Figure 4 that, among the first 171 elements of both y_1 and y_2, many have magnitudes less than 10^{-10}. This observation suggests that one may potentially reduce the dimension of the subspace (6) by excluding eigenvectors of (K_{ii}, M_{ii}) that are associated with these small entries from S_i. We will pursue this idea further in a follow-up paper on mode selection strategies.

We will end this example by pointing out that the large gap between the leading 361 eigenvalues of (K, M) and the rest of the spectrum is a highly favorable feature of this problem. This gap, which also manifests itself in the ρ-factor plots displayed in Figure 2, allows an algebraic substructuring algorithm to easily construct a subspace that contains accurate approximations to the leading 361 eigenvalues of (K, M). Figure 5 shows that, by setting $k_i = 171$, the leading 361 Ritz values extracted from the subspace $\mathcal{S}$ spanned by columns of (15) all have at least 7 digits of accuracy.

4.2. Example 2. Short traveling wave accelerating structure. We show in this example that algebraic substructuring can be used to compute approximate cavity resonance frequencies and the electromagnetic field associated with a small accelerator structure. The matrix pencil used in this example is obtained from a finite element model of a five-cell traveling wave accelerating structure. The three-dimensional geometry of the model is shown in Figure 6. The model contains three cavity cells and two couplers. The dimension of the pencil (K, M) is $n = 1898$. The stiffness matrix K has 336 zero rows and columns. These zero rows and columns are produced by a particular hierarchical vector finite element discretization scheme. Because the null space of K has a special structure, it can be effectively deflated in the algebraic substructuring algorithm. The details of the deflation scheme can be found in [30, 3].

In order to deflate the null space of (K, M) associated with these zero rows and columns, which has no physical significance, we perform the following two-stage matrix reordering:

- A single-level dissection is applied to $|K| + |M|$ first using the METIS [19] software. The dissection produces two substructures of sizes $n_1 = 995$ and $n_2 = 887$, respectively. These substructures are connected by a small separa-

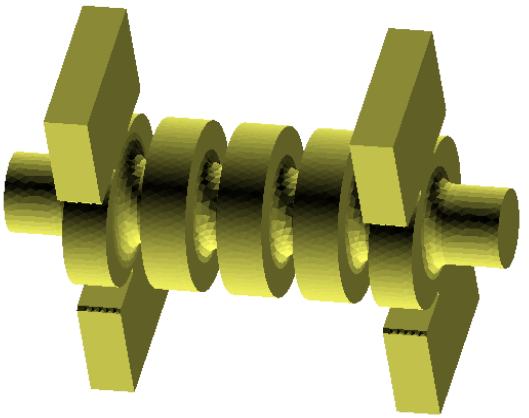


FIG. 6. *The finite element model corresponding to a five-cell traveling wave accelerating structure.*

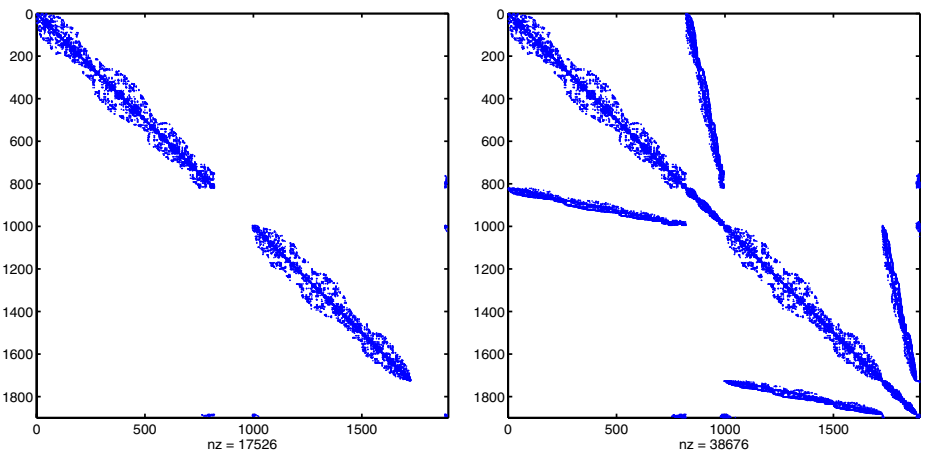


FIG. 7. *The nonzero pattern of the permuted stiffness matrix K (left) and the mass matrix M (right) associated with the traveling wave accelerating structure.*

tor (an interface block) which contains only 16 rows and columns. The K_{11} block of the permuted K contains 175 zero rows and columns, the K_{22} block contains 157 zero rows and columns, and the K_{33} block contains 6 zero rows and columns.

- The nonzero rows and columns of K_{11}, K_{22}, and K_{33} are permuted to the leading blocks of these submatrices. The matrix M is permuted accordingly.

The nonzero patterns of the permuted K and M are displayed in Figure 7. The distribution of the nonzero eigenvalues of (K, M) is shown in Figure 8. We are interested in the smallest nonzero eigenvalues, which appear to be relatively well separated from the large end of the spectrum. In addition to the spectrum of (K, M), we also plot the spectra of (K_{ii}, M_{ii}) $(i = 1, 2)$ in Figure 8. Notice that the spectra of both substructures show a distribution pattern similar to that of (K, M).

In Figure 9 we plot the $\widehat{\rho}$-factors associated with the smallest eigenvalue of the deflated problem. We observe that the $\widehat{\rho}$-factors associated with this example decrease

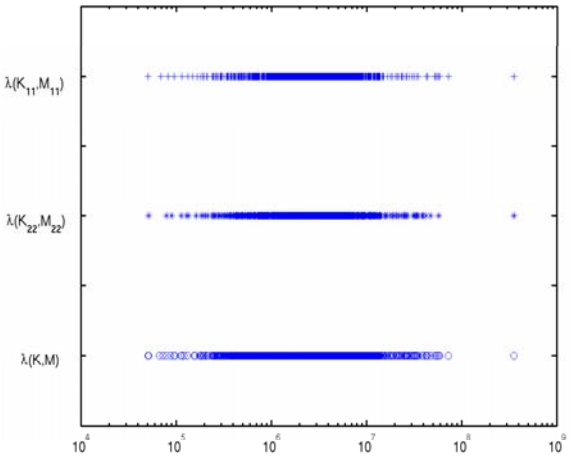


FIG. 8. *The spectra of the pencils* (K_{11}, M_{11}), (K_{22}, M_{22}), *and* (K, M) *associated with the traveling wave accelerating structure.*

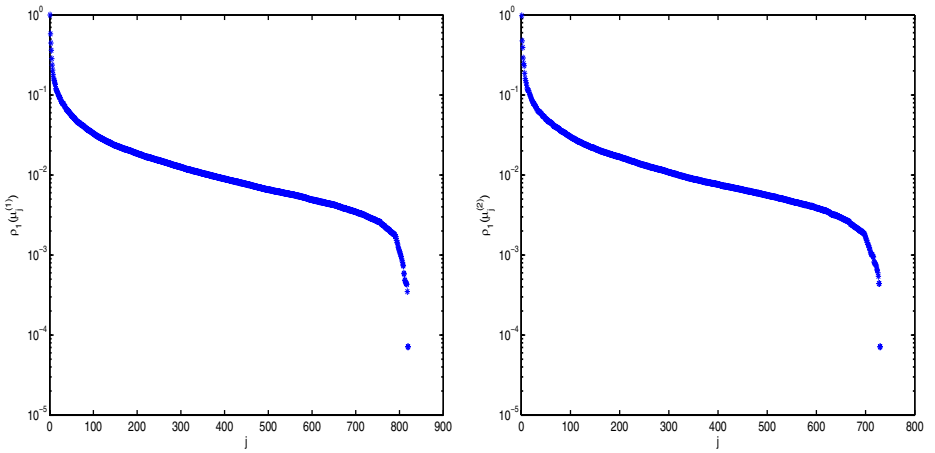


FIG. 9. *The approximate* ρ*-factors associated with each substructure of the traveling wave accelerating structure.*

at a somewhat slower rate. Three different choices of τ values were used as the thresholds ($\tau = 0.1, 0.05, 0.01$) for selecting substructure modes. The relative accuracy of the 50 smallest nonzero Ritz values extracted from the subspaces constructed with these choices of τ values is displayed in Figure 10.

We observe that with $\tau = 0.1$, θ_1 has roughly four digits of accuracy, which is quite sufficient for this particular discretized model. If we decrease τ down to 0.01, most of the smallest 50 nonzero Ritz values have at least four digits of accuracy.

The least upper bound for $|g_j^{(i)}|$ used in (12) is $\gamma = 0.02$. Thus the ρ-factor gives an overestimate of $|e_j^T y_i|$ in this case. In Figure 11, we plot $|e_j^T y_1|$ and $|e_j^T y_2|$, where $(y_1^T, y_2^T, y_3^T)^T$ is the eigenvector associated with the smallest nonzero eigenvalue of (10). For simplicity, we excluded the values of $|e_j^T y_1|$ and $|e_j^T y_2|$ corresponding to the null space of (K_{11}, M_{11}) and (K_{22}, M_{22}), which have been deflated from our calculations (see section 4). We observe that $|e_j^T y_i|$ is much smaller compared to

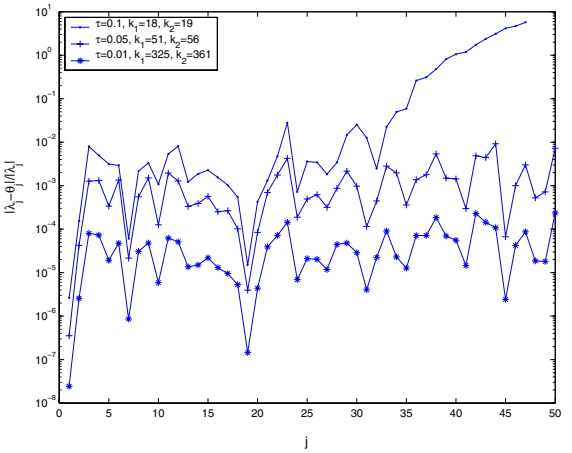


FIG. 10. *The relative error of the smallest* 50 *Ritz values extracted from three subspaces constructed by using different choices of the* ρ*-factor thresholds* (τ *values) for the traveling wave accelerating problem.*

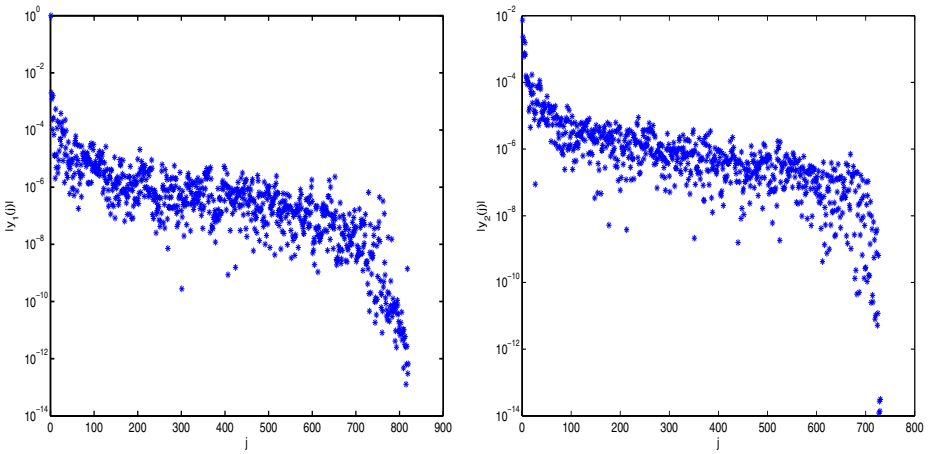


FIG. 11. *The magnitude of* $e_j^T y_1$ *(left) and* $e_j^T y_2$ *(right), where* $(y_1^T, y_2^T, y_3^T)^T$ *is the eigenvector corresponding to the smallest eigenvalue of the canonical problem* (10) *associated with the traveling wave accelerating structure.*

TABLE 2

The effect of τ *on the number of selected modes associated with the traveling wave accelerating structure, the relative accuracy of the smallest Ritz value, and the relative error bound defined by* (21).

τ	k_1	k_2	$(\theta_1 - \lambda_1)/\lambda_1$	Relative error bound
0.1	18	19	1.4×10^{-4}	1.7×10^{-3}
0.05	51	56	1.2×10^{-5}	2.6×10^{-4}
0.01	325	361	2.4×10^{-8}	2.5×10^{-6}

$\widehat{\rho}_1(\mu_j^{(i)})$, and it decays much faster than the $\widehat{\rho}$-factor also.

We conclude this example by listing in Table 2 the mode selection threshold τ,

the number of modes selected from each substructure (k_i), the relative accuracy of θ_1, and the error estimate computed directly from the right-hand side of (21).

5. Concluding remarks. A purely algebraic analysis of a single-level substructuring algorithm for large-scale eigenvalue calculation is developed in this paper. By applying a sequence of special congruence transformations to (K, M), we transform the original generalized eigenvalue (2) into a canonical problem (10) with a simpler structure. We observed that the desired eigenvector y of the canonical problem (10) often contains only a few large entries. The magnitude of these entries ultimately determines which eigenvectors (modes) of each substructure should be included in the subspace (6), from which approximations to the eigenpairs of (K, M) are extracted. All other substructure modes can essentially be truncated from (9) without sacrificing the required level of accuracy in our approximation. We provided an explicit a priori error estimate for the smallest Ritz pair in terms of the small components of y that are truncated from (9). We also suggested a practical way to estimate the magnitude of each component of y by exploiting its relationship with the "ρ-factor" defined in (13). This estimation leads to a practical way to select substructure modes by specifying a threshold value τ for the ρ-factor. We showed that the accuracy of the smallest Ritz pair is proportional to the size of τ under some mild conditions. A number of numerical examples are provided to confirm our theoretical analysis. Moreover, we demonstrated that an algebraic substructuring algorithm can be an effective tool for computing cavity resonance frequencies and the electromagnetic field generated by a linear accelerator structure.

Our analysis of a simple algebraic substructuring algorithm can be extended to a multilevel setting. Our error estimate can be made for nonextreme Ritz pairs as well. These topics will be pursued in our future research. Another interesting area that would require further research is the development of a better strategy for selecting substructuring modes.

Our presentation has focused on the theoretical aspects of the algebraic substructuring algorithm. Implementation details and comparsion of a multilevel algebraic substructuring algorithm with other methods for large-scale eigenvalue computation will be reported elsewhere.

Acknowledgments. We would like to thank the anonymous referees for their careful reading and helpful comments.

REFERENCES

[1] A. Abramov, *On the separation of the principal part of certain algebraic problems*, Zh. Vychisl. Mat. Mat. i Fiz., 2 (1962), pp. 141–145 (in Russian); USSR Comput. Math. Math. Phys., 1963, pp. 147–151 (in English).

[2] A. Abramov, *Remarks on finding the eigenvalues and eigenvectors of matrices which arise in the application of Ritz's method or in the difference method*, Zh. Vychisl. Mat. i Mat. Fiz., 7 (1962), pp. 644–647 (in Russian); USSR Comput. Math. Math. Phys., 7 (1967), pp. 234–240 (in English).

[3] P. Arbenz and Z. Drmač, *On positive semidefinite matrices with known null space*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 132–149.

[4] C. Bekas and Y. Saad, *Computation of Smallest Eigenvalues Using Spectral Schur Complements*, Technical report UMSI-2004-6, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2004.

[5] J. K. Bennighof, *Adaptive multi-level substructuring method for acoustic radiation and scattering from complex structures*, in Computational Methods for Fluid/Structure Interaction, A. J. Kalinowski, ed., AMSE, New York, 1993, pp. 25–38.

[6] J. K. Bennighof and C. K. Kim, *An adaptive multi-level substructuring method for efficient modeling of complex structures*, in Proceedings of the 33rd AIAA SDM Conference, Dallas, TX, 1992, pp. 1631–1639.

[7] J. K. Bennighof and R. B. Lehoucq, *An automated multilevel substructuring method for eigenspace computation in linear elastodynamics*, SIAM J. Sci. Comput., 25 (2004), pp. 2084–2106.

[8] F. Bourquin, *Analysis and comparison of several component mode synthesis methods on one-dimensional domains*, Numer. Math., 58 (1990), pp. 11–34.

[9] F. Bourquin, *Component mode synthesis and eigenvalues of second order operators: Discretization and algorithm*, Math. Model. Numer. Anal., 26 (1992), pp. 385–423.

[10] V. S. Chichov, *A method for partitioning a high order matrix into blocks in order its eigenvalues*, Zh. Vychisl. Mat. i Mat. Fiz., 1 (1961), pp. 169–173 (in Russian); USSR Comput. Math. Math. Phys., 1962, pp. 186–190 (in English).

[11] R. R. Craig and M. C. C. Bampton, *Coupling of substructures for dynamic analysis*, AIAA J., 6 (1968), pp. 1313–1319.

[12] R. R. Craig and C-J. Chang, *A review of substructure coupling methods for dynamic analysis*, in Advances in Engineering Science, Vol. 2, National Aeronautics and Space Administration, Washington, DC, 1976, pp. 393–408.

[13] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Users' Guide for the Harwell–Boeing Sparse Matrix Collection (Release* 1*)*, Technical report RAL-92-086, Rutherford Appleton Laboratory, Oxfordshire, UK, 1992.

[14] K. Elssel and H. Voss, *An A Priori Bound for Automated Multi-level Substructuring*, Technical report 81, Arbeitsbereich Mathematik, TU Hamburg-Harburg, Germany, 2004.

[15] A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[16] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[17] W. C. Hurty, *Vibrations of structure systems by component-mode synthesis*, J. Engrg. Mech., 86 (1960), pp. 51–69.

[18] M. F. Kaplan, *Implementation of Automated Multilevel Substructuring for Frequency Response Analysis of Structures*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 2001.

[19] G. Karypis and V. Kumar, MeTiS—*A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices—Version* 4.0, University of Minnesota, Duluth, MN, 1998.

[20] K. Ko, N. Folwell, L. Ge, A. Guetz, V. Ivanov, L. Lee, Z. Li, I. Malik, W. Mi, C. Ng, and M. Wolf, *Electromagnetic Systems Simulation—from Simulation to Fabrication*, SciDAC report, Stanford Linear Accelerator Center, Menlo Park, CA, 2003.

[21] A. Kropp and D. Heiserer, *Efficient broadband vibro-acoustic analysis of passenger car bodies using an FE-based component mode synthesis approach*, J. Comput. Acoust., 11 (2003), pp. 139–157.

[22] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.

[23] R. H. MacNeal, *Vibrations of Composite Systems*, Technical report OSRTN-55-120, Air Force Office of Scientific Research, Air Research and Development Command, Arlington, VA, 1954.

[24] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, New York, 1980.

[25] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford University Press, Oxford, UK, 1999.

[26] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halsted Press, New York, 1992.

[27] G. L. G. Sleijpen, J. van den Eshof, and P. Smit, *Optimal a priori error bounds for the Rayleigh-Ritz method*, Math. Comp., 72 (2003), pp. 677–684.

[28] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition*, Cambridge University Press, Cambridge, UK, 1996.

[29] G. W. Stewart, *Matrix Algorithms, Volume* II: *Eigensystems*, SIAM, Philadelphia, 2001.

[30] C. Yang, W. Gao, Z. Bai, X. Li, L. Lee, P. Husbands, and E. Ng, *An Algebraic Substructuring Algorithm for Large-Scale Eigenvalue Calculation*, Technical report LBNL-55055, Lawrence Berkeley National Lab, Berkeley, CA, 2004.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 893–913
© 2005 Society for Industrial and Applied Mathematics

GAUSSIAN-TYPE QUADRATURE RULES FOR MÜNTZ SYSTEMS*

GRADIMIR V. MILOVANOVIĆ† AND ALEKSANDAR S. CVETKOVIĆ

Abstract. A method for constructing the generalized Gaussian quadrature rules for Müntz polynomials on $(0,1)$ is given. Such quadratures possess several properties of the classical Gaussian formulae (for polynomial systems), such as positivity of the weights, rapid convergence, etc. They can be applied to the wide class of functions, including smooth functions, as well as functions with end-point singularities, such as those in boundary-contact value problems, integral equations with singular kernels, complex analysis, potential theory, etc. The constructive method is based on an application of orthogonal Müntz polynomials introduced by Badalyan [*Akad. Nauk Armyan. SSR. Izv. Fiz.-Mat. Estest. Tehn. Nauk*, 8 (1955), pp. 1–28; 9 (1956), pp. 3–22 (Russian; Armenian summary)] and studied intensively by Borwein, Erdélyi, and Zhang [*Trans. Amer. Math. Soc.*, 342 (1994), pp. 523–542], as well as by Milovanović on a numerical procedure for evaluation of such polynomials with high precision [*Müntz orthogonal polynomials and their numerical evaluation*, in Applications and Computation of Orthogonal Polynomials, Internat. Ser. Numer. Math. 131, W. Gautschi, G. H. Golub, and G. Opfer, eds., Birkhäuser, Basel, 1999, pp. 179–194]. The method is numerically stable. Some numerical examples are included.

Key words. numerical integration, generalized Gaussian quadrature, Müntz polynomials, zeros, nodes, weights

AMS subject classifications. 41A55, 65D30, 65D32

DOI. 10.1137/040621533

1. Introduction.

Gauss's famous method of approximate integration from 1814 can be extended in a natural way to nonpolynomial functions, taking a system of linearly independent functions

$$\{P_0(x), P_1(x), P_2(x), \ldots\} \qquad (x \in [a,b]), \tag{1.1}$$

usually chosen to be complete in some suitable space of functions (cf. Gautschi [6], Karlin and Studden [12], Harris and Evans [11], Ma, Rokhlin, and Wandzura [13]). If $d\sigma(x)$ is a given nonnegative measure on $[a,b]$ and the quadrature rule

$$\int_a^b f(x)d\sigma(x) = \sum_{k=1}^{n} A_k f(x_k) + R_n(f) \tag{1.2}$$

is such that it integrates exactly the first $2n$ functions in (1.1), we call rule (1.2) [illegible] (1.1). The existence and uniqueness of a Gaussian quadrature rule (1.2) with respect to the system (1.1), or, for brevity, a [illegible], is always guaranteed if the first $2n$ functions of this system constitute a Chebyshev system on $[a,b]$. Then, all the weights $A_1, \ldots, A_n$ in (1.2) are positive. In terms of moment spaces, the Gaussian rule corresponds to the unique [illegible] of the measure $d\sigma(x)$ (see Karlin and Studden [12]).

*Received by the editors December 26, 2004; accepted for publication (in revised form) May 10, 2005; published electronically November 22, 2005. This work was supported in part by the Swiss National Science Foundation (SCOPES Joint Research Project New Methods for Quadrature) and the Serbian Ministry of Science and Environmental Protection (Project Applied Orthogonal Systems, Constructive Approximation and Numerical Methods, grant 2002).

http://www.siam.org/journals/sisc/27-3/62153.html

†Department of Mathematics, Faculty of Electronic Engineering, University of Niš, P. O. Box 73, 18000 Niš, Serbia (grade@ni.ac.yu, aca@elfak.ni.ac.yu).

The generalized Gaussian quadratures for Müntz systems goes back to Stieltjes's paper [22] of 1884. Taking $P_k(x) = x^{\lambda_k}$ on $[a,b] = [0,1]$, where $0 \le \lambda_0 < \lambda_1 < \cdots$, he showed the existence of Gaussian formulae. In his short note he considered also Gauss–Radau formulae.

A numerical algorithm for constructing the generalized Gaussian quadratures was recently investigated by Ma, Rokhlin, and Wandzura [13]. They take a Chebyshev system of functions $\{P_0, P_1, \dots, P_{2n-1}\}$ on $[a,b]$ with the following properties:

(1) $P_k \in C^1[a,b]$ $(k = 0,1,\dots,2n-1)$;

(2) the determinants

$$\begin{vmatrix} P_0(x_1) & \cdots & P_0(x_n) & P_0'(x_1) & \cdots & P_0'(x_n) \\ P_1(x_1) & & P_1(x_n) & P_1'(x_1) & & P_1'(x_n) \\ \vdots & & & & & \\ P_{2n-1}(x_1) & & P_{2n-1}(x_n) & P_{2n-1}'(x_1) & & P_{2n-1}'(x_n) \end{vmatrix}$$

are nonzero for any set of n points $x_1, \dots, x_n \in [a,b]$ ($x_i \ne x_j$ for $i \ne j$).

Such a system will be referred to as an [illegible].

The procedure given in [13] requires the construction of the functions

$$\xi_i(x) = \sum_{j=1}^{2n} \alpha_{ij} P_{j-1}(x), \qquad \eta_i(x) = \sum_{j=1}^{2n} \beta_{ij} P_{j-1}(x) \qquad (i = 1,\dots,n) \tag{1.3}$$

such that

$$\begin{cases} \xi_i(x_k) = 0, \\ \xi_i'(x_k) = \delta_{ik}, \end{cases} \qquad \begin{cases} \eta_i(x_k) = \delta_{ik}, \\ \eta_i'(x_k) = 0, \end{cases} \tag{1.4}$$

for all $i = 1,\dots,n$ and all $k = 1,\dots,n$. The algorithm is ill conditioned (see [13, Remark 6.2]). In order to obtain the double precision results (REAL*8), the authors performed the computations in extended precision (Q-arithmetic, i.e., REAL*16) for generating Gaussian quadratures up to order 20, and in MATHEMATICA (120-digit operations) for generating Gaussian quadratures of higher orders ($n \le 40$). In particular, they considered the following important cases of EH systems:

$$\left\{1, \log x, x, x\log x, \dots, x^{n-1}, x^{n-1}\log x\right\} \tag{1.5}$$

and

$$\left\{1, x^s, x, x^{s+1}, \dots, x^{n-1}, x^{s+n-1}\right\} \tag{1.6}$$

for $s = 1/3$, $s = -1/3$, $s = -2/3$. The case (1.5) was also considered by Andronov [1] in order to solve certain boundary-contact value problems.

For a given sequence of real numbers $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \dots\}$ given in nondecreasing order, i.e., $\lambda_k \le \lambda_{k+1}$, $k \in \mathbb{N}_0$, in this paper we consider Müntz polynomials as linear combinations of the Müntz system

$$\left\{x^{\lambda_0}, x^{\lambda_1}, \dots, x^{\lambda_n}\right\}.$$

By $M_n(\Lambda)$ we denote the set of all such polynomials, i.e.,

$$M_n(\Lambda) = \operatorname{span}\left\{x^{\lambda_0}, x^{\lambda_1}, \dots, x^{\lambda_n}\right\},$$

where the linear span is over the real numbers.

Here, we present an alternatively numerical method for constructing the generalized Gaussian quadratures for Müntz polynomials

$$\int_0^1 f(x)d\sigma(x) = \sum_{k=1}^{n} A_k f(x_k) + R_n(f), \tag{1.7}$$

which are exact for each $f \in M_{2n-1}(\Lambda)$. Our method is rather stable and simpler than the previous one. It performs calculations in double precision arithmetic to get double precision results.

As is well known (see [14], [28]), the Gaussian quadrature rule is unique, provided the measure σ has a nonnegative absolutely continuous part and has finitely many atoms on $[0,1]$. Such quadratures possess several properties of the classical Gaussian formulae (for polynomial systems), such as positivity of the weights, rapid convergence, etc. They can be applied to the wide class of functions, including smooth functions, as well as functions with end-point singularities, such as those in boundary-contact value problems (see [1]), integral equations, complex analysis, potential theory, and several other fields (see [13]).

This paper is organized as follows. Section 2 is devoted to some properties of orthogonal Müntz polynomials on $(0,1)$ and their connection with orthogonal rational functions, including numerical evaluation of such generalized polynomials. A numerical method for constructing the quadrature rules (1.7) is presented in section 3. Finally, in section 4 we give numerical results.

2. Orthogonal Müntz–Legendre polynomials. We are interested in the class of Müntz polynomials introduced by Badalyan [2], and we refer to it as Müntz–Legendre polynomials.

Let a complex sequence $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$ be such that $\operatorname{Re}(\lambda_k) > -1/2$ for every $k \in \mathbb{N}_0$ and let $\Lambda_n = \{\lambda_0, \lambda_1, \ldots, \lambda_n\}$. If Γ is a simple contour surrounding all the zeros of the denominator in the rational function

$$W_n(s) = \prod_{k=0}^{n-1} \frac{s + \bar{\lambda}_k + 1}{s - \lambda_k} \cdot \frac{1}{s - \lambda_n} \qquad (n \in \mathbb{N}_0), \tag{2.1}$$

then the Müntz–Legendre polynomials are defined by (see [2], [3], [4], [15], [25])

$$P_n(x) = P_n(x; \Lambda_n) = \frac{1}{2\pi i} \oint_\Gamma W_n(s) x^s \, ds. \tag{2.2}$$

In the case $n = 0$, an empty product in (2.1) should be taken to be equal to 1.

If $\lambda_k \neq \lambda_j$ $(k \neq j)$, by the Cauchy residue theorem, these generalized polynomials can be expressed in a power form as

$$P_n(x) = \sum_{k=0}^{n} C_{nk} x^{\lambda_k}. \tag{2.3}$$

If there exists a pair with the property $\lambda_k = \lambda_j$, $k \neq j$, an application of Cauchy residue theorem produces terms with log function included. For example, we consider the important special case, where

$$\lambda_{2k} = \lambda_{2k+1} = k \qquad (k = 0, 1, \ldots).$$

Namely, we take $\lambda_{2k} = k$ and $\lambda_{2k+1} = k + \varepsilon$ $(k = 0, 1, \ldots)$, where ε decreases to zero. The corresponding limit process leads to orthogonal Müntz polynomials with logarithmic terms. Then, (2.1) becomes

$$W_n(s) = \begin{cases} \displaystyle\prod_{\nu=0}^{m-1} \Big(\frac{s+\nu+1}{s-\nu}\Big)^2 \frac{1}{s-m} & \text{when } n = 2m, \\ \displaystyle\prod_{\nu=0}^{m} \Big(\frac{s+\nu+1}{s-\nu}\Big)^2 \frac{1}{s+m+1} & \text{when } n = 2m+1. \end{cases}$$

Applying the Cauchy residue theorem to the integral in (2.2), with this rational function, we obtain the following representation for the corresponding Müntz polynomials:

$$P_n(x) = R_n(x) + S_n(x) \log x \qquad (n = 0, 1, \ldots), \tag{2.4}$$

where $R_n(x)$ and $S_n(x)$ are algebraic polynomials of degree $[n/2]$ and $[(n-1)/2]$, respectively, i.e.,

$$R_n(x) = \sum_{\nu=0}^{[n/2]} a_\nu^{(n)} x^\nu, \qquad S_n(x) = \sum_{\nu=0}^{[(n-1)/2]} b_\nu^{(n)} x^\nu.$$

Notice that $P_n(1) = R_n(1) = 1$. A general expression for the polynomial P_n, $n \in \mathbb{N}_0$, was given in [17]. These Müntz polynomials can be used in the proof of the irrationality of $\zeta(3)$ and of other familiar numbers (see [4, pp. 372–381] and [27]).

In the special case when $\lambda_0 = \lambda_1 = \cdots = \lambda$, (2.2) gives

$$P_n(x; \Lambda_n) = x^\lambda L_n(-(\lambda + \bar{\lambda} + 1) \log x), \tag{2.5}$$

where $L_n(x)$ is the Laguerre polynomial orthogonal with respect to e^{-x} on $[0, \infty)$ and such that $L_n(0) = 1$ (cf. [5, p. 145]).

THEOREM 2.1. [illegible] (2.2) [illegible]

$$(P_n, P_m) = \int_0^1 P_n(x)\overline{P_m(x)}\, dx = \frac{\delta_{nm}}{\lambda_n + \bar{\lambda}_n + 1}. \tag{2.6}$$

According to this theorem we can construct orthonormal sequence of Müntz–Legendre polynomials, denoted by $p_n(x) := p_n(x; \Lambda) = (\lambda_n + \bar{\lambda}_n + 1)^{1/2} P_n(x)$, $n \in \mathbb{N}_0$.

For a general nondecreasing sequence Λ, the case when some of λ_k, $k \in \mathbb{N}_0$, are the same, it is complicated to express the basis functions in a closed form. For example, if all λ_k, $k \in \mathbb{N}_0$, are different, we have that the system of basis functions is given by $\{x^{\lambda_0}, x^{\lambda_1}, \ldots\}$. In the case $\lambda_k = \lambda_{k+1} = k$, $k \in \mathbb{N}_0$, we have that the system of basis functions is given by $\{1, \log x, x, x \log x, \ldots\}$. To make notation simple, we use the notation x^Λ to denote the set of basis functions for the general nondecreasing sequence Λ. Also, we introduce the notation x^{Λ_n} for the first n functions in the system x^Λ.

Putting $\lambda_k + \beta/2$ instead of λ_k, $k = 0, 1, \ldots$, in the sequence Λ, we can define a kind of Müntz–Jacobi polynomials $P_n^{(\beta)}(x)$ by

$$P_n^{(\beta)}(x) = \frac{x^{-\beta/2}}{2\pi i} \oint_\Gamma W_n^{(\beta)}(s) x^s\, ds, \tag{2.7}$$

where

$$W_n^{(\beta)}(s) = \prod_{k=0}^{n-1} \frac{s + \bar{\lambda}_k + \beta/2 + 1}{s - \lambda_k - \beta/2} \cdot \frac{1}{s - \lambda_n - \beta/2}.$$

Then the following result holds.

THEOREM 2.2. [illegible] $\beta \in \mathbb{R}$ [illegible] $\operatorname{Re}\lambda_k > -(\beta+1)/2$ [illegible] $k \in \mathbb{N}_0$ [illegible]

$$(P_n^{(\beta)}, P_m^{(\beta)}) = \int_0^1 P_n^{(\beta)}(x)\overline{P_m^{(\beta)}(x)}\, x^\beta\, dx = \frac{\delta_{nm}}{\lambda_n + \bar{\lambda}_n + \beta + 1}.$$

Let $P_n(\cdot;\Lambda)$, $n \in \mathbb{N}_0$, denote the Müntz–Legendre polynomials for a nondecreasing sequence $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$. Then it is very easy to verify that $P_n^{(\beta)}(x) = x^{-\beta/2}P_n(x;\Lambda+\beta/2)$, where $P_n(\cdot;\Lambda+\beta/2)$ are really Müntz–Legendre polynomials orthogonal with respect to the Legendre measure. This means that an orthogonality with respect to x^β is obtained straightforwardly from Theorem 2.1.

Also, some recurrence relations exist for Müntz–Legendre polynomials, e.g.,

$$xP_n'(x) - xP_{n-1}'(x) = \lambda_n P_n(x) + (1 + \bar{\lambda}_{n-1})P_{n-1}(x) \tag{2.8}$$

and

$$P_n(x) = P_{n-1}(x) - (\lambda_n + \bar{\lambda}_{n-1} + 1)x^{\lambda_n}\int_x^1 t^{-\lambda_n - 1}P_{n-1}(t)\, dt \qquad (x \in (0,1]).$$

It is easy to prove that

$$P_n(1) = 1 \qquad \text{and} \qquad P_n'(1) = \lambda_n + \sum_{k=0}^{n-1}(\lambda_k + \bar{\lambda}_k + 1).$$

Taking $x = e^{-t}$, the Müntz–Legendre polynomials can be expressed in terms of a Laplace transform. Namely, we can prove the following (cf. [17]).

THEOREM 2.3. [illegible] $W_n(s)$ [illegible] (2.1) [illegible]

$$G_n(s) = -W_n(-s) = \prod_{k=0}^{n-1} \frac{s - (\bar{\lambda}_k + 1)}{s + \lambda_k} \cdot \frac{1}{s + \lambda_n},$$

[illegible] $P_n(e^{-t})$ [illegible] $G_n(s)$ [illegible]

$$P_n(e^{-t}) = \mathcal{L}^{-1}[G_n(s)].$$

In the proof of this result we can take, for example, $\alpha > 1/2$, and then prove that

$$P_n(e^{-t}) = \frac{1}{2\pi i}\int_{\alpha - i\infty}^{\alpha + i\infty} G_n(s)e^{st}\, ds.$$

An interesting question is connected with the zero distribution of the Müntz–Legendre polynomials for a real sequence Λ. A nice proof of the following result was given in [3].

THEOREM 2.4. [illegible] $\lambda_\nu > -1/2$ $(\nu = 0, 1, \ldots)$ [illegible] $P_n(x;\Lambda_n)$ [illegible] n [illegible] $(0,1)$ [illegible]

$$P_{n-1}(x;\Lambda_{n-1}) \qquad\qquad P_n(x;\Lambda_n)$$

[illegible] $(0,1)$ [illegible]

3. Numerical construction of the generalized Gaussian quadrature rule. Applying the theory of orthogonality for Müntz systems and using a method for evaluating orthogonal Müntz polynomials, in this section we present the method for the numerical construction of the generalized Gaussian quadrature rule for Müntz systems given by (1.7), i.e.,

$$\int_0^1 f(x)d\sigma(x) = \sum_{k=1}^{n} A_k f(x_k) + R_n(f), \tag{3.1}$$

being exact for each $f \in M_{2n-1}(\Lambda)$. The presented method is rather stable and performs calculations in double precision arithmetic to obtain double precision results.

Let $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$ be nondecreasing real sequence such that $\lambda_\nu > -1/2$ for every $\nu \in \mathbb{N}_0$. The orthogonal Müntz–Legendre system $\{P_0, P_1, \ldots, P_{2n-1}\}$, defined by (2.2), is an EH system. It can be recognized as a linear span of the system of basis functions x^Λ, which is an EH system.

In what follows, by $P_n(x) := P_n(x;\Lambda)$ we denote Müntz–Legendre polynomials given by (2.2), constructed for the nondecreasing real sequence $\Lambda = \{\lambda_0, \lambda_1, \ldots\}$. By σ we denote a positive measure supported on the interval $[0,1]$. The use of orthogonal Müntz–Legendre polynomials in our procedure is very important. Our construction is quite different not only from the procedure for the classical Gaussian integration formulae with an algebraic degree of precision but also from the corresponding algorithm given in [13].

In the polynomial case the well-known Golub and Welsch procedure [9] is applied when the quadrature nodes x_k are eigenvalues of the three-diagonal Jacobi matrix, i.e., the zeros of the (algebraic) polynomial $P_n(x)$ orthogonal with respect to the measure $d\sigma(x)$, and weights A_k are expressed in terms of first component of the normalized eigenvectors corresponding to the eigenvalues x_k, respectively (cf. the routine GAUSS in the package ORTHPOL given by Gautschi [7]).

The algorithm in [13] for finding nodes and weights in (3.1) needs the construction of the functions ξ_i and η_i, $i = 1, \ldots, n$, defined by (1.3) and (1.4). The nodes $x_1, \ldots, x_n$ are Gaussian if and only if (see [13, Theorem 4.1])

$$\int_0^1 \xi_i(x)\, d\sigma(x) = \sum_{j=1}^{2n} \alpha_{ij} \int_0^1 P_{j-1}(x)\, d\sigma(x) = 0, \quad i = 1, \ldots, n.$$

In this case, the Gaussian weights are given by

$$A_i = \int_0^1 \eta_i(x)\, d\sigma(x) = \sum_{j=1}^{2n} \beta_{ij} \int_0^1 P_{j-1}(x)\, d\sigma(x), \quad i = 1, \ldots, n.$$

Our algorithm can be summarized in the following steps.

We start with the initial system of nonlinear equations given by

$$\sum_{k=1}^{n} A_k P_\nu(x_k) = \mu_\nu = \int_0^1 P_\nu(x)\, d\sigma, \quad \nu = 0, 1, \ldots, 2n-1. \tag{3.2}$$

Note that the vector of moments μ_ν becomes quite simple, provided $d\sigma(x) = x^{\lambda_0} dx$. In this special case, according to the orthogonality condition of the Müntz–Legendre polynomials (see (2.6)), we have

$$\mu_\nu = \frac{\delta_{\nu,0}}{2\lambda_0 + 1}, \quad \nu \in \mathbb{N}_0.$$

We rewrite the system of equations (3.2) in the form

$$U_n \boldsymbol{A} - \boldsymbol{c} = \boldsymbol{0}, \quad V_n \boldsymbol{A} - \boldsymbol{d} = \boldsymbol{0},$$

where $\boldsymbol{A} = [A_1\ A_2\ \cdots\ A_n]^T$, $\boldsymbol{c} = [\mu_0\ \mu_1\ \cdots\ \mu_{n-1}]^T$, and $\boldsymbol{d} = [\mu_n\ \mu_{n+1}\ \cdots\ \mu_{2n-1}]^T$, and we apply the Newton–Kantorovich method in order to solve it.

Let $\Delta \boldsymbol{A} = \boldsymbol{A} - \hat{\boldsymbol{A}}$ and $\Delta \boldsymbol{x} = \boldsymbol{x} - \hat{\boldsymbol{x}}$, where $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{x}}$ are the unique solutions of the starting nonlinear system. Then we can formulate the Newton–Kantorovich method as

$$U_n \Delta \boldsymbol{A} + Y_n \Delta \boldsymbol{x} = U_n \boldsymbol{A} - \boldsymbol{c},$$

$$V_n \Delta \boldsymbol{A} + Z_n \Delta \boldsymbol{x} = V_n \boldsymbol{A} - \boldsymbol{d},$$

or, equivalently,

$$W_n \begin{bmatrix} \Delta \boldsymbol{A} \\ \Delta \boldsymbol{x} \end{bmatrix} = \begin{bmatrix} U_n \boldsymbol{A} - \boldsymbol{c} \\ V_n \boldsymbol{A} - \boldsymbol{d} \end{bmatrix}, \tag{3.3}$$

where

$$W_n = \begin{bmatrix} U_n & Y_n \\ V_n & Z_n \end{bmatrix}, \qquad \boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T, \tag{3.4}$$

$$U_n = U_n(\boldsymbol{x}) = \begin{bmatrix} P_0(x_1) & P_0(x_2) & \cdots & P_0(x_n) \\ P_1(x_1) & P_1(x_2) & & P_1(x_n) \\ \vdots & & & \\ P_{n-1}(x_1) & P_{n-1}(x_2) & & P_{n-1}(x_n) \end{bmatrix},$$

$$V_n = V_n(\boldsymbol{x}) = \begin{bmatrix} P_n(x_1) & P_n(x_2) & \cdots & P_n(x_n) \\ P_{n+1}(x_1) & P_{n+1}(x_2) & & P_{n+1}(x_n) \\ \vdots & & & \\ P_{2n-1}(x_1) & P_{2n-1}(x_2) & & P_{2n-1}(x_n) \end{bmatrix},$$

$$Y_n = Y_n(\boldsymbol{x}) = \begin{bmatrix} A_1 P_0'(x_1) & A_2 P_0'(x_2) & \cdots & A_n P_0'(x_n) \\ A_1 P_1'(x_1) & A_2 P_1'(x_2) & & A_n P_1'(x_n) \\ \vdots & & & \\ A_1 P_{n-1}'(x_1) & A_2 P_{n-1}'(x_2) & & A_n P_{n-1}'(x_n) \end{bmatrix},$$

$$Z_n = Z_n(\boldsymbol{x}) = \begin{bmatrix} A_1 P_n'(x_1) & A_2 P_n'(x_2) & \cdots & A_n P_n'(x_n) \\ A_1 P_{n+1}'(x_1) & A_2 P_{n+1}'(x_2) & & A_n P_{n+1}'(x_n) \\ \vdots & & & \\ A_1 P_{2n-1}'(x_1) & A_2 P_{2n-1}'(x_2) & & A_n P_{2n-1}'(x_n) \end{bmatrix}.$$

Now, our iterative process becomes

$$\hat{\boldsymbol{x}} = \boldsymbol{x} - \left[V_n U_n^{-1} Y_n - Z_n\right]^{-1} (\boldsymbol{d} - V_n U_n^{-1} \boldsymbol{c}),$$

$$\hat{\boldsymbol{A}} = U_n^{-1} \left[Y_n (\boldsymbol{x} - \hat{\boldsymbol{x}}) + \boldsymbol{c}\right]. \tag{3.5}$$

We can prove that our method for this construction is convergent. The basic theorem on the convergence of the Newton–Kantorovich method can be summarized as follows: Suppose we are given some differentiable function $\boldsymbol{f} : \mathbb{R}^m \mapsto \mathbb{R}^m$, $m \in \mathbb{N}$, with the property that at the solution of $\boldsymbol{f}(\boldsymbol{x}) = 0$, we have that $|J(\boldsymbol{x})| \neq 0$, i.e., the Jacobian of $\boldsymbol{f}$ at the solution is not singular. Then, there exists some $\varepsilon > 0$, such that for any $\boldsymbol{y}_0$, with the property $||\boldsymbol{y}_0 - \boldsymbol{x}|| < \varepsilon$, we have that the sequence $\boldsymbol{y}_k$, $k \in \mathbb{N}$, defined by

$$\boldsymbol{y}_{k+1} = \boldsymbol{y}_k - (J(\boldsymbol{y}_k))^{-1}\boldsymbol{f}(\boldsymbol{y}_k), \quad k \in \mathbb{N}_0,$$

tends to $\boldsymbol{x}$ with a quadratic speed.

LEMMA 3.1. [illegible] (3.5) [illegible]. The Jacobian for the starting nonlinear system can be easily calculated in the form

$$J(\hat{\boldsymbol{x}}, \hat{\boldsymbol{A}}) = W_n.$$

It is easy to see that the determinant $|W_n|$ can be written as

$$|W_n| = \left(\prod_{k=1}^{n} A_k\right) \begin{vmatrix} P_0(x_1) & \cdots & P_0(x_n) & P_0'(x_1) & \cdots & P_0'(x_n) \\ P_1(x_1) & & P_1(x_n) & P_1'(x_1) & & P_1'(x_n) \\ \vdots & & & & & \\ P_{2n-1}(x_1) & & P_{2n-1}(x_n) & P_{2n-1}'(x_1) & & P_{2n-1}'(x_n) \end{vmatrix}.$$

Since our system of functions is EH and the weights A_k, $k = 1, \ldots, n$, are positive, this determinant cannot be zero, and thus the Jacobian has full rank. Also, since the system of functions x^Λ is Chebyshev, the matrix U_n is invertible. It is enough to apply the theorem on the matrix inversion in the block form presented in [18, p. 205] (see also [10, p. 13] and [26, p. 201]) to finish the proof. □

The standard problem with the Newton–Kantorovich method is a problem of determining starting values. One way to ensure the convergence, for $d\sigma(x) = x^{\lambda_0}dx$, is based on the continuation of the classical Gaussian quadrature rule.

LEMMA 3.2. [illegible] $\Lambda = \{\lambda_k \mid k \in \mathbb{N}_0\}$ $\lambda_k > -1/2$ $k \in \mathbb{N}_0$ [illegible] $A_k := A_k(\alpha)$ $x_k := x_k(\alpha)$ $\alpha \in [0, 1]$ [illegible]

$$\sum_{k=1}^{n} A_k P_\nu^\alpha(x_k) = \frac{\delta_{\nu,0}}{2\alpha\lambda_0 + 1}, \quad \nu = 0, 1, \ldots, 2n-1,$$

[illegible]

$$P_\nu^\alpha(x) := P_\nu(x; (1-\alpha)\mathbb{N}_0 + \alpha\Lambda) = \frac{1}{2\pi i}\oint_\Gamma G_\nu(s, \alpha, \Lambda)x^s\, ds$$

[illegible]

$$G_\nu(s, \alpha, \Lambda) = \prod_{k=0}^{\nu-1} \frac{s + (1-\alpha)k + \alpha\lambda_k + 1}{s - (1-\alpha)k - \alpha\lambda_k} \frac{1}{s - (1-\alpha)\nu - \alpha\lambda_\nu},$$

[illegible]

. It is easy to verify that the function

$$\boldsymbol{F}(\boldsymbol{x},\boldsymbol{A},\alpha)=\begin{bmatrix}\sum\limits_{k=1}^{n}A_kP_0^{\alpha}(x_k)\\ \vdots\\ \sum\limits_{k=1}^{n}A_kP_{2n-1}^{\alpha}(x_k)\end{bmatrix}$$

has derivatives in x_k, A_k, $k=1,\dots,n$, and α for $0<x_1<x_2<\cdots<x_n<1$, $A_k>0$, $k=1,\dots,n$, and $\alpha\in(0,1)$. We have

$$\partial_{x_j}\left(\sum_{k=1}^{n}A_kP_\nu^{\alpha}(x_k)\right)=A_j(P_\nu^{\alpha})'(x_j),\quad \nu=0,1,\dots,2n-1,\quad j=1,\dots,n,$$

$$\partial_{A_j}\left(\sum_{k=1}^{n}A_kP_\nu^{\alpha}(x_k)\right)=P_\nu^{\alpha}(x_j),\quad \nu=0,1,\dots,2n-1,\quad j=1,\dots,n,$$

$$\partial_{\alpha}\left(\sum_{k=1}^{n}A_kP_\nu^{\alpha}(x_k)\right)=\frac{1}{2\pi i}\oint_{\Gamma}\partial_{\alpha}\left\{\big(G_\nu(s,\alpha,\Lambda)\big)\left(\sum_{k=1}^{n}A_kx_k^s\right)\right\}ds.$$

Since for every $\alpha\in[0,1]$, there exists the unique solution of

$$\boldsymbol{F}(\boldsymbol{A},\boldsymbol{x},\alpha)=\frac{\boldsymbol{e}_0}{2\alpha\lambda_0+1} \tag{3.6}$$

in $\boldsymbol{A}$ and $\boldsymbol{x}$, we can define the functions $A_k:=A_k(\alpha)$, $x_k:=x_k(\alpha)$, $k=1,\dots,n$. Here, $\boldsymbol{e}_0$ is the fist coordinate vector, i.e., $\boldsymbol{e}_0=[1\ 0\ \cdots\ 0]^T$.

Using the system of equations (3.6), we can calculate derivatives with respect to α in the form

$$\partial_{\alpha}\boldsymbol{F}+\sum_{k=1}^{n}\left(\partial_{x_k}\boldsymbol{F}\frac{dx_k}{d\alpha}+\partial_{A_k}\boldsymbol{F}\frac{dA_k}{d\alpha}\right)=-\frac{2\lambda_0}{(2\alpha\lambda_0+1)^2}\boldsymbol{e}_0.$$

Now, from here we get

$$W_n\begin{bmatrix}\dfrac{dA_1}{d\alpha}\\ \vdots\\ \dfrac{dx_n}{d\alpha}\end{bmatrix}=-\frac{2\lambda_0}{(2\alpha\lambda_0+1)^2}\boldsymbol{e}_0-\partial_{\alpha}\boldsymbol{F},$$

where the matrix W_n is the same as in (3.4), formed with P_ν replaced by P_ν^{α}. Since the matrix W_n is invertible, it is easy to conclude that the functions $A_k:=A_k(\alpha)$, $x_k:=x_k(\alpha)$, $k=1,\dots,n$, are differentiable. □

Since the functions $A_k(\alpha)$ and $x_k(\alpha)$, $k=1,\dots,n$, are continuous in $\alpha\in[0,1]$, we start with $\alpha=0$ and construct the classical Gaussian quadrature rule with parameters $A_k(0)$ and $x_k(0)$. Now, if we increase α by a small amount, we can use the obtained values for $A_k(0)$ and $x_k(0)$ as starting values for $A_k(\alpha)$ and $x_k(\alpha)$ in the Newton–Kantorovich method. Again, we can increase α by a small amount and try to solve the system with starting values obtained from the previous step. We repeat the procedure until $\alpha=1$ is reached.

There is still another possibility of continuation. Namely, we have the following lemma.

LEMMA 3.3. [illegible] $\boldsymbol{\mu} = [\mu_0\ \mu_1\ \cdots\ \mu_{2n-1}]^T$ [illegible] (3.2) [illegible] $A_k := A_k(\boldsymbol{\mu})$ [illegible] $x_k := x_k(\boldsymbol{\mu})$ $k = 1, \ldots, n$ [illegible] (3.2) [illegible]. Note that by the fact that there exists a unique solution to the system of equations (3.2), we have that A_k and x_k are single valued functions. Since the Jacobi matrix W_n is invertible at any solution, according to the inverse function theorem, we have that x_k and A_k, $k = 1, \ldots, n$, are continuous in $\boldsymbol{\mu}$. □

Using this lemma and the following idea we can construct the generalized Gaussian quadrature rule for some measure σ. Denote moments for the measure $x^{\lambda_0}dx$ by $\delta_{\nu,0}/(2\lambda_0+1)$, $\nu = 0, 1, \ldots, 2n-1$. By $\boldsymbol{\mu}$ integrals as in (3.2), we can introduce the moments

$$\hat{\mu}_\nu = (1-\alpha)\frac{\delta_{\nu,0}}{2\lambda_0+1} + \alpha\mu_\nu, \quad \nu = 0, 1, \ldots, 2n-1, \quad \alpha \in [0,1]. \tag{3.7}$$

Then x_k and A_k, $k = 1, \ldots, n$, are continuous functions of $\alpha \in [0,1]$. Thus, we increase α by a small amount and solve the systems of nonlinear equations (3.2) in order to achieve $\alpha = 1$ starting with $\alpha = 0$.

According to the previous two lemmas, we can construct the solution using the following steps:

- First, using Lemma 3.2, starting from the classical Gauss–Legendre quadrature rule, we construct the generalized quadrature rule for the $x^{\lambda_0}dx$ measure and any given system Λ.
- Once the generalized quadrature rule is constructed for the $x^{\lambda_0}dx$ measure, we introduce the moments, as given in (3.7), to construct the generalized quadrature rule for the targeting measure σ, with the moment vector $\boldsymbol{\mu}$.

We next address numerical properties of the nonlinear system of equations (3.2). For example, we have to give the algorithm to calculate the values of the polynomials P_ν, $\nu = 0, 1, \ldots, 2n-1$, at the points x_k, $k = 1, \ldots, n$. This is already solved (see [17]), and we address it in subsection 3.3. For derivatives $P_k'(x)$ we can use the recurrence relation (2.8), i.e.,

$$xP_n'(x) - xP_{n-1}'(x) = \lambda_n P_n(x) + (1+\bar{\lambda}_{n-1})P_{n-1}(x).$$

It is easy to verify that our method depends strongly on the condition number of the matrix U_n. In fact, assuming the Newton–Kantorovich method is converging, it is easy to verify that the weights $\boldsymbol{A}$ are the solution of the system of linear equations

$$U_n\boldsymbol{A} = \frac{\boldsymbol{e}_0}{2\lambda_0+1},$$

where the matrix $U_n = [P_\nu(x_k)]_{\nu,k=1}^n$ is calculated at the solution for the nodes x_k, $k = 1, \ldots, n$. Hence, the precision in constructed weights A_k, $k = 1, \ldots, n$, can be determined according to the condition number of the matrix U_n. Note that the precision in the constructed nodes x_k, $k = 1, \ldots, n$, is also determined by the precision in the constructed weights A_k. To present this simple fact, we consider our system of equations for the Newton–Kantorovich method (3.5). We have

$$\begin{aligned}\hat{x} &= x - (V_nU_n^{-1}Y_n - Z_n)^{-1}(\boldsymbol{d} - V_nU_n^{-1}\boldsymbol{c}) \\ &= x - (V_nU_n^{-1}Y_n - Z_n)^{-1}(\boldsymbol{d} - V_n\bar{\boldsymbol{A}}) = x - (V_nU_n^{-1}Y_n - Z_n)^{-1}(\boldsymbol{d} - \bar{\boldsymbol{d}}),\end{aligned}$$

where

$$\bar{\boldsymbol{d}} = V_n \bar{\boldsymbol{A}}$$

is a vector of the calculated moments $[\bar{\mu}_n \cdots \bar{\mu}_{2n-1}]^T$ with weights $\bar{\boldsymbol{A}}$. Obviously we have $\boldsymbol{d} \neq \bar{\boldsymbol{d}}$, which has as a consequence that nodes $\boldsymbol{x}$ are constructed with an error originating from the fact that weights cannot be constructed without an error.

3.1. Condition number of U_n. The algorithm strongly depends on the condition number of the matrix U_n. Assuming all nodes $\boldsymbol{x}$ are known, it is obvious that we can calculate the weights $\boldsymbol{A}$ as $U_n^{-1}\boldsymbol{c}$. Although it is hard to give precise results about the condition number of U_n, for some special cases linked with polynomials, we can give some theoretical results concerning the condition number of this matrix.

We define the condition number of a matrix B to be $||B||||B^{-1}||$, where, as it is known, any matrix norm will do. Following [8, p. 64], we restrict our attention to the Frobenius norm, i.e., $||\,[b_{i,j}]_{i,j=1}^{n}\,|| = (\sum_i \sum_j b_{i,j}^2)^{1/2}$. We consider the slightly modified matrix U_n, i.e., we put the normalized version of Müntz–Legendre polynomials into the system of nonlinear equations (3.2) so that we have

$$\sum_{k=1}^{n} A_k p_\nu(x_k) = \frac{\mu_\nu}{||P_\nu||} = \int_0^1 p_\nu(x)d\sigma(x), \quad \nu = 0, 1, \ldots, 2n-1.$$

This has as a consequence that in matrices U_n, V_n, Y_n, and Z_n, P_ν is replaced with p_ν, $\nu = 0, 1, \ldots, 2n-1$, and vectors $\boldsymbol{d}$ and $\boldsymbol{c}$ are filled with the sequence of moments $\mu_\nu/||P_\nu||$, $\nu \in \mathbb{N}_0$. This renormalization of the system does not change anything, as is explained in [8, pp. 65–66].

In this case, we can express

$$||U_n||^2 = \sum_{k=1}^{n} \sum_{\nu=0}^{n-1} p_\nu(x_k)^2.$$

In order to get the inverse matrix, we define Müntz–Lagrange polynomials as

$$\ell_k(x) = \sum_{\nu=0}^{n-1} \alpha_k^\nu p_\nu(x), \quad k = 1, \ldots, n,$$

with the property

$$\ell_k(x_\nu) = \delta_{k,\nu}, \quad k, \nu = 1, \ldots, n.$$

The Müntz–Lagrange polynomials ℓ_k, $k = 1, \ldots, n$, are defined correctly since p_ν, $\nu = 0, 1, \ldots, n-1$, makes a Chebyshev system. Also, it is easy to prove that $\operatorname{span}\{\ell_k \mid k = 1, \ldots, n\}$ is exactly the same as $\operatorname{span}\{p_\nu \mid \nu = 0, 1, \ldots, n-1\}$.

Now, we can identify the matrix U_n^{-1} as the matrix $[\alpha_k^\nu]_{\nu=0,1,\ldots,n-1,k=1,\ldots,n}$, and then we can calculate the norm of the matrix U_n^{-1} in the form

$$||U_n^{-1}||^2 = \sum_{\nu=0}^{n-1} \sum_{k=1}^{n} (\alpha_k^\nu)^2.$$

Using the simple identity

$$\int_0^1 \ell_k(x)^2 dx = \int_0^1 \left\{ \sum_{\nu=0}^{n-1} \alpha_k^\nu p_\nu(x) \sum_{\mu=0}^{n-1} \alpha_k^\mu p_\mu(x) \right\} dx = \sum_{\nu=0}^{n-1} (\alpha_k^\nu)^2,$$

we get

$$||U_n^{-1}||^2 = \int_0^1 \left(\sum_{k=1}^{n} \ell_k(x)^2 \right) dx.$$

In total we express the condition number of the matrix U_n as

$$||U_n||||U_n^{-1}|| = \left(\sum_{k=1}^{n} \sum_{\nu=0}^{n-1} p_\nu(x_k)^2 \right)^{1/2} \left(\int_0^1 \left(\sum_{k=1}^{n} \ell_k(x)^2 \right) dx \right)^{1/2}.$$

It is not easy to give some estimates of this number; hence, we are going to discuss two cases connected to the orthogonal polynomials. The first is for $\Lambda = \{k \mid k \in \mathbb{N}_0\}$, where our Müntz–Legendre polynomials are really Legendre polynomials. The second case is for $\Lambda = \{0 \mid k \in \mathbb{N}_0\}$, where our Müntz–Legendre polynomials are Laguerre polynomials in $-\log x$, as given in (2.5). As experiments show, the behavior for the general nondecreasing sequence Λ can be understood using these two special cases. Hence, we discuss these two cases.

According to the fact that in these cases Müntz–Lagrange polynomials can be expressed in the form

$$\ell_\nu(x) = \frac{p_n(x)}{(x - x_\nu) p_n'(x_\nu)}, \quad \nu = 1, \ldots, n,$$

for the Legendre case, and as

$$\ell_\nu(x) = \frac{p_n(x)}{(-\log x - x_\nu) p_n'(x_\nu)}, \quad \nu = 1, \ldots, n,$$

in the Laguerre case, we have an orthogonality of the Müntz–Lagrange polynomials, i.e.,

$$\int_0^1 \ell_\nu(x) \ell_k(x) dx = ||\ell_k||^2 \delta_{\nu,k}, \quad \nu, k = 1, \ldots, n.$$

Then, according to the fact that $1 = \sum_\nu \ell_\nu(x)$, and using the mentioned orthogonality, we find

$$\int_0^1 \left(\sum_{\nu=1}^{n} \ell_\nu(x)^2 \right) dx = \int_0^1 \left(\sum_{\nu=1}^{n} \ell_\nu(x) \sum_{k=1}^{n} \ell_k(x) \right) dx = \int_0^1 \left(\sum_{\nu=1}^{n} \ell_\nu(x) \right)^2 dx = 1,$$

so that $||U_n^{-1}|| = 1$.

Using the Shohat formula (see [19], [23]), we can express the norm $||U_n||$ in the form

$$||U_n||^2 = \sum_{k=1}^{n} \frac{1}{w_k},$$

where we denoted by w_k, $k = 1, \ldots, n$, the weights in the classical Gaussian quadrature rule associated with the sequence of polynomials p_ν, $\nu \in \mathbb{N}_0$. Using the Markov inequality (see [24, p. 50])

$$\sum_{k=\nu+1}^{n} w_k \leq \int_{x_\nu}^{+\infty} d\mu(x), \quad \nu = 1, \ldots, n-1,$$

we have, for the Laguerre measure, $\exp(x_{n-1}) \le (w_n)^{-1}$, which means that in the Laguerre case we have $||U_n|| \ge \exp(\sqrt{n}/2)$, where we have used an estimate for zeros of Laguerre polynomials given in [24, p. 129]. Hence, for the sequence $\Lambda = \{0 \mid k \in \mathbb{N}_0\}$, the condition number of the matrix U_n grows exponentially with n.

In what follows, for two sequences A_n and B_n we write $A_n \sim B_n$ if and only if there exist two positive constants C_1 and C_2 such that $C_1 \le |A_n/B_n| \le C_2$.

Now, for the Legendre measure we use the following asymptotic estimate $w_k \sim \pi(1-x_k^2)^{1/2}/n$, $k = 1, \dots, n$, given in [21, p. 85] (see also Szegő [24, p. 253]), to get

$$||U_n||^2 = \sum_{k=1}^{n} \frac{1}{w_k} \sim \frac{n}{\pi} \sum_{k=1}^{n} (1-x_k^2)^{-1/2}.$$

Using the estimate for the zeros $x_k = -\cos\theta_k$, $k = 1, \dots, n$, of the Legendre polynomials (see [24, p. 125])

$$\frac{k-1/2}{n+1/2} < \theta_k < \frac{k}{n+1/2}, \quad k = 1, \dots, n,$$

and Jordan inequality $2\theta/\pi < \sin\theta < \theta$, $\theta \in (0, \pi/2)$, we have simply

$$||U_n||^2 \sim \frac{n}{\pi} \sum_{k=1}^{n} \frac{1}{\sin\theta_k} \sim \frac{2n}{\pi} \sum_{k=1}^{[n/2]} \frac{1}{\sin\theta_k} \sim n \sum_{k=1}^{[n/2]} \frac{1}{\theta_k} \sim n\left(n+\frac{1}{2}\right) \sum_{k=1}^{[n/2]} \frac{1}{k} \sim n^2 \log n,$$

so that, for the Legendre case, i.e., $\Lambda = \{k \mid k \in \mathbb{N}_0\}$, we have that the condition number of the matrix U_n grows like $n \log^{1/2} n$.

LEMMA 3.4. , $\Lambda = \{k \mid k \in \mathbb{N}_0\}$, . . $||U_n||||U_n^{-1}|| \sim n \log^{1/2} n$, $\Lambda = \{0 \mid k \in \mathbb{N}_0\}$, . . $||U_n||||U_n^{-1}|| \sim \exp(\sqrt{n}/2)$

This lemma shows that in the cases when a nondecreasing sequence Λ is close to the Legendre case, we should expect the condition number of U_n to grow relatively slowly with respect to n. In the opposite case for the almost constant sequence Λ, we should expect the condition number to grow rather rapidly with respect to n. This behavior is going to be illustrated in the last section.

3.2. Construction for the nonintegrable basis. In what follows we discuss the construction of the generalized Gaussian quadrature rule in the case when a nondecreasing sequence Λ is such that for some λ_k we have $2\lambda_k \le -1$. It is easy to verify that in this case we cannot construct the sequence of Müntz–Legendre polynomials orthogonal with respect to the Legendre measure since the functions $x^{\lambda_k} x^{\lambda_k}$, for which $2\lambda_k \le -1$, are not integrable with respect to the Legendre measure.

What we can do is use Müntz polynomials given by (2.7), which are orthogonal with respect to the measure $x^\beta dx$, where β is chosen such that we have $2\lambda_k + \beta > -1$, $k \in \mathbb{N}_0$. If we introduce a new nondecreasing sequence $\Lambda + \beta/2$, we can construct the sequence of Müntz polynomials $P_n(\cdot; \Lambda + \beta/2)$, $n \in \mathbb{N}_0$, generated by $\Lambda + \beta/2$, orthogonal with respect to Legendre measure, and then, using the previously described method, we are able to solve the system of equations

$$\sum_{k=1}^{n} A_k P_\nu\left(x_k; \Lambda + \frac{\beta}{2}\right) = \frac{\delta_{\nu,0}}{2\lambda_0 + \beta + 1} = \int_0^1 x^{\lambda_0 + \beta/2} P_\nu\left(x; \Lambda + \frac{\beta}{2}\right) dx, \; 0 \le \nu \le 2n-1.$$

Then, if we rewrite the system in the form

$$\sum_{k=1}^{n}\left(A_k x_k^{\beta/2}\right)\left(x_k^{-\beta/2}P_\nu\left(x_k;\Lambda+\frac{\beta}{2}\right)\right)$$
$$=\frac{1}{2\lambda_0+\beta+1}\delta_{\nu,0}=\int_0^1 x^{\lambda_0+\beta}\left(x^{-\beta/2}P_\nu\left(x;\Lambda+\frac{\beta}{2}\right)\right)dx,\quad \nu=0,1,\ldots,2n-1,$$

and if we choose

$$A_k^*=A_k x_k^{\beta/2},\quad x_k^*=x_k,\quad k=1,\ldots,n,$$

it can be verified that the quadrature rule

$$\sum_{k=1}^{n}A_k^* f(x_k^*)$$

is exact in the linear span of $x^{\Lambda_{2n-1}}$ with respect to the measure $x^{\lambda_0+\beta}dx$. Thus, we have the following result.

LEMMA 3.5. $\Lambda+\beta/2\quad\beta\in\mathbb{R}$ $x_k\quad A_k\quad k=1,\ldots,n$ (3.2) $x^{\lambda_0+\beta/2}dx$ (3.2) $x^{\lambda_0+\beta}dx$ $x^{\Lambda_{2n-1}}$

$$x_k^*=x_k,\quad A_k^*=A_k x_k^{\beta/2},\quad k=1,\ldots,n.$$

A consequence of this simple observation is the possibility of using the described algorithm for the construction of the generalized Gaussian quadrature rule, for example, for the set of functions

$$x^{k-2/3},\quad x^k,\quad k=0,1,\ldots,2n-1.$$

Some numerical values for this case are presented in the last section.

3.3. Numerical computation of Müntz–Legendre polynomials. A direct evaluation of Müntz polynomials $P_n(x)$ in the power form (2.3) can be a big problem in finite arithmetic, especially when n is a large number and x is close to 1. These problems have been addressed in [17].

In this subsection we give a stable numerical method for evaluating the values of the Müntz–Legendre polynomials defined by (2.1) and (2.2), i.e.,

$$P_n(x)=\frac{1}{2\pi i}\oint_\Gamma W_n(s)x^s\,ds,\qquad W_n(s)=\prod_{\nu=0}^{n-1}\frac{s+\bar{\lambda}_\nu+1}{s-\lambda_\nu}\cdot\frac{1}{s-\lambda_n}. \tag{3.8}$$

For a detailed discussion, see [17].

Our method is based on a direct evaluation of the contour integral in (3.8). At first we take the contour $\Gamma=\Gamma_R=C_R\cup L_R$ (see Figure 3.1 (left)). As we can see, C_R is a semicircle with the center at $\sigma<-1/2$, its radius R is such that all poles of $W_n(s)$ are inside the contour Γ_R, and L_R is a part of the straight line $s=\sigma+it$, $-R\le t\le R$. (Notice that the function $W_n(s)$ in Figure 3.1 has only real poles.)

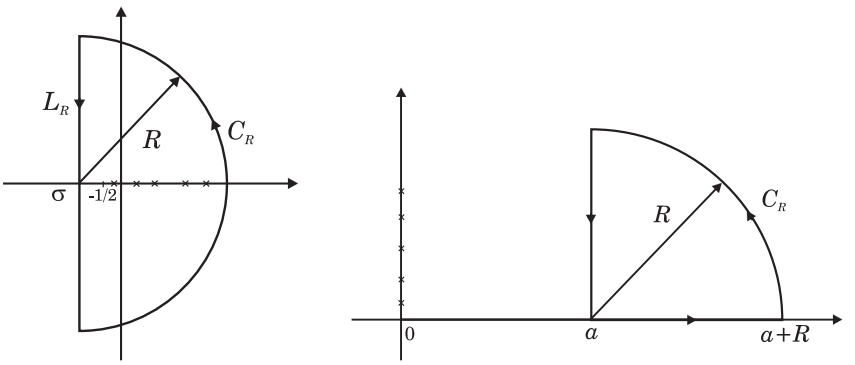


FIG. 3.1. *The contour of integration for the integral in* (3.1) (*left*) *and for the integral* $L_2(f_n(\cdot;\omega))$ (*right*).

We have a simple lemma.

LEMMA 3.6. $\int_{C_R} W_n(s)x^s\,ds \to 0$ $R \to +\infty$

Thus, when $R \to +\infty$, an integration along the contour Γ_R reduces to the integration over the line L_R, so that

$$P_n(x) = \frac{x^\sigma}{2\pi i}\int_{-\infty}^{+\infty} f_n(t;\omega)e^{it}\,dt,$$

where $\omega = \log(1/x) > 0$ and

$$f_n(t;\omega) = \prod_{\nu=0}^{n-1} \frac{t + i(\sigma + \bar{\lambda}_\nu + 1)\omega}{t + i(\sigma - \lambda_\nu)\omega} \cdot \frac{1}{t + i(\sigma - \lambda_n)\omega}. \tag{3.9}$$

This gives the following result.

THEOREM 3.7. $\sigma < -1/2$ $\omega = \log(1/x)$ $f_n(t;\omega)$ (3.9)

$$\varphi_n(t;\omega) = \frac{1}{2i}\Big(f_n(t;\omega)e^{it} + f_n(-t;\omega)e^{-it}\Big). \tag{3.10}$$

$$P_n(x) = \frac{x^\sigma}{\pi}\int_0^{+\infty} \varphi_n(t;\omega)\,dt. \tag{3.11}$$

In what follows we consider the case when the sequence Λ is real. An important corollary of Theorem 3.7 is the following result.

THEOREM 3.8. $\Lambda = \{\lambda_0, \lambda_1, \lambda_2, \ldots\}$ $\lambda_\nu > -1/2$ $k \in \mathbb{N}_0$ $f_n(t;\omega)$ (3.9) $\omega = \log(1/x)$ $\sigma < -1/2$

$$P_n(x) = \frac{x^\sigma}{\pi}\mathrm{Im}\left\{\int_0^{+\infty} f_n(t;\omega)e^{it}\,dt\right\}. \tag{3.12}$$

In this case we have $f_n(-t;\omega) = \overline{f_n(t;\omega)}$, then (3.10) becomes $\varphi_n(t;\omega) = \mathrm{Im}\{f_n(t;\omega)e^{it}\}$, and (3.11) gives (3.12). The poles of $f_n(t;\omega)$ are then purely imaginary, $i(\lambda_\nu - \sigma)\omega$, $\nu = 0, 1, \ldots, n$, and located on the positive part of the imaginary axes, because of $\lambda_\nu > \sigma$ and $\omega = \log(1/x) > 0$.

In order to calculate the integral in (3.12) (or in (3.11)) we use an idea from complex integration (see [16, Thm. 2.2]). We select a positive number $a = m\pi$ ($m \in \mathbb{N}$) and put

$$\int_0^{+\infty} f_n(t;\omega)e^{it}\,dt = \int_0^{a} f_n(t;\omega)e^{it}\,dt + \int_a^{+\infty} f_n(t;\omega)e^{it}\,dt$$
$$= L_1(f_n(\cdot;\omega)) + L_2(f_n(\cdot;\omega)).$$

Here,

$$L_1(f_n(\cdot;\omega)) = \int_0^{a=m\pi} f_n(t;\omega)e^{it}\,dt = \sum_{k=1}^{m}\int_{(k-1)\pi}^{k\pi} f_n(t;\omega)e^{it}\,dt,$$

i.e.,

$$L_1(f_n(\cdot;\omega)) = \pi\int_0^1 \left[\sum_{k=1}^{m} f_n\big(\pi(\xi+k-1);\omega\big)e^{i\pi(\xi+k-1)}\right] d\xi, \tag{3.13}$$

and $L_2(f_n(\cdot;\omega)) = \int_a^{+\infty} f_n(t;\omega)e^{it}\,dt$.

Since $f_n(z;\omega)$ is a holomorphic function in the region $D = \big\{z \in \mathbb{C} \mid \operatorname{Re} z \geq a > 0,\ \operatorname{Im} z \geq 0\big\}$ and $|f_n(z;\omega)| \leq A/|z|$ when $|z| \to +\infty$, for some positive constant A, we can prove

$$L_2(f_n(\cdot;\omega)) = ie^{ia}\int_0^{+\infty} f_n(a+iy;\omega)e^{-y}\,dy. \tag{3.14}$$

For $a = m\pi$, (3.14) becomes

$$L_2(f_n(\cdot;\omega)) = (-1)^m\int_0^{+\infty} \psi_n(y;\omega)e^{-y}\,dy, \tag{3.15}$$

where

$$\psi_n(y;\omega) = if_n(a+iy;\omega) = \prod_{\nu=0}^{n-1}\frac{y+(\sigma+\lambda_\nu+1)\omega-ia}{y+(\sigma-\lambda_\nu)\omega-ia}\cdot\frac{1}{y+(\sigma-\lambda_n)\omega-ia}.$$

THEOREM 3.9. [illegible] 3.8 [illegible]

$$P_n(x) = \frac{x^\sigma}{\pi}\operatorname{Im}\Big\{L_1(f_n(\cdot;\omega)) + L_2(f_n(\cdot;\omega))\Big\},$$

[illegible] $L_1(f_n(\cdot;\omega))$ [illegible] $L_2(f_n(\cdot;\omega))$ [illegible] (3.13) [illegible] (3.15) [illegible]

In the numerical implementation we use the Gauss–Legendre rule on $(0,1)$ and the Gauss–Laguerre rule for calculating $L_1(f_n(\cdot;\omega))$ and $L_2(f_n(\cdot;\omega))$, respectively. A convenient choice for the parameter σ is $\lambda_{\min} - \pi/\omega$, where $\lambda_{\min} = \min\{\lambda_0, \lambda_1, \ldots\}$.

4. Numerical examples. In this section we present some numerical examples. Here we want to emphasize that in all examples, as starting values for the Newton–Kantorovich method (3.2), we used the zeros of the Müntz–Legendre polynomial P_n. According to Theorem 2.4 there are exactly n different zeros of P_n in the interval $(0,1)$.

TABLE 4.1

Nodes and weights in the generalized Gaussian quadrature rule for $\lambda_{2k} = k$ and $\lambda_{2k+1} = k+1/3$, $k \in \mathbb{N}_0$ (left) and $\lambda_{2k} = \lambda_{2k+1} = k$, $k \in \mathbb{N}_0$ (right) for 30 points.

Nodes	Weights	Nodes	Weights
.11890626221894(−4)	.41297515794865(−4)	.73237974427261(−5)	.27989215430955(−4)
.14403587843195(−3)	.26627367672116(−3)	.11004470045777(−3)	.21736552650254(−3)
.65503119332169(−3)	.82242421680232(−3)	.54691832618397(−3)	.72070358653439(−3)
.19430375893262(−2)	.18402261193412(−2)	.17018575191016(−2)	.16744609650550(−2)
.45251469339122(−2)	.34260127408358(−2)	.40838636097144(−2)	.31912824064115(−2)
.90102163603464(−2)	.56560992521664(−2)	.83000411768823(−2)	.53537883135293(−2)
.16066432118835(−1)	.85721821462782(−2)	.15022978156080(−1)	.82096213680820(−2)
.26385011600158(−1)	.12178245807910(−1)	.24953923615755(−1)	.11768029213085(−1)
.40641627763273(−1)	.16439118767071(−1)	.38783386171063(−1)	.15998143504891(−1)
.59457247937461(−1)	.21280752356760(−1)	.57150898481176(−1)	.20829041093624(−1)
.83360115784155(−1)	.26592222426836(−1)	.80605741472655(−1)	.26151597661309(−1)
.11275056965256(−1)	.32229382586004(−1)	.10957039423452	.31822068269456(−1)
.14787028400295	.38020028095164(−1)	.14430837300150	.37667255999807(−1)
.18877734727908	.43770365881357(−1)	.18489794942753	.43491062263223(−1)
.23532835662211	.49272530816407(−1)	.23121300154826	.49082153228403(−1)
.28716842664143	.54312843701643(−1)	.28291196019721	.54222428661263(−1)
.34372968768336	.58680474177812(−1)	.33943548119085	.58695944290977(−1)
.40423850186626	.62176153382847(−1)	.40001311310945	.62297918114936(−1)
.46773126684306	.64620577387348(−1)	.46367885693931	.64843445707233(−1)
.53307832255314	.65862153426135(−1)	.52929514274104	.66175559851254(−1)
.59901513975232	.65783766285597(−1)	.59558439532565	.66172295277200(−1)
.66417966475440	.64308280886562(−1)	.66116704039692	.64752458922921(−1)
.72715443518779	.61402547573361(−1)	.72460452817603	.61879858349955(−1)
.78651187749469	.57079736900636(−1)	.78444573473494	.57565803663442(−1)
.84086105709630	.51399898611490(−1)	.83927495147866	.51869769192466(−1)
.88889408329736	.44468713374991(−1)	.88775959761734	.44898178495567(−1)
.92943037965232	.36434488347266(−1)	.92869579596323	.36801362500369(−1)
.96145713651391	.27483573132827(−1)	.96105005918741	.27768870377414(−1)
.98416462586884	.17834924980349(−1)	.98399570352129	.18023873784161(−1)
.99697836948100	.77447054256862(−2)	.99694595867976	.78276701954968(−2)

The first example is concerned with $\lambda_{2k} = k$ and $\lambda_{2k+1} = k + 1/3$, $k \in \mathbb{N}$. Using the demonstrated algorithm we are able to safely construct generalized Gaussian quadrature rules with 30 points. The corresponding results are presented in Table 4.1 (left). Numbers in parentheses indicate decimal exponents. The presented precision is achieved in six iterations. Figure 4.1 (left) presents the condition number of the matrix U_n as a function of the number of nodes n in the generalized Gaussian quadrature rule (1.7). As we can see, the condition number grows as n^2, which makes the

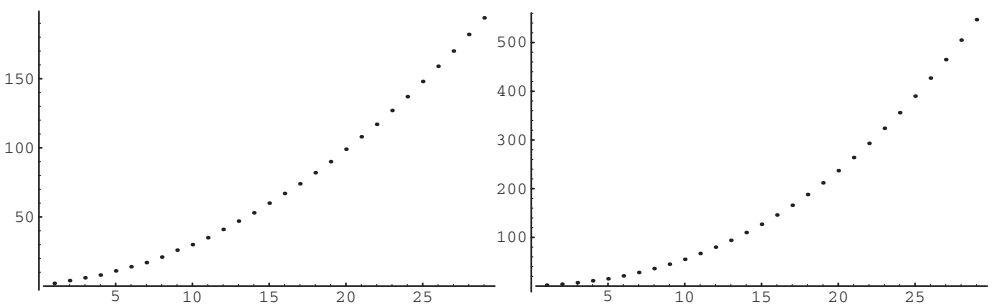


FIG. 4.1. *The condition number of the matrices U_n (left) and $V_n U_n^{-1} Y_n - Z_n$ (right) for $n = 2(1)30$ and $\lambda_{2k} = k$, $\lambda_{2k+1} = k + 1/3$, $k \in \mathbb{N}_0$.*

TABLE 4.2

Nodes and weights in the generalized Gaussian quadrature rule for $\lambda_{3k} = \lambda_{3k+1} = \lambda_{3k+2} = k$, $k \in \mathbb{N}_0$, and 25 points (left) and for $\lambda_{2k} = k - 2/3$, $\lambda_{2k+1} = k$, $k \in \mathbb{N}_0$, and 30 points (right).

Nodes	Weights	Nodes	Weights
.19071312857447(−6)	.89820559783363(−6)	.13087228815024(−5)	.83137155735974(−5)
.62344778034991(−5)	.16484108838767(−4)	.54198486279011(−4)	.13031036416570(−3)
.55318345803585(−4)	.10072125141830(−3)	.35417216474237(−3)	.52866204357169(−3)
.26885580904675(−3)	.37100307224423(−3)	.12534406542994(−2)	.13520067889260(−2)
.92211316431026(−3)	.10179277384384(−2)	.32421307701854(−2)	.27261843607855(−2)
.25153686640161(−2)	.22982801041688(−2)	.69217972803109(−2)	.47471314476665(−2)
.58293105313034(−2)	.45106638074320(−2)	.12972528592304(−1)	.74751509252983(−2)
.11948207244089(−1)	.79560124465681(−2)	.22115108819058(−1)	.10930789003913(−1)
.22237933600721(−1)	.12887776775257(−1)	.35069941343596(−1)	.15092506726768(−1)
.38271779896341(−1)	.19458551169345(−1)	.52514592060155(−1)	.19896260127508(−1)
.61704355074579(−1)	.27670975632577(−1)	.75041883938197(−1)	.25237021047441(−1)
.94101825972594(−1)	.37340754197965(−1)	.10312046455617	.30972186656745(−1)
.13674411641697	.48078533727343(−1)	.13705967503275	.36926743841072(−1)
.19042046600597	.59295303951325(−1)	.17698037618878	.42899978314872(−1)
.25524304731076	.70233159794501(−1)	.22279314304338	.48673450869232(−1)
.33050359479737	.80020063322645(−1)	.27418493220982	.54019907442052(−1)
.41459502770349	.87744065586239(−1)	.33061497138402	.58712748099728(−1)
.50501408859338	.92539713525291(−1)	.39132023098003	.62535654329125(−1)
.59845270059371	.93677440023386(−1)	.45533043158372	.65291965366787(−1)
.69097602175889	.90645885312278(−1)	.52149213454311	.66813403026136(−1)
.77827520264508	.83217463141567(−1)	.58850107398115	.66967770271901(−1)
.85597385719824	.71489061053272(−1)	.65494153338288	.65665290563282(−1)
.91996040353326	.55892462615736(−1)	.71933126384264	.62863311010376(−1)
.96671495434044	.37173422115917(−1)	.78017019720500	.58569160378130(−1)
.99360559333526	.16363377320650(−1)	.83599103637665	.52841030340611(−1)
		.88540971524184	.45786833033451(−1)
		.92717371952416	.37561082160968(−1)
		.96020636853232	.28359983739178(−1)
		.98364554126283	.18415501918852(−1)
		.99687876726025	.79996620858817(−2)

construction possible. The condition number of the matrix $V_n U_n^{-1} Y_n - Z_n$ is given in Figure 4.1 (right). The presented condition number of the matrix U_n suggests that we should be ready to lose certain digits of precision with which we are working. Actually, experiments show that the presented algorithm, in double precision arithmetic (REAL*8), gives exact results with 14 significant digits. Thus, the results presented in Table 4.1 (left) have all digits exact.

Another example is presented in Table 4.1 (right) that is the generalized Gaussian quadrature rule for the sequence $\lambda_{2k} = \lambda_{2k+1} = k$, $k \in \mathbb{N}_0$, or for the system of functions x^k, $x^k \log x$, $k \in \mathbb{N}_0$. In this case, the condition numbers of U_n, as well as of $V_n U_n^{-1} Y_n - Z_n$ (see Figure 4.2), are growing faster than in the previous example. The same behavior we encounter in the example for $\lambda_{3k} = \lambda_{3k+1} = \lambda_{3k+2} = k$, $k \in \mathbb{N}_0$, given in Table 4.2, with condition numbers presented in Figure 4.3. In the latter case, the condition numbers are bigger. This behavior can be understand according to Lemma 3.4. In these two cases the Λ sequence is quite different from the Λ sequence in the Legendre case. It turns out that, using the presented algorithm in double precision arithmetic (REAL*8), we can safely calculate 14 digits.

In the final example we present an application of our algorithm to a computation of the generalized Gaussian quadrature formula (1.7) in the case when the basis functions are nonintegrable. Consider the system of functions $x^{k-2/3}\ x^k$, $k \in \mathbb{N}_0$. For this system of functions our algorithm cannot be applied directly, since the sequence

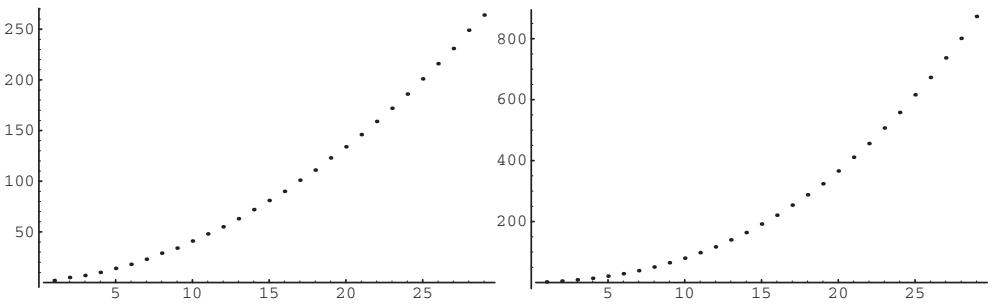


FIG. 4.2. *The condition number of the matrices* U_n *and* $V_n U_n^{-1} Y_n - Z_n$ *for* $n = 2(1)30$ *and* $\lambda_{2k} = \lambda_{2k+1} = k$, $k \in \mathbb{N}_0$.

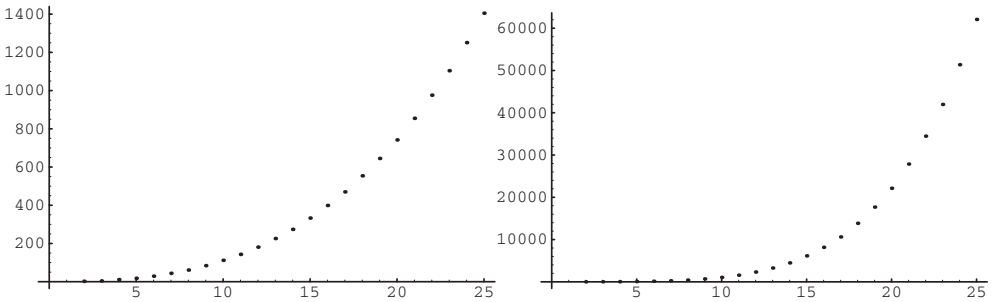


FIG. 4.3. *The condition number of the matrices* U_n *and* $V_n U_n^{-1} Y_n - Z_n$ *for* $n = 2(1)30$ *and* $\lambda_{3k} = \lambda_{3k+1} = \lambda_{3k+2} = k$, $k \in \mathbb{N}_0$.

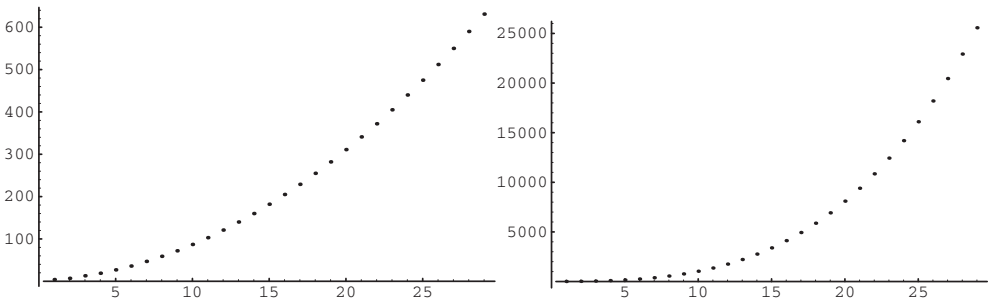


FIG. 4.4. *The condition number of the matrices* U_n *and* $V_n U_n^{-1} Y_n - Z_n$ *for* $n = 2(1)30$ *and* $\lambda_{2k} = k - 2/3$, $\lambda_{2k+1} = k$, $k \in \mathbb{N}_0$.

$\lambda_{2k} = k - 2/3$, $\lambda_{2k+1} = k$, $k \in \mathbb{N}_0$, does not satisfy the condition $\mathrm{Re}\,(\lambda_k) > -1/2$ and orthogonal Müntz–Legendre polynomials do not exist. However, we can apply Lemma 3.5. For example, we can choose $\beta = 2/3$ and apply our algorithm for constructing the generalized Gaussian quadrature rule (1.7) for the sequence $\lambda_{2k} = k - 1/3$, $\lambda_{2k+1} = k + 1/3$, $k \in \mathbb{N}_0$. Using Lemma 3.5, once our algorithm is applied, we can construct the generalized Gaussian quadrature rule for the system of functions $x^{k-2/3}$, x^k, $k \in \mathbb{N}_0$, with respect to the Legendre measure. The results for $n = 30$ nodes in the generalized Gaussian quadrature rule (1.7) are presented in Table 4.2 (right). The corresponding condition numbers are given in Figure 4.4.

TABLE 4.3

Relative errors for the classical Gauss–Legendre quadrature, with $n = 100(100)600$ *points, in evaluation of* (4.1).

n	100	200	300	400	500	600
r. err	1.2(−3)	3.0(−4)	1.3(−4)	7.4(−5)	4.7(−5)	3.3(−5)

TABLE 4.4

Relative errors for the generalized Gaussian quadrature rule, with $n = 5$, 10, *and* 15 *nodes, in evaluation of* (4.1), *for the sequence* $\lambda_{2k} = \lambda_{2k+1} = k$, $k \in \mathbb{N}_0$.

n	5	10	15
r. err	9.0(−7)	1.3(−15)	m.p.

At the end of this section we present an application of the derived quadrature formulae. We consider the integral

$$\int_0^1 J_0(x)(1+\log x)dx = -0.05310803758951187304684861869781 72\ldots, \tag{4.1}$$

where $J_0(x)$ is the Bessel function. Table 4.3 lists relative errors for the classical Gauss–Legendre quadrature rule, shifted to the interval $(0,1)$, applied to the previous integral. This is an example of the well-known saturation phenomenon (see [20]). However, using the generalized Gaussian quadrature rule derived for the sequence $\lambda_{2k} = \lambda_{2k+1} = k$, $k \in \mathbb{N}_0$, the machine precision (m.p. $\approx 2.26(-16)$) can be achieved with 15 nodes, as presented in Table 4.4.

REFERENCES

[1] I. V. ANDRONOV, *Application of orthogonal polynomials in the boundary-contact value problems,* in Proceeding VIII Simposium Sobre Polinomios Ortogonales y Sus Aplicaciones (Sevilla, Sept. 22–26, 1997), Book of Abstracts, Facultad de Matemáticas, Universidad de Sevilla, 1997, p. 32.

[2] G. V. BADALYAN, *Generalization of Legendre polynomials and some of their applications,* Akad. Nauk Armyan. SSR. Izv. Fiz.-Mat. Estest. Tehn. Nauk, 8 (1955), pp. 1–28; 9 (1956), pp. 3–22 (in Russian; Armenian summary).

[3] P. BORWEIN, T. ERDÉLYI, AND J. ZHANG, *Müntz systems and orthogonal Müntz–Legendre polynomials,* Trans. Amer. Math. Soc., 342 (1994), pp. 523–542.

[4] P. BORWEIN AND T. ERDÉLYI, *Polynomials and Polynomial Inequalities,* Graduate Texts in Math. 161, Springer-Verlag, New York, 1995.

[5] T. S. CHIHARA, *An Introduction to Orthogonal Polynomials*, Gordon and Breach, New York, 1978.

[6] W. GAUTSCHI, *A survey of Gauss-Christoffel quadrature formulae,* in E. B. Christoffel. The Influence of His Work on Mathematics and the Physical Sciences, P. L. Butzer and F. Fehér, eds., Birkhäuser, Basel, 1981, pp. 72–147.

[7] W. GAUTSCHI, *Algorithm* 726: *ORTHPOL—a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules*, ACM Trans. Math. Software, 20 (1994), pp. 21–62.

[8] W. GAUTSCHI, *Orthogonal Polynomials, Computation, and Approximation*, Oxford University Press, New York, 2004.

[9] G. GOLUB AND J. H. WELSCH, *Calculation of Gauss quadrature rules*, Math. Comp., 44 (1969), pp. 221–230.

[10] A. S. HOUSEHOLDER, *The Theory of Matrices in Numerical Analysis*, Dover, New York, 1975.

[11] C. G. HARRIS AND W. A. B. EVANS, *Extension of numerical quadrature formulae to cater for end point singular behaviours over finite intervals,* Internat. J. Comput. Math., 6 (1977), pp. 219–227.

[12] S. KARLIN AND W. J. STUDDEN, *Tchebycheff Systems with Applications in Analysis and Statistics,* Pure and Appl. Math. XV, Wiley Interscience, New York, 1966.

[13] J. Ma, V. Rokhlin, and S. Wandzura, *Generalized Gaussian quadrature rules for systems of arbitrary functions,* SIAM J. Numer. Anal., 33 (1996), pp. 971–996.
[14] A. A. Markov, *Selected Papers on Continued Fractions and the Theory of Functions Deviating Least from Zero*, OGIZ, Moscow, Leningrad, 1948 (in Russian).
[15] P. C. McCarthy, J. E. Sayre, and B. L. R. Shawyer, *Generalized Legendre polynomials*, J. Math. Anal. Appl., 177 (1993), pp. 530–537.
[16] G. V. Milovanović, *Numerical calculation of integrals involving oscillatory and singular kernels and some applications of quadratures,* Comput. Math. Appl., 36 (1998), pp. 19–39.
[17] G. V. Milovanović, *Müntz orthogonal polynomials and their numerical evaluation,* in Applications and Computation of Orthogonal Polynomials, Internat. Ser. Numer. Math. 131, W. Gautschi, G. H. Golub, and G. Opfer, eds., Birkhäuser, Basel, 1999, pp. 179–194.
[18] G. V. Milovanović, *Numerical Analysis, Part* I, 3rd ed., Naučna Knjiga, Belgrade, 1991 (in Serbo-Croatian).
[19] G. V. Milovanović and A. S. Cvetković, *Note on a construction of weights in Gauss-type quadrature rule*, Facta Univ. Ser. Math. Inform., 15 (2000), pp. 69–83.
[20] G. V. Milovanović and A. S. Cvetković, *Numerical integration of functions with logarithmic end point singularity*, Facta Univ. Ser. Math. Inform., 17 (2002), pp. 57–74.
[21] P. G. Nevai, *Orthogonal Polynomials*, Mem. Amer. Math. Soc. 18, AMS, Providence, RI, 1979.
[22] T. J. Stieltjes, *Sur une généralisation de la théorie des quadratures mécaniques,* C.R. Acad. Sci. Paris, 99 (1884), pp. 850–851.
[23] J. A. Shohat, *On a certain formula of mechanical quadratures with non-equidistant ordinates*, Trans. Amer. Math. Soc., 31 (1929), pp. 448–463.
[24] G. Szegő, *Orthogonal Polynomials,* 4th ed., Amer. Math. Soc. Colloq. Publ. 23, AMS, Providence, RI, 1975.
[25] A. K. Taslakyan, *Some properties of Legendre quasipolynomials with respect to a Müntz system,* in Mathematics, vol. 2, Èrevan University, Èrevan, 1984, pp. 179–189 (Russian, Armenian summary).
[26] C. W. Ueberhuber, *Numerical Computation* 2: *Methods, Software, and Analysis*, Springer-Verlag, Berlin, 1997.
[27] W. Van Assche, *Approximation theory and analytic number theory,* Special Functions and Differential Equations (Madras, 1997), Allied Publ., New Delhi, 1998, pp. 336–355.
[28] N. Yarvin and V. Rokhlin, *Generalized Gaussian quadratures and singular value decompositions of integral operators*, SIAM J. Sci. Comput., 20 (1998), pp. 699–718.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 914–936
© 2005 Society for Industrial and Applied Mathematics

A SEMI-IMPLICIT RELAXATION SCHEME FOR MODELING TWO-PHASE FLOW IN A PIPELINE*

MICHAËL BAUDIN†, FRÉDÉRIC COQUEL‡, AND QUANG-HUY TRAN†

Abstract. This work is devoted to the numerical approximation of the solutions to the system of conservation laws which arises in the modeling of two-phase flows in pipelines. The PDEs are closed by two highly nonlinear algebraic relations, namely, a pressure law and a hydrodynamic law. We have previously proposed an explicit relaxation scheme which allows us to cope with these nonlinearities. But the system considered has eigenvalues which are of very different orders of magnitude, which prevents the explicit scheme from being effective, since the time step has to be very small. In order to solve this effectiveness problem, we now proceed to construct a scheme which is explicit with respect to the small eigenvalues and linearly implicit with respect to the large eigenvalues. Numerical evidences are provided.

Key words. two-phase flows, relaxation scheme, semi-implicit, Riemann solvers

AMS subject classifications. 76T10, 76N15, 35L65, 65M06

DOI. 10.1137/030601624

Introduction. We are interested in the simulation of two-phase flows in petroleum pipelines by a specific kind of models called [illegible] [13]. The peculiarity of such a model lies in the existence of two algebraic closure laws, that is, a thermodynamic law and a hydrodynamic law. The complexity of these closure laws makes classical numerical schemes such as Godunov or Roe schemes very difficult to use. Our feeling is that only a "rough" scheme would be able to successfully meet the challenge of nonlinearities.

An essential property of our system is that it possesses fast characteristic speeds (acoustic waves) and slow ones (kinematic waves). From the engineer's standpoint, however, the most interesting phenomena to be observed are those associated with kinematic waves, since these reflect mass transportation. With such a purpose in mind, an explicit scheme appears not to be suitable, not only because the time required for the simulation would be prohibitive due to the large characteristic speeds, but also because a small time step would induce too much dissipation on kinematic waves.

Inspired by [9], Faille and Heintzé have proposed in [6] a VFRoe-type scheme which is sufficiently rough (diffusive) to handle very stiff industrial cases. The scheme is linearly implicit with respect to the large eigenvalues and explicit with respect to the small eigenvalues [6]. Then this semi-implicit scheme will combine (i) accuracy on kinematic waves and (ii) CPU-time reduction insofar as the CFL limitation only applies to small eigenvalues.

But this scheme also suffers from a few drawbacks. For one, it is still CPU-time consuming because it needs to numerically assess the eigenvalues of the Jacobian matrix of the system. For another, these eigenvalues are not necessarily real; that is,

*Received by the editors November 3, 2003; accepted for publication (in revised form) March 8, 2005; published electronically December 8, 2005.
http://www.siam.org/journals/sisc/27-3/60162.html
†Département Mathématiques Appliquées, Institut Français du Pétrole, 1 et 4 avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex, France (baudinm@yahoo.com, q-huy.tran@ifp.fr).
‡Laboratoire Jacques-Louis Lions, Université Pierre et Marie Curie, Boîte courrier 187, 75252 Paris Cedex 5, France (coquel@ann.jussieu.fr).

the system is not always hyperbolic. As a matter of fact, it is observed [2] that the system is hyperbolic only if the slip between the phases is not too large.

Since Xin and Jin's pioneering work [14] on relaxation schemes, several ideas have been put forward in order to improve the original relaxation procedure. For instance, Evje and Fjelde [5] attempted to achieve a more accurate approximation by splitting the flux into a convective part and a pressure part, thus reducing the stability condition on the relaxation coefficient. In [1], the authors have taken another road to construct an explicit relaxation scheme. The idea was to partially relax the most "prominent" nonlinearities of the system, namely, the hydrodynamic and thermodynamic closure laws. Those nonlinearities are first spotted by means of a Lagrangian change of coordinates. Then, these are relaxed. Finally, the equations are brought back to the Eulerian frame. The resulting relaxation system is automatically hyperbolic and has all its fields linearly degenerated. This scheme is less CPU-time consuming than the VFRoe-type because the most algorithmically complex step is the computation of only two relaxation coefficients. Moreover, the most interesting aspect of this relaxation procedure is that these coefficients can be optimally adjusted [1] in order to satisfy physical properties such as the positivity of the density and, most of all, the maximum principle on the gas mass fraction.

The goal of this paper is to work out a semi-implicit version of the explicit scheme [1] via some mathematically sound extension procedures. In essence, the extension is possible and sound because of the linear degeneracy property of the relaxation system.

The paper is organized as follows. In section 1, we introduce the two-phase flow model together with the boundary conditions and the characteristics of this model. In section 2, we develop the second-order explicit relaxation scheme with the computation of new relaxation coefficients and boundary conditions. In section 3, we present the semi-implicit scheme and section 4 is devoted to the numerical results.

1. Two-phase flow model.

1.1. Equations. In the flow, the gas (resp., liquid) is characterized by its density ρ_{G} (resp., ρ_{L}), its velocity v_{G} (resp., v_{L}), and its surface fraction $R_{\mathrm{G}} \in [0,1]$ (resp., $R_{\mathrm{L}} \in [0,1]$) with the property $R_{\mathrm{L}} + R_{\mathrm{G}} = 1$. The model is governed by the following system of conservation laws:

$$\left\{\begin{array}{lll} \partial_t(\rho_{\mathrm{L}} R_{\mathrm{L}}) & + \partial_x(\rho_{\mathrm{L}} R_{\mathrm{L}} v_{\mathrm{L}}) & = 0, \\ \partial_t(\rho_{\mathrm{G}} R_{\mathrm{G}}) & + \partial_x(\rho_{\mathrm{G}} R_{\mathrm{G}} v_{\mathrm{G}}) & = 0, \\ \partial_t(\rho_{\mathrm{L}} R_{\mathrm{L}} v_{\mathrm{L}} + \rho_{\mathrm{G}} R_{\mathrm{G}} v_{\mathrm{G}}) & + \partial_x(\rho_{\mathrm{L}} R_{\mathrm{G}} v_{\mathrm{L}}^2 + \rho_{\mathrm{G}} R_{\mathrm{G}} v_{\mathrm{G}}^2 + p) & = S(\mathbf{w}) \end{array}\right. \tag{1.1}$$

for $x \in \mathbb{R}$ and $t \geq 0$. The unknowns are $\mathbf{w} := (\rho_{\mathrm{L}} R_{\mathrm{L}}, \rho_{\mathrm{G}} R_{\mathrm{G}}, \rho_{\mathrm{L}} R_{\mathrm{L}} v_{\mathrm{L}} + \rho_{\mathrm{G}} R_{\mathrm{G}} v_{\mathrm{G}})$. Here the source term S includes gravity and wall friction terms which are given functions of the unknowns. We will consider a perfect gas and a compressible liquid with the state equations

$$\rho_{\mathrm{G}} = p/a_{\mathrm{G}}^2, \quad \rho_{\mathrm{L}} = \rho_{\mathrm{L}}^0 + (p - p^0)/a_{\mathrm{L}}^2, \tag{1.2}$$

where p^0 is the atmospheric pressure, ρ_{L}^0 is the liquid density for $p = p^0$, and a_{G} (resp., a_{L}) is the sound speed in the gas (resp., the liquid). Sometimes, we will use the noncompressible liquid state equation $\rho_{\mathrm{L}} = \rho_{\mathrm{L}}^0$. Anyhow, the pressure law can be put in the general form $p = p(\mathbf{w}_1, \mathbf{w}_2)$ and is a smooth function. We consider a general algebraic hydrodynamic law of the type

$$v_{\mathrm{L}} - v_{\mathrm{G}} = \Phi(\mathbf{w}), \quad \text{for } \mathbf{w} \in (\mathbb{R}^+)^2 \times \mathbb{R}, \tag{1.3}$$

in order to close (1.1). In (1.3), the mapping Φ is assumed to be smooth enough. In practical situations, Φ turns out to be nonlinear in the unknown $\mathbf{w}$ [2, 15].

1.2. Boundary and initial conditions. At the inlet of the pipe ($x = 0$), the mass flow rates are given as functions of time, i.e.,

$$(\rho_\alpha R_\alpha v_\alpha)(0,\ t) = q_\alpha^0(t),\ t \geq 0,\ \alpha = \mathrm{L}, \mathrm{G},$$

and at the outlet of the pipe ($x = L$) the pressure is a given function of time, i.e.,

$$p(L,\ t) = p^L(t), \quad t \geq 0.$$

We will treat cases in which the flow is induced only by variations of the boundary conditions: in such experiments, the initial condition is the steady state, computed by the values of the boundary conditions at time $t = 0$.

1.3. Characteristics of the model. There is no analytical expression for the physical flux of the considered system, except for very simple hydrodynamic laws. Therefore the eigenvalues of the system (1.1) are not known in full generality. However, in most common situations, i.e., for usual values of $\mathbf{w}$, the system is hyperbolic and has three real eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$, with $\lambda_1 < 0$ and $\lambda_3 > 0$.

Typically, $|\lambda_{1,3}| \gg |\lambda_2|$ and the large eigenvalues are 10–100 times bigger than the small eigenvalue. The extreme eigenvalues $\lambda_{1,3}$ correspond to the acoustic waves that we do not really care about, while the middle eigenvalue λ_2, which has a variable sign, corresponds to the kinematic waves that we wish to capture accurately.

2. Explicit relaxation scheme. Most computational details of this section are given in [1] and only the main results will be reported here.

2.1. The relaxation model. Let us consider the total density of the mixing $\rho = \rho_\mathrm{L} R_\mathrm{L} + \rho_\mathrm{G} R_\mathrm{G}$, the total momentum $\rho v = \rho_\mathrm{L} R_\mathrm{L} v_\mathrm{L} + \rho_\mathrm{G} R_\mathrm{G} v_\mathrm{G}$, and ρY, where Y denotes the gas mass fraction $Y = \frac{\rho_\mathrm{G} R_\mathrm{G}}{\rho}$ (so that $1 - Y = \frac{\rho_\mathrm{L} R_\mathrm{L}}{\rho}$). The natural phase space associated with such variables then reads

$$\Omega = \left\{ \mathbf{u} = (\rho,\ \rho v,\ \rho Y) \in \mathbb{R}^3;\ \rho > 0,\ v \in \mathbb{R},\ Y \in [0, 1] \right\}.$$

With some little abuse, both the pressure law p and the hydrodynamic closure Φ will keep their previous notations when expressed in terms of the new variable $\mathbf{u}$. In order to simplify the notations, let us introduce the two functions

$$\sigma(\mathbf{u}) = \rho Y(1 - Y)\Phi(\mathbf{u}), \quad P(\mathbf{u}) = p(\mathbf{u}) + \rho Y(1 - Y)\Phi(\mathbf{u})^2 \tag{2.1}$$

of the unknown $\mathbf{u}$. Weak solutions of the system (1.1)–(1.3) equivalently obey the conservative system

$$\left\{ \begin{array}{lllll} \partial_t(\rho) & + & \partial_x(\rho v) & = & 0, \\ \partial_t(\rho v) & + & \partial_x(\rho v^2 + P(\mathbf{u})) & = & S(\mathbf{u}), \\ \partial_t(\rho Y) & + & \partial_x(\rho Y v - \sigma(\mathbf{u})) & = & 0. \end{array} \right. \tag{2.2}$$

To determine the main nonlinearities of the equilibrium system (2.2), we proceed to a Lagrangian change of coordinates. Let us set $\tau = 1/\rho$, and define the Lagrangian mass coordinate by $dy = \rho dx - \rho v dt$. The Lagrangian set of variables is $\mathbf{z} = (\tau, v, Y)$. For the sake of simplicity and with some abuse in the notations, nonlinear functions

(like the pressure or the hydrodynamic law) will be given the same notation when expressed in both types of variables. The system (2.2) then rewrites as

$$\left\{\begin{array}{lll} \partial_t \tau & - \partial_y v & = 0, \\ \partial_t v & + \partial_y P(\mathbf{z}) & = S(\mathbf{z}), \\ \partial_t Y & - \partial_y \sigma(\mathbf{z}) & = 0. \end{array}\right. \tag{2.3}$$

We introduce two new state variables Σ and Π which are intended to coincide, respectively, with $\sigma(\mathbf{z})$ and $P(\mathbf{z})$ in the limit of an infinite relaxation parameter λ. We propose as a relaxation model the system

$$\left\{\begin{array}{lll} \partial_t \tau & - \partial_y v & = 0, \\ \partial_t v & + \partial_y \Pi & = S(\mathbf{u}), \\ \partial_t \Pi & + a^2 \partial_y v & = \lambda(P(\mathbf{u}) - \Pi), \\ \partial_t Y & - \partial_y \Sigma & = 0, \\ \partial_t \Sigma & - b^2 \partial_y Y & = \lambda(\sigma(\mathbf{u}) - \Sigma), \end{array}\right. \tag{2.4}$$

where a and b are two real positive parameters, the so-called relaxation coefficients to be properly prescribed to protect the relaxation procedure from instabilities in the regime $\lambda \to \infty$ (see [1] and the references therein). Returning back to Eulerian coordinates, we end up with

$$\left\{\begin{array}{lll} \partial_t \rho & + \partial_x (\rho v) & = 0, \\ \partial_t (\rho v) & + \partial_x (\rho v^2 + \Pi) & = S(\mathbf{u}), \\ \partial_t (\rho \Pi) & + \partial_x (\rho \Pi v + a^2 v) & = \lambda\rho(P(\mathbf{u}) - \Pi), \\ \partial_t (\rho Y) & + \partial_x (\rho Y v - \Sigma) & = 0, \\ \partial_t (\rho \Sigma) & + \partial_x (\rho \Sigma v - b^2 Y) & = \lambda\rho(\sigma(\mathbf{u}) - \Sigma). \end{array}\right. \tag{2.5}$$

The above relaxation system will be given hereafter the convenient abstract form

$$\partial_t \mathbf{v} + \partial_x \mathcal{G}(\mathbf{v}) = \lambda \mathcal{R}(\mathbf{v}) + \mathcal{S}(\mathbf{v}), \quad \text{for } t > 0,\ x \in \mathbb{R}, \tag{2.6}$$

in which the flux function $\mathcal{G}$ receives a clear definition and

$$\mathcal{R}(\mathbf{v}) = (0,\ 0,\ \lambda\rho\,(P(\mathbf{u}) - \Pi)\,,\ 0,\ \lambda\rho\,(\sigma(\mathbf{u}) - \Sigma))^T\,, \tag{2.7}$$

$$\mathcal{S}(\mathbf{v}) = (0,\ S(\mathbf{u}),\ 0,\ 0,\ 0)^T\,. \tag{2.8}$$

The associated admissible state space $\mathcal{V}$ reads

$$\begin{aligned} \mathcal{V} = \big\{ \mathbf{v} = (\rho,\ \rho v,\ \rho\Pi,\ \rho Y,\ \rho\Sigma)^T \in \mathbb{R}^5; \\ \rho > 0,\ v \in \mathbb{R},\ \Pi \in \mathbb{R},\ Y \in [0,1],\ \Sigma \in \mathbb{R} \big\}\,. \end{aligned} \tag{2.9}$$

The following lemma has been proven in [1].

LEMMA 2.1. (a, b) $\mathbf{v} \in \mathcal{V}$ (2.5)

$$v,\ v \pm a\tau,\ v \pm b\tau,$$

(2.5) $\mathcal{V}$

2.2. First-order explicit scheme. The pipeline is made of cells, denoted by $(M_i)_{i=1,I}$. Let x_i be the center of the cell and Δx its length. We also denote $x_{i+1/2} = (x_i + x_{i+1})/2$ the interface between two cells, $x_{1/2} = 0$ the inlet boundary interface, and $x_{I+1/2} = L$ the outlet boundary interface. Let $\Delta t^n = t^{n+1} - t^n$ be the time step. Let $\mathbf{u}_i^n \approx \mathbf{u}(x_i, t^n)$ be the discrete unknown. On the grounds of the relaxation model (2.5), we suggest the two-step procedure below to advance in time the approximate solution.

1. In order to prescribe the initial data $\mathbf{v}_i^n \simeq \mathbf{v}(x_i, t^n)$, we advocate the equilibrium condition by setting

$$\Pi_i^n := P(\mathbf{u}_i^n), \qquad \Sigma_i^n := \sigma(\mathbf{u}_i^n). \tag{2.10}$$

2. Choosing $\lambda = 0$, we solve the Cauchy problem $\partial_t \mathbf{v} + \partial_x \mathcal{G}(\mathbf{v}) = \mathcal{S}(\mathbf{v})$, $\mathbf{v}(0,x) = \mathbf{v}(t^n, x)$ for small time $t \in [0, \Delta t^n]$. This strategy leads to the finite volume scheme

$$\frac{\mathbf{v}_i^{n+1} - \mathbf{v}_i^n}{\Delta t^n} + \frac{\mathbf{H}_{i+1/2}^{n+1/2} - \mathbf{H}_{i-1/2}^{n+1/2}}{\Delta x} = \mathcal{S}(\mathbf{v}_i^n) \tag{2.11}$$

with $\mathbf{H}_{i+1/2}^{n+1/2}$ the numerical flux based on the Godunov scheme. We designate by $\mathbf{v_L} = \mathbf{v}_i^n$ and $\mathbf{v_R} = \mathbf{v}_{i+1}^n$ the left and right states of the Riemann problem and $\mathbf{v}^\star(\mathbf{v_L}, \mathbf{v_R})$ the value of the solution of this Riemann problem at $x = x_{i+1/2}$. The Godunov numerical flux is

$$\mathbf{H}_{i+1/2}^{n+1/2} = \mathbf{H}^{\mathrm{God}}(\mathbf{v_L}, \mathbf{v_R}) = \mathcal{G}(\mathbf{v}^\star(\mathbf{v_L}, \mathbf{v_R})). \tag{2.12}$$

The explicit computations for $\mathbf{v}^\star$ are given in Proposition 2.2 at the end of this section. For a proof of this proposition, see [1]. Let us underline the fact that the linear degeneracy of all fields allows for very simple expressions. For any quantity φ, we introduce the average $\overline{\varphi} = (\varphi_{\mathbf{L}} + \varphi_{\mathbf{R}})/2$ and the midjump $\langle\varphi\rangle = (\varphi_{\mathbf{L}} - \varphi_{\mathbf{R}})/2$ between the two states $\mathbf{v_L}$ and $\mathbf{v_R}$.

PROPOSITION 2.2. (a,b) (2.5)

$$\mathbf{v}(x,\,0) = \begin{cases} \mathbf{v_L} & x < 0, \\ \mathbf{v_R} & x > 0 \end{cases} \tag{2.13}$$

$$\mathbf{v}(x,t) = \begin{cases} \mathbf{v_L} = \mathbf{v}_0 & \dfrac{x}{t} < \mu_1, \\ \mathbf{v}_j & \mu_j < \dfrac{x}{t} < \mu_{j+1}, \quad 1 \le j \le 4, \\ \mathbf{v_R} = \mathbf{v}_5 & \dfrac{x}{t} > \mu_5. \end{cases} \tag{2.14}$$

$$\begin{aligned} \tau_{\mathbf{L}}^\star &= \tau_{\mathbf{L}} - \frac{\langle v\rangle}{a} + \frac{\langle\Pi\rangle}{a^2}, & \tau_{\mathbf{R}}^\star &= \tau_{\mathbf{R}} - \frac{\langle v\rangle}{a} - \frac{\langle\Pi\rangle}{a^2}, \\ v^\star &= \overline{v} + \frac{\langle\Pi\rangle}{a}, & \Pi^\star &= \overline{\Pi} + a\langle v\rangle, \\ Y^\star &= \overline{Y} - \frac{\langle\Sigma\rangle}{b}, & \Sigma^\star &= \overline{\Sigma} - b\langle Y\rangle. \end{aligned} \tag{2.15}$$

a $\tau^\star_{\mathbf{L}}$ $\tau^\star_{\mathbf{R}}$, (2.15)
(2.29)
$a > b$

$$\underbrace{v_{\mathbf{L}} - a\tau_{\mathbf{L}}}_{\mu_1(\mathbf{v_L})} \le v^\star - b\tau^\star_{\mathbf{L}} \le v^\star \le v^\star + b\tau^\star_{\mathbf{R}} \le \underbrace{v_{\mathbf{R}} + a\tau_{\mathbf{R}}}_{\mu_5(\mathbf{v_R})}, \tag{2.16}$$

$$\mathbf{v}_1 = \begin{pmatrix} \rho^\star_{\mathbf{L}} \\ \rho^\star_{\mathbf{L}} v^\star \\ \rho^\star_{\mathbf{L}} \Pi^\star \\ \rho^\star_{\mathbf{L}} Y_{\mathbf{L}} \\ \rho^\star_{\mathbf{L}} \Sigma_{\mathbf{L}} \end{pmatrix}, \ \mathbf{v}_2 = \begin{pmatrix} \rho^\star_{\mathbf{L}} \\ \rho^\star_{\mathbf{L}} v^\star \\ \rho^\star_{\mathbf{L}} \Pi^\star \\ \rho^\star_{\mathbf{L}} Y^\star \\ \rho^\star_{\mathbf{L}} \Sigma^\star \end{pmatrix}, \ \mathbf{v}_3 = \begin{pmatrix} \rho^\star_{\mathbf{R}} \\ \rho^\star_{\mathbf{R}} v^\star \\ \rho^\star_{\mathbf{R}} \Pi^\star \\ \rho^\star_{\mathbf{R}} Y^\star \\ \rho^\star_{\mathbf{R}} \Sigma^\star \end{pmatrix}, \ \mathbf{v}_4 = \begin{pmatrix} \rho^\star_{\mathbf{R}} \\ \rho^\star_{\mathbf{R}} v^\star \\ \rho^\star_{\mathbf{R}} \Pi^\star \\ \rho^\star_{\mathbf{R}} Y_{\mathbf{R}} \\ \rho^\star_{\mathbf{R}} \Sigma_{\mathbf{R}} \end{pmatrix}.$$

2.3. Relaxation coefficients. We present in this section the computation of the relaxation coefficients a and b. Our approach is based on the Chapman–Enskog expansions

$$\Pi = P + \lambda^{-1}\Pi_1 + \mathcal{O}(\lambda^{-2}), \tag{2.17}$$

$$\Sigma = \sigma + \lambda^{-1}\Sigma_1 + \mathcal{O}(\lambda^{-2}). \tag{2.18}$$

In order to get a first estimate for the relaxation coefficients (a, b), we decided to weaken the procedure introduced in [1]. This amounts to thinking of the Lagrangian relaxation system (2.4) as being made of two "uncoupled" subsystems: the first one, in (τ, v, Π), consists of the first three equations and is called the subsystem; the second one, in (Y, Σ), consists of the last two equations and is called the subsystem.

According to this decoupling strategy, we temporarily assume that the mappings P and σ depend only on the equilibrium variables involved in their respective subsystems, i.e., (τ, v) for P and Y for σ. This yields a first estimate $(a^\flat, b^\flat)$, which is then given a fairly simple correction $(a^\sharp, b^\sharp)$ so as to enforce—in the framework of general P and σ laws—two central stability properties, that is, the positivity of the density and the maximum principle on the mass fraction.

2.3.1. Two stability theorems. Let us consider the first subsystem

$$\begin{cases} \partial_t \tau - \partial_y v = 0, \\ \partial_t v + \partial_y \Pi = 0, \\ \partial_t \Pi + a^2 \partial_y v = \lambda(P - \Pi). \end{cases} \tag{2.19}$$

THEOREM 2.3. $P = P(\tau, v)$ (2.19)

$$\partial_t \mathbf{w}^1 + \Gamma^1(\mathbf{w}^1)\partial_y \mathbf{w}^1 = \lambda^{-1}\partial_y \left[\Delta_a(\mathbf{w}^1)\partial_y \mathbf{w}^1\right], \tag{2.20}$$

$\mathbf{w}^1 = (\tau, v)^T$

$$\Delta_a(\mathbf{w}^1) = \begin{pmatrix} 0 & 0 \\ E & a^2 - A \end{pmatrix} \quad E = -P_v P_\tau, \ \ A = -P_\tau + P_v^2. \tag{2.21}$$

2.3. Substituting (2.17) into (2.19), we end up with the first-order asymptotic system

$$\left\{ \begin{array}{l} \partial_t \tau - \partial_y v = 0, \\ \partial_t v + \partial_y P = -\lambda^{-1} \partial_y \Pi_1, \end{array} \right. \tag{2.22}$$

where the first-order corrector Π_1 reads $\Pi_1 = -(\partial_t P + a^2 \partial_y v)$. Let us express the time derivative Π_1 by means of space derivatives. To do this, we start from the hypothesis $P = P(\tau, v)$ in order to have the identity $\partial_t P = (P_\tau \quad P_v)\, \partial_t \mathbf{w}^1$. Second, the equilibrium system (2.3) gives $\partial_t \mathbf{w}^1 = \partial_y (v \quad -P)^T$. Third, we differentiate P using the classical chain rule, which yields

$$\partial_y (v \quad -P)^T = \begin{pmatrix} 0 & 1 \\ -P_\tau & -P_v \end{pmatrix} \partial_y \mathbf{w}^1. \tag{2.23}$$

Finally, the combination of the last three identities leads to $-\Pi_1 = \partial_t P + a^2 \partial_y \mathbf{w}^1 = \begin{pmatrix} E & a^2 - A \end{pmatrix} \partial_y \mathbf{w}^1$, which completes the proof. ∎

Let us consider the second subsystem

$$\left\{ \begin{array}{l} \partial_t Y - \partial_y \Sigma = 0, \\ \partial_t \Sigma - b^2\, \partial_y Y = \lambda(\sigma - \Sigma). \end{array} \right. \tag{2.24}$$

THEOREM 2.4. $\sigma = \sigma(Y)$ (2.24)

$$\partial_t Y - \partial_y \sigma = \lambda^{-1} \partial_y \left[\Delta_b(Y) \partial_y Y \right], \tag{2.25}$$

$$\Delta_b(Y) = b^2 - B \qquad B = \sigma_Y^2. \tag{2.26}$$

2.4. Substituting (2.18) into (2.24), we obtain

$$\partial_t Y - \partial_y \sigma = \lambda^{-1} \partial_y \Sigma_1, \tag{2.27}$$

where the first-order corrector Σ_1 reads $\Sigma_1 = -(\partial_t v - b^2 \partial_y Y)$. The hypothesis and the equilibrium system (2.3) give easily $\partial_t \sigma = B \partial_y Y$. This implies $-\Sigma_1 = (B - b^2) \partial_y Y$, which completes the proof. ∎

Then, requiring that the second-order operator of the asymptotic systems (2.22) and (2.25) be dissipative (so as to heuristically ensure stability), we are led to the following conditions.

COROLLARY 2.5. $\Delta_a(w)$ $\Delta_b(Y)$ $a^2 - A \geq 0$ $b^2 - B \geq 0$

REMARK 2.6. P σ $a > b$ a b

2.3.2. Practical computation of the relaxation coefficients. We present in this section the computation of the relaxation coefficients a and b for general pressure and slip laws. We consider a local pair $(a_{i+1/2},\, b_{i+1/2})$ on each interface $x_{i+1/2}$ in order to minimize the numerical dissipation. The relaxation coefficients are designed in order to ensure the following.

1. The stability of the first-order asymptotic equilibrium system thanks to the Chapman–Enskog expansion. On the grounds of Corollary 2.5, we opt for the following first estimates:

$$\begin{aligned} a^{\flat}_{i+1/2} &= \sqrt{\max(A(\mathbf{u}_i), A(\mathbf{u}_{i+1}))}, \qquad A(\mathbf{u}) = -P_\tau(\mathbf{u}) + P_v^2(\mathbf{u}), \\ b^{\flat}_{i+1/2} &= \sqrt{\max(B(\mathbf{u}_i), B(\mathbf{u}_{i+1}))}, \qquad B(\mathbf{u}) = \sigma_Y^2(\mathbf{u}). \end{aligned} \tag{2.28}$$

2. The positivity of the density and the mass fractions. As explained in [1], we have to correct the above first estimates in order to comply with two additional but fundamental physical properties: the positivity of the density and, above all, the maximum principle on the gas mass fraction, i.e., $Y_i^n \in [0, 1]$. This is done by setting [1]

(2.29)

$$a^{\sharp}_{i+1/2} = \frac{(v_i - v_{i+1}) + \sqrt{(v_i - v_{i+1})^2 + 8\max(\tau_i, \tau_{i+1})|\Pi_i - \Pi_{i+1}|}}{4\min(\tau_i, \tau_{i+1})},$$

$$b^{\sharp}_{i+1/2} = \frac{|\rho_i\Phi(\mathbf{u}_i) + \rho_{i+1}\Phi(\mathbf{u}_{i+1})|}{2} + \frac{|\rho_i\Phi(\mathbf{u}_i) - \rho_{i+1}\Phi(\mathbf{u}_{i+1})|}{2}$$

and by finally requiring that

$$a_{i+1/2} \geq \max(a^{\flat}_{i+1/2}, a^{\sharp}_{i+1/2}) \quad \text{and} \quad b_{i+1/2} \geq \max(b^{\flat}_{i+1/2}, b^{\sharp}_{i+1/2}). \tag{2.30}$$

The lower bounds (2.29) to be met by the relaxation coefficients have been proven to be almost optimal in [1]: a failure to satisfy those conditions generally results in a violation of our positivity requirements. Again, the bounds are valid for general pressure and slip laws.

The above strategy, simpler than the one in [1], has been discovered in a pragmatic way, considering a numerical experiment (a contact discontinuity) with real-life slip laws in which the old fashion [1] led to overestimated coefficients $b_{i+1/2}$. The resulting numerical solutions were very dissipated and therefore unacceptable, as shown in Appendix A, which the reader is referred to for more details. The new procedure for deriving the relaxation coefficients significantly lowers the numerical dissipation while still achieving stability properties.

2.4. Second order in space. The scheme is extended to second-order accuracy in space by using the classical MUSCL (monotonic upstream scheme for conservation laws) technique [11, 12]. The limited slopes are those of the "physical variable" (p, Y, v) rather than those of the conservative variable $\mathbf{u}$ [6, 7].

2.5. Second order in time. We use the following Runge–Kutta second-order procedure. Let us suppose that we denote $\mathbb{L}$ the explicit operator, which is second order in space, first order in time so that $\mathbf{u}^{n+1} = \mathbb{L}(\mathbf{u}^n)$. Then the explicit second order both in space and time is

$$\mathbf{u}^{(1)} = \mathbb{L}(\mathbf{u}^n), \quad \mathbf{u}^{(2)} = \mathbb{L}(\mathbf{u}^{(1)}), \quad \mathbf{u}^{n+1} = \frac{1}{2}(\mathbf{u}^n + \mathbf{u}^{(2)}). \tag{2.31}$$

2.6. Time step. The time step of the relaxation explicit scheme is limited by the CFL condition

$$\frac{\Delta t^n}{\Delta x} \max_{i=1,I} |V_i^n| \leq \mu, \tag{2.32}$$

where $\mu = 0.5$ is the CFL and V_i^n is a characteristic speed given—under the natural ordering (2.16)—by

$$V_i^n = \max(|v_{i-1/2}^{\mathbf{R}} + a_{i-1/2}\tau_{i-1/2}^{\mathbf{R}}|,\ |v_{i+1/2}^{\mathbf{L}} - a_{i+1/2}\tau_{i+1/2}^{\mathbf{L}}|). \tag{2.33}$$

This condition enables us to obtain a sequence of noninteracting Riemann problems located at each cell interface $x_{i+1/2}$.

2.7. Treatment of boundary conditions. In order to take the boundary conditions into account, we introduce two ghost cells: $\mathbf{u}_0^n$ at the inlet of the pipe and $\mathbf{u}_{I+1}^n$ at the outlet. The computation of these two states must be explicit, due to the specific implementation of the linearly implicit scheme.

It is well known that the number of scalar physical conditions to specify on the boundaries is equal to the number of entering characteristics [2]. We have already noticed that our system is conditionally hyperbolic and, for most cases, admits three real eigenvalues $\lambda_1 < \lambda_2 < \lambda_3$, with $\lambda_1 < 0$ and $\lambda_3 > 0$. Moreover, [illegible] at the inlet, $\lambda_2 \geq 0$. Since our system is of dimension 3, this justifies the two physical conditions (q_{L} and q_{G}) at the inlet and just one physical condition (the pressure) at the outlet.

2.7.1. Inlet ($x = 0$). We compute the state $\mathbf{v}_0^n$ in order to satisfy the five following conditions. First, the relaxation state $\mathbf{v}_0^n$ is at equilibrium, which leads to the two identities (2.10). Second, we specify the total mass flow rate (which corresponds to the first component ρv) and the gas mass flow rate (which corresponds to the fourth component $\rho Y v - \Sigma$) which leads to the two identities

$$\mathbf{H}^1(\mathbf{v}_0^n,\ \mathbf{v}_1^n) = q_{\mathrm{L}}^0(t^n) + q_{\mathrm{G}}(t^n), \quad \mathbf{H}^4(\mathbf{v}_0^n,\ \mathbf{v}_1^n) = q_{\mathrm{G}}^0(t^n). \tag{2.34}$$

Third, let us introduce ℓ_1, the left eigenvector corresponding to the first eigenvalue of $\nabla\mathcal{G}(\mathbf{u}_1^n)$. We then specify the characteristic projection

$$\ell_1.\mathbf{v}_0^n = \ell_1.\mathbf{v}_1^n. \tag{2.35}$$

The nonlinear system made up of the five equations (2.10), (2.34), and (2.35) is solved by a classical nonlinear solver (for instance, Newton's method).

2.7.2. Outlet ($x = L$). We compute the state $\mathbf{v}_{I+1}^n$ in order to satisfy the five following conditions. First, the relaxation state $\mathbf{v}_{I+1}^n$ is at equilibrium, which leads to the two identities (2.10). Second, we specify the pressure, which leads to the identity

$$p(\mathbf{u}_{I+1}^n) = p^L(t^n). \tag{2.36}$$

Third, we could have chosen, as at the inlet, two characteristic projections based on the left eigenvectors corresponding to the fourth and fifth eigenvalues of $\nabla\mathcal{G}(\mathbf{u}_I^n)$. But the experience (based on numerous numerical experiments) shows that it suffices to write the identities

$$Y_{I+1}^n = Y_I^n, \quad v_{I+1}^n = v_I^n. \tag{2.37}$$

The nonlinear system made up of the five equations (2.10), (2.36), and (2.37) is solved exactly.

The no-return condition. On offshore petroleum installations, the mixture coming out of a vertical pipe ("riser") usually falls into a separator whose goal is to separate the liquid and gas phases. This is why we would like to enforce the additional condition: the liquid flow rate on the outlet must remain positive. Note that this condition is independent of the change of sign of whatever eigenvalue. The numerical treatment that we have chosen is therefore the following. After having computed the ghost state $\mathbf{v}_{I+1}^n$ satisfying the previous conditions, we compute the numerical flux on the last interface $\mathbf{H}_{I+1/2} = \mathbf{H}(\mathbf{v}_{I+1}^n,\, \mathbf{v}_{I+1}^n)$. We then modify the first component of the flux by

$$\overline{\mathbf{H}}_{I+1/2}^1 = \begin{cases} \mathbf{H}_{I+1/2}^3 & \text{if} \quad \mathbf{H}_{I+1/2}^1 - \mathbf{H}_{I+1/2}^3 \leq 0, \\ \mathbf{H}_{I+1/2}^1 & \text{otherwise.} \end{cases} \tag{2.38}$$

This type of boundary condition is analyzed in [8].

3. Semi-implicit relaxation scheme.

3.1. First-order linearly implicit scheme.

3.1.1. The linear system. Let us consider the five-equation relaxation hyperbolic system (2.4)

$$\partial_t \mathbf{v} + \partial_x \mathcal{G}(\mathbf{v}) = \mathcal{S}(\mathbf{v}). \tag{3.1}$$

In the previous section, we have constructed the explicit scheme

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n - \frac{\Delta t^n}{\Delta x} \left(\mathbf{H}(\mathbf{v}_i^n,\, \mathbf{v}_{i+1}^n) - \mathbf{H}(\mathbf{v}_{i-1}^n,\, \mathbf{v}_i^n) \right) + \Delta t^n\, \mathcal{S}(\mathbf{v}_i^n), \tag{3.2}$$

where the right-hand side contains only the approximate solution at time t^n. Such a time discretization implies that the time step will be sufficiently small. A classical technique to get rid of this limitation is to use the implicit time discretization

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n - \frac{\Delta t^n}{\Delta x} \left(\mathbf{H}\left(\mathbf{v}_i^{n+1},\, \mathbf{v}_{i+1}^{n+1}\right) - \mathbf{H}\left(\mathbf{v}_{i-1}^{n+1},\, \mathbf{v}_i^{n+1}\right) \right) + \Delta t^n\, \mathcal{S}\left(\mathbf{v}_i^{n+1}\right), \tag{3.3}$$

where the computation of the approximate solution at time t^{n+1} is based on the inversion of a nonlinear problem. This inversion is expensive in terms of CPU time. On the other hand, numerical experiments show that the scheme is stable with just one Newton iteration with a far less restrictive CFL condition. Therefore we choose this strategy which gives a [illegible] scheme for which the flux $\mathbf{H}$ and the source term $\mathcal{S}$ are approximated by the first-order Taylor expansions

$$\mathbf{H}\left(\mathbf{v}_i^{n+1},\, \mathbf{v}_{i+1}^{n+1}\right) \approx \mathbf{H}\left(\mathbf{v}_i^n,\, \mathbf{v}_{i+1}^n\right) + \partial_1 \mathbf{H}\, \delta \mathbf{v}_i + \partial_2 \mathbf{H}\, \delta \mathbf{v}_{i+1}, \tag{3.4}$$

$$\mathcal{S}\left(\mathbf{v}_i^{n+1}\right) \approx \mathcal{S}\left(\mathbf{v}_i^n\right) + \nabla \mathcal{S}_i\, \delta \mathbf{v}_i, \tag{3.5}$$

where $\delta \mathbf{v}_i = \mathbf{v}_i^{n+1} - \mathbf{v}_i^n$; the partial derivatives of $\mathbf{H}$, which are designated by $\partial_1 \mathbf{H}$ and $\partial_2 \mathbf{H}$, are evaluated at $(\mathbf{v}_i^n,\, \mathbf{v}_{i+1}^n)$; and the Jacobian matrix $\mathcal{S}$ is evaluated at $\mathbf{v}_i^n$. Let us denote

$$\alpha_i = \frac{\Delta t^n}{\Delta x} \partial_1 \mathbf{H}(\mathbf{v}_i^n,\, \mathbf{v}_{i+1}^n) \quad \text{and} \quad \gamma_{i+1} = \frac{\Delta t^n}{\Delta x} \partial_2 \mathbf{H}(\mathbf{v}_i^n,\, \mathbf{v}_{i+1}^n) \tag{3.6}$$

the partial derivatives of $\mathbf{H}$. Plugging (3.4) and (3.5) into (3.3), one obtains the numerical scheme

$$\begin{aligned}\mathbf{v}_i^{n+1} = \mathbf{v}_i^n - \frac{\Delta t^n}{\Delta x}\left(\mathbf{H}(\mathbf{v}_i^n,\ \mathbf{v}_{i+1}^n) - \mathbf{H}(\mathbf{v}_{i-1}^n,\ \mathbf{v}_i^n)\right) + \Delta t^n\ \mathcal{S}(\mathbf{v}_i^n)\\ - (\alpha_i\delta\mathbf{v}_i + \gamma_{i+1}\delta\mathbf{v}_{i+1}) + (\alpha_{i-1}\delta\mathbf{v}_{i-1} + \gamma_i\delta\mathbf{v}_i) + \Delta t^n\ \nabla\mathcal{S}_i\ \delta\mathbf{v}_i.\end{aligned} \tag{3.7}$$

The first three terms of the right-hand side of (3.7) correspond to the explicit scheme. This is why the linearly implicit scheme is classically split into three steps.

1. In the physical step, one computes a "predictor" using the explicit scheme

$$\mathbf{v}_i^\sharp = \mathbf{v}_i^n - \frac{\Delta t^n}{\Delta x}\left(\mathbf{H}(\mathbf{v}_i^n, \mathbf{v}_{i+1}^n) - \mathbf{H}(\mathbf{v}_{i-1}^n, \mathbf{v}_i^n)\right) + \Delta t^n\ \mathcal{S}(\mathbf{v}_i^n). \tag{3.8}$$

2. In the mathematical step, one solves the linear equation in $\delta\mathbf{v}$,

$$-\alpha_{i-1}\delta\mathbf{v}_{i-1} + \beta_i\delta\mathbf{v}_i + \gamma_{i+1}\delta\mathbf{v}_{i+1} = \mathbf{v}_i^\sharp - \mathbf{v}_i^n, \tag{3.9}$$

   where β_i is the matrix $\beta_i = \mathbb{I} + \alpha_i - \gamma_i - \Delta t^n\ \nabla\mathcal{S}_i$ with $\mathbb{I}$ the 5×5 identity matrix. The mathematical step involves a nonsymmetric tridiagonal by block matrix which is solved by a classical linear solver.
3. Now we can update the conservative variable of the cell i by

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \delta\mathbf{v}_i. \tag{3.10}$$

Now, the remaining question is: how to construct the matrices α and γ? It is easy [6] to compute these matrices from the definition (3.6) when the explicit scheme is a Roe-type scheme. The main objective of the next section is therefore to put the relaxation explicit scheme under Roe's form. Afterwards, we will give the explicit computations of the matrices α and γ for the linearly implicit relaxation scheme.

3.1.2. Roe's form of the explicit relaxation scheme. We consider the Riemann problem for the relaxation system (2.5) with $\lambda = 0$, no source terms, and the left and right states $\mathbf{v}^{\mathbf{L}} = \mathbf{v}_0$ and $\mathbf{v}^{\mathbf{R}} = \mathbf{v}_5$. Let $(\mathbf{v}_j)_{j=1,4}$ be the intermediate states and $(\mu_j)_{j=1,5}$ the eigenvalues defined in Proposition 2.2. It is interesting to compare the following results with the one obtained by Godlewski and Raviart [10, Chap. III, section 3.4] while considering the gas dynamic system and rewriting Godunov's method in terms of a Roe-type linearization. Such a rewriting is based on a shock curve decomposition and is possible when the solution of the Riemann problem is only made of discontinuities (see also Dubois and Mehlmann [4]).

LEMMA 3.1. [illegible]

$$\mathbf{v}_j - \mathbf{v}_{j-1} = \delta_j r_j, \quad j = 1, 5, \tag{3.11}$$

[illegible] δ_j [illegible]

$$\begin{aligned}&\delta_1 = \rho_{\mathbf{L}}^\star - \rho_{\mathbf{L}}, && \delta_2 = \rho_{\mathbf{L}}^\star(Y^\star - Y_{\mathbf{L}}), && \delta_3 = \rho_{\mathbf{R}}^\star - \rho_{\mathbf{L}}^\star,\\ &\delta_4 = \rho_{\mathbf{R}}^\star(Y_{\mathbf{R}} - Y^\star), && \delta_5 = \rho_{\mathbf{R}} - \rho_{\mathbf{R}}^\star,\end{aligned}$$

$R = (r_j)_{j=1,5}$ [illegible]

$$R = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ v_{\mathbf{L}} - a\tau_{\mathbf{L}} & 0 & v^\star & 0 & v_{\mathbf{R}} + a\tau_{\mathbf{R}} \\ \Pi_{\mathbf{L}} + a^2\tau_{\mathbf{L}} & 0 & \Pi^\star & 0 & \Pi_{\mathbf{R}} + a^2\tau_{\mathbf{R}} \\ Y_{\mathbf{L}} & 1 & Y^\star & 1 & Y_{\mathbf{R}} \\ \Sigma_{\mathbf{L}} & b & \Sigma^\star & -b & \Sigma_{\mathbf{R}} \end{pmatrix}. \tag{3.12}$$

With this lemma, it is easy to construct Roe's matrix.

PROPOSITION 3.2. [illegible] A^{Roe} [illegible] $A^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R}) = R\Lambda L$ [illegible] $\Lambda = \mathrm{Diag}(\mu_j)_{j=1,5}$ [illegible] (2.16) R [illegible] (3.12) [illegible] $L = R^{-1}$ [illegible] A^{Roe} [illegible]

The right eigenvectors of A^{Roe} are obviously the vectors $(r_j)_{j=1,5}$, since

$$A^{\mathrm{Roe}} r_j = \mu_j r_j, \quad j = 1, 5. \tag{3.13}$$

Let us notice that for $\mathbf{v^L} \neq \mathbf{v^R}$, the Roe matrix $A^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R})$ is not the value of the Jacobian of the relaxed flux $\mathcal{G}$ at an "average" state. We define the absolute value of the Roe matrix by $|A^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R})| = R|\Lambda|L$. The Roe matrix allows us to construct Roe's numerical scheme and to conclude, thanks to the forthcoming theorem.

THEOREM 3.3. [illegible]

$$\mathbf{H}^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R}) = \frac{1}{2}\left(\mathcal{G}(\mathbf{v^L}) + \mathcal{G}(\mathbf{v^R})\right) - \frac{1}{2}|A^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R})|(\mathbf{v^R} - \mathbf{v^L}) \tag{3.14}$$

[illegible]

$$\mathbf{H}^{\mathrm{Roe}}(\mathbf{v^L}, \mathbf{v^R}) = \mathbf{H}^{\mathrm{God}}(\mathbf{v^L}, \mathbf{v^R}), \tag{3.15}$$

[illegible] $\mathbf{H}^{\mathrm{God}}$ [illegible] (2.12)

Such a scheme can be constructed when all the fields of the relaxation system are linearly degenerated, which is our case, or when the solution of the Riemann problem is only made up of shocks.

[illegible] 3.1. The main ideas involved in the following computations are the use of the Rankine–Hugoniot condition and the conservation of an appropriate Riemann invariant. The proof of the lemma is composed of the proof of the five equalities (3.11), as each of these requires a specific treatment.

In a first step, we show that $\mathbf{v}_1 - \mathbf{v}_0 = \delta_1 r_1$. By definition,

$$\begin{aligned} \mathbf{v}_1 - \mathbf{v}_0 = (&\rho^\star_{\mathbf{L}} - \rho_{\mathbf{L}},\ \rho^\star_{\mathbf{L}} v^\star - \rho_{\mathbf{L}} v_{\mathbf{L}},\ \rho^\star_{\mathbf{L}} \Pi^\star - \rho_{\mathbf{L}} \Pi_{\mathbf{L}}, \\ &(\rho^\star_{\mathbf{L}} - \rho_{\mathbf{L}}) Y_{\mathbf{L}},\ (\rho^\star_{\mathbf{L}} - \rho_{\mathbf{L}}) \Sigma_{\mathbf{L}})^T. \end{aligned} \tag{3.16}$$

First, the Rankine–Hugoniot condition applied on the Eulerian equation $\partial_t \rho + \partial_x(\rho v) = 0$ with the first discontinuity gives

$$\rho^\star_{\mathbf{L}} v^\star - \rho_{\mathbf{L}} v_{\mathbf{L}} = (v_{\mathbf{L}} - a\tau_{\mathbf{L}})(\rho^\star_{\mathbf{L}} - \rho_{\mathbf{L}}). \tag{3.17}$$

Second, in Lagrangian coordinates, $\mathcal{I} = \Pi + a^2\tau$ is a Riemann invariant for the characteristic speed 0, i.e., $\partial_t \mathcal{I} = 0$. This is a straightforward consequence of the two Lagrangian conservation laws $\partial_t \tau - \partial_y v = 0$ and $\partial_t \Pi + a^2 \partial_y v = 0$. Back in Eulerian

coordinates, $\mathcal{I}$ is a Riemann invariant for the characteristic speed v. A direct application is $\Pi^\star + a^2\tau_{\mathbf{L}}^\star = \Pi_{\mathbf{L}} + a^2\tau_{\mathbf{L}}$, which implies $\rho_{\mathbf{L}}^\star\left(\Pi^\star + a^2\tau_{\mathbf{L}}^\star\right) - \rho_{\mathbf{L}}\left(\Pi_{\mathbf{L}} + a^2\tau_{\mathbf{L}}\right) = \left(\rho_{\mathbf{L}}^\star - \rho_{\mathbf{L}}\right)\left(\Pi_{\mathbf{L}} + a^2\tau_{\mathbf{L}}\right)$. The expansion of the left-hand side of the last equality gives

$$\rho_{\mathbf{L}}^\star\Pi^\star - \rho_{\mathbf{L}}\Pi_{\mathbf{L}} = \left(\rho_{\mathbf{L}}^\star - \rho_{\mathbf{L}}\right)\left(\Pi_{\mathbf{L}} + a^2\tau_{\mathbf{L}}\right) \tag{3.18}$$

because $\rho_{\mathbf{L}}^\star \times (a^2\tau_{\mathbf{L}}^\star) - \rho_{\mathbf{L}} \times (a^2\tau_{\mathbf{L}}) = 0$. Substituting equalities (3.17) and (3.18) into the jump (3.16) and using the definitions of δ_1 and r_1 gives $\mathbf{v}_1 - \mathbf{v}_0 = \delta_1 r_1$.

In a second step, the definition of the intermediate states gives

$$\mathbf{v}_2 - \mathbf{v}_1 = \left[0,\ ,\ 0,\ 0,\ \rho_{\mathbf{L}}^\star\left(Y^\star - Y_{\mathbf{L}}\right),\ \rho_{\mathbf{L}}^\star\left(\Sigma^\star - \Sigma_{\mathbf{L}}\right)\right]^T. \tag{3.19}$$

Here, we use the Lagrangian equation $\partial_t Y - \partial_y \Sigma = 0$. We then apply the Rankine–Hugoniot condition with the second discontinuity, which gives $-(\Sigma^\star - \Sigma_{\mathbf{L}}) = -b\,(Y^\star - Y_{\mathbf{L}})$. Multiplying each side by $\rho_{\mathbf{L}}^\star$, one easily obtains

$$\rho_{\mathbf{L}}^\star(\Sigma^\star - \Sigma_{\mathbf{L}}) = b\,\rho_{\mathbf{L}}^\star(Y^\star - Y_{\mathbf{L}}). \tag{3.20}$$

Substituting the equality (3.20) into the jump (3.19) and using the definitions of δ_2 and r_2 gives $\mathbf{v}_2 - \mathbf{v}_1 = \delta_2 r_2$.

In a third step, the definitions of δ_3 and r_3 give directly $\mathbf{v}_3 - \mathbf{v}_2 = \delta_3 r_3$. Similarly, one can prove that $\mathbf{v}_4 - \mathbf{v}_3 = \delta_4 r_4$ and $\mathbf{v}_5 - \mathbf{v}_4 = \delta_5 r_5$, which completes the proof. ∎

3.2. We must show that the matrix A^{Roe} satisfies the three Roe conditions. This will be done by using the definition of this matrix and some easy computations, as we are going to see.

1. By definition, the matrix $A^{\mathrm{Roe}}(\mathbf{v}^{\mathbf{L}}, \mathbf{v}^{\mathbf{R}})$ is diagonalizable with real eigenvalues $(\mu_j)_{j=1,5}$. Its right (resp., left) eigenvectors are $(r_j)_{j=1,5}$, the columns of R (resp., $(l_j)_{j=1,5}$, the lines of L). The five right eigenvectors form a basis of $\mathbb{R}^5$.
2. An easy but tedious calculation enables us to check that $A^{\mathrm{Roe}}(\mathbf{v}, \mathbf{v}) = \nabla\mathcal{G}(\mathbf{v})$.
3. Let us now prove that $A^{\mathrm{Roe}}(\mathbf{v}^{\mathbf{L}}, \mathbf{v}^{\mathbf{R}})(\mathbf{v}^{\mathbf{R}} - \mathbf{v}^{\mathbf{L}}) = \mathcal{G}(\mathbf{v}^{\mathbf{R}}) - \mathcal{G}(\mathbf{v}^{\mathbf{L}})$. First, one can notice that, by definition (3.11), $\mathbf{v}^{\mathbf{L}} - \mathbf{v}^{\mathbf{R}} = \sum_{j=1,5}(\mathbf{v}_j - \mathbf{v}_{j-1}) = \sum_{j=1,5}\delta_j r_j$. It is therefore easy to see that, thanks to (3.13) and (3.11),

$$A^{\mathrm{Roe}}(\mathbf{v}^{\mathbf{L}},\ \mathbf{v}^{\mathbf{R}})(\mathbf{v}^{\mathbf{L}} - \mathbf{v}^{\mathbf{R}}) = \sum_{j=1,5}\delta_j\mu_j r_j = \sum_{j=1,5}\mu_j(\mathbf{v}_j - \mathbf{v}_{j-1}).$$

On the other hand, the Rankine–Hugoniot condition gives the jump through the jth discontinuity as

$$\mathcal{G}(\mathbf{v}_j) - \mathcal{G}(\mathbf{v}_{j-1}) = \mu_j(\mathbf{v}_j - \mathbf{v}_{j-1}), \quad j = 1, \ldots, 5. \tag{3.21}$$

It follows that

$$A^{\mathrm{Roe}}(\mathbf{v}^{\mathbf{L}},\ \mathbf{v}^{\mathbf{R}})(\mathbf{v}^{\mathbf{L}} - \mathbf{v}^{\mathbf{R}}) = \sum_{j=1,5}\mathcal{G}(\mathbf{v}_j) - \mathcal{G}(\mathbf{v}_{j-1}) = \mathcal{G}(\mathbf{v}^{\mathbf{L}}) - \mathcal{G}(\mathbf{v}^{\mathbf{R}}),$$

which is the desired result. ∎

3.3. We use the two equalities [11, Chap. 3, Lemma 4.3]

$$\frac{2}{\Delta x}\int_0^{\Delta x/2}\mathbf{v}\left(\frac{x}{\Delta t^n}\right)dx = \mathbf{v}_5 - 2\frac{\Delta t^n}{\Delta x}\left(\mathcal{G}(\mathbf{v}_5) - \mathbf{H}(\mathbf{v}_0,\ \mathbf{v}_5)\right), \tag{3.22}$$

$$\frac{2}{\Delta x}\int_{-\Delta x/2}^{0}\mathbf{v}\left(\frac{x}{\Delta t^n}\right)dx = \mathbf{v}_0 - 2\frac{\Delta t^n}{\Delta x}\left(\mathbf{H}(\mathbf{v}_0,\ \mathbf{v}_5) - \mathcal{G}(\mathbf{v}_0)\right), \tag{3.23}$$

with Δt^n satisfying the CFL condition (2.32). One denotes $\mu_j^+ = \max(\mu_j, 0)$ and $\mu_j^- = \min(\mu_j, 0)$ for $j = 1, 5$. The main idea of the proof is to compute explicitly the left-hand sides of (3.22)–(3.23) and to obtain two distinct expressions for the Godunov numerical flux. As the CFL condition is satisfied, we are sure that $\mu_5^+ \Delta t \leq \frac{\Delta x}{2}$ and $\mu_1^- \Delta t \geq -\frac{\Delta x}{2}$, which justify such computations. The half-sum of these equalities will give a flux involving the eigenvalues and eigenvectors of the Roe matrix, and that will easily be transformed into the Roe numerical flux (3.14).

We want to compute the left-hand side of (3.22) by integrating the solution of the Riemann problem on the first domain $[0, \Delta x/2] \times [0, \Delta t^n]$ so as to get

$$\begin{aligned}\int_0^{\Delta x/2} \mathbf{v}\left(\frac{x}{\Delta t^n}\right) dx = \mathbf{v}_0 \mu_1^+ \Delta t^n + \mathbf{v}_1(\mu_2^+ - \mu_1^+)\Delta t^n + \cdots \\ + \mathbf{v}_4(\mu_5^+ - \mu_4^+)\Delta t^n + \mathbf{v}_5\left(\frac{\Delta x}{2} - \mu_5^+ \Delta t^n\right).\end{aligned} \tag{3.24}$$

This process leads to

$$\frac{2}{\Delta x}\int_0^{\Delta x/2} \mathbf{v}\left(\frac{x}{\Delta t^n}\right) dx = 2\frac{\Delta t^n}{\Delta x}\sum_{j=1,5}(\mathbf{v}_{j-1} - \mathbf{v}_j)\mu_j^+ + \mathbf{v}_5.$$

By comparison with (3.22), we infer that

$$\mathbf{H}(\mathbf{v}_0,\ \mathbf{v}_5) = \mathcal{G}(\mathbf{v}_5) - \sum_{j=1,5}(\mathbf{v}_j - \mathbf{v}_{j-1})\mu_j^+. \tag{3.25}$$

Similarly, we compute the left-hand side of (3.23) by integrating the solution of the Riemann problem on the second domain $[-\Delta x/2, 0] \times [0, \Delta t^n]$ to get

$$\mathbf{H}(\mathbf{v}_0, \mathbf{v}_5) = \mathcal{G}(\mathbf{v}_0) + \sum_{j=1,5}(\mathbf{v}_j - \mathbf{v}_{j-1})\mu_j^-. \tag{3.26}$$

The equality $|\mu| = \mu^+ - \mu^-$ combined with the half-sum of (3.25) and (3.26) gives

$$\mathbf{H}(\mathbf{v}_0, \mathbf{v}_5) = \frac{1}{2}(\mathcal{G}(\mathbf{v}_0) + \mathcal{G}(\mathbf{v}_5)) - \frac{1}{2}\sum_{j=1,5}|\mu_j|(\mathbf{v}_j - \mathbf{v}_{j-1}). \tag{3.27}$$

But we have already noticed that $\mathbf{v}_5 - \mathbf{v}_0 = \sum_{j=1,5}\delta_j r_j$ and therefore

$$\begin{aligned}|A^{\mathrm{Roe}}(\mathbf{v}_0, \mathbf{v}_5)|(\mathbf{v}_5 - \mathbf{v}_0) &= \sum_{j=1,5}\delta_j R|\Lambda|L r_j = \sum_{j=1,5}\delta_j|\mu_j|r_j \\ &= \sum_{j=1,5}|\mu_j|(\mathbf{v}_j - \mathbf{v}_{j-1}),\end{aligned}$$

which yields the equality (3.15). □

3.1.3. The first-order linearly implicit relaxation scheme. We are now in a position to compute the partial derivatives of the numerical flux defined by (3.14). They are obtained by assuming that Roe's matrix is constant. We obtain the matrices α and γ as

$$\alpha_i \approx \frac{\Delta t^n}{2\Delta x}\left(A^{\mathbf{L}} + |A^{\mathrm{Roe}}|\right), \quad \gamma_{i+1} \approx \frac{\Delta t^n}{2\Delta x}\left(A^{\mathbf{R}} - |A^{\mathrm{Roe}}|\right), \tag{3.28}$$

where $A^{\mathbf{L},\mathbf{R}} = \nabla\mathcal{G}(\mathbf{v}^{\mathbf{L},\mathbf{R}})$ and $A^{\mathrm{Roe}} = A^{\mathrm{Roe}}(\mathbf{v}^{\mathbf{L}}, \mathbf{v}^{\mathbf{R}})$.

The first-order linearly implicit relaxation scheme consists, at each time step, in solving the linear system $\mathbb{A}\,\delta\mathbf{v} = b$. The matrix $\mathbb{A}$ is a tridiagonal by block matrix with each block of size 5×5; the resulting matrix is a band matrix with nine extradiagonal terms.

We have already shown the interior lines (3.8)–(3.9), but we have not described the first and the last lines, which are related to the boundary conditions. For the states $\mathbf{v}_0$ and $\mathbf{v}_{I+1}$, we specify $\delta\mathbf{v}_0 = 0$ and $\delta\mathbf{v}_{I+1} = 0$.

3.2. First-order semi-implicit relaxation scheme. The relaxation scheme is constructed so as to be

- linearly implicit with respect to the fast acoustic waves of speed $v \pm a\tau$,
- explicit with respect to the slow kinematic waves of speed v and the associated relaxation waves of speed $v \pm b\tau$.

Accordingly, when one computes the partial derivatives of the numerical flux, only the terms associated with the largest eigenvalues are kept. We take the diagonal matrix $\tilde{\Lambda} = \mathrm{Diag}(v_{\mathbf{L}} - a\tau_{\mathbf{L}},\, 0,\, 0,\, 0,\, v_{\mathbf{R}} + a\tau_{\mathbf{R}})$ which modifies Roe's matrix by $\tilde{A}^{\mathrm{Roe}} = R\tilde{\Lambda}L$ with the same matrices R and L as previously. Like Roe's matrix, the eigenvalues of the Jacobian matrices are modified in the same way by $\tilde{A}^{\mathbf{L},\mathbf{R}} = R^{\mathbf{L},\mathbf{R}}\tilde{\Lambda}^{\mathbf{L},\mathbf{R}}L^{\mathbf{L},\mathbf{R}}$. The matrices α and β are then computed with (3.28) and the previous modified matrices.

3.2.1. Time step of the semi-implicit relaxation scheme. At each time step, one computes two auxiliary time steps.

1. The explicit time step is

$$\Delta t^{\mathrm{Void}} = \mu^{\mathrm{Void}}\frac{\Delta x}{\max_{i=1,I}|V_i^{\mathrm{Void}}|},$$

   with $\mu^{\mathrm{Void}} = 0.5$ and $|V_i^{\mathrm{Void}}| = \max\left(|v^\star|,\, |v^\star - b\tau_{\mathbf{L}}^\star|,\, |v^\star + b\tau_{\mathbf{R}}^\star|\right)$.
2. The linearly implicit time step is

$$\Delta t^{\mathrm{Press}} = \mu^{\mathrm{Press}}\frac{\Delta x_i}{\max_{i=1,I}|V_i^{\mathrm{Press}}|},$$

   with $\mu^{\mathrm{Press}} = 20$ and $|V_i^{\mathrm{Press}}| = \max\left(|v_{\mathbf{L}} - a\tau_{\mathbf{L}}|,\, |v_{\mathbf{L}} + a\tau_{\mathbf{R}}|\right)$.

The selected time step is $\Delta t^n = \min\left(\Delta t^{\mathrm{Void}}, \Delta t^{\mathrm{Press}}\right)$. The CFL number $\mu^{\mathrm{Press}} = 20$, experimentally determined, turns out to be a good compromise between a small CPU time and an acceptable amount of smearing out for the numerical solution.

3.2.2. Implicit projection. We follow the ideas of Chalons [3]. The use of a linearly implicit scheme implies that the projection step is giving steady states that are not accurate. The solution is to link the evolutions of the relaxed variables associated with an implicit field to the evolutions of the equilibrium variables. In our case, the only relaxed variable associated with an implicit field is $\rho\Pi$; the relaxation variable $\rho\Sigma$ is associated with the eigenvalues $v \pm b\tau$, which are explicit. We therefore write the implicit projection equation

$$\delta(\rho\Pi)_i = \left(\frac{\partial\mathcal{I}}{\partial\rho}\right)_i^n \delta\rho_i + \left(\frac{\partial\mathcal{I}}{\partial(\rho Y)}\right)_i^n \delta(\rho Y)_i + \left(\frac{\partial\mathcal{I}}{\partial(\rho v)}\right)_i^n \delta(\rho v)_i,$$

with $\mathcal{I}(\mathbf{u}) = \rho P(\mathbf{u})$. The partial derivatives of $\mathcal{I}$ are computed on $\mathbf{u}_i^n$. The previous identity gives an equation that replaces the fifth line of (3.9), i.e., modifies the linear

system to be solved in the mathematical step. It enables us to reduce the size of each block from 5×5 to 4×4 and therefore reduce the size of the global linear system.

Numerical experiments show that not only are the steady solutions more accurate [3], but even transient solutions are more accurate.

REMARK 3.4.

$(a_{i+1/2}, b_{i+1/2})$

1.

2.

3.

3.3. Second-order semi-implicit scheme. The slope limiters used to build the second-order explicit scheme lead to a nondifferentiable expression. That is why we choose a simplified version in which we do not differentiate the nonlinear operator involved in the second-order correction. In the physical step, we first compute the second-order correction of the states $\mathbf{u_{L,R}}$ and evaluate the numerical flux. In the mathematical step, we solve the first-order linear system, but the derivatives are computed with the corrected states $\mathbf{u_{L,R}}$.

In order to have a second order in time scheme, we use again the Runge–Kutta 2 procedure. Since the scheme is only semi-implicit, the order 2 in time is only achieved on the explicit waves and the linearly implicit waves are solved with order 1 in time.

4. Numerical results. In this section, we show the numerical results of the semi-implicit relaxation scheme: shock tube test (without boundary conditions) and real-life cases (with boundary conditions). The first two test-cases are Riemann problems. The length of the domain is 100 meters and the discontinuity between the two initial constant states is located at $x = 50$ m. The last two test-cases are real-life problems in which the solution is driven by the changes of the boundary conditions.

4.1. Experiment 1: No-slip law. The details of this test-case are shown in Table 4.1 and the results are displayed in Figure 4.1. We see that the second-order correction leads to a more accurate scheme.

4.2. Experiment 2: Zuber–Findlay law. The details of this test-case are shown in Table 4.2 and the results are displayed in Figure 4.2. We used the VFRoeTACITE scheme described in [6] with the same CFL numbers. The two schemes are semi-implicit second-order schemes. The numerical results exhibit a good agreement between the two schemes.

4.3. Experiment 3: No-slip law. The details of this test-case are shown in Table 4.3 and the results are given in Figure 4.3. This is the simplest real-life problem we can imagine: we just double the inlet gas flow rate. There are no one-phase states. The numerical results show a good agreement between the two schemes.

4.4. Experiment 4: Zuber–Findlay-like law. The details of this test-case are shown in Table 4.4 and the results are given in Figure 4.4.

TABLE 4.1
Data for Experiment 1*, no-slip law.*

Thermodynamic laws	incompressible liquid, $\rho_{\mathrm{L}}^0 = 1000$ kg/m^3 perfect gas, $a_{\mathrm{G}} = 100$ m/s
Hydrodynamic law	no-slip law $\Phi = 0$
Source terms	none
Exact solution	1-rarefaction wave
Final time	$T = 0.8$ s
Length of the pipe	$L = 100$ m
Cell size	$\Delta x = 0.5$ m
Left state	$(\rho, Y, v)_{\mathbf{L}} = (500,\ 0.2,\ 34.4233)$
Right state	$(\rho, Y, v)_{\mathbf{R}} = (400,\ 0.2,\ 50)$
Characteristic speeds	$\lambda_{\mathbf{L}}^- = -40.12$ m/s and $\lambda_{\mathbf{R}}^- = -15.77$ m/s

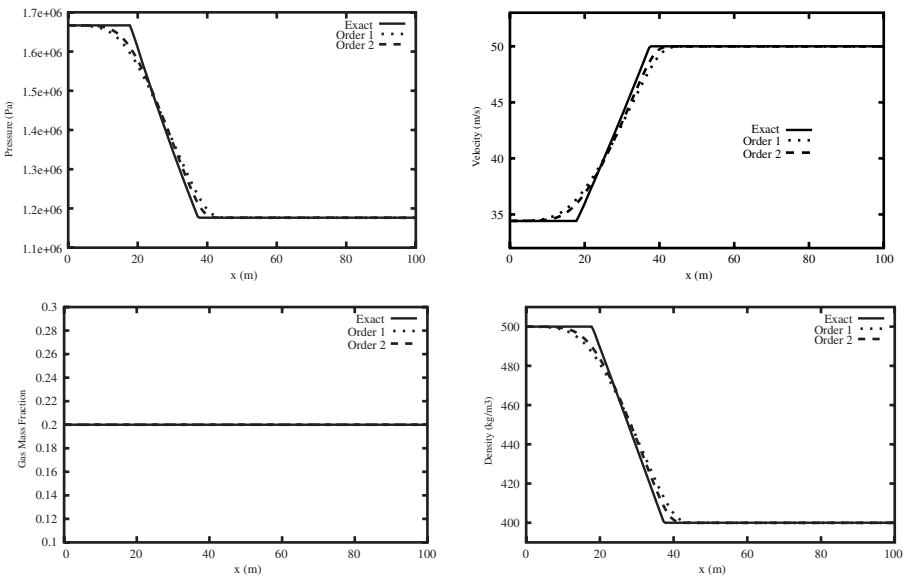


FIG. 4.1. *Numerical results of Experiment* 1*, no-slip law.*

TABLE 4.2
Data for Experiment 2*, Zuber–Findlay law.*

Thermodynamic laws	incompressible liquid, $\rho_{\mathrm{L}}^0 = 1000$ kg/m^3 perfect gas, $a_{\mathrm{G}} = 300$ m/s
Hydrodynamic law	Zuber–Findlay law [15] $\Phi = \dfrac{(c_0 - 1)v + c_1}{c_0[XR_{\mathrm{G}} - YR_{\mathrm{L}}] - X}$ with $c_0 = 1.07$, $c_1 = 0.2162$ m/s
Source terms	none
Exact solution	1-shock, 2-contact discontinuity, 3-shock
Final time	$T = 0.5$ s
Length of the pipe	$L = 100$ m
Cell size	$\Delta x = 0.5$ m
Left state	$(\rho, Y, v)_{\mathbf{L}} = (453.19,\ 0.00705,\ 24.80)$
Right state	$(\rho, Y, v)_{\mathbf{R}} = (454.19,\ 0.0108,\ 1.746)$
Shock speeds	$\sigma^- = -40.03$ m/s, $\sigma^0 = 10$ m/s, and $\sigma^+ = 67.24$ m/s

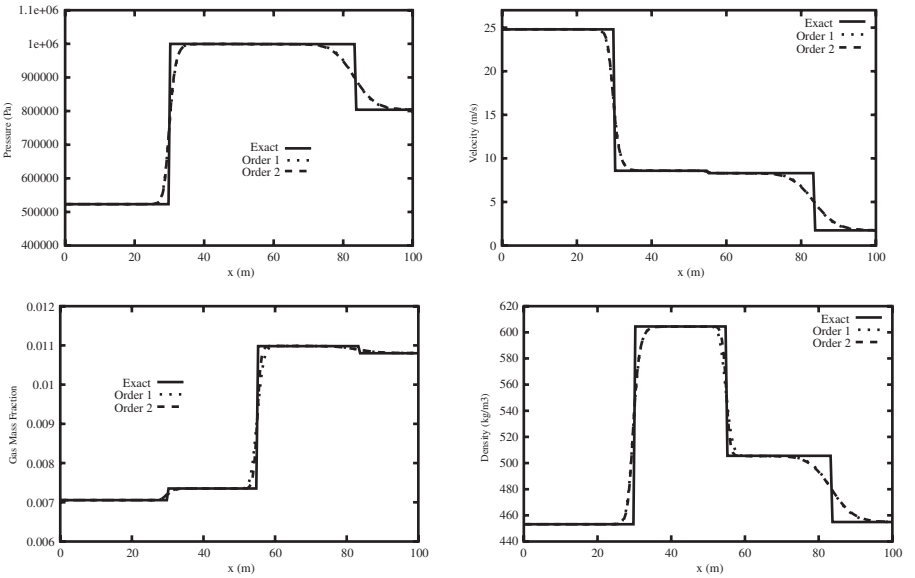


FIG. 4.2. *Numerical results of Experiment* 2*, Zuber–Findlay law.*

TABLE 4.3
Data for Experiment 3*, no-slip law with boundary conditions.*

Thermodynamic laws	incompressible liquid, perfect gas
Hydrodynamic law	no-slip law
Source terms	friction
Stabilization period	3000 s
Transient period	1 s
Length of the pipe	$L = 10000$ m, horizontal
Cell size	$\Delta x = 200$ m
Diameter	0.146 m
Inlet condition on gas flow rate	0.2 to 0.4 kg/s
Inlet condition on liquid flow rate	20 kg/s
Outlet condition on pressure	10^6 Pa

In this test-case [6], the inlet gas flow rate is decreased from 0.114 to 0 kg/s. As the mass flow rates are small, the decrease in the inlet gas mass flow rate gives rise to negative oil velocities in the upper part of the pipeline. Therefore a void fraction between a "single-phase gas" state and a "two-phase" state propagates from the outlet down the pipe. Simultaneously, the change in the inlet gas mass flow rates induces another void fraction front which propagates from the inlet up the pipe. These two fronts meet around $x = 47$ m at time $t = 260$ s to form a unique discontinuity wave which propagates towards the outlet. At time $t = 200$ s, this discontinuity wave reaches the outlet, and the pipe turns to a single-phase liquid steady state.

This test-case is quite stiff. During the simulation, the scheme must handle two-phase states, liquid state and gas state. Moreover, the discontinuity propagating at the end on the simulation is between two one-phase states. The classical VFRoe scheme is not rough enough to handle this case and that is why the authors of [6] introduced more numerical diffusion in their VFRoe-TACITE scheme. The numerical

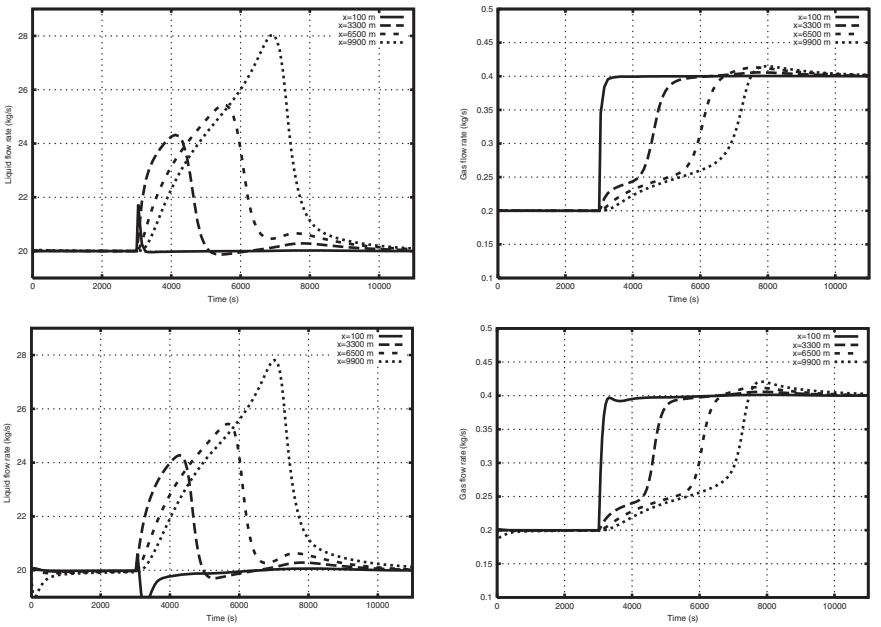


FIG. 4.3. *Experiment* 3, *TACITE results (top panels) and relaxation results (bottom panels).*

TABLE 4.4
Data for Experiment 4, *Zuber–Findlay-like law with boundary conditions.*

Thermodynamic laws	incompressible liquid, perfect gas
Hydrodynamic law	no-slip law
Source terms	friction and gravity
Stabilization period	10 s
Transient period	100 s
Length of the pipe	$L = 80$ m, vertical
Cell size	$\Delta x = 1.6$ m
Diameter	0.146 m
Inlet condition on gas flow rate	0.114 to 0 kg/s
Inlet condition on liquid flow rate	1.628 kg/s
Outlet condition on pressure	10^6 Pa

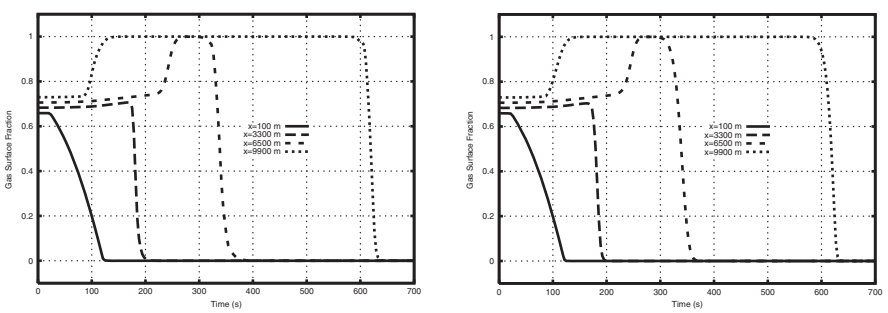


FIG. 4.4. *Experiment* 4, *gas fraction. TACITE results (left) and relaxation results (right).*

results show a good agreement between the two schemes.

5. Conclusion. A second-order semi-implicit relaxation scheme has been obtained as soon as we have succeeded in expressing the explicit scheme in Roe's form, that is to say, in computing the Roe matrix. The semi-implicit relaxation scheme is then explicit for the slow waves and linearly implicit for the fast waves and enables us to reduce the CPU time as well as to be more accurate on slow waves. Numerical experiments show a good agreement with the exact solution on shock tube tests and with a VFRoe-type scheme on more realistic problems.

There are a great number of prospects and open issues to this work: extension to the multicomponent case and to the energy equation, investigation into the link between the relaxation coefficients and the implicit projection, boundary conditions that are consistent with positivity properties for the inner scheme, and so on.

Appendix A. Additional benchmarks. In this appendix, we provide a few more critical numerical experiments so the reader can have a broader grasp of various aspects of the explicit and semi-implicit schemes.

A.1. Old vs. new relaxation coefficients. In [1], the relaxation coefficients a and b were assessed in a "coupled" way. In this paper, these are derived from the formulae detailed in section 2.3, based on a rough "decoupling" strategy. We wish to compare the performance of the explicit relaxation scheme based on the conditions of [1], called "old" (a, b), and the performance of the explicit relaxation scheme based on the conditions of section 2.3, called "new" (a, b).

Figure A.1 plots the pressure, the velocity, the gas mass fraction, and the density at the final time for a test-case consisting merely of a transport of the mass-fraction. First-order explicit approximations are considered. It can be seen that the "old" coefficients give rise to a slightly more smoothed out front. The difference between the two ways of computing (a, b) is hardly noticeable here, because in both cases b is very small, and the difference brought about by $a_{\text{new}} < a_{\text{old}}$ essentially lies in acoustic waves, which do not exist here. The discrepancy is much more impressive in Figure A.2, where we plot the results for a similar test-case with the Zuber–Findlay slip law. Here, $b_{\text{new}} \ll b_{\text{old}}$, which accounts for the excessively smeared out front in the curves associated with the old coefficients. This example highlights the great benefit of the new procedure for adjusting the relaxation coefficients.

A.2. CFL ratio for the semi-implicit scheme. In order to get a better understanding of the basic properties of the semi-implicit scheme regarding accuracy and stability properties, we plot in Figure A.3 a comparison of three first-order schemes, namely,

1. the explicit scheme with CFL = 0.5,
2. the semi-implicit scheme with CFL = 20 (CFL corresponding to fast waves),
3. the semi-implicit scheme with CFL = 0.5 (CFL based on fast wave),

for a Riemann problem consisting of a single transport of the slow wave. Note that since the Zuber–Findlay closure law is used, the pressure p is not preserved at the contact jump (which would be the case for a no-slip closure law).

It turns out that scheme 1 has the same amount of diffusion as scheme 3 and gives results better than scheme 2 at the pressure jump. Nevertheless, both are less sharp than scheme 2 as far as the mass-fraction is concerned. It is a well-known fact for an explicit scheme that the larger the CFL is, the less the computed solution is smeared out. It is thus tempting to take the "largest" μ^{Void} possible. However, we are limited by the fact that a large CFL ratio introduces errors via the implicit part

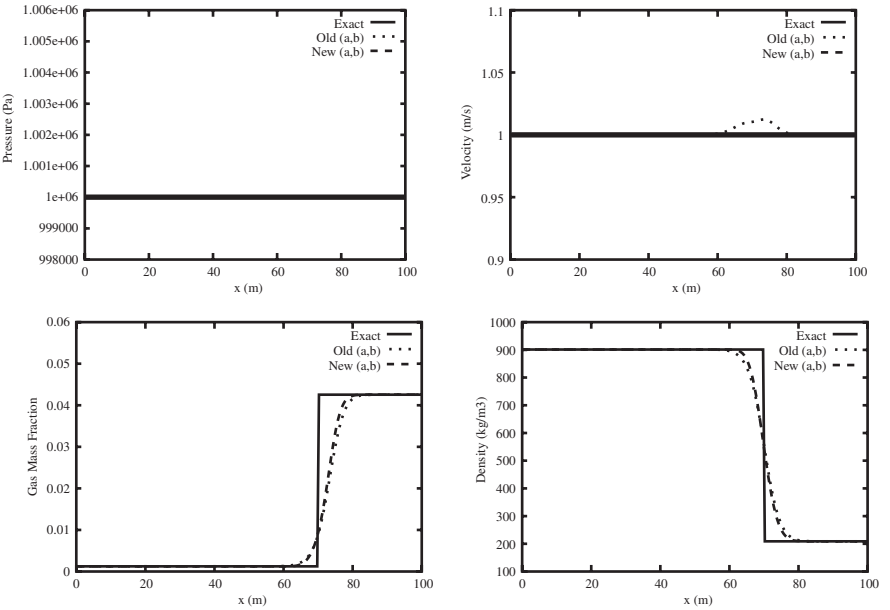


FIG. A.1. *Results for old and new relaxation coefficients with the no-slip law.*

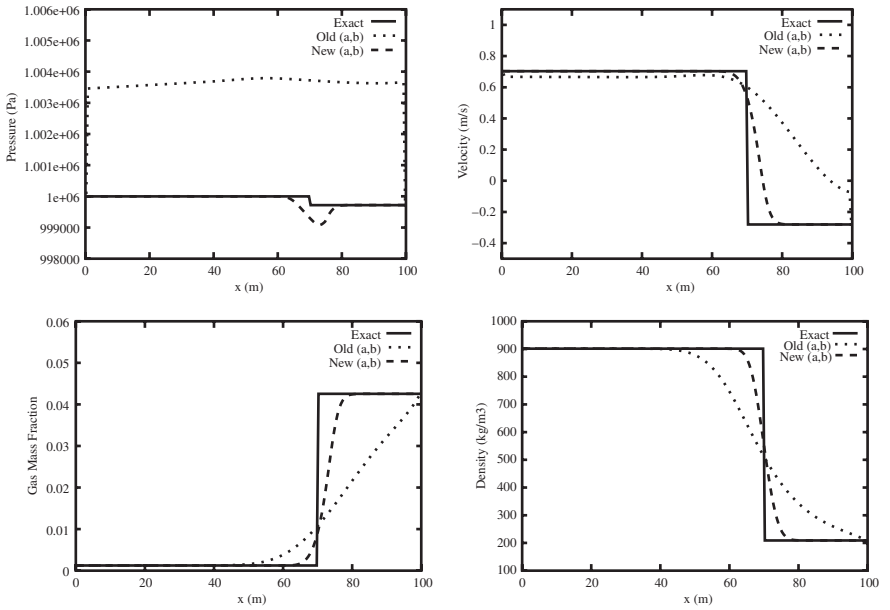


FIG. A.2. *Results for old and new relaxation coefficients with the Zuber–Findlay law.*

of the scheme (since only one Newton iteration is carried out). In this respect, the CFL number $\mu^{\text{Press}} = 20$ appears to be a good experimental compromise.

The fact that we do not observe any obvious difference between the three curves is due to the nature of the experiment at issue, in which only a slow wave is involved. The semiexplicit scheme has been purportedly designed to achieve good accuracy

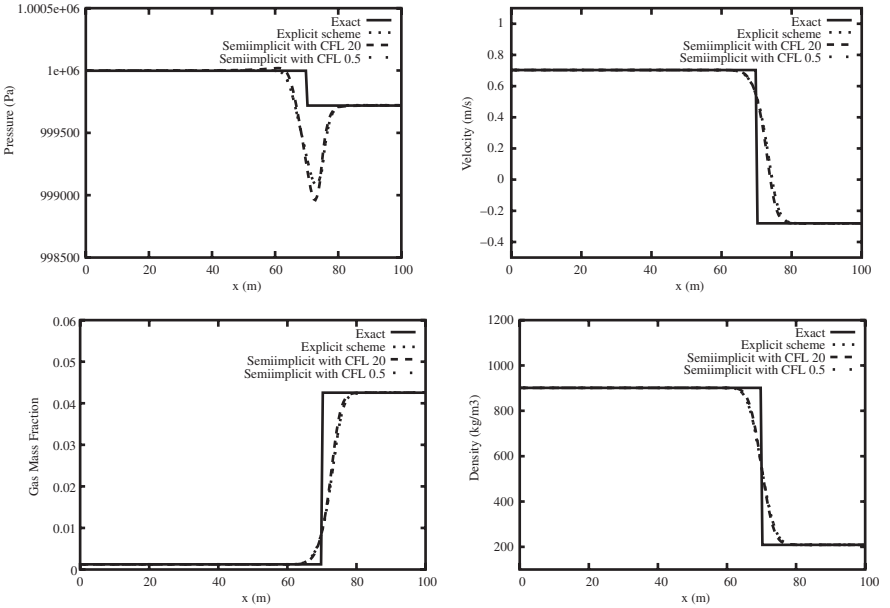


FIG. A.3. *Various first-order schemes, Zuber–Findlay law.*

for these slow waves. Had we considered a case with many types of waves, as in Experiment 3, we would have seen a relatively poor performance of the semi-implicit scheme on acoustic waves. Therefore, Figure A.3 should be interpreted as good news, insofar as it confirms our claim that the new semi-implicit scheme is able to achieve high accuracy on slow waves while saving a lot of computational time.

Acknowledgments. The authors wish to thank I. Faille and F. Willien for the numerous helpful discussions we had. We are also grateful to the anonymous reviewer who suggested several improvements to the original text.

REFERENCES

[1] M. BAUDIN, C. BERTHON, F. COQUEL, R. MASSON, AND Q.-H. TRAN, *A relaxation method for two-phase flow models with hydrodynamic closure law*, Numer. Math., 99 (2005), pp. 411–440.

[2] S. BENZONI-GAVAGE, *Analyse Numérique des Modèles Hydrodynamiques d'Écoulements Diphasiques Instationnaires dans les Réseaux de Production Pérolière*, Ph.D. dissertation, École Normale Supérieure de Lyon, Lyon cedex, France, 1991.

[3] C. CHALONS, *Bilans d'entropie discrets dans l'approximation numérique des chocs non classiques. Application aux équations de Navier-Stockes multi-pression 2D et à quelques systèmes visco-capillaires*, thèse de doctorat, Université Pierre et Marie Curie, Paris VI, Paris, France, 2002.

[4] F. DUBOIS AND G. MEHLMANN, *A non-parameterized "entropy fix" for Roe's method*, in Proceedings of the 10th AIAA CFD Conference, Honolulu, HI, 1991.

[5] S. EVJE AND K. K. FJELDE, *Relaxation schemes for the calculation of two-phase flow in pipes*, Math. Comput. Modelling, 36 (2002), pp. 535–567.

[6] I. FAILLE AND E. HEINTZÉ, *A rough finite volume scheme for modeling two phase flow in a pipeline*, Comput. & Fluids, 28 (1999), pp. 213–241.

[7] K.-K. FJELDE AND K.-H. KARLSEN, *A Comparative Study of Some Fully Numerical Shock-Capturing Schemes for Simulating Two-Phase Flow*, preprint.

[8] T. GALLOUËT, *Équations satisfaites par des limites de solution approchées*, in Proceedings of the Congrès National d'Analyse Numérique, Biarritz, France, 2002.

[9] T. Gallouët and J.-M. Masella, *A rough Godunov scheme*, C. R. Acad. Sci. Paris Ser. I Math., 323 (1996), pp. 77–84.
[10] E. Godlewski and P. Raviart, *Numerical Approximation of Hyperbolic Systems of Conservation Laws*, Appl. Math. Sci. 118, Springer-Verlag, New York, 1996.
[11] E. Godlewski and P.-A. Raviart, *Hyperbolic Systems of Conservation Laws*, Math. Appl. (Paris) 3/4, Ellipses, Paris, 1991.
[12] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in Mathematics ETH Zürich, Birkhäuser-Basel Verlag, 1992.
[13] C. Pauchon, H. Dhulésia, G. Binh-Cirlot, and J. Fabre, Tacite*: A transient tool for multiphase pipeline and well simulation*, in Proceedings of the SPE Annual Technical Conference, New Orleans, LA, 1994.
[14] Z. Xin and S. Jin, *The relaxation schemes for systems of conservation laws in arbitrary space dimensions*, Comm. Pure Appl. Math., 48 (1995), pp. 235–276.
[15] N. Zuber and J. Findlay, *Average volumetric concentration in two-phase flow systems*, J. Heat Transfer, C87 (1965), pp. 453–458.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 937–966
© 2005 Society for Industrial and Applied Mathematics

ANALYSIS OF HALF-QUADRATIC MINIMIZATION METHODS FOR SIGNAL AND IMAGE RECOVERY*

MILA NIKOLOVA† AND MICHAEL K. NG‡

Abstract. We address the minimization of regularized convex cost functions which are customarily used for edge-preserving restoration and reconstruction of signals and images. In order to accelerate computation, the *multiplicative* and the *additive* half-quadratic reformulation of the original cost-function have been pioneered in Geman and Reynolds [*IEEE Trans. Pattern Anal. Machine Intelligence*, 14 (1992), pp. 367–383] and Geman and Yang [*IEEE Trans. Image Process.*, 4 (1995), pp. 932–946]. The alternate minimization of the resultant (augmented) cost-functions has a simple explicit form. The goal of this paper is to provide a systematic analysis of the convergence rate achieved by these methods. For the multiplicative and additive half-quadratic regularizations, we determine their upper bounds for their root-convergence factors. The bound for the multiplicative form is seen to be always smaller than the bound for the additive form. Experiments show that the number of iterations required for convergence for the multiplicative form is always less than that for the additive form. However, the computational cost of each iteration is much higher for the multiplicative form than for the additive form. The global assessment is that minimization using the additive form of half-quadratic regularization is faster than using the multiplicative form. When the additive form is applicable, it is hence recommended. Extensive experiments demonstrate that in our MATLAB implementation, both methods are substantially faster (in terms of computational times) than the standard MATLAB OPTIMIZATION TOOLBOX routines used in our comparison study.

Key words. convergence analysis, half-quadratic regularization, inverse problems, maximum a posteriori estimation, optimization, preconditioning, signal and image restoration, variational methods

AMS subject classifications. 35A15, 49N45, 65K10, 65F22, 65Y20, 94A08, 94A12

DOI. 10.1137/030600862

1. Introduction.

This work addresses a wide class of problems where a sought vector $\hat{x} \in R^p$ (e.g., an image or a signal) is estimated based on degraded data $y \in \mathbb{R}^q$ by minimizing a cost function $J : \mathbb{R}^p \to \mathbb{R}$ which combines a quadratic data-fidelity term and a regularization term Φ, weighted by a parameter $\beta > 0$:

$$\hat{x} = \min_{x \in \mathbb{R}^p} J(x), \tag{1.1}$$

$$J(x) = \|Ax - y\|^2 + \beta\Phi(x). \tag{1.2}$$

In a statistical framework, this data-fidelity term assumes that data y are obtained from the original x by a linear transform, modeled by $A \in \mathbb{R}^{q \times p}$, and that they are contaminated by white Gaussian noise. Such data-fidelity terms are popular in denoising, in deblurring, and in numerous inverse problems such as seismic imaging,

*Received by the editors September 5, 2003; accepted for publication (in revised form) March 29, 2005; published electronically December 8, 2005. The work was supported by a grant from the PROCORE - France/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong; the Consulate General of France in Hong Kong F-HK 30/04T; RGC grants HKU 7130/02P, 7046/03P, 7035/04P; and HKU CRCG grants 10205775 and 10204437.

http://www.siam.org/journals/sisc/27-3/60086.html

†Centre de Mathématiques et de Leurs Applications (CNRS-UMR 8536), ENS de Cachan, 61 avenue du President Wilson, 94235 Cachan Cedex, France (nikolova@cmla.ens-cachan.fr).

‡Department of Mathematics, The University of Hong Kong, Pokfulam Road, Hong Kong and Department of Mathematics, Hong Kong Baptist University, Kowloon Tong, Hong Kong (mng@maths.hku.hk).

nondestructive evaluation, and X-ray tomography [1, 12, 28, 29, 31]. We focus on regularization terms Φ of the form

$$\Phi(x) = \sum_{i=1}^{r} \phi(g_i^T x), \tag{1.3}$$

where $g_i^T : \mathbb{R}^p \to \mathbb{R}$, for $i = 1, \ldots, r$, are linear operators. Typically, $\{g_i^T x\}$ are first- or second-order differences between neighboring samples in x. For example, if x is a one-dimensional signal, usually $g_i^T x = x_i - x_{i+1}$, for $i = 1, \ldots, p-1$. Let G denote the $r \times p$ matrix whose ith row is g_i^T, for $i = 1, \ldots, r$. Some basic requirements are

$$\phi \not\equiv 0 \quad \text{and} \quad \ker(A^T A) \cap \ker(G^T G) = \{0\}. \tag{1.4}$$

We focus on convex, [illegible] potential functions $\phi : \mathbb{R} \to \mathbb{R}$ in (1.3), because they give rise to image and signal estimates of high quality, involving edges and homogeneous regions. Examples of such functions are [4, 5, 8, 17, 22]

$$\phi(t) = |t|^\alpha, \ 1 < \alpha \leq 2, \tag{1.5}$$

$$\phi(t) = \sqrt{\alpha + t^2}, \tag{1.6}$$

$$\phi(t) = \log(\cosh(\alpha t)), \tag{1.7}$$

$$\phi(t) = |t|/\alpha - \log\left(1 + |t|/\alpha\right), \tag{1.8}$$

$$\phi(t) = \begin{cases} t^2/2 & \text{if} \quad |t| \leq \alpha, \\ \alpha|t| - \alpha^2/2 & \text{if} \quad |t| > \alpha, \end{cases} \tag{1.9}$$

where $\alpha > 0$ is a parameter. We will systematically consider that ϕ is convex, even, and $\mathcal{C}^2$ and that

$$A^T A \ \text{ is invertible} \quad \text{and/or} \quad \phi''(t) > 0 \ \ \forall t \in \mathbb{R}. \tag{1.10}$$

Remark 1. [illegible] (1.10) [illegible] (1.4) [illegible] $y \in \mathbb{R}^p$ [illegible] J [illegible]

However, the minimizers $\hat{x}$ of cost-functions J involving edge-preserving regularization are nonlinear with respect to y and their computation is costly, especially when A has many nonzero entries and/or A is ill-conditioned. In order to cope with the computation, [illegible] reformulation of J was pioneered, in two different ways, in [13, 14]. The idea is to construct an [illegible] $\mathcal{J} : \mathbb{R}^p \times \mathbb{R}^r \to \mathbb{R}$ which involves an auxiliary variable $b \in \mathbb{R}^r$,

$$\mathcal{J}(x, b) = \|Ax - y\|^2 + \beta \sum_{i=1}^{r} \left(\, Q(g_i^T x, \ b_i) + \psi(b_i) \, \right), \tag{1.11}$$

where $Q(., s) : \mathbb{R} \to \mathbb{R}$ is quadratic for any $s \in \mathbb{R}$ and $\psi : \mathbb{R} \to \mathbb{R}$ satisfies

$$\phi(t) = \min_{s \in \mathbb{R}} \{Q(t, \ s) + \psi(s)\} \quad \forall t \in \mathbb{R}. \tag{1.12}$$

Such a [illegible] potential function ψ is determined using the theory of convex conjugacy [18, 27]. Condition (1.12) ensures that

$$J(x) = \min_{b \in \mathbb{R}^r} \mathcal{J}(x, b) \quad \forall x \in \mathbb{R}^p. \tag{1.13}$$

The regularization term involved in $\mathcal{J}$ is half-quadratic, and hence the name of the method. The minimizer $(\hat{x}, \hat{b})$ of $\mathcal{J}$ is calculated using [illegible]. Let the solution at iteration $(k-1)$ read $(x^{(k-1)}, b^{(k-1)})$. At iteration k we calculate

$$b^{(k)} \text{ such that } \mathcal{J}(x^{(k-1)}, b^{(k)}) \leq \mathcal{J}(x^{(k-1)}, b) \quad \forall b \in \mathbb{R}^r, \tag{1.14}$$

$$x^{(k)} \text{ such that } \mathcal{J}(x^{(k)}, b^{(k)}) \leq \mathcal{J}(x, b^{(k)}) \quad \forall x \in \mathbb{R}^p. \tag{1.15}$$

This amounts to finding $b^{(k)}$ and $x^{(k)}$ according to

$$\begin{aligned} b^{(k)} &= \left[\sigma(g_i^T x^{(k-1)})\right]_{i=1}^r, \\ x^{(k)} &= \chi(b^{(k)}), \end{aligned} \tag{1.16}$$

where $\sigma : \mathbb{R} \to \mathbb{R}$ and $\chi : \mathbb{R}^r \to \mathbb{R}^p$ are minimizer functions which are defined in sections 2 and 3. For every $t \in \mathbb{R}$, the scalar $\sigma(t)$ is such that

$$Q(t, \sigma(t)) + \psi(\sigma(t)) \leq Q(t, s) + \psi(s) \quad \forall s \in \mathbb{R}. \tag{1.17}$$

For every $b \in \mathbb{R}^r$, the vector $\chi(b) \in \mathbb{R}^p$ is such that

$$\|A\chi(b) - y\|^2 + \beta \sum_{i=1}^r Q(g_i^T \chi(b),\ b_i) \leq \|Ax - y\|^2 + \beta \sum_{i=1}^r Q(g_i^T x,\ b_i)\ \forall x \in \mathbb{R}^p. \tag{1.18}$$

The key points are that each component $b_i^{(k)}$ is calculated using a scalar function σ and that $x^{(k)}$ is calculated using an affine function of y, namely χ. Moreover, both functions σ and χ have an explicit form. Thus $b^{(k)}$ and $x^{(k)}$ are easy to compute and allow J to be decreased.

REMARK 2. [illegible] (1.14)–(1.15) [illegible] $k \in \mathbb{N}$ [illegible]

$$\mathcal{J}(x^{(k+1)}, b^{(k+1)}) \leq \mathcal{J}(x^{(k)}, b^{(k+1)}) \leq \mathcal{J}(x^{(k)}, b^{(k)}). \tag{1.19}$$

[illegible] $\mathcal{J}$ [illegible]

$$\{\ldots, \mathcal{J}(x^{(k)}, b^{(k)}), \mathcal{J}(x^{(k)}, b^{(k+1)}), \mathcal{J}(x^{(k+1)}, b^{(k+1)}), \ldots\}$$

[illegible] $k \to \infty$ [illegible] $J(x^{(k+1)}) \leq J(x^{(k)})$ [illegible] k [illegible] $\{J(x^{(k)})\}$ [illegible]

By extending the notion of line-variable [15], Geman and Reynolds [13] first considered quadratic terms of the [illegible],

$$Q(t, s) = \frac{1}{2} t^2 s \quad \text{for} \quad t \in \mathbb{R}, \quad s \in \mathbb{R}_+. \tag{1.20}$$

Later, Geman and Yang proposed in [14] the [illegible] for Q,

$$Q(t, s) = (t - s)^2 \quad \text{for} \quad t \in \mathbb{R}, \quad s \in \mathbb{R}. \tag{1.21}$$

In [13, 14], half-quadratic regularization was used to simplify simulated annealing minimization in cases when ϕ is nonconvex and A has many nonzero entries. These fruitful ideas have been pursued and deepened by many authors. The multiplicative form (1.20) was considered in [7, 8, 11, 19, 21, 30]. The additive form (1.21) seems to have suggested a narrower interest. It was considered in [3, 7, 10, 19]. In [10],

the auxiliary variable is introduced in a nonconvex data-fidelity term. Extensions of the multiplicative form to interacting line variables b were proposed in [30, 19]. The convergence of the alternate minimization (1.14)–(1.15) of $\mathcal{J}$ towards the sought minimizer $\hat{x}$ of the original J for the multiplicative form is considered in [8, 11, 19] and for the additive form in [3] in a continuous setting.

In [3, 7, 10, 13, 14], the numerical results have shown that minimization using half-quadratic (HQ) regularization can speed up computation compared with the steepest descent method. However, the convergence rate of HQ minimization methods has never been analyzed in a systematic way. The effectiveness of the two HQ formulations (1.20) and (1.21) has never been compared theoretically. The main objective of this paper is to characterize the speed of convergence and estimate the bound on the convergence rate relevant to each form of HQ regularization and to compare it with standard minimization methods. We show that minimization using HQ regularization amounts to finding the fixed point $\hat{x}$ of a mapping $F : \mathbb{R}^p \to \mathbb{R}^p$ of the form

$$F(x) = x - (\mathcal{H}(x))^{-1} DJ(x),$$

where $\mathcal{H}(x) \in \mathbb{R}^{p\times p}$ performs a pertinent correction of the steepest descent direction $-DJ(x)$. We analyze the contraction properties of the mapping F corresponding to the multiplicative and the additive forms of HQ regularization. In order to compare the iterations corresponding to these two forms, we focus on the root-convergence factor [9, 26], namely

$$C(F,\hat{x}) = \sup\left\{ \limsup_{k\to\infty} \left\| x^{(k)} - \hat{x} \right\|^{\frac{1}{k}} \;:\; x^{(0)} \in \mathbb{R}^p \right\}. \tag{1.22}$$

Let us emphasize that $C(F,\hat{x})$ characterizes the convergence speed for any norm used to measure the residual $\|x^{(k)} - \hat{x}\|$; see, e.g., [26]. Using the linear convergence theorem [26, p. 301], there are conditions ensuring that

$$C(F,\hat{x}) = \text{Spectral Radius}(DF(\hat{x})).$$

For each form we determine an upper bound of the spectral radius of DF at $\hat{x}$ which is shown to be strictly smaller than 1. The bound for the multiplicative form is revealed to be smaller than the bound for the additive form. This result suggests that the multiplicative form (1.20) needs fewer iterations to find the minimum than the additive form (1.21). However, the cost per iteration is much smaller for the additive form than for the multiplicative form. By combining these two factors, we demonstrate that minimization using the additive form is faster than using the multiplicative form. When the additive form is applicable, it is more attractive. We also propose a fast method to effectively recover signals and images using regularization with the Huber function ϕ—cf. (1.9)—and a nonsmooth augmented cost-function $\mathcal{J}$. Furthermore, we discuss different ways to improve the computational cost of HQ minimization. Extensive experiments based on the MATLAB computing environment demonstrate that the additive form achieves minimization faster than the multiplicative form, and that both forms in our MATLAB implementation substantially outperform in terms of computational times the routines in the MATLAB Optimization Toolbox. (These results must be qualified by the observation that MATLAB run times for the same operations can vary by orders of magnitude depending on the particular coding used.)

Organization of the paper. Section 2 is dedicated to the speed of convergence of the multiplicative form. In section 3, we study the speed of convergence relevant

to the additive form. The convergence properties of the two forms, multiplicative and additive, are compared in section 4. Section 5 summarizes the experimental results. Concluding remarks are presented in section 6.

Notations. When necessary, a vector $x \in \mathbb{R}^p$ is also written as $[x_i]_{i=1}^p$. Given $x \in \mathbb{R}^p$, $\mathrm{diag}(x) \equiv \mathrm{diag}\left([x_i]_{i=1}^p\right)$ is the diagonal $p \times p$ matrix whose diagonal elements are the entries of x. For a function $f : \mathbb{R} \to \mathbb{R}$, $f'(t^-)$ and $f'(t^+)$ are its left-side and right-side derivatives at t, respectively. By $\lim_{t \searrow 0} f(t)$ we mean the limit of f when t converges to 0 by positive values. The symbol $\mathbb{R}_+$ addresses the nonnegative reals and $\mathbb{R}_+^*$ the positive reals. If $f : \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ depends on two variables, $D_j^k f$ is its kth differential with respect to the jth variable. The Euclidean norm of a vector $u \in \mathbb{R}^p$ is denoted $\|u\|$. If H is a $p \times p$ real matrix, its largest-in-magnitude eigenvalue is denoted by $\rho(H)$. Furthermore, $|\!|\!|H|\!|\!| = \sqrt{\rho(H^T H)}$, which is in fact the largest singular value of H. We write $H \succ 0$ if H is symmetric and positive definite. The symbol I stands for identity matrix and $\mathbb{1}$ for a vector whose entries are equal to 1. Given $x \in \mathbb{R}^p$ and $\delta > 0$, we denote $B(x, \delta) = \{u \in \mathbb{R}^p : \|x - u\| \leq \delta\}$. Furthermore, $S = \{x \in \mathbb{R}^p : \|x\| = 1\}$. We also systematically denote

$$\nu^2 = \min\left\{\mu \in \mathbb{R}_+ : \mu \text{ is an eigenvalue of } A^T A\right\},$$

where A is the matrix involved in (1.2).

2. Multiplicative form of half-quadratic regularization.

2.1. Preliminary facts.

Based on previous papers, we suppose that

$$\begin{array}{ll} \text{(a)} & t \to \phi(t) \ \text{ is convex on } \ \mathbb{R}, \\ \text{(b)} & t \to \phi(\sqrt{t}) \ \text{ is concave on } \ \mathbb{R}_+, \\ \text{(c)} & \phi(t) = \phi(-t) \ \forall t \in \mathbb{R}, \\ \text{(d)} & \phi \ \text{ is } \ \mathcal{C}^1 \text{ on } \mathbb{R}, \\ \text{(e)} & \phi'(0^+) > 0, \\ \text{(f)} & \lim\limits_{t \to \infty} \phi(t)/t^2 = 0. \end{array} \tag{2.1}$$

The dual function ψ is convex and reads

$$\psi(s) = \sup_{t \in \mathbb{R}} \left\{ -\frac{1}{2} s t^2 + \phi(t) \right\}. \tag{2.2}$$

As required in (1.12), we have reciprocally (cf. section 7.1 in the appendix)

$$\phi(t) = \inf_{s \in \mathbb{R}} \left\{ \frac{1}{2} s t^2 + \psi(s) \right\}. \tag{2.3}$$

The relation between ϕ and ψ under different assumptions on ϕ were analyzed in [8, 13, 19]. Using (1.11) and (1.20), the augmented cost-function $\mathcal{J}$ reads

$$\begin{aligned} \mathcal{J}(x, b) &= \|Ax - y\|^2 + \beta \sum_{i=1}^{r} \left(\frac{b_i}{2} \, (g_i^T x)^2 + \psi(b_i) \right) \\ &= \|Ax - y\|^2 + \frac{\beta}{2} (Gx)^T \mathrm{diag}(b) \, Gx + \beta \sum_{i=1}^{r} \psi(b_i). \end{aligned} \tag{2.4}$$

The minimizer function σ, defined by (1.17), admits an explicit form which was first determined in [8]:

$$\sigma(t) = \begin{cases} \phi''(0^+) & \text{if} \quad t = 0, \\ \dfrac{\phi'(t)}{t} & \text{if} \quad t \neq 0. \end{cases} \tag{2.5}$$

For its proof under (2.1), see section 7.1 in the appendix. The expression of σ for several functions ϕ is given in Table 4.1. It is useful to notice its main properties.

REMARK 3. [illegible] ϕ [illegible] (2.1)

1. [illegible] $t \to \sigma(t)$ [illegible] $\mathbb{R}_+$ [illegible] (b) [illegible] $t \to \frac{d}{dt}\phi(\sqrt{t}) = \phi'(\sqrt{t})/(2\sqrt{t}) = \sigma(\sqrt{t})/2$ [illegible] $\mathbb{R}_+$
2. [illegible] (a), (c), [illegible] (e), [illegible] $t > 0$ [illegible] $\phi'(t) > 0$ [illegible] $\sigma(t) > 0$ [illegible] $t \in \mathbb{R}$ [illegible]

$$\sigma(t) \in \big(\, 0, \phi''(0^+) \,\big] \quad \forall t \in \mathbb{R}. \tag{2.6}$$

Hence, we can consider b in (2.4) only on $(0, \phi''(0^+)\,]^r$. For any b fixed, the minimizer function χ, introduced in (1.18), is determined by solving with respect to x the equation $D_1\mathcal{J}(x, b) = 0$, that is, $H(b)x = 2A^T y$, where

$$H(b) = 2A^T A + \beta G^T \mathrm{diag}(b) G. \tag{2.7}$$

LEMMA 2.1. [illegible] $H : \mathbb{R}^r \to \mathbb{R}^{p\times p}$ [illegible] (2.7) [illegible] (1.4) [illegible] $\zeta > 0$ [illegible] $\lambda_\zeta > 0$ [illegible] $b \in [\zeta, \infty)^r$ [illegible] $H(b)$ [illegible] λ_ζ [illegible]. Given $\zeta > 0$, let λ_ζ be the smallest eigenvalue of $H(\zeta \mathbb{1})$ and $u_0 \in S$ be the corresponding eigenvector. Clearly, $\lambda_\zeta \geq 0$. We have

$$u_0^T H(\zeta \mathbb{1}) u_0 = \lambda_\zeta = 2\|Au_0\|^2 + \beta\zeta\|Gu_0\|^2.$$

If $u_0 \in \ker(A^T A)$, then (1.4) guarantees that $\|Gu_0\|^2 > 0$. By (1.4) again, if $u_0 \in \ker(G^T G)$, we have $\|Au_0\|^2 > 0$. In all cases, $\lambda_\zeta > 0$. Furthermore, for any $b \in [\zeta, \infty)^r$ and $u \in S$,

$$u^T H(b) u = 2\|Au\|^2 + \beta \sum_{i=1}^{r} b_i (g_i^T u)^2 \geq 2\|Au\|^2 + \beta\zeta\|Gu\|^2 = u^T H(\zeta \mathbb{1}) u \geq \lambda_\zeta.$$

Hence we have the conclusion. □

If (1.4) and (1.10) hold, for every $b \in (0, \infty)^r$ fixed, $H(b)$ is invertible and χ reads

$$\chi(b) = (H(b))^{-1} 2A^T y. \tag{2.8}$$

Although $\mathcal{J}(x, .)$ and $\mathcal{J}(., b)$ are convex functions, $(x, b) \to \mathcal{J}(x, b)$ is generally nonconvex.[1] The convergence of $\{x^{(k)}\}$ in the alternate minimization sequence

[1] For $x \in \mathbb{R}$, $y \in \mathbb{R}$, $A = 1$, and $g_1 = 1$, we have $J(x) = (x - y)^2 + \beta\phi(x)$, and hence $\mathcal{J}(x, b) = (x - y)^2 + \frac{\beta}{2} b x^2 + \beta\psi(b)$. Consider that ψ is twice differentiable on $\mathbb{R}_+^*$. The second differential of $\mathcal{J}$ at an arbitrary (x, b) reads

$$D^2\mathcal{J}(x, b) = \begin{bmatrix} 2 + \beta b & \beta x \\ \beta x & \beta\psi''(b) \end{bmatrix}.$$

$D^2\mathcal{J}(x, b)$ is not positive definite for all $(x, b) \in \mathbb{R} \times \mathbb{R}_+^*$ since for any $b > 0$ and $|x| > (2/\beta + b)\psi''(b)$ we have $\det D^2\mathcal{J}(x, b) < 0$.

$\{(x^{(k)}, b^{(k)})\}$ towards the minimizer $\hat{x}$ of J was initially proven in [8] under strong assumptions on ϕ and later in [19] under conditions similar to (2.1). Convergence towards local minimizers of a nonconvex J was addressed in [11]. It is easy to see that under our assumptions, the sequence $\{(x^{(k)}\}$ is convergent.

LEMMA 2.2. [illegible] J [illegible] (1.2)–(1.3) [illegible] (1.4) [illegible] (2.1) [illegible] $\mathcal{J}$ [illegible] (2.2)–(2.4) [illegible] $\{(x^{(k)}, b^{(k)})\}$ [illegible] (1.14)–(1.15) [illegible]

$$\|x^{(k)} - x^{(k+1)}\| \to 0 \quad k \to \infty.$$

[illegible] Since the function $\mathcal{J}(., b)$ is quadratic, for any $k \in \mathbb{N}$ we can write that

$$\mathcal{J}(x^{(k-1)}, b^{(k)}) - \mathcal{J}(x^{(k)}, b^{(k)}) = D_1\mathcal{J}(x^{(k)}, b^{(k)}) \left(x^{(k-1)} - x^{(k)}\right) + \frac{1}{2} \left(x^{(k-1)} - x^{(k)}\right)^T H(b^{(k)}) \left(x^{(k-1)} - x^{(k)}\right),$$

where the matrix $H(b^{(k)})$ reads as in (2.7). Since $x^{(k)}$ minimizes $\mathcal{J}(., b^{(k)})$, we have $D_1\mathcal{J}(x^{(k)}, b^{(k)}) = 0$. Then

$$\mathcal{J}(x^{(k-1)}, b^{(k)}) - \mathcal{J}(x^{(k)}, b^{(k)}) = \frac{1}{2} \left(x^{(k-1)} - x^{(k)}\right)^T H(b^{(k)}) \left(x^{(k-1)} - x^{(k)}\right). \tag{2.9}$$

The rest of the proof uses the following simple observation.

REMARK 4. [illegible] (1.4) [illegible] ϕ [illegible] J [illegible] 0 [illegible] $\lim_{\|x\| \to \infty} J(x) = \infty$ [illegible] 2 [illegible] $\{J(x^{(k)})\}$ [illegible] $\delta \in (0, \infty)$ [illegible]

$$x^{(k)} \in B(0, \delta) \quad \forall k \in \mathbb{N}. \tag{2.10}$$

For this $\delta > 0$, put

$$\zeta = \sigma(\delta \, |||G|||); \tag{2.11}$$

then $\zeta > 0$ by Remark 3 condition 2. Then we have

$$\begin{aligned} x \in B(0, \delta) \quad &\Rightarrow \quad |g_i^T x| \le \delta \, |||G||| \qquad \forall i \in \{1, \dots, r\}, \\ &\Rightarrow \quad \sigma(g_i^T x) \ge \zeta > 0 \qquad \forall i \in \{1, \dots, r\}, \end{aligned} \tag{2.12}$$

where the second implication comes from Remark 3 condition 1. Since $b_i^{(k)} = \sigma(g_i^T x^{(k-1)})$ for all i and k, combining (2.10) and (2.12) yields

$$b_i^{(k)} \ge \zeta > 0 \quad \forall i \in \{1, \dots, r\}, \quad \forall k \in \mathbb{N}. \tag{2.13}$$

Let $\lambda_\zeta > 0$ be the constant exhibited in Lemma 2.1 relevant to ζ. Then (2.9) yields

$$\mathcal{J}(x^{(k-1)}, b^{(k)}) - \mathcal{J}(x^{(k)}, b^{(k)}) \ge \frac{\lambda_\zeta}{2} \left\|x^{(k-1)} - x^{(k)}\right\|^2 \quad \forall k \in \mathbb{N}.$$

By Remark 2, $\mathcal{J}(x^{(k-1)}, b^{(k)}) - \mathcal{J}(x^{(k)}, b^{(k)}) \to 0$ as $k \to \infty$, and hence we have the conclusion. □

2.2. Convergence analysis. We will assume that in addition to (2.1),

$$\phi \text{ is } \mathcal{C}^2 \text{ on } \mathbb{R} \text{ and } \mathcal{C}^3 \text{ near zero.} \tag{2.14}$$

REMARK 5. ϕ (2.1) (2.14) σ (2.5) $\mathcal{C}^1$ $\mathbb{R}$ $\sigma'(0) = 0$ $t \in \mathbb{R} \setminus \{0\}$

$$\sigma'(t) = \frac{\phi''(t)t - \phi'(t)}{t^2} \tag{2.15}$$

$t \to 0$

$$\lim_{t\to 0} \sigma'(t) = \lim_{t\to 0} \frac{\frac{d}{dt}\left(\phi''(t)t - \phi'(t)\right)}{\frac{d}{dt}\left(t^2\right)} = \lim_{t\to 0} \frac{\phi'''(t)}{2} = 0.$$

Notice that $\phi'''(0) = 0$ because ϕ is even and $\mathcal{C}^3$ near zero.

THEOREM 2.3. J (1.2)–(1.3) (1.4) (1.10) ϕ (2.1) (2.14) $\{(x^{(k)}, b^{(k)})\}$ (1.16) σ χ (2.5) (2.7)–(2.8)

(i) $k \in \mathbb{N}$

$$x^{(k)} = F(x^{(k-1)}), \tag{2.16}$$

$F : \mathbb{R}^p \to \mathbb{R}^p$ $\mathcal{C}^1$ $B(0,\delta)$ $\delta \in (0,\infty)$

$$F(x) = x - (\mathcal{H}(x))^{-1} DJ(x), \tag{2.17}$$

$$\mathcal{H}(x) = 2A^TA + \beta G^T \operatorname{diag}\left(\left[\sigma(g_i^T x)\right]_{i=1}^r\right) G \succ 0, \tag{2.18}$$

DJ J (1.2)–(1.3).

(ii) $k \to \infty$ $\{x^{(k)}\}$ $\hat{x}$ J.

(iii) DF $\hat{x}$

$$\rho\left(DF(\hat{x})\right) \le K \max_{1\le i\le r} \mathcal{R}(|g_i^T\hat{x}|) \le K \sup_{0\le t\le \mu} \mathcal{R}(t) < 1, \tag{2.19}$$

$\mu = \max_{1\le i\le r} |g_i^T\hat{x}|$ $K \in [0,1]$ $\mathcal{R} : \mathbb{R}_+ \to [0,1]$

$$K = \frac{\beta\phi''(0)\left\|\!\left|G\right|\!\right\|^2}{2\nu^2 + \beta\phi''(0)\left\|\!\left|G\right|\!\right\|^2}, \tag{2.20}$$

$$\mathcal{R}(t) = 1 - \frac{\phi''(t)}{\sigma(t)}. \tag{2.21}$$

The shape of $\mathcal{R}$ for different functions ϕ can be seen in Table 4.2 and Figures 4.1 and 4.2. Notice that $\mathcal{R}$ is monotone increasing provided that ϕ'' is monotone decreasing on $\mathbb{R}_+$ and that the latter is true for all potential functions cited in (1.6)–(1.9).

. Let $\delta > 0$ be the radius exhibited in Remark 4 and let ζ be defined as in (2.11). The first step of iteration k is to calculate

$$b_i^{(k)} = \sigma(g_i^T x^{(k-1)}) \quad \forall i \in \{1,\dots,r\}. \tag{2.22}$$

Noticing that $b^{(k)}$ satisfies (2.13), the matrix $H(b^{(k)})$ defined by (2.7) is invertible (cf. Lemma 2.1). The second step of iteration k is to calculate $x^{(k)}$ using (2.8):

$$x^{(k)} = \left(H(b^{(k)})\right)^{-1} 2A^T y = \left(2A^T A + \beta G^T \mathrm{diag}(b^{(k)}) G\right)^{-1} 2A^T y. \tag{2.23}$$

Introducing (2.22) into (2.23) yields

$$x^{(k)} = F(x^{(k-1)}),$$

where $F : \mathbb{R}^p \to \mathbb{R}^p$ reads

$$F(x) = (\mathcal{H}(x))^{-1} 2A^T y, \tag{2.24}$$

and $\mathcal{H}$ is as given in (2.18). From (1.2)–(1.3),

$$DJ(x) = 2A^T Ax + \beta G^T \left[\phi'(g_i^T x)\right]_{i=1}^r - 2A^T y. \tag{2.25}$$

By (2.5), for every $i \in \{1, \ldots, r\}$ we can write

$$\phi'(g_i^T x) = \sigma(g_i^T x)\, g_i^T x.$$

Then (2.25) yields

$$\begin{aligned} 2A^T y &= 2A^T Ax + \beta G^T \mathrm{diag}\left(\left[\sigma(g_i^T x)\right]_{i=1}^r\right) Gx - DJ(x) \\ &= \mathcal{H}(x)\, x - DJ(x). \end{aligned}$$

Inserting this expression into (2.24) yields (2.17).

Let us fix $\delta > 0$ arbitrarily. Then define ζ according to (2.11). Based on Lemma 2.1, $b \to (H(b))^{-1}$ is $\mathcal{C}^1$ on $[\zeta, \phi''(0)\,]^r$ since $b \to H(b)$ is $\mathcal{C}^\infty$ and

$$D(H(b))^{-1} = -(H(b))^{-1} DH(b) (H(b))^{-1}$$

is continuous. Using (2.12) we have $[\sigma(g_i^T x)]_{i=1}^r \in [\zeta, \phi''(0)\,]^r$ for all $x \in B(0, \delta)$ and, since σ is $\mathcal{C}^1$ on $\mathbb{R}$ (see Remark 5), it follows that

$$x \to (\mathcal{H}(x))^{-1} = \left(H\left([\sigma(g_i^T x)]_{i=1}^r\right)\right)^{-1}$$

is $\mathcal{C}^1$ on $B(0, \delta)$. Combining the latter with the fact that J is $\mathcal{C}^2$ shows that F is $\mathcal{C}^1$ on $B(0, \delta)$. The proof of (i) is complete.

By Lemma 2.2, the sequence $\{x^{(k)}\}$ converges; let $\hat{x}$ denote its limit as $k \to \infty$. Since F is continuous, considering (2.16) for $k \to \infty$ yields

$$\hat{x} = F(\hat{x}). \tag{2.26}$$

Introducing this into (2.17) shows that $(\mathcal{H}(\hat{x}))^{-1} DJ(\hat{x}) = 0$ and consequently that $DJ(\hat{x}) = 0$. Since J has a unique minimizer (cf. Remark 1), it follows that J reaches its minimum at $\hat{x}$. Hence we prove (ii).

Next, we determine DF at an arbitrary x. Using (2.24),

$$\mathcal{H}(x) F(x) = 2A^T y \quad \forall x \in B(0, \delta).$$

Differentiating both sides of this equation with respect to x yields

$$D\mathcal{H}(x)F(x) + \mathcal{H}(x)DF(x) = 0 \quad \forall x \in B(0,\delta). \tag{2.27}$$

Then $DF(x) = -(\mathcal{H}(x))^{-1} D\mathcal{H}(x)F(x)$, where

$$\begin{aligned} D\mathcal{H}(x)F(x) &= \lim_{\epsilon\to 0} \frac{\mathcal{H}(x+\epsilon F(x)) - \mathcal{H}(x)}{\epsilon} = \beta G^T \mathrm{diag}\left(\left[\sigma'(g_i^T x)\; g_i^T F(x)\right]_{i=1}^r\right) G \\ &= -\beta G^T \mathrm{diag}\left(\left[\,\left|\sigma'(g_i^T x)\; g_i^T x\right|\,\right]_{i=1}^r\right) G \\ &\quad -\beta G^T \mathrm{diag}\left(\left[\sigma'(g_i^T x)\; g_i^T (x - F(x))\right]_{i=1}^r\right) G, \end{aligned}$$

where we use the fact that $\sigma'(t)t \le 0$ for all $t \in \mathbb{R}$. It follows that

$$DF(x) = M(x) + E(x), \tag{2.28}$$

where

$$M(x) = (\mathcal{H}(x))^{-1}\, \beta G^T \mathrm{diag}\left(\left[\;\left|\;\sigma'(g_i^T x)\; g_i^T x\;\right|\;\right]_{i=1}^r\right) G, \tag{2.29}$$

$$E(x) = (\mathcal{H}(x))^{-1}\, \beta G^T \mathrm{diag}\left(\left[\sigma'(g_i^T x)\; g_i^T\, (x - F(x))\right]_{i=1}^r\right) G. \tag{2.30}$$

The eigenvalues of $M(x)$, for $x \in \mathbb{R}^p$, are considered next.

LEMMA 2.4. [illegible] 2.3 [illegible] $M : \mathbb{R}^p \to \mathbb{R}^{p\times p}$ [illegible] (2.29) [illegible]

$$\rho(M(x)) \le K \max_{1\le i\le r} \mathcal{R}(|g_i^T x|) \quad \forall x \in \mathbb{R}^p, \tag{2.31}$$

[illegible] K [illegible] $\mathcal{R}$ [illegible] (2.20) [illegible] (2.21) [illegible]

2.4. Let us first focus on the function $\mathcal{R}$ defined in (2.21). Clearly, $\mathcal{R}(t) \le 1$ for all $t \in \mathbb{R}_+$ since $\phi''(t) \ge 0$ and $\sigma(t) > 0$ for all $t \in \mathbb{R}$. Using (2.5), (2.6), and (2.15), for any $t \ge 0$ we can write

$$\mathcal{R}(t) = \frac{1}{\sigma(t)}\left(\frac{\phi'(t)}{t} - \phi''(t)\right) = \frac{-\sigma'(t)\, t}{\sigma(t)}, \tag{2.32}$$

where in particular $\mathcal{R}(0) = 0$. For all $t \in \mathbb{R}_+$, Remark 3 condition 1 says that $\sigma'(t) \le 0$, and hence $\mathcal{R}(t) \ge 0$. This allows us to write down that

$$\mathcal{R}(|t|) = \frac{|\sigma'(t)t|}{\sigma(t)} \quad \forall t \in \mathbb{R}. \tag{2.33}$$

Given $x \in \mathbb{R}^p$, let $\lambda \in \mathbb{C}$ be the largest-in-magnitude eigenvector of $M(x)$. Let $u \in \mathbb{C}^p$ be such that $\|u\| = 1$ and $M(x)\; u = \lambda\; u$. Using the expression for M, we obtain

$$\beta(Gu)^T \mathrm{diag}\left(\left[\;\left|\sigma'(g_i^T x)\; g_i^T x\right|\;\right]_{i=1}^r\right) Gu = \lambda\; u^T\; \mathcal{H}(x)\; u \ge 0. \tag{2.34}$$

The last inequality comes from the fact that $G^T \mathrm{diag}\left(\left[\;\left|\sigma'(g_i^T x)\; g_i^T x\right|\;\right]_{i=1}^r\right) G$ is semi-positive definite and that $\mathcal{H}(x)$ is positive definite. It implies that λ is real with $\lambda \ge 0$

and that $u \in S$. If $\lambda = 0$, then (2.31) is satisfied. In the following, consider that $\lambda > 0$. Inserting (2.18) into (2.34) yields

$$\beta(Gu)^T \operatorname{diag}\Big(\big[\ |\sigma'(g_i^T x)\ g_i^T x| - \lambda\sigma(g_i^T x)\ \big]_{i=1}^r \Big)\, Gu = 2\lambda\|Au\|^2 \geq 2\lambda\nu^2 \geq 0.$$

Using (2.33), this is equivalent to

$$\beta \sum_{i=1}^r \sigma(g_i^T x)\left(\mathcal{R}(g_i^T x) - \lambda\right)\ \left(g_i^T u\right)^2 = 2\lambda\|Au\|^2 \geq 2\lambda\nu^2 \geq 0. \tag{2.35}$$

Let $j \in \{1, \ldots, r\}$ be such that

$$\sigma(g_j^T x)\left(\mathcal{R}(g_j^T x) - \lambda\right) \geq \sigma(g_i^T x)\left(\mathcal{R}(g_i^T x) - \lambda\right) \quad \forall i \in \{1, \ldots, r\}.$$

From (2.35) we find that $\sigma(g_j^T x)\left(\mathcal{R}(g_j^T x) - \lambda\right) \geq 0$. Using (2.35) yet again,

$$\beta\sigma(g_j^T x)\left(\mathcal{R}(g_j^T x) - \lambda\right)\|\!|G|\!\|^2 \geq \beta\sigma(g_j^T x)\left(\mathcal{R}(g_j^T x) - \lambda\right)\|Gu\|^2 \geq 2\lambda\nu^2.$$

Since $\sigma(t) \leq \phi''(0)$ for all $t \in \mathbb{R}$ (cf. Remark 3 condition 2), we obtain

$$\mathcal{R}(|g_j^T x|) - \lambda \geq \frac{2\lambda\nu^2}{\beta\phi''(0)\,\|\!|G|\!\|^2}.$$

It follows that

$$\lambda \leq K\mathcal{R}(|g_j^T x|) \leq K \max_{1\leq i\leq r} \mathcal{R}(|g_i^T x|),$$

where $K \leq 1$ is the constant given in (2.20). □

We pursue the proof of the theorem. From (2.26) it follows that $E(\hat{x}) = 0$, and hence

$$DF(\hat{x}) = M(\hat{x}). \tag{2.36}$$

On the other hand, the function $\mathcal{R}$ is continuous on $\mathbb{R}_+$ with $\mathcal{R}(0) = 0$. The first two inequalities in (2.19) are an immediate consequence of (2.36) and Lemma 2.4. If $\phi''(t) > 0$, for all $t \in \mathbb{R}$, (2.6) and (2.21) show that $\mathcal{R}(t) < 1$, for all $t \in \mathbb{R}_+$, and hence

$$\sup_{0\leq t\leq\mu} \mathcal{R}(t) < 1,$$

since $[0, \mu]$ is compact. Thus, the assumption in (1.10) ensures that

$$K < 1 \quad \text{and/or} \quad \sup_{0\leq t\leq\mu} \mathcal{R}(t) < 1,$$

and hence (2.19). □

REMARK 6. F (2.28) DF M E $\|\!|E(x^{(k)})|\!\|$ $\|x^{(k)} - F(x^{(k)})\|$ $x^{(k)}$ $\hat{x}$ $\rho\left(DF(x^{(k)})\right) > 1$ $\|\!|E(x^{(k)}|\!\|$ M DF $\hat{x}$ (2.19) $\eta > 0$

$$\sup_{x\in B(\hat{x},\eta)} \rho\left(DF(x)\right) < 1.$$

$k_0 \in \mathbb{N}$

$$x^{(k)} \in B(\hat{x}, \eta) \quad \forall k \geq k_0.$$

F $B(\hat{x}, \eta)$ $\hat{x}$

The assumption in (2.14) that ϕ is $\mathcal{C}^3$ near zero is only technical. The proof is easy to extend to functions ϕ which are $\mathcal{C}^2$ and have $\phi''(0^+)$ finite. Even if σ is nondifferentiable at zero, its side derivatives there are then finite. Noticing that the function $t \to \sigma(t)t$ is continuous, the analysis of M in Lemma 2.4 remains unchanged. An example of a function ϕ with σ nonsmooth at zero is $\phi(t) = |t|/\alpha - \log(1 + |t|/\alpha)$ since we have $\sigma'(0^+) = \alpha = -\sigma'(0^-)$. Extending the proof to functions ϕ as discussed above would considerably complicate the presentation.

COROLLARY 2.5. 2.3 $C(F, \hat{x})$ (1.22)

$$C(F, \hat{x}) \leq K \max_{1 \leq i \leq r} \mathcal{R}(|g_i^T \hat{x}|) \leq K \sup_{0 \leq t \leq \mu} \mathcal{R}(t) < 1,$$

$||.||$ (1.22)

It is easy to check that F satisfies all conditions required in the linear convergence theorem [26, p. 301]. By the latter theorem, $C(F, \hat{x}) = \rho(DF(\hat{x}))$. The result follows from Theorem 2.3. □

3. Additive form of half-quadratic regularization.

3.1. Preliminary facts.

This form is considered under the following conditions:

$$\begin{array}{ll} \text{(a)} & \phi \text{ is convex,} \\ \text{(b)} & c > 0 \text{ is such that } t \to ct^2/2 - \phi(t) \text{ is convex,} \\ \text{(c)} & \phi(t) = \phi(-t) \quad \forall t \in \mathbb{R}, \\ \text{(d)} & \phi \text{ is continuous on } \mathbb{R}, \\ \text{(e)} & \lim_{|t| \to \infty} \phi(t)/t^2 < c/2. \end{array} \tag{3.1}$$

By (a)–(b), ϕ is differentiable.[2] The dual function ψ is given by

$$\psi(s) = \sup_{t \in R} \left\{ -\frac{1}{2} \left(t\sqrt{c} - \frac{s}{\sqrt{c}} \right)^2 + \phi(t) \right\}, \tag{3.2}$$

where $c > 0$ is as specified in (3.1). Under (3.1), ψ is convex and the requirement in (1.12) is satisfied since

$$\phi(t) = \inf_{s \in R} \left\{ \frac{1}{2} \left(t\sqrt{c} - \frac{s}{\sqrt{c}} \right)^2 + \psi(s) \right\}. \tag{3.3}$$

The connections between ϕ and ψ under different assumptions on ϕ were considered in [3, 10, 14, 19]. For completeness, these are sketched in section 7.2 in the appendix.

[2] By (a), $\phi'(t^-) \leq \phi'(t^+)$ for all t. If for some t the latter inequality is strict, (b) cannot be satisfied for any c. If follows that $\phi'(t^-) = \phi'(t^+) = \phi'(t)$. For example, if we apply (3.2) to $\phi(t) = |t|$ with $c = 1$, we get $\psi(s) = |s| + 1/2$. Then (3.3) yields $\min_s \left(\psi(s) + (s-t)^2/2\right) \neq \phi(t)$.

According to (1.11) and (1.21), $\mathcal{J}$ reads

$$\begin{aligned} \mathcal{J}(x,b) &= \|Ax-y\|^2 + \beta \sum_{i=1}^{r} \left(\frac{1}{2} \left(\sqrt{c}\, g_i^T x - \frac{1}{\sqrt{c}}\, b_i \right)^2 + \psi(b_i) \right) \\ &= \|Ax-y\|^2 + \frac{\beta}{2} \left\| \sqrt{c}\, Gx - \frac{1}{\sqrt{c}}\, b \right\|^2 + \beta \sum_{i=1}^{r} \psi(b_i). \end{aligned} \tag{3.4}$$

The expression of σ for several functions ϕ is presented in Table 4.1. The minimizer function σ defined by (1.17) admits an explicit form [7, 10, 3] and its derivation is recalled in section 7.2 in the appendix:

$$\sigma(t) = ct - \phi'(t). \tag{3.5}$$

If (1.4) holds, the minimizer function χ defined by (1.18) reads

$$\chi(b) = \mathcal{H}^{-1} \left(2A^T y + \beta G^T b \right), \tag{3.6}$$

where

$$\mathcal{H} = 2A^T A + \beta c\, G^T G \succ 0 \tag{3.7}$$

is invertible thanks to (1.4).

As exhibited in [19], $\mathcal{J}$ is convex on $\mathbb{R}^p \times \mathbb{R}^r$, and hence the convergence of alternate minimization towards the minimizer $\hat{x}$ of J. Convergence when x and y are defined on a bounded domain of $\mathbb{R}^2$—in the context of image restoration—was established in [3] under appropriate assumptions on ϕ.

3.2. Convergence analysis. The analysis here is developed in a similar fashion as for the multiplicative form.

THEOREM 3.1. [illegible] J [illegible] (1.2)–(1.3) [illegible] (1.4) [illegible] (1.10) [illegible] ϕ [illegible] $\mathcal{C}^2$ [illegible] $\mathbb{R}$ [illegible] (3.1) [illegible] $\{(x^{(k)}, b^{(k)})\}$ [illegible] (1.16) [illegible] σ [illegible] χ [illegible] (3.5) [illegible] (3.6)–(3.7) [illegible]

(i) [illegible] $k \in \mathbb{N}$ [illegible]

$$x^{(k)} = F(x^{(k-1)}), \tag{3.8}$$

[illegible] $F : \mathbb{R}^p \to \mathbb{R}^p$ [illegible] $\mathcal{C}^1$ [illegible]

$$F(x) = x - \mathcal{H}^{-1} DJ(x), \tag{3.9}$$

$$\mathcal{H} = 2A^T A + \beta c G^T G \succ 0; \tag{3.10}$$

(ii) [illegible] $x \in \mathbb{R}^p$

$$\rho\left(DF(x)\right) \leq K \max_{1 \leq i \leq r} \mathcal{R}(|g_i^T x|) \leq K \sup_{0 \leq t \leq \mu} \mathcal{R}(t) < 1, \tag{3.11}$$

[illegible] $\mu = \max_{1 \leq i \leq r} |g_i^T x|$ $K \in [0,1]$ [illegible] $\mathcal{R} : \mathbb{R}_+ \to [0,1]$ [illegible]

$$K = \frac{\beta c \left\| G \right\|^2}{2\nu^2 + \beta c \left\| G \right\|^2}, \tag{3.12}$$

$$\mathcal{R}(t) = 1 - \frac{\phi''(t)}{c}; \tag{3.13}$$

(iii) [illegible] $k \to \infty$ $\{x^{(k)}\}$ [illegible] $\hat{x}$ [illegible] J

The function $\mathcal{R}$ for several potential functions ϕ is presented in Table 4.2 and Figures 4.1 and 4.2. For all potential functions cited in (1.6)–(1.9), ϕ'' is monotone decreasing on $\mathbb{R}_+$, and hence $\mathcal{R}$ is monotone increasing on $\mathbb{R}_+$.

[illegible] Given $x^{(k-1)}$, iterate $b^{(k)}$ is calculated using (3.5):

$$b^{(k)} = \left[\sigma(g_i^T x^{(k-1)})\right]_{i=1}^r = cGx^{(k-1)} - \left[\phi'(g_i^T x^{(k-1)})\right]_{i=1}^r. \tag{3.14}$$

Next, $x^{(k)}$ is calculated using (3.6)–(3.7):

$$x^{(k)} = \mathcal{H}^{-1}(2A^T y + \beta G^T b^{(k)}).$$

Inserting (3.14) into the expression given above yields

$$x^{(k)} = F(x^{(k-1)}), \tag{3.15}$$

where $F : \mathbb{R}^p \to \mathbb{R}^p$ reads

$$F(x) = \mathcal{H}^{-1}\left(2A^T y + \beta c G^T G x - \beta G^T \left[\phi'(g_i^T x)\right]_{i=1}^r\right). \tag{3.16}$$

The expression for DJ given in (2.25) yields

$$\begin{aligned} 2A^T y &= 2A^T A x + \beta G^T \left[\phi'(g_i^T x)\right]_{i=1}^r - DJ(x) \\ &= \mathcal{H}x - \beta c G^T G x + \beta G^T \left[\phi'(g_i^T x)\right]_{i=1}^r - DJ(x). \end{aligned}$$

Introducing this into (3.16) leads to $F(x) = \mathcal{H}^{-1}\left(\mathcal{H}x - DJ(x)\right)$, and hence to (3.9). Since J is $\mathcal{C}^2$, it is straightforward that F is $\mathcal{C}^1$ on $\mathbb{R}^p$. The proof of (i) is complete.

The differential of F at an arbitrary $x \in \mathbb{R}^p$ reads

$$\begin{aligned} DF(x) &= I - \mathcal{H}^{-1}\left(2A^T A + \beta G^T \operatorname{diag}\left(\left[\phi''(g_i^T x)\right]_{i=1}^r\right) G\right) \\ &= \beta c \mathcal{H}^{-1} G^T \operatorname{diag}\left(\left[\mathcal{R}(g_i^T x)\right]_{i=1}^r\right) G, \end{aligned} \tag{3.17}$$

where $\mathcal{R}$ was introduced in (3.13). The eigenvalues of $DF(x)$, for any $x \in \mathbb{R}^p$, are analyzed below.

LEMMA 3.2. [illegible] 3.1 [illegible]

$$\rho(DF(x)) \le K \max_{1 \le i \le r} \mathcal{R}(|g_i^T x|) \quad \forall x \in \mathbb{R}^p, \tag{3.18}$$

[illegible] K [illegible] $\mathcal{R}$ [illegible] (3.12) [illegible] (3.13) [illegible]

3.2. Consider first the function $\mathcal{R}$ introduced in (3.13). Since ϕ is $\mathcal{C}^2$, the constant c in (3.1)(b) satisfies

$$\phi''(t) \le c \quad \forall t \in \mathbb{R}, \tag{3.19}$$

and hence $0 \le \mathcal{R}(t) \le 1$ for all $t \in \mathbb{R}$.

For $x \in \mathbb{R}^p$, let $\lambda \in \mathbb{C}$ be the largest-in-magnitude eigenvector of $DF(x)$. Let $u \in \mathbb{C}^p$ be such that $\|u\| = 1$ and $DF(x)u = \lambda u$. Then using (3.17) we get

$$\beta c (Gu)^T \operatorname{diag}\left(\left[\mathcal{R}(g_i^T x)\right]_{i=1}^r\right) Gu = \lambda u^T \mathcal{H} u \ge 0. \tag{3.20}$$

The last inequality above comes from the facts that $\mathcal{H}$ is positive definite and that $G^T \operatorname{diag}\left(\left[\mathcal{R}(g_i^T x)\right]_{i=1}^r\right) G$ is semipositive definite. It shows that $\lambda \geq 0$ and that $u \in S$. If $\lambda = 0$, then (3.18) is satisfied. In the following we consider that $\lambda > 0$. Introducing (3.10) into (3.20) yields

$$\beta c \sum_{i=1}^{r} \left(\mathcal{R}(g_i^T x) - \lambda\right) \left(g_i^T u\right)^2 = 2\lambda \|Au\|^2 \geq 2\lambda \nu^2 \geq 0. \tag{3.21}$$

Let $j \in \{1, \ldots, r\}$ be such that

$$\mathcal{R}(g_j^T x) - \lambda \geq \mathcal{R}(g_i^T x) - \lambda \quad \forall i \in \{1, \ldots, r\}.$$

Then (3.21) shows that $\mathcal{R}(g_j^T x) - \lambda \geq 0$ and we can write down that

$$\beta c \left(\mathcal{R}(g_j^T x) - \lambda\right) \|\!|G|\!\|^2 \geq \beta c \left(\mathcal{R}(g_j^T x) - \lambda\right) \|Gu\|^2 \geq 2\lambda \|Au\|^2 \geq 2\lambda \nu^2.$$

It follows that

$$\lambda \leq \frac{\beta c \|\!|G|\!\|^2}{2\nu^2 + \beta c \|\!|G|\!\|^2} \mathcal{R}(|g_j^T x|) \leq K \max_{1 \leq i \leq r} \mathcal{R}(|g_i^T x|).$$

The lemma is proved. □

We pursue the proof of the theorem. If $\phi''(t) > 0$ for all $t \in \mathbb{R}$, noticing that $\mathcal{R}$ is continuous on $\mathbb{R}$ and that $[0, \mu]$ is compact, (3.13) and (3.19) show that $\sup_{0 \leq t \leq \mu} \mathcal{R}(t) < 1$. By the assumption given in (1.10) we have

$$K < 1 \quad \text{and/or} \quad \sup_{0 \leq t \leq \mu} \mathcal{R}(t) < 1.$$

Combining this with Lemma 3.2 proves (ii).

Let $\delta \in (0, \infty)$ be the radius exhibited in Remark 4. Define $\eta = \delta \|\!|G|\!\|$; then

$$|g_i^T x| \leq \eta \quad \forall i \in \{1, \ldots, r\}, \ \forall x \in B(0, \delta).$$

Using Lemma 3.2 yet again,

$$\sup_{x \in B(0,\delta)} \rho(DF(x)) \leq K \sup_{x \in B(0,\delta)} \max_{1 \leq i \leq r} \mathcal{R}(|g_i^T x|) \leq K \sup_{0 \leq t \leq \eta} \mathcal{R}(t) < 1. \tag{3.22}$$

It follows that the sequence $\{x^{(k)}\}$ converges; let $\lim_{k \to \infty} x^{(k)} = \hat{x}$. Since F is continuous, letting $k \to \infty$ in (3.8) shows that $\hat{x} = F(\hat{x})$. Introducing the latter result into (3.9) shows that $DJ(\hat{x}) = 0$. Since J has a unique minimizer (Remark 1), it follows that J reaches its minimum at $\hat{x}$. Hence we prove (iii). □

COROLLARY 3.3. [illegible] 3.1 [illegible] $C(F, \hat{x})$ [illegible] (1.22) [illegible]

$$C(F, \hat{x}) \leq K \max_{1 \leq i \leq r} \mathcal{R}(|g_i^T \hat{x}|) \leq K \sup_{0 \leq t \leq \mu} \mathcal{R}(t) < 1,$$

[illegible] $\|.\|$ [illegible] (1.22)

[illegible]. It is easy to see that the linear convergence theorem [26] can be applied, which shows that $C(F, \hat{x}) = \rho(DF(\hat{x}))$. The result follows from Theorem 3.1. □

REMARK 7 (the optimal value of c). (3.19) $c \geq \sup_{t\in\mathbb{R}} \phi''(t)$ ψ (3.2) (3.12) (3.13) K $\mathcal{R}(t)$ $t \in \mathbb{R}$ c c c

$$c = \sup_{t\in\mathbb{R}} \phi''(t). \tag{3.23}$$

Noticing that edge-preserving functions ϕ satisfy $\phi''(0) = \sup_{t\in\mathbb{R}} \phi''(t)$, (3.23) becomes

$$c = \phi''(0). \tag{3.24}$$

This recommendation is corroborated by the experiments in section 5.

3.3. Fast additive form based on nonsmooth dual function. We now concentrate on the Huber potential function ϕ:

$$\phi(t) = \begin{cases} \frac{1}{2}t^2 & \text{if} \quad |t| \leq \alpha, \\ \alpha|t| - \frac{1}{2}\alpha^2 & \text{if} \quad |t| > \alpha. \end{cases} \tag{3.25}$$

This potential function is edge preserving, since it is affine beyond α. Unlike total variation regularization $\phi(t) = |t|$ which yields a stair-casing effect, the quadratic part of ϕ in (3.25) near 0 allows smoothly varying regions in $\hat{x}$ to be restored [25]. Homogeneous regions in $\hat{x}$ correspond with $|g_i^T \hat{x}| \leq \alpha$ while edges correspond with $|g_i^T \hat{x}| > \alpha$.

REMARK 8. $A^T A$ $y \in \mathbb{R}^q$ J $\hat{x}$ $D^2 J$ $\hat{x}$ J $A^T A$ ϕ'' $-\alpha$ α $A^T A$ $y \in \mathbb{R}^q$ $\hat{x}$ $|g_i^T \hat{x}| = \alpha$ i $\mathbb{R}^q$ y J $\mathcal{C}^\infty$ $\hat{x}$ $A^T A$

From (3.25) we have $\max_{t\in\mathbb{R}}\{\phi''(t^-), \phi''(t^+)\} = 1$. For $c > 1$, the dual ψ reads

$$\psi(s) = \begin{cases} \dfrac{s^2}{(2c(c-1))} & \text{if} \quad |s| \leq (c-1)\alpha, \\ \dfrac{\alpha|s|}{c - \alpha^2(c-1)/(2c)} & \text{if} \quad |s| > (c-1)\alpha. \end{cases}$$

Previous papers [7, 19] recommend taking $c > 1$ in order to ensure that ψ is $\mathcal{C}^1$. Using (3.5), the minimizer function σ is

$$\sigma(t) = \begin{cases} (c-1)t & \text{if} \quad |t| \leq \alpha, \\ ct - \alpha \operatorname{sign}(t) & \text{if} \quad |t| > \alpha. \end{cases} \tag{3.26}$$

Instead, for $c = 1$, we get

$$\psi(s) = \alpha|s|. \tag{3.27}$$

It is easy to check[3] that applying (3.3) leads to ϕ as given in (3.25). The function σ now takes a particularly simple form:

$$\sigma(t) = \begin{cases} 0 & \text{if} \quad |t| \le \alpha, \\ t - \alpha \ \text{sign}(t) & \text{if} \quad |t| > \alpha. \end{cases} \tag{3.28}$$

The computation of each $b_i^{(k)} = \sigma(g_i^T x^{(k)})$ using (3.28) is much easier than using (3.26). Typically we have $|g_i^T x^{(k)}| \le \alpha$ for most of the indexes $i \in \{1, \dots, r\}$ since most of the pixels are in homogeneous regions. The relevant $b_i^{(k)}$ are set to zero after verifying that $|g_i^T x^{(k)}| \le \alpha$: thus, they need no calculation.

The augmented cost-function $\mathcal{J}$ now reads

$$\mathcal{J}(x, s) = \|Ax - y\|^2 + \beta \sum_{i=1}^{r} \left(\frac{1}{2}(g_i^T x - s_i)^2 + |s_i| \right). \tag{3.29}$$

The convergence of alternate minimization is easily deduced from Theorem 1.4, p. 73 in [16]. Furthermore, Theorem 3.1 can be extended to potential functions ϕ which satisfy (3.1) and which at several points α_i have $\phi''(\alpha_i^-)$ and $\phi''(\alpha_i^+)$ different but finite. Instead of DF, the subdifferential of F must be considered in (3.17). The inequalities presented in the proof of Theorem 3.1 remain basically the same.

4. Theoretical comparisons of two forms.

The expressions for σ and $\mathcal{R}$ for both forms are displayed in Tables 4.1 and 4.2.

4.1. Extension to alternative regularization terms. In some image processing applications, the regularization term Φ in (1.2) reads [2]

$$\Phi(x) = \sum_{i=1}^{p} \phi(\|G_i x\|), \tag{4.1}$$

where every $G_i x$ is a vector approximating the gradient of x at pixel i. We wish to know whether the resultant J can be minimized using HQ regularization methods. According to (1.11)–(1.12), an auxiliary variable b_i is associated with each $\|G_i x\|$.

- In the multiplicative form of HQ regularization, the augmented cost-function $\mathcal{J}$ is similar to (2.4):

$$\mathcal{J}(x, b) = \|Ax - y\|^2 + \beta \sum_{i=1}^{p} \left(\frac{b_i}{2} \ \|G_i x\|^2 + \psi(b_i) \right).$$

  Notice that $\mathcal{J}(x, .)$ is separable and the minimization over b is given by p scalar minimizer functions σ of the form (2.5). Since the function $\mathcal{J}(., b)$ is quadratic, the minimizer function χ, defined in (1.18), is linear in y and has an explicit form similar to (2.7)–(2.8). Minimization using the multiplicative form of HQ regularization is straightforward.

---

[3] Given $t \in \mathbb{R}$, define $h_t : \mathbb{R} \to \mathbb{R}$ by $h_t(s) = (s - t)^2/2 + \alpha|s|$. The function h_t reaches its minimum at $\hat{s} \in \mathbb{R}$ if and only if [18]

$$\begin{array}{lll} \hat{s} = 0 & \Rightarrow & h_t'(0^-) \le 0 \le h_t'(0^+), \quad \text{i.e.,} \quad -t - \alpha \le 0 \le -t + \alpha; \\ \hat{s} \neq 0 & \Rightarrow & h_t'(\hat{s}) = 0, \quad \text{i.e.,} \quad \hat{s} - t + \alpha \text{sign}(\hat{s}) = 0. \end{array}$$

It is easily deduced that

$$\begin{array}{lllll} |t| \le \alpha & \Rightarrow & \hat{s} = 0 & \Rightarrow & \inf_s h_t(\hat{s}) = t^2/2, \\ |t| > \alpha & \Rightarrow & \hat{s} = t - \alpha \text{sign(t)} & \Rightarrow & \inf_s h_t(\hat{s}) = \alpha|t| - \alpha^2/2. \end{array}$$

In other words, $\inf_{s \in R} \left((s - t)^2/2 + \alpha|s| \right) = \phi(t)$ where ϕ is given in (3.25).

TABLE 4.1
Minimizer function σ relevant to multiplicative and additive form of HQ regularization (cf. (2.5) and (3.5)) for different potential functions ϕ.

Potential function	Multiplicative form	Additive form
$\phi(t) = \lvert t\rvert^{\alpha}, \quad \alpha \in (1,2]$	$\sigma(t) = \alpha\lvert t\rvert^{\alpha-2}$	not applicable
$\phi(t) = \sqrt{\alpha + t^2}$	$\sigma(t) = \dfrac{1}{\sqrt{\alpha + t^2}}$	$\sigma(t) = ct - \dfrac{t}{\sqrt{\alpha + t^2}}$
$\phi(t) = \dfrac{\lvert t\rvert}{\alpha} - \log\left(1 + \dfrac{\lvert t\rvert}{\alpha}\right)$	$\sigma(t) = \dfrac{1}{\alpha(\alpha + \lvert t\rvert)}$	$\sigma(t) = ct - \dfrac{t}{\alpha(\alpha + \lvert t\rvert)}$
$\phi(t) = \begin{cases} \dfrac{t^2}{2}, & \lvert t\rvert \le \alpha \\ \alpha\lvert t\rvert - \dfrac{\alpha^2}{2}, & \lvert t\rvert > \alpha \end{cases}$	$\sigma(t) = \begin{cases} 1, & \lvert t\rvert \le \alpha \\ \dfrac{\alpha}{\lvert t\rvert}, & \lvert t\rvert > \alpha \end{cases}$	$\sigma(t) = \begin{cases} (c-1)t, & \lvert t\rvert \le \alpha \\ ct - \alpha\,\mathrm{sign(t)}, & \lvert t\rvert > \alpha \end{cases}$
$\phi(t) = \log(\cosh(\alpha t))$	$\sigma(t) = \alpha\dfrac{\tanh(\alpha t)}{t}$	$\sigma(t) = ct - \alpha\tanh(\alpha t)$

- In the additive form of HQ regularization, $\mathcal{J}$ reads

$$\mathcal{J}(x,b) = \|Ax - y\|^2 + \beta \sum_{i=1}^{p} \left(\frac{1}{2}\left(\|G_i x\| - b_i\right)^2 + \psi(b_i) \right),$$

  where for simplicity we take $c = 1$. Now the function $\mathcal{J}(.,b)$ is nonquadratic since it is nonsmooth. The minimizer function χ defined by (1.18) is nonlinear in y and is difficult to calculate numerically. There is no advantage to consider minimization using the additive form of HQ regularization when Φ is of the form (4.1).

4.2. The bound on convergence rate. The minimization of a J of the form (1.2)–(1.3) using either the multiplicative or the additive form of HQ regularization amounts to constructing a sequence of iterates $\{x^{(k)}\}$ as

$$x^{(k)} = F(x^{(k-1)}), \tag{4.2}$$

where $F : \mathbb{R}^p \to \mathbb{R}^p$ is of the form

$$F(x) = x - (\mathcal{H}(x))^{-1} DJ(x).$$

The matrix $\mathcal{H}(x)$ here provides a correction of the steepest descent direction $-DJ(x)$. Recall that the classical Newton method corresponds to $\mathcal{H}(x) = D^2J(x)$, where

$$D^2J(x) = 2A^TA + \beta G^T \mathrm{diag}\left(\left[\phi''(g_i^T x)\right]_{i=1}^{r} \right) G. \tag{4.3}$$

In the multiplicative form, the matrix $\mathcal{H}(x)$ given in (2.18) differs from $D^2J(x)$ only in that the entries of the diagonal matrix above are $\sigma(g_i^T x) = \phi'(g_i^T x)/g_i^T x$. Since $\sigma(t) \approx \phi''(t)$ if $t \approx 0$, the entries relevant to small differences $|g_i^T x|$ are almost the

same for $\mathcal{H}$ and for D^2J. For $|g_i^T x|$ large, $\phi''(g_i^T x) \approx 0$ because ϕ is edge preserving; such entries heavily deteriorate the conditioning of D^2J. On the other hand, (2.21) shows that $\sigma(t) > \phi''(t)$ if $t \neq 0$. In fact, $\sigma(g_i^T x)$ remains larger than $\phi''(g_i^T x)$ for large differences $|g_i^T x|$, thus preventing $\mathcal{H}(x)$ from being as ill-conditioned as $D^2J(x)$.

In the additive form, $\mathcal{H}$ is constant for all x; cf. (3.10). If we choose $c = \phi''(0)$, as recommended in (3.24), then $\mathcal{H} \approx D^2J(\mathbb{1})$. Here again, the entries of the diagonal matrix in (4.3) relevant to small differences $|g_i^T x| \approx 0$ are the same for D^2J and $\mathcal{H}$.

The calculation of $x^{(k)}$, as given by (4.2), can also be put into the form

$$x^{(k)} = F^k(x^{(0)}), \quad \text{where} \quad F^k = \underbrace{F \circ \cdots \circ F}_{k \text{ times}}.$$

In the multiplicative form of HQ regularization, F is not necessarily a contraction except in the vicinity of the minimizer $\hat{x}$—see Remark 6. We can expect that in early iterations the decrease of J is relatively slow. In the additive form of HQ regularization, Theorem 3.1 and Lemma 3.2 show that F is a contraction.

By (2.19) and (3.11), the convergence rate relevant to the multiplicative and the additive forms of HQ regularization is upper bounded by a term of the form

$$K \sup_{|t| \leq \mu} \mathcal{R}(t). \tag{4.4}$$

If in the additive form we choose $c = \phi''(0)$, as recommended in (3.24), then the constant K is the same for both forms; cf. (2.20) and (3.12). We notice that K decreases when ν^2 increases, which means that convergence is faster if $A^T A$ is well-conditioned. For $A^T A$ singular, $K = 1$. Furthermore, comparing (2.21) and (3.13) shows that the function $\mathcal{R}$ has a similar expression for both the multiplicative and the additive forms of HQ regularization. Moreover, for all edge-preserving functions ϕ used in practice $\mathcal{R}$ is monotone increasing on $\mathbb{R}_+$ since ϕ'' is monotone decreasing on $\mathbb{R}_+$. In the multiplicative form, using that $\mathcal{R}(t) \geq 0$ for all $t \in \mathbb{R}_+$ and Remark 3, we find that

$$\phi''(t) \leq \sigma(t) \leq \phi''(0) \quad \forall t \in \mathbb{R}.$$

These inequalities are strict in the usual situation when $t \to \phi(\sqrt{t})$ is strictly concave and $\phi''(t) > 0$ for all $t \in \mathbb{R}$. In such a case, (2.21) and (3.13) lead to

$$\underset{\text{multipl. form}}{\mathcal{R}(t)} < \underset{\text{additive form}}{\mathcal{R}(t)} \qquad \forall t \in \mathbb{R} \setminus \{0\},$$

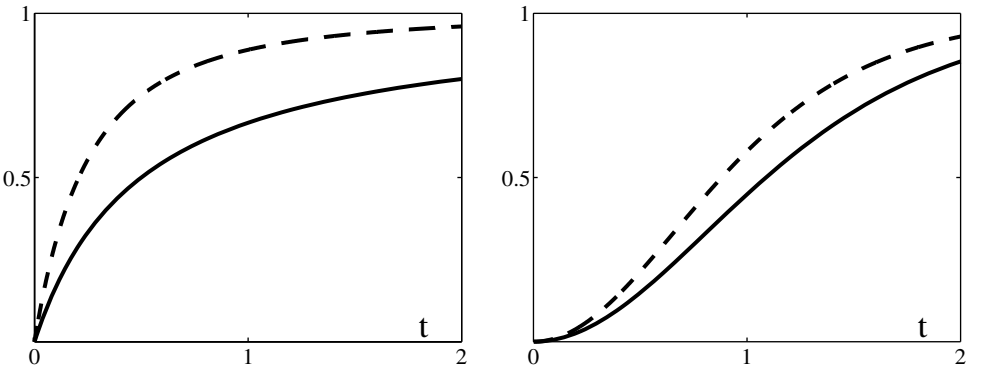
with $\mathcal{R}(0) = 0$ for both forms. Table 4.2 and Figures 4.1 and 4.2 illustrate this result. This suggests that the multiplicative form needs fewer iterations than the additive form in order to reach the minimizer $\hat{x}$. This is corroborated by the experiments presented in section 5. In Figure 4.2, we show $\mathcal{R}$ corresponding to $\phi(t) = \sqrt{\alpha + t^2}$, for $\alpha = 0.5$ and $\alpha = 0.05$. It is worth noticing that for both forms, $\mathcal{R}$ decreases when α increases, hence convergence is faster if α is larger. The constant $\mu \in (0, \infty)$ in (4.4) is determined essentially by the magnitude of the differences $|g_i^T x|$. Since $\mathcal{R}$ is monotone increasing on $\mathbb{R}_+$, convergence is slower if $\hat{x}$ involves large differences.

4.3. Computational cost. In both the multiplicative and the additive forms of HQ regularization, the calculation of $b^{(k)}$ has basically the same complexity. In the multiplicative form, the new iterate $x^{(k)}$ is the solution of a linear system

$$\mathcal{H}(b^{(k)})x = 2A^T y, \tag{4.5}$$

TABLE 4.2
The convergence factor $\mathcal{R}$ for multiplicative and additive HQ regularization; cf. (2.21) *and* (3.13).

Potential function	$\mathcal{R}$ for the multiplicative form	$\mathcal{R}$ for the additive form
$\phi(t) = \sqrt{\alpha + t^2}$	$\dfrac{t^2}{\alpha + t^2}$	$1 - \dfrac{\alpha}{c(\alpha + t^2)\sqrt{\alpha + t^2}}$
$\phi(t) = \lvert t\rvert/\alpha - \log(1 + \lvert t\rvert/\alpha)$	$\dfrac{t}{\alpha + t}$	$1 - \dfrac{1}{c(\alpha + t)^2}$
$\phi(t) = \begin{cases} \dfrac{t^2}{2}, & \lvert t\rvert \le \alpha \\ \alpha\lvert t\rvert - \dfrac{\alpha^2}{2}, & \lvert t\rvert > \alpha \end{cases}$	$\begin{cases} 0, & t \le \alpha \\ 1, & t > \alpha \end{cases}$	$\begin{cases} 1 - \dfrac{1}{c}, & t \le \alpha \\ 1, & t > \alpha \end{cases}$
$\phi(t) = \log(\cosh(\alpha t))$	$1 - \dfrac{\alpha t}{\sinh(\alpha t)\cosh(\alpha t)}$	$1 - \dfrac{\alpha^2(1 - \tanh^2(\alpha t))}{c}$



$\phi(t) = \dfrac{|t|}{\alpha} - \log\left(1 + \dfrac{|t|}{\alpha}\right)$, $\alpha = 0.5$. $\phi(t) = \log(\cosh(\alpha t))$ for $\alpha = 1$.

FIG. 4.1. *The shape of $\mathcal{R}$ for multiplicative (solid lines) and for additive (dotted lines) form of HQ regularization corresponding to different functions ϕ. In the additive form, $c = \phi''(0)$.*

where $\mathcal{H}(b^{(k)})$ is given in (2.7). We note that the matrix $\mathcal{H}(b^{(k)})$ is changing, but the right-hand side is fixed at each iteration. This system must be solved at each iteration. When p is small, (4.5) can be solved using Gaussian elimination. The complexity is then $O(p^3)$. In signal and image deblurring, A is a Toeplitz matrix. However, $\mathcal{H}(b^{(k)})$ is not necessarily Toeplitz: no benefit can be made from the special structure of A. When p is large, the conjugate gradient method can be used to solve (4.5). Although invertible, the matrix $\mathcal{H}(b^{(k)})$ can be ill-conditioned and/or there are a number of clusters in the small eigenvalues of the Hessian matrix. Solving (4.5) then needs a large number of iterations.

In the additive form, $x^{(k)}$ is the solution of

$$\mathcal{H}\, x = 2A^T y + \beta G^T b^{(k)}, \tag{4.6}$$

where $\mathcal{H}$ is given in (3.7). Now $\mathcal{H}$ is fixed, but the right-hand side is changing at each iteration. If p is small, we can compute $\mathcal{H}^{-1}$ before starting iterations and

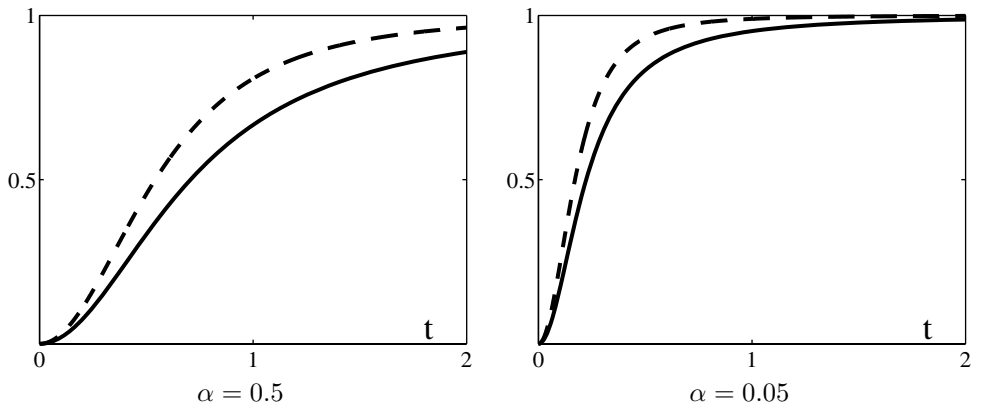


FIG. 4.2. *The shape of $\mathcal{R}$ as a function of α for $\phi(t) = \sqrt{\alpha + t^2}$. Multiplicative form (solid lines) and additive form (dotted lines) for $c = \phi''(0)$.*

then, at each iteration, $x^{(k)} = \mathcal{H}^{-1}\left(2A^T y + \beta G^T b^{(k)}\right)$. A stable way to solve (4.6) is to first compute the Cholesky factorization of $\mathcal{H}$ and then solve the corresponding triangular systems at each iteration. The factorization requires $O(p^3)$ operations, but the complexity of solving a triangular system is only $O(p^2)$. Observe that if A is a Toeplitz matrix, $\mathcal{H}$ is a Toeplitz-like matrix [20, 6] as well. We note that $A^T A$ is a Toeplitz-like matrix and G is the discretization matrix of the first-order or the second-order differential operator. There are many direct or iterative Toeplitz solvers with costs ranging from $O(p \log p)$ to $O(p^2)$ [20, 6]. When p is large, the conjugate gradient method can be used to solve (4.6) at each iteration. Convergence can be improved using preconditioning techniques. When A is a Toeplitz-type or block-Toeplitz-Toeplitz-block-type matrix, transform-based preconditioning techniques have been proven to be very successful [24]. For example, if A is a blurring matrix generated by a symmetric point spread function, $\mathcal{H}$ can be diagonalized by a fast transform matrix; then (4.6) is solved by using three fast transforms in $O(p \log p)$ operations [24].

In conclusion, the computational cost of each iteration for the additive form of HQ regularization is smaller than for the multiplicative form. Moreover, it can substantially be improved using fast solvers and preconditioning.

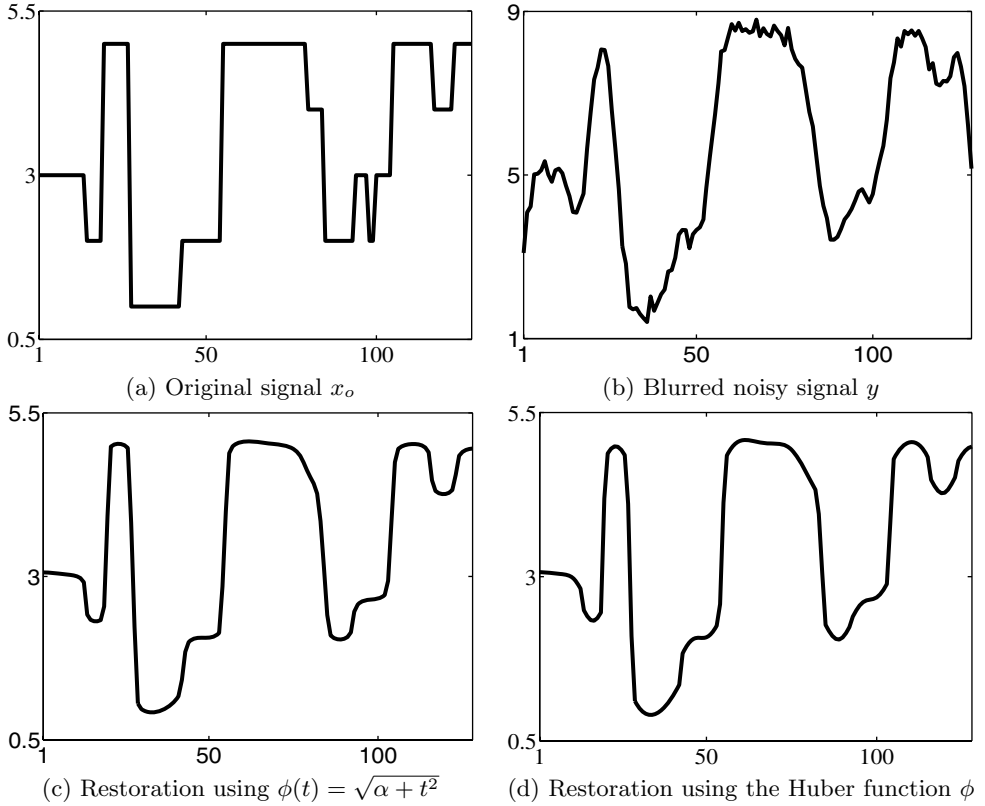
5. Numerical results.

5.1. One-dimensional signal. Our goals are first to compare the convergence properties of the two forms of HQ regularization and second to compare them with standard minimization methods. All the computations are done in MATLAB 6.0. The MATLAB optimization routines are compared with our written MATLAB codes. Their computational times are reported in the tables in this section.

We give average results based on the restoration of 1000 generated original 128-length signals from blurred noisy data. If x_o is such an original signal, observed data read $y = Ax_o + n$, where

$$A_{i,j} = \gamma e^{-0.1(i-j)^2}, \quad i, j = 1, 2, \ldots, p,$$

γ is a normalization constant such that $\sum_{i=1}^{p} A_{i,1} = 1$, and n is white Gaussian noise yielding a signal-to-noise ratio (SNR) of 40dB. In Figure 5.1(a) we show an example of a randomly generated original signal x_o and in Figure 5.1(b) data y generated as explained above. For all minimization methods, the stopping criterion is



(a) Original signal x_o (b) Blurred noisy signal y

(c) Restoration using $\phi(t) = \sqrt{\alpha + t^2}$ (d) Restoration using the Huber function ϕ

FIG. 5.1. *Example of restoration of a randomly generated original signal* (a) *from blurred noisy data* (b). *Restored signals:* (c) $\phi(t) = \sqrt{\alpha + t^2}$, *and* $\alpha = 0.001$, $\beta = 0.8$; (d) $\phi(t) = t^2/2$ *if* $|t| \leq \alpha$, $\phi(t) = \alpha|t| - \alpha^2/2$ *if* $|t| > \alpha$, *where* $\alpha = 0.1$, $\beta = 10$.

$||x^{(k)} - x^{(k-1)}||_2 < 1 \times 10^{-4}/||x_o||$, while the maximum iteration number is 1000. For the restoration of each signal, we minimize cost-functions J defined using different potential functions ϕ and different sets of parameters (α, β). Figures 5.1(c) and (d) show two restorations obtained from the data in Figure 5.1(b). Each cost-function J is minimized using the multiplicative form of HQ regularization, the additive form of HQ regularization where $c = \phi''(0)$, as well as using MATLAB optimization routines "fminunc.m." In the latter optimization procedure, we use quasi-Newton updating schemes "bfgs" or "dfp," the line search algorithm "cubicpoly," as well as the steepest descent scheme "steepdesc." Tables 5.1–5.5 summarize the average number of iterations (the first row of each set of parameters) and the average computational times in seconds (the second row of each set of parameters) for each minimization method. Below we briefly comment on the obtained results.

- The multiplicative form of HQ regularization allows us to minimize J using a smaller number of iterations than the additive form of HQ regularization.
- The overall computational times used by the additive form are less than those used by the multiplicative form.
- The performance of MATLAB optimization routines are poorer than those of both forms of HQ regularization in our MATLAB implementation. These results must be qualified by the observation that MATLAB run times for the

TABLE 5.1
Average number of iterations (upper row) and computational times (lower row) for $\phi(t) = \sqrt{\alpha + t^2}$.

β	α	Multiplicative	Additive	BFGS	DFP	Steepest
0.8	0.00025	79.6	228.2	122.8	110.3	> 1000
		5.28	2.63	11.12	8.77	$> 36.\ 73$
0.8	0.0005	74.6	207.4	110.5	101.6	> 1000
		4.94	2.39	9.96	8.08	> 36.71
0.8	0.001	68.8	182.9	100.1	89.4	990.2
		4.55	2.17	9.13	7.11	37.17
0.8	0.002	63.3	188.3	85.2	77.9	835.6
		4.15	2.19	7.79	6.23	30.81
0.8	0.004	56.4	181.4	76.0	65.4	697.5
		3.76	2.14	6.96	5.28	25.87
0.2	0.001	64.1	258.6	83.0	74.2	775.6
		4.24	2.90	7.62	6.03	28.49
0.4	0.001	64.7	226.3	92.9	81.7	886.5
		4.30	2.56	8.48	6.60	32.36
0.8	0.001	63.3	188.3	85.2	77.9	835.6
		4.15	2.19	7.79	6.23	30.81
1.6	0.001	72.6	174.0	111.8	100.4	> 1000
		4.78	2.00	10.06	7.98	> 36.74
3.2	0.001	84.7	168.0	126.4	112.4	> 1000
		5.58	1.93	11.30	8.85	36.89

TABLE 5.2
Average number of iterations and computational times for $\phi(t) = t^2/2$, $|t| \leq \alpha$ *and* $\phi(t) = \alpha|t| - \alpha^2/2$, $|t| > \alpha$.

β	α	Multiplicative	Additive	BFGS	DFP	Steepest
10	0.025	75.7	282.6	133.0	97.6	894.4
		5.03	3.23	11.67	7.68	32.65
10	0.05	75.8	267.4	109.8	86.6	721.8
		5.04	3.27	9.89	6.85	26.52
10	0.1	71.2	232.4	89.5	72.1	529.2
		4.74	2.81	7.96	5.61	19.49
10	0.2	54.3	147.2	63.7	47.3	329.1
		3.59	1.20	5.19	3.66	12.16
10	0.4	22.7	39.2	38.5	31.4	77.2
		1.54	0.62	3.49	2.43	2.92
2.5	0.1	62.3	195.6	56.5	59.5	323.0
		4.15	2.41	5.09	4.83	11.86
5	0.1	62.1	220.4	71.6	61.5	384.7
		4.13	2.68	6.40	4.87	14.08
10	0.1	71.2	232.4	89.5	72.1	529.2
		4.74	2.81	7.96	5.61	19.49
20	0.1	78.3	198.2	109.4	84.2	688.3
		5.19	2.46	9.75	6.56	25.47
40	0.1	96.0	154.6	132.4	105.6	961.7
		6.34	1.93	11.73	8.17	35.13

same operations can vary by orders of magnitude depending on the particular coding used. This result may show that HQ regularization is a powerful tool for the minimization of cost functions of the form (1.2)–(1.3).

- We have computed the condition numbers of the matrices $\mathcal{H}(b^{(k)})$ and $\mathcal{H}$ arising from the multiplicative form and the additive form of HQ regularization, respectively. As expected, the condition number of $\mathcal{H}(b^{(k)})$ is always larger

TABLE 5.3
Average number of iterations and computational times for $\phi(t) = |t|/\alpha - \log(1 + |t|/\alpha)$.

β	α	Multiplicative	Additive	BFGS	DFP	Steepest
0.005	0.025	45.4	182.9	68.1	52.1	608.3
		2.96	2.13	6.16	4.17	21.71
0.005	0.05	32.9	170.3	44.9	54.3	206.2
		2.19	2.00	4.19	4.46	7.55
0.005	0.1	25.1	111.4	42.6	54.7	224.1
		1.69	1.39	4.36	4.50	8.31
0.005	0.2	19.0	65.8	53.1	48.1	209.0
		1.27	0.90	4.89	3.85	7.49
0.005	0.4	13.8	36.6	59.4	58.3	252.8
		0.94	0.56	5.50	4.70	7.12
0.00125	0.1	24.8	117.4	69.6	61.6	290.5
		1.65	1.45	6.48	4.99	10.49
0.0025	0.1	24.2	114.7	55.6	54.6	260.7
		1.61	1.43	5.19	4.44	9.42
0.005	0.1	25.1	111.4	42.6	54.7	224.1
		1.69	1.39	4.36	4.50	8.31
0.01	0.1	26.2	111.9	38.4	44.9	161.3
		1.76	1.37	3.58	3.62	5.81
0.02	0.1	27.2	102.9	34.7	32.7	140.8
		1.82	1.28	3.24	2.69	5.14

TABLE 5.4
Average number of iterations and computational times for $\phi(t) = \log(\cosh(\alpha t))$.

β	α	Multiplicative	Additive	BFGS	DFP	Steepest
0.5	0.25	3.0	3.0	32.2	324.1	132.6
		0.30	0.18	3.04	14.91	5.10
0.5	0.5	3.8	4.0	23.7	120.3	59.5
		0.34	0.20	2.17	5.65	2.36
0.5	1	5.5	6.1	213.5	211.7	30.8
		0.44	0.22	9.23	8.78	1.23
0.5	2	13.9	20.1	707.5	510.5	335.4
		0.98	0.40	30.70	22.74	12.49
0.5	4	30.5	63.7	> 1000	809.8	155.6
		2.07	0.90	42.08	36.91	6.02
0.125	1	6.9	8.2	220.2	120.6	66.5
		0.55	0.24	10.04	5.75	2.59
0.25	1	5.8	7.2	218.3	509.1	43.5
		0.47	0.24	9.72	21.45	1.72
0.5	1	5.5	6.1	213.5	211.7	30.8
		0.44	0.22	9.23	8.78	1.23
1	1	4.9	5.4	606.9	605.6	119.4
		0.41	0.22	25.75	25.32	4.51
2	1	4.5	4.9	902.7	803.3	804.9
		0.37	0.20	44.10	34.86	30.22

than that of $\mathcal{H}$. Some average results are reported in Table 5.6.

We have tested other examples where the blurred signal is corrupted with white Gaussian noise with SNR of 30dB or 50dB. We have also considered different point spread functions for A, such as

$$A_{i,j} = \begin{cases} \gamma e^{-0.1(i-j)^2}, & i,j = 1,2,\ldots,10, \\ 0 & \text{otherwise}, \end{cases}$$

TABLE 5.5
Average number of iterations and computational times for $\phi(t) = |t|^\alpha$.

β	α	Multiplicative	BFGS	DFP	Steepest
5	1.1	12.6	>1000	> 1000	> 1000
		1.36	38.39	37.89	37.93
5	1.15	18.9	>1000	> 1000	> 1000
		1.60	39.10	37.69	37.48
5	1.2	16.8	>1000	> 1000	> 1000
		1.26	38.76	38.48	38.12
5	1.25	14.4	>1000	> 1000	> 1000
		1.06	38.13	38.16	37.55
5	1.3	13.6	>1000	> 1000	> 1000
		0.98	37.91	37.56	37.12
1.25	1.2	18.2	>1000	> 1000	> 1000
		1.36	39.16	38.84	39.13
2.5	1.2	18.0	>1000	> 1000	> 1000
		1.28	39.01	38.45	38.56
5	1.2	16.8	>1000	> 1000	> 1000
		1.26	38.99	38.98	37.90
10	1.2	16.8	>1000	> 1000	> 1000
		1.17	38.59	38.39	38.67
20	1.2	16.3	>1000	> 1000	> 1000
		1.17	39.01	38.87	38.03

TABLE 5.6
Average condition numbers of $\mathcal{H}(b^{(k)})$ *in multiplicative form and* $\mathcal{H}$ *in additive form; cf.* (2.7) *and* (3.7). *Left:* $\phi(t) = \sqrt{\alpha + t^2}$. *Right:* $\phi(t) = t^2/2$ *if* $|t| \le \alpha$, $\phi(t) = \alpha|t| - \alpha^2/2$ *if* $|t| > \alpha$.

β	α	Multiplicative	Additive
0.8	0.00025	58.68	25.34
0.8	0.0005	47.80	15.92
0.8	0.001	42.01	10.78
0.8	0.002	73.85	43.40
0.8	0.004	117.26	78.82
0.2	0.001	43.35	19.89
0.4	0.001	30.74	15.91
0.8	0.001	42.01	10.78
1.6	0.001	85.80	32.89
3.2	0.001	120.79	43.40

β	α	Multiplicative	Additive
10	0.025	51.37	13.83
10	0.05	34.12	13.83
10	0.1	20.98	13.83
10	0.2	15.02	13.83
10	0.4	13.83	13.83
2.5	0.1	14.93	7.15
5	0.1	17.30	9.61
10	0.1	20.98	13.83
20	0.1	28.62	21.46
40	0.1	42.88	35.89

where γ is a normalization constant, and

$$A_{i,j} = \gamma_{i,j} e^{-0.1(i-j)^2}, \quad i, j = 1, 2, \ldots, p,$$

where $\gamma_{i,j} \in [0, 1]$ are random numbers. In the former case A is banded and in the latter A is not Toeplitz. All observations stated above have been corroborated.

We focused on the role of the constant c involved in the definition of ψ in the additive form of HQ regularization—see (3.1) and (3.2). The results presented in Figure 5.2 show that the convergence rate depends linearly on the value of c. This nicely corroborates our Remark 7 on the optimal choice for c.

5.2. Two-dimensional images. The original image and the data, degraded by blur and noise, are shown in Figure 5.3. We consider spatial-invariant blurring, in which case A is a Toeplitz-like matrix [23]. In the regularization term, G is the discretization matrix of the first-order differentiation operator. The matrix $\mathcal{H}$ in the additive form can be diagonalized by the discrete cosine transform matrix. Thus the

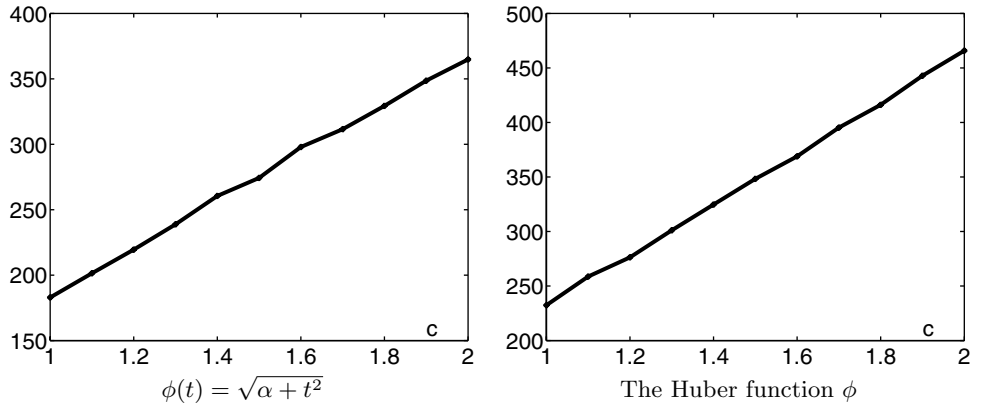


FIG. 5.2. *Average number of iterations as a function of the values of c in the additive form: (left)* $\phi(t) = \sqrt{\alpha + t^2}$ *(*$\alpha = 0.001$*); (right)* $\phi(t) = t^2/2$ *if* $|t| \leq \alpha$ *and* $\phi(t) = \alpha|t| - \alpha^2/2$ *if* $|t| > \alpha$ *(*$\alpha = 0.1$*).*

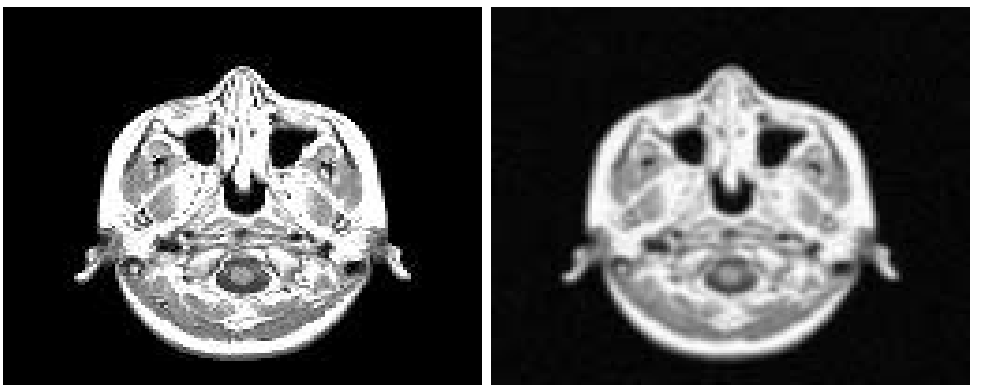


FIG. 5.3. *Original image (left) and blurred and noisy image (right).*

computational complexity to solve (4.6) at each iteration is $O(p^2 \log p)$ operations for a p-by-p image. In the multiplicative form, the matrix $\mathcal{H}(b)$ cannot be diagonalized efficiently. At each iteration, we solve (4.5) using conjugate gradient methods. The computational complexity is $O(p^2 \log p)$ per iteration.

The restored images using the Huber potential function are displayed in Figure 5.4 for the additive and the multiplicative forms with $\alpha = \beta = 1$. The stopping criterion of the HQ regularization iterations is $||x^{(k)} - x^{(k-1)}||_2 < 1 \times 10^{-3}\|y\|_2$. For the inner conjugate gradient iterations, we set the stopping criterion such that the $\mathcal{H}(b)$-norm of the solution error $||x^{(k)} - x_j^{(k)}||_{\mathcal{H}(b)}$ is less than 1×10^{-4}, where $x^{(k)}$ is the solution of the linear system (4.5) and $x_j^{(k)}$ is the jth iterate of the conjugate gradient iteration [6]. Visually, two restored images using the additive and the multiplicative form are almost the same. From Table 5.7, we see that the multiplicative form is more effective than the additive form in terms of the number of iterations, the objective function values, and the relative errors. However, the differences are not significant. The computational times required by using the additive form are significantly less than those using the multiplicative form.

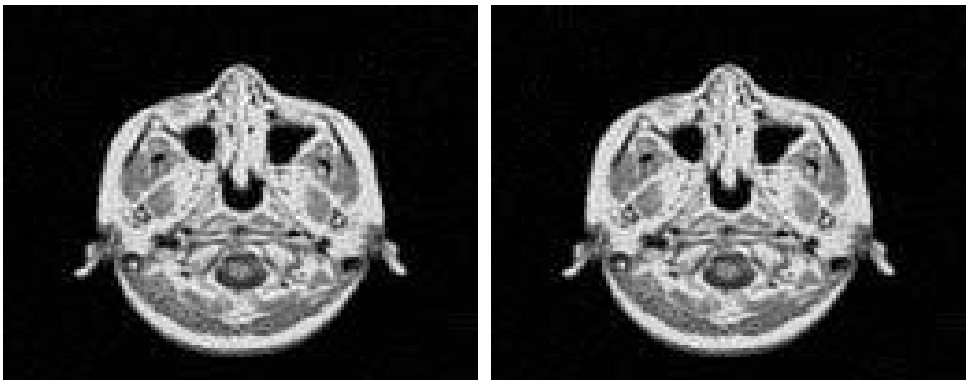


FIG. 5.4. *Restored image using the multiplicative form (left) and using the additive form (right).*

TABLE 5.7
The performance of the multiplicative form and the additive form for the Huber potential function.

$\beta = 1$ and $\alpha = 1$	Multiplicative	Additive
Number of iterations required	3	3
Number of inner conjugate gradient iterations required	79	–
Objective function value	457.1436	457.3545
Relative error of the restored image	0.1240	0.1241
CPU times required in seconds	16.23	0.54
$\beta = 1$ and $\alpha = 0.5$	Multiplicative	Additive
Number of iterations required	3	3
Number of inner conjugate gradient iterations required	78	–
Objective function value	456.4929	456.1902
Relative error of the restored image	0.1223	0.1237
CPU times required in seconds	16.02	0.54
$\beta = 1$ and $\alpha = 0.25$	Multiplicative	Additive
Number of iterations required	3	4
Number of inner conjugate gradient iterations required	79	–
Objective function value	430.4126	430.5825
Relative error of the restored image	0.1173	0.1229
CPU times required in seconds	16.20	0.71
$\beta = 0.5$ and $\alpha = 1$	Multiplicative	Additive
Number of iterations required	3	3
Number of inner conjugate gradient iterations required	80	–
Objective function value	325.2088	325.3879
Relative error of the restored image	0.1275	0.1339
CPU times required in seconds	16.01	0.54
$\beta = 2$ and $\alpha = 1$	Multiplicative	Additive
Number of iterations required	3	3
Number of inner conjugate gradient iterations required	78	–
Objective function value	682.3296	682.9721
Relative error of the restored image	0.1222	0.1224
CPU times required in seconds	16.06	0.54

6. Concluding remarks. We showed that the minimization of J using both the multiplicative and the additive form of HQ regularization amounts to finding the fixed point of a mapping involving a pertinent correction of the steepest descent direction. We presented a theoretical analysis of the convergence rate for each one of these minimization methods. More precisely, for both forms we derived an upper bound on the root-convergence factor. The bound for the multiplicative form was found to be smaller than the bound for the additive form. Our experimental results

show that minimization using the multiplicative form needs fewer iterations than the additive form, but that it requires a higher computation time. When applicable, the additive form is more attractive to use than the multiplicative form. Extensive experiments demonstrate that in our MATLAB implementation, minimization using HQ regularization is substantially faster (in terms of computational times) than standard routines, such as those in the MATLAB Optimization Toolbox. These results must be qualified by the observation that MATLAB run times for the same operations can vary by orders of magnitude depending on the particular coding used.

7. Appendix. From the theory of convex conjugated functions [27, 18], given $f : \mathbb{R} \to \mathbb{R}$, its conjugated function g is defined by $g(s) = \sup_{t\in\mathbb{R}}\{st - f(t)\}$ and is convex on $\mathbb{R}$. If f is continuous, convex and $f \not\equiv \infty$ on $\mathbb{R}$, we have reciprocally $\sup_{s\in\mathbb{R}}\{st - g(s)\} = f(t)$.

7.1. Functions ψ and σ for the multiplicative form. We suppose that ϕ satisfies all conditions given in (2.1). Define $\theta : \mathbb{R} \to \mathbb{R}$ as

$$\theta(t) = \begin{cases} \phi(\sqrt{t}) & \text{if} \quad t \geq 0, \\ -\infty & \text{if} \quad t < 0. \end{cases} \tag{7.1}$$

By (2.1)(b), the function $-\theta$ is convex on $\mathbb{R}$. Using convex conjugacy, if

$$\psi(s) = \sup_{t\in\mathbb{R}_+} \{\ h(t,s)\ \}, \quad \text{where} \quad h(t,s) = \left(-\frac{1}{2}s\right) t - (-\theta(t)), \tag{7.2}$$

we have reciprocally

$$\theta(t) = \inf_{s\in\mathbb{R}_+^*} \left\{\frac{1}{2}st + \psi(s)\right\}, \tag{7.3}$$

where the infimum is taken over $\mathbb{R}_+^*$ because $\psi(s) = +\infty$ if $s \leq 0$. Since $t \in \mathbb{R}_+$ in (7.2) and (7.3), we can write t^2 in place of t. Using that $\theta(t^2) = \phi(t)$, these expressions yield (2.2) and (2.3), respectively.

Since for any $s > 0$ the function $t \to h(t,s)$ is concave on $\mathbb{R}_+$ by (2.1)(b) and satisfies $\lim_{t\to\infty} h(t,s) = -\infty$ by (2.1)(f), it follows that $h(.,s)$ reaches its supremum for a finite $t \in [0,\infty)$. We determine σ by exhibiting all the pairs $(\hat{t},\hat{s})$ for which the supremum in (7.2) is reached, i.e., $\psi(\hat{s}) = h(\hat{t},\hat{s})$.

- For any $\hat{s} \geq 2\theta'(0^+)$, the function $h(.,\hat{s})$ reaches its maximum at $\hat{t} = 0$ since $\lim_{t\searrow 0} D_1h(t,\hat{s}) \leq 0$ shows that $h(.,\hat{s})$ is decreasing on the right side of zero.
- Consider that $\hat{s} \in (0, 2\theta'(0^+))$. Since $\lim_{t\searrow 0} D_1h(t,\hat{s}) > 0$, the function $h(.,\hat{s})$ increases on the right of zero. The supremum in (7.2) is then reached for a $\hat{t} \in (0,\infty)$ such that $D_1h(\hat{t},\hat{s}) = 0$, hence $\hat{s} = 2\theta'(\hat{t}) = \phi'(\sqrt{\hat{t}})/\sqrt{\hat{t}}$.

Using $\theta(t^2) = \phi(t)$ again shows that σ is as given in (2.5).

7.2. Functions ψ and σ for the additive form. We assume that ϕ satisfies (3.1). Using (3.1)(b), convex conjugacy shows that if

$$\psi(s) + \frac{1}{2c}s^2 = \sup_{t\in\mathbb{R}} \{\ h(t,s)\ \}, \quad \text{where} \quad h(t,s) = ts - \left(\frac{c}{2}t^2 - \phi(t)\right), \tag{7.4}$$

we have reciprocally

$$\frac{c}{2}t^2 - \phi(t) = \sup_{s\in\mathbb{R}} \left(ts - \left(\psi(s) + \frac{1}{2c}s^2\right)\ \right).$$

The equation in (7.4) is equivalent to (3.2) while the latter is equivalent to (3.3). Here again, the function σ is obtained by determining all the pairs $(\hat{t}, \hat{s})$ which achieve the supremum in (7.4). Using (3.1)(b) and (3.1)(e), for any $s \in \mathbb{R}$, the function $h(., s)$ is concave and satisfies $\lim_{|t|\to\infty} h(t, s) = -\infty$. It follows that for any $\hat{s}$, the supremum in (7.4) is reached for a finite $\hat{t}$. Since h is differentiable, any such $\hat{t}$ satisfies $D_1 h(\hat{t}, \hat{s}) = 0$. Hence $\hat{s} = \sigma(\hat{t})$ if σ reads as in (3.5).

REFERENCES

[1] R. Acar and C. Vogel, *Analysis of bounded variation penalty methods for ill-posed problems*, IEEE Trans. Image Process., 10 (1994), pp. 1217–1229.

[2] G. Aubert and P. Kornprobst, *Mathematical Problems in Image Processing*, Springer-Verlag, Berlin, 2002.

[3] G. Aubert and L. Vese, *A variational method in image recovery*, SIAM J. Numer. Anal., 34 (1997), pp. 1948–1979.

[4] M. Black and A. Rangarajan, *On the unification of line processes, outlier rejection, and robust statistics with applications to early vision*, Internat. J. Computer Vision, 19 (1996), pp. 57–91.

[5] C. Bouman and K. Sauer, *A generalized Gaussian image model for edge-preserving* map *estimation*, IEEE Trans. Image Process., 2 (1993), pp. 296–310.

[6] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[7] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, *Two deterministic half-quadratic regularization algorithms for computed imaging*, in Proceedings of IEEE ICIP, Vol. 2, 1994, pp. 168–172.

[8] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, *Deterministic edge-preserving regularization in computed imaging*, IEEE Trans. Image Process., 6 (1997), pp. 298–311.

[9] P. G. Ciarlet, *Introduction à l'analyse numérique matricielle et à l'optimisation*, Collection mathématiques appliquées pour la maîtrise, Masson, Paris, 1990.

[10] L. D. Cohen, *Auxiliary variables and two-step iterative algorithms in computer vision problems*, J. Math. Imaging Vision, 6 (1996), pp. 376–386.

[11] A. H. Delaney and Y. Bresler, *Globally convergent edge-preserving regularized reconstruction: An application to limited-angle tomography*, IEEE Trans. Image Process., 7 (1998), pp. 204–221.

[12] G. Demoment, *Image reconstruction and restoration: Overview of common estimation structures and problems*, IEEE Trans. Acoustics Speech Signal Process., 37 (1989), pp. 2024–2036.

[13] D. Geman and G. Reynolds, *Constrained restoration and the recovery of discontinuities*, IEEE Trans. Pattern Anal. Machine Intelligence, 14 (1992), pp. 367–383.

[14] D. Geman and C. Yang, *Nonlinear image recovery with half-quadratic regularization*, IEEE Trans. Image Process., 4 (1995), pp. 932–946.

[15] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Machine Intelligence, 6 (1984), pp. 721–741.

[16] R. Glowinski, J. Lions, and R. Trémolières, *Analyse numérique des inéquations variationnelles*, Vol. 1, 1st ed., Dunod, Paris, 1976.

[17] P. J. Green, *Bayesian reconstructions from emission tomography data using a modified* em *algorithm*, IEEE Trans. Medical Imaging, 9 (1990), pp. 84–93.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms*, Vol. I, Springer-Verlag, Berlin, 1996.

[19] J. Idier, *Convex half-quadratic criteria and interacting auxiliary variables for image restoration*, IEEE Trans. Image Process., 10 (2001), pp. 1001–1009.

[20] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386.

[21] P. Kornprobst, R. Deriche, and G. Aubert, *Image sequence analysis via partial differential equations*, J. Math. Imaging Vision, 11 (1999), pp. 5–26.

[22] S. Li, *Markov Random Field Modeling in Computer Vision*, 1st ed., Springer-Verlag, New York, 1995.

[23] M. Ng, R. Chan, T. Chan, and A. Yip, *Cosine transform preconditioners for high resolution image reconstruction*, Linear Algebra Appl., 316 (2000), pp. 89–104.

[24] M. K. Ng, R. H. Chan, and W.-C. Tang, *A fast algorithm for deblurring models with Neumann boundary conditions*, SIAM J. Sci. Comput., 21 (1999), pp. 851–866.
[25] M. Nikolova, *Local strong homogeneity of a regularized estimator*, SIAM J. Appl. Math., 61 (2000), pp. 633–658.
[26] J. Ortega and W. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[27] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.
[28] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[29] A. Tarantola, *Inverse Problem Theory: Methods for Data Fitting and Model Parameter Estimation*, Elsevier–North Holland, Amsterdam, 1987.
[30] S. Teboul, L. Blanc-Féraud, G. Aubert, and M. Barlaud, *Variational approach for edge-preserving regularization using coupled PDE's*, IEEE Trans. Image Process., 7 (1998), pp. 387–397.
[31] A. Tikhonov and V. Arsenin, *Solutions of Ill-Posed Problems*, Winston, Washington, DC, 1977.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 967–994
© 2005 Society for Industrial and Applied Mathematics

A NONLINEAR GALERKIN SCHEME INVOLVING VECTOR AND TENSOR SPHERICAL HARMONICS FOR SOLVING THE INCOMPRESSIBLE NAVIER–STOKES EQUATION ON THE SPHERE*

M. J. FENGLER† AND W. FREEDEN†

Abstract. This work is concerned with a nonlinear Galerkin method for solving the incompressible Navier–Stokes equation on the sphere. It extends the work of [A. Debussche, T. Dubois, and R. Temam, *Theoret. Comput. Fluid Dyn.,* 7 (1995), pp. 279–315; M. Marion and R. Temam, *SIAM J. Numer. Anal.,* 26 (1989), pp. 1139–1157; J. Shen and R. Temam, *Proceedings of the International Conference on Nonlinear Evolution Partial Differential Equations*, AMS, Providence, RI, 1997, pp. 363–376] from one-dimensional or toroidal domains to the spherical geometry. In the first part, the method based on type 3 vector spherical harmonics is introduced and convergence is indicated. Further it is shown that the occurring coupling terms involving three vector spherical harmonics can be expressed algebraically in terms of Wigner-3j coefficients. To improve the numerical efficiency and economy we introduce an FFT-based pseudospectral algorithm for computing the Fourier coefficients of the nonlinear advection term. The resulting method scales with $O(N^3)$ if N denotes the maximal spherical harmonic degree. The latter is demonstrated in an extensive numerical example.

Key words. incompressible spherical Navier–Stokes equation, nonlinear Galerkin scheme, vector spherical harmonics, pseudospectral algorithm, transform method

AMS subject classifications. 76D05, 37L65, 33C55, 42A99, 85A20

DOI. 10.1137/040612567

1. Motivation.

Spectral methods are well known for solving different types of partial differential equations such as Laplace, Helmholtz, and heat or wave equations, but also for nonlinear equations such as the Burger, shallow water, or full Navier–Stokes equation or even more complex coupled partial differential equations, e.g., those known from dynamo theory, magnetohydrodynamics, or meteorology (cf. [1, 2, 3, 9, 12, 15, 20, 21, 22, 25]). Spectral methods are usually implemented in Galerkin schemes, which are applied to, for example, complex flows but in simple geometries, e.g., a two-dimensional rectangular, periodic cell. These geometries provide an easy access to fast basis transforms by exploiting the periodicity of the basis functions, leading to FFT-based algorithms. Moreover, recent publications due to Temam et al. [7, 17, 23, 24] extend the idea of linear Galerkin approximations to nonlinear Galerkin methods. The key idea is that nonlinear expressions such as the nonlinear advection term can be treated more efficiently and also more economically than in a linear Galerkin ansatz. Nonlinear Galerkin methods are able to capture the flow behavior with fewer test functions than are needed within a linear Galerkin approach.

This article deals with a formulation of a nonlinear Galerkin method for solving the Navier–Stokes equations on the sphere involving vector and tensor spherical harmonics. The method itself is mainly motivated by the work of Marion and Temam [17], who have given a proof of convergence within a general Hilbert space concept. In

*Received by the editors July 30, 2004; accepted for publication (in revised form) April 20, 2005; published electronically December 8, 2005.

http://www.siam.org/journals/sisc/27-3/61256.html

†Geomathematics Group, Department of Mathematics, University of Kaiserslautern, D-67653 Kaiserslautern, Germany (fengler@rhrk.uni-kl.de, freeden@mathematik.uni-kl.de).

detail, this work extends the results of [7, 23] from one-dimensional and toroidal domains to the spherical geometry, which is, for example, of great interest in meteorology and oceanography. Moreover, special emphasis is given on a fast computation of the nonlinear advection. Importantly, the techniques introduced later on differ from that used in the torus geometry due to the loss of periodicity in latitudinal direction. The idea is based on the so-called [illegible] or [illegible] originally introduced by Orszag [20]; see also the related works of [2, 12, 25]. However, the main difference between [2, 12, 20, 25] and this work is that the new method proposed below follows intrinsically a vectorial and tensorial formulation of the advection term. Thus, we are concerned with the fast transformation of (physically meaningful) vectorial and tensorial spherical harmonics which are not reduced (componentwise) to scalar ones. In doing so, we can easily separate the involved vector fields into poloidal and toroidal parts. This is important since we are interested here in flows, which are free of surface divergences. Finally, it should be noted that the term [illegible] is often used in meteorology to denote an exact variant of the so-called [illegible], while the latter is then meant to be approximative (see, for instance, [20, 25]). However, throughout this work we follow the standard textbook of Boyd [3] and use the term [illegible] synonymously; i.e., we use this method as an exact replacement of an algebraic evaluation of the advection term. In that sense, the term [illegible] is understood to denote the general idea behind, which is applied to many other special functions and geometries, e.g., Chebyshev polynomials on finite intervals (see [3] and the references therein).

The layout of the paper is as follows: First, we introduce reference function spaces. Within this framework various estimates are derived to show that the theorem of convergence as proposed by [17] can be applied. Second, we discuss the occurring coupling terms in the nonlinear Galerkin scheme, in particular, the nonlinear advection term. To improve the algorithmic efficiency, we extend the techniques known for pseudospectral algorithms to toroidal vector and tensor fields. Finally, we numerically illustrate our method for an extensive example, which involves the rotational Coriolis terms, the nonlinear advection, the viscosity, and the time-dependent flow driving forces.

2. Preliminaries. In what follows we adopt the nomenclature proposed by Freeden, Gervens, and Schreiner [11]:

$\mathbb{R}^3$	Three-dimensional Euclidean space
x, y, z	Elements of $\mathbb{R}^3$
$x \cdot y$	Scalar product of vectors $x, y \in \mathbb{R}^3$
$x \wedge y$	Vector product of vectors $x, y \in \mathbb{R}^3$
$x \otimes y$	Tensor product of vectors $x, y \in \mathbb{R}^3$
$\varepsilon^i, i = 1, 2, 3$	Canonical orthonormal basis in $\mathbb{R}^3$
∇	Gradient in $\mathbb{R}^3$
Δ	Laplacian in $\mathbb{R}^3$
r, t, ϕ	Polar coordinates in $\mathbb{R}^3$
Ω	Unit sphere in $\mathbb{R}^3$
ξ, η	Elements of the unit sphere $\Omega \subset \mathbb{R}^3$
$\varepsilon^r, \varepsilon^\phi, \varepsilon^t$	Moving orthonormal triad on Ω
∇^*	Surface gradient on Ω
L^*	Surface curl gradient on Ω
$\nabla^* \cdot$	Surface divergence on Ω
$L^* \cdot$	Surface curl on Ω
Δ^*	Scalar Beltrami operator on Ω
$\mathbf{\Delta}^*$	Vectorial Beltrami operator on Ω

$C^{(k)}_{\mathbb{K}}(\Omega), L^p_{\mathbb{K}}(\Omega)$ (respectively, $c^{(k)}_{\mathbb{K}}(\Omega), l^p_{\mathbb{K}}(\Omega)$) denote the following classes of scalar-valued (respectively, vector-valued) functions on Ω (with $\mathbb{K} = \mathbb{R}$ or $\mathbb{C}$):

$C^{(k)}_{\mathbb{K}}(\Omega)$	Class of scalar functions $F : \Omega \to \mathbb{K}$ possessing k continuous derivatives on Ω
$L^p_{\mathbb{K}}(\Omega)$	Class of scalar functions $F : \Omega \to \mathbb{K}$ which are measurable and for which $\|F\|_{L^p(\Omega)} = (\int_\Omega \|F(\xi)\|^p dS(\xi))^{1/p} < \infty$ (dS is the surface element)
$c^{(k)}_{\mathbb{K}}(\Omega)$	Class of vector functions $f : \Omega \to \mathbb{K}^3$ possessing k continuous derivatives on Ω
$l^p_{\mathbb{K}}(\Omega)$	Class of vector functions $f : \Omega \to \mathbb{K}^3$ which are measurable and for which $\|f\|_{l^p(\Omega)} = (\int_\Omega \|f(\xi)\|^p dS(\xi))^{1/p} < \infty$

We denote by $Y_{n,k}$ the complex scalar spherical harmonics as introduced by Edmonds [8]. Moreover, we denote by $y^{(i)}_{n,k}(\xi)$ vector spherical harmonics for $n = 0_i, \ldots; k = -n, \ldots, n$ and $i = 1, 2, 3$ with $0_1 = 0$ and $0_2 = 0_3 = 1$. To be more specific, we let

$$\begin{aligned} y^{(1)}_{n,k}(\xi) &= \xi Y_{n,k}(\xi), \\ y^{(2)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \nabla^*_\xi Y_{n,k}(\xi), \\ y^{(3)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} L^*_\xi Y_{n,k}(\xi) \end{aligned}$$

for $n = 0_i, \ldots; k = -n, \ldots, n$. Finally, the scalar spherical harmonics can be used to introduce tensor spherical harmonics by

(2.1)

$$\begin{aligned} \mathbf{y}^{(1,1)}_{n,k}(\xi) &= \xi \otimes \xi Y_{n,k}(\xi), \\ \mathbf{y}^{(1,2)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \xi \otimes \nabla^*_\xi Y_{n,k}(\xi), & \mathbf{y}^{(1,3)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \xi \otimes L^*_\xi Y_{n,k}(\xi), \\ \mathbf{y}^{(2,1)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \nabla^*_\xi Y_{n,k}(\xi) \otimes \xi, & \mathbf{y}^{(2,2)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \nabla^*_\xi \otimes \nabla^*_\xi Y_{n,k}(\xi), \\ \mathbf{y}^{(2,3)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} \nabla^*_\xi \otimes L^*_\xi Y_{n,k}(\xi), & \mathbf{y}^{(3,1)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} L^*_\xi Y_{n,k}(\xi) \otimes \xi, \\ \mathbf{y}^{(3,2)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} L^*_\xi \otimes \nabla^*_\xi Y_{n,k}(\xi), & \mathbf{y}^{(3,3)}_{n,k}(\xi) &= \frac{1}{\sqrt{n(n+1)}} L^*_\xi \otimes L^*_\xi Y_{n,k}(\xi) \end{aligned}$$

for $n = 0_{i,j}, \ldots; k = -n, \ldots, n$ with

$$0_{i,j} = \begin{cases} 0 & \text{if } (i,j) = (1,1), \\ 1 & \text{else.} \end{cases}$$

Furthermore, in analogy to [11], we introduce the function spaces

$$\begin{aligned} \mathrm{Harm}_{p,\ldots,q}(\Omega) &= \mathrm{span}_{n=p,\ldots,q;k=-n,\ldots,n}\{Y_{n,k}\}, \\ \mathrm{harm}^{(i)}_{p,\ldots,q}(\Omega) &= \mathrm{span}_{n=p,\ldots,q;k=-n,\ldots,n}\{y^{(i)}_{n,k}\}, \quad i = 1, 2, 3, \\ l^2_{(i)}(\Omega) &= \overline{\mathrm{harm}^{(i)}_{0_i,\ldots,\infty}(\Omega)}^{\|\cdot\|_{l^2(\Omega)}}. \end{aligned}$$

From now on, we simply let $H = l^2_{(3)}(\Omega)$ if no confusion is likely to arise.

Finally, we introduce certain Sobolev spaces:

$$\mathcal{H}(\{A_n\};\Omega) = \left\{ F \in L^2(\Omega) | \sum_{n=0}^{\infty} \sum_{k=-n}^{n} \langle F, Y_{n,k}\rangle^2_{L^2(\Omega)} A_n^2 < \infty \right\},$$

$$h^{(i)}(\{A_n\};\Omega) = \left\{ f \in l^2(\Omega) | \sum_{n=0_i}^{\infty} \sum_{k=-n}^{n} \langle f, y^{(i)}_{n,k}\rangle^2_{l^2(\Omega)} A_n^2 < \infty \right\},$$

$$h(\{A_n\};\Omega) = h^{(1)}(\{A_n\};\Omega) \oplus h^{(2)}(\{A_n\};\Omega) \oplus h^{(3)}(\{A_n\};\Omega)$$

for sequences $\{A_n\}$ with $A_n \in \mathbb{R}$ for all n.

Following the terminology of [17], we let $A = -\boldsymbol{\Delta}^*$; i.e., A is the negative vectorial Beltrami operator. This basic dissipative operator A is a linear self-adjoint unbounded operator in H with domain of definition $D(A)$ dense in H (see [11]). Moreover, we know from [11] that A is positive and that A^{-1} is compact. Thus, one can define powers A^s of A for $s \in \mathbb{R}$; A^s maps $D(A^s)$ into H and $D(A^s)$ is a Sobolev space when endowed with the norm $\|A^s \cdot \|_{l^2_{(\Omega)}}$. We set $\mathcal{V} = D(A^{1/2})$ and denote by

$$\|\cdot\|_{\mathcal{V}} = \|(-\boldsymbol{\Delta}^*)^{1/2} \cdot \|_{l^2(\Omega)}$$

the norm on $\mathcal{V}$. Hence, we find by [11] the relation

$$\mathcal{V} = h^{(3)}(\{\sqrt{n(n+1)}\};\Omega).$$

Thus we arrive at

$$A : H \supset D(A) \to H, \qquad u \mapsto Au = -\boldsymbol{\Delta}^* u. \tag{2.2}$$

Moreover, we define the bilinear advection operator B by

$$B : \mathcal{V} \times \mathcal{V} \to \mathcal{V}', \qquad (u,v) \mapsto ((u \cdot \nabla^*)v)\,, \tag{2.3}$$

where $\mathcal{V}'$ is understood to be the dual space of $\mathcal{V}$. The Coriolis operator C is defined by

$$C : H \to H, \qquad u \mapsto 2\omega \wedge u, \tag{2.4}$$

where ω denotes the rotational axis pointing in direction of ε^3. By $b(u,v,w) = \langle B(u,v), w\rangle_{\mathcal{V}',\mathcal{V}}$ we denote the trilinear form defined by

$$b(u,v,w) = \int_\Omega ((u \cdot \nabla^*)v) \cdot w \; dS \tag{2.5}$$

for all $u, v, w \in \mathcal{V}$. Finally, we introduce the space

$$L^2_\Omega(0,T;\mathcal{V}) = \left\{ f(t) \in \mathcal{V} : [0,T] \to \mathbb{R}^3 \,\middle|\, \left(\int_0^T |f(t)|^2 \, dt \right)^{1/2} < \infty \right\}.$$

The Navier–Stokes equations. Throughout this work we denote by u the fluid velocity, by T the temperature, by p the pressure, by g the apparent gravity, and by ρ the fluid's density. Additional flow driving terms are given by f, whereas friction

and dissipation are induced by a viscous term $\nu\boldsymbol{\Delta}^*$. In the sense of meteorology, we introduce Q as the total heating rate per unit mass, c_v as the specific heat at constant volume, and c_R as the gas constant per unit mass. From [18, 19] we recapitulate the set of forecasting equations for a pure tangential, atmospherical stream as follows:

$$\frac{\partial u}{\partial t} = -(u \cdot \nabla^*)u - 2\omega \wedge u - 1/\rho\nabla^* p - \nabla^* g + \nu\boldsymbol{\Delta}^* u + f, \tag{2.6}$$

$$c_v \frac{\partial T}{\partial t} = -c_v(u \cdot \nabla^*)T - \frac{p}{\rho}\nabla^* \cdot u + Q, \tag{2.7}$$

$$\frac{\partial \rho}{\partial t} = -(u \cdot \nabla^*)\rho - \rho\nabla^* \cdot u, \tag{2.8}$$

$$p = \rho c_R T, \tag{2.9}$$

$$u(0) = u_0, \qquad p(0) = p_0, \qquad T(0) = T_0, \qquad \rho(0) = \rho_0. \tag{2.10}$$

We restrict ourselves to pure tangential [illegible] flows on a sphere. This leads us to $\nabla^* \cdot u = 0$ and $\rho = \text{const.}$, which simplifies (2.6)–(2.10). To be more specific, in our Galerkin approach, the pressure and gravity terms drop out in a weak formulation. Namely, let ϕ be a smooth toroidal vector-valued function; then

$$\langle \nabla^* p(t), \phi \rangle_{l^2(\Omega)} = 0$$

(by standard theorems; see, for example, [11]). The latter decouples the system (2.6)–(2.10) such that we have to consider "only" (in the weak sense) the equations

$$\begin{aligned} \frac{\partial u}{\partial t} &= -(u \cdot \nabla^*)u - 2\omega \wedge u + \nu\boldsymbol{\Delta}^* u + f, \\ \nabla^* \cdot u &= 0, \\ u(0) &= u_0. \end{aligned} \tag{2.11}$$

In the formulation used above, we can write (2.11) as follows:

$$\frac{\partial u}{\partial t} + \nu Au + B(u, u) + Cu - f = 0, \tag{2.12}$$

where f is understood to be projected onto $l^2_{(3)}(\Omega)$.

Existence and uniqueness. For the Navier–Stokes equations on the 2-sphere Il'in [13] and Il'in and Filatov [14] have verified existence and uniqueness of a generalized (weak) solution, provided that $f \in L^2_\Omega(0, T; \mathcal{V}')$ and $u_0 \in H$. Moreover, they have proven that the solution is continuous and a member of class $L^2_\Omega(0, T; \mathcal{V})$.

3. Operator estimates. This section is concerned with some operator estimates involving (2.12). Based on these results we are able to assure the convergence of the nonlinear Galerkin method. As a matter of fact, we present the estimates for the spherical case the same way as [17]. It should be noted that $l^2(\Omega)$ is written instead of $l^2_{\mathbb{R}}(\Omega)$ if no confusion is likely to arise.

First some results are borrowed from [14].

LEMMA 3.1 (antisymmetry of the trilinear form). [illegible] $u, v, w \in \mathcal{V}$ [illegible] b [illegible] (2.5) [illegible]

$$b(u, v, w) = -b(u, w, v).$$

LEMMA 3.2. [illegible] $u \in \mathcal{V}$

$$\|u\|_{l^4(\Omega)} \le c_1 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2}. \tag{3.1}$$

From [10] we have the following important embedding theorem.

LEMMA 3.3 (embedding $\mathcal{V}$). [illegible] $2 \le p \le 4$ [illegible]

(i) $\mathcal{H}(\{n(n+1)\};\Omega) \subset \mathcal{H}(\{\sqrt{n(n+1)}\};\Omega) \subset L^p(\Omega) \subset L^2(\Omega)$
(ii) $h(\{n(n+1)\};\Omega) \subset h(\{\sqrt{n(n+1)}\};\Omega) \subset l^p(\Omega) \subset l^2(\Omega)$
(iii) $h^{(3)}(\{n(n+1)\};\Omega) \subset h^{(3)}(\{\sqrt{n(n+1)}\};\Omega) \subset l^p_{(3)}(\Omega) \subset l^2_{(3)}(\Omega)$
(iv) $\mathcal{V} \subset l^p_{(3)}(\Omega) \subset l^2_{(3)}(\Omega)$

Consequently, we can deduce a corollary relating the norms of the involved Sobolev spaces and $l^2(\Omega)$.

COROLLARY 3.4. [illegible] u [illegible] $h^{(3)}(\{n(n+1)\};\Omega)$ [illegible]

$$\|u\|_{l^2(\Omega)} \le \|u\|_{\mathcal{V}} \le \|u\|_{h^{(3)}(\{n(n+1)\};\Omega)}.$$

Based on these results we are able to deduce the following estimates.

ESTIMATE 3.5 (estimating the trilinear form). [illegible] b [illegible] $u, v, w \in \mathcal{V}$ [illegible]

(i)

$$|b(u,v,w)| \le c_1 \|u\|_{\mathcal{V}} \|v\|_{\mathcal{V}} \|w\|_{\mathcal{V}},$$

(ii)

$$|b(u,v,w)| \le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} \|w\|_{l^2(\Omega)}^{1/2} \|w\|_{\mathcal{V}}^{1/2}. \tag{3.2}$$

[illegible]. The first statement has already been proven in [14]. The second statement can be derived by expanding the trilinear form in its Cartesian components: By keeping in mind that u, v, w are tangential vector fields, we find

$$b(u,v,w) = \int_\Omega [(u \cdot \nabla^*)v] \cdot w \, dS = \int_\Omega [(u \cdot \nabla)v] \cdot w \, dS. \tag{3.3}$$

Since ∇ is the gradient in $\mathbb{R}^3$, we are able to expand (3.3) componentwise in Cartesian coordinates, i.e., $v_i = \varepsilon^i \cdot v$ and $\nabla_i = \varepsilon^i \cdot \nabla$, such that

$$b(u,v,w) = \int_\Omega \sum_{i=1}^3 \sum_{j=1}^3 u_i (\nabla_i v_j) w_j \, dS.$$

By applying Cauchy–Schwarz twice we estimate the absolute value of b by

$$\begin{aligned} |b(u,v,w)| &\le \sum_{i=1}^3 \sum_{j=1}^3 \int_\Omega |u_i (\nabla_i v_j) w_j| \, dS \le \sum_{i=1}^3 \sum_{j=1}^3 \|u_i\|_{L^4(\Omega)} \|\nabla_i v_j\|_{L^2(\Omega)} \|w_j\|_{L^4(\Omega)} \\ &\le \left(\sum_{i=1}^3 \|u_i\|_{L^4(\Omega)} \right)^{1/2} \left[\sum_{j=1}^3 \left(\sum_{i=1}^3 \|\nabla_i v_j\|_{L^2(\Omega)} \right)^{1/2} \|w_j\|_{L^4(\Omega)} \right] \\ &\le \left(\sum_{i=1}^3 \|u_i\|_{L^4(\Omega)} \right)^{1/2} \left(\sum_{i=1}^3 \sum_{j=1}^3 \|\nabla_i v_j\|_{L^2(\Omega)} \right)^{1/2} \left(\sum_{j=1}^3 \|w_j\|_{L^4(\Omega)} \right)^{1/2}. \end{aligned} \tag{3.4}$$

Now the first and third sums can be estimated by the aid of (3.1)

$$\|u_i\|^2_{L^4(\Omega)} = \|u_i\varepsilon_i\|^2_{l^4(\Omega)} \le \|u\|^2_{l^4(\Omega)} \le c_1^2\|u\|_{l^2(\Omega)}\|u\|_{\mathcal{V}}.$$

This leads us to

$$\left(\sum_{i=1}^{3}\|u_i\|^2_{L^4(\Omega)}\right)^{1/2} \le \sqrt{3}c_1\|u\|^{1/2}_{l^2(\Omega)}\|u\|^{1/2}_{\mathcal{V}}. \tag{3.5}$$

Observing the estimate

$$\begin{aligned}\sum_{i=1}^{3}\sum_{j=1}^{3}\|\nabla_i v_j\|^2_{L^2(\Omega)} &= \sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega \nabla_i v_j\nabla_i v_j\,dS = -\sum_{i=1}^{3}\sum_{j=1}^{3}\int_\Omega v_j(\nabla_i)^2 v_j\,dS\\ &= -\sum_{i=1}^{3}\int_\Omega v\cdot(\nabla_i)^2 v\,dS = -\int_\Omega v\cdot\Delta v\,dS\\ &= -\int_\Omega v\cdot\Delta^* v\,dS = \|v\|^2_{\mathcal{V}}\end{aligned}$$

we finally get the desired result. □

In the following we are concerned with the well-known Poincaré inequality on the sphere.

ESTIMATE 3.6 (Poincaré inequality). [illegible] $u \in \mathcal{V}$ [illegible]

$$\|u\|_{l^2(\Omega)} \le \frac{1}{\sqrt{2}}\|u\|_{\mathcal{V}}.$$

[illegible] Let us expand u in type 3 vector spherical harmonics such that $u = \sum_{n=1}^{\infty} u^{(3)}_{n,k}y^{(3)}_{n,k}$. Then $\|u\|^2_{\mathcal{V}}$ reads as

$$\|u\|^2_{\mathcal{V}} = \sum_{n=1}^{\infty} n(n+1)\left(u^{(3)}_{n,k}\right)^2.$$

Since $n(n+1) \ge 2$ for all $n \ge 1$ we get the desired result. □

ESTIMATE 3.7 (estimating the Coriolis term). [illegible] $u \in \mathcal{V}$ [illegible] $\omega \in \mathbb{R}^3$

$$\|Cu\|_{l^2(\Omega)} \le \sqrt{2}|\omega|\|u\|_{\mathcal{V}}.$$

[illegible] First we see that

$$\|Cu\|_{l^2(\Omega)} \le 2\|\omega\wedge u\|_{l^2(\Omega)} = 2\left[\int_\Omega(\omega\wedge u)^2\,dS\right]^{1/2}.$$

Applying the Lagrange identity we obtain

$$\begin{aligned}2\left[\int_\Omega(\omega\wedge u)^2\,dS\right]^{1/2} &= 2\left[\int_\Omega(\omega\cdot\omega)(u\cdot u)-(u\cdot\omega)(\omega\cdot u)\,dS\right]^{1/2}\\ &\le 2\left[\int_\Omega|\omega|^2(u\cdot u)\,dS\right]^{1/2}\\ &= 2|\omega|\|u\|_{l^2(\Omega)}.\end{aligned}$$

By aid of the Poincaré inequality from Estimate 3.6 we have

$$2|\omega|\|u\|_{l^2(\Omega)} \leq \sqrt{2}|\omega|\|u\|_{\mathcal{V}},$$

which proves our estimate. □

ESTIMATE 3.8 (positive definiteness of $\nu A + C$). A [illegible] (2.2) [illegible] C [illegible] (2.4) [illegible] $\nu \in \mathbb{R}^+$ [illegible] $u \in \mathcal{V}$,

$$\int_\Omega [(\nu A + C)u] \cdot u \; dS \geq \nu \|u\|_{\mathcal{V}}^2.$$

[illegible]. Let us drop for a moment the operator νA. We expand u in type 3 vector spherical harmonics such that $u = \sum_{n=1}^{\infty} \sum_{k=-n}^{n} u_{n,k}^{(3)} y_{n,k}^{(3)}$ in the sense of $l^2(\Omega)$. Then, we get for the Coriolis operator the following identity:

$$\int_\Omega Cu \cdot u \; dS = 2 \int_\Omega (\omega \wedge u) \cdot u \; dS = 0.$$

By using the linearity of the integral operator, we are able to deduce that

$$\int_\Omega [(\nu A + C)u] \cdot u \; dS = \int_\Omega \nu Au \cdot u \; dS = \nu \|u\|_{\mathcal{V}}^2.$$

This guarantees Estimate 3.8. □

Finally we are concerned with some estimates for the nonlinear advection operator B.

ESTIMATE 3.9 (estimating the advection operator). [illegible] B [illegible] (2.3) [illegible]

(i) [illegible] $v \in D(A), u \in \mathcal{V}$ [illegible]

$$\|B(u,v)\|_{l^2(\Omega)} \leq c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}^{1/2} \|Av\|_{l^2(\Omega)}^{1/2}. \tag{3.6}$$

(ii) [illegible] $u \in D(A)$ [illegible]

$$\|B(u,u)\|_{l^2(\Omega)} \leq c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}} \|Au\|_{l^2(\Omega)}^{1/2}.$$

(iii) [illegible] $v \in D(A), u \in \mathcal{V}$ [illegible]

$$\|B(u,v)\|_{\mathcal{V}'} \leq c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}^{1/2} \|v\|_{l^2(\Omega)}^{1/2}.$$

(iv) [illegible] $u, v \in D(A)$ [illegible]

$$\|B(u,v)\|_{\mathcal{V}'} \leq c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}} \|v\|_{\mathcal{V}}^{1/2}.$$

(v) [illegible] $u, v \in D(A)$ [illegible]

$$\|B(u,v)\|_{l^2(\Omega)} \leq c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}} \|v\|_{\mathcal{V}}^{1/2}.$$

(vi) [illegible] $u, v \in D(A)$ [illegible]

$$\|B(u,v)\|_{l^2(\Omega)} \leq c_2 \|u\|_{\mathcal{V}}^{1/2} \|Au\|_{l^2(\Omega)}^{1/2} \|v\|_{\mathcal{V}}.$$

Before entering into detailed steps for the proof of these five estimates, we need a preliminary result.

LEMMA 3.10. [illegible] $v \in h^{(3)}(\{A_n\};\Omega) \subset l^2_{(3)}(\Omega)$ [illegible] $\{A_n\}$ [illegible]

$$\|v \cdot \varepsilon_i\|_{\mathcal{H}(\{A_n\};\Omega)} \le \|v\|_{h(\{A_n\};\Omega)}$$

[illegible] $= 1, 2, 3$

[illegible] It is clear that $\|v \cdot \varepsilon_i\|_{L^2(\Omega)} \le \|v\|_{l^2(\Omega)}$. As $y^{(3)}_{n,k} \in \oplus^3_{i=1}\mathrm{Harm}_n(\Omega)\,\varepsilon_i$ for all $k = -n, \dots, n$ (see [11]). Since $v \in l^2_{(3)}(\Omega)$ we are able to verify that

$$\sum_{k=-n}^{n} \left\langle v, y^{(3)}_{n,k} \right\rangle_{l^2(\Omega)} y^{(3)}_{n,k} = \sum_{i=1}^{3} \sum_{k=-n}^{n} \langle v, Y_{n,k}\varepsilon_i \rangle_{l^2(\Omega)}\, Y_{n,k}\varepsilon_i.$$

From the right-hand side we get an estimate for the Fourier coefficients of the component functions $v_i, i = 1, 2, 3$,

$$\sum_{k=-n}^{n} |(v \cdot \varepsilon_i)^\wedge(n,k)|^2 \le \sum_{i=1}^{3} \sum_{k=-n}^{n} |(v \cdot \varepsilon_i)^\wedge(n,k)|^2 = \sum_{k=-n}^{n} |v^{(3)}(n,k)|^2.$$

By similar arguments we obtain

$$A_n^2 \sum_{k=-n}^{n} |(v \cdot \varepsilon_i)^\wedge(n,k)|^2 \le A_n^2 \sum_{k=-n}^{n} |v^{(3)}(n,k)|^2$$
$$\|v \cdot \varepsilon_i\|_{\mathcal{H}(\{A_n\};\Omega)} \le \|v\|_{h(\{A_n\};\Omega)}. \qquad \square$$

Lemmas 3.3 and 3.10 enable us to establish the proof of Estimate 3.9.

[illegible] (i) The first estimate can be derived from (3.4) and (3.5):

$$|b(u,v,w)| \le \left(\sum_{i=1}^{3} \|u_i\|_{L^4(\Omega)}\right)^{1/2} \left(\sum_{i=1}^{3}\sum_{j=1}^{3} \|\nabla_i v_j\|_{L^2(\Omega)}\right)^{1/2} \left(\sum_{j=1}^{3} \|w_j\|_{L^4(\Omega)}\right)^{1/2}$$

$$\le \left(\sum_{i=1}^{3}\sum_{j=1}^{3} \|\nabla_i v_j\|_{L^2(\Omega)}\right)^{1/2} \sqrt{3}c_1 \|u\|^{1/2}_{l^2(\Omega)} \|u\|^{1/2}_{\mathcal{V}} \|w\|_{l^2(\Omega)}. \tag{3.7}$$

It remains to look at the first term. Due to our assumption we have $v \in D(A) \subset h(\{n(n+1)\};\Omega)$. Thus, $\nabla_i v_j \in \mathcal{H}(\{\sqrt{n(n+1)}\};\Omega) \subset L^4(\Omega) \subset L^2(\Omega)$ by the aid of Lemma 3.3. This yields $\|\nabla_i v_j\|_{L^2(\Omega)} \le \|\nabla_i v_j\|_{L^4(\Omega)}$. We observe

$$\|\nabla_i v_j\|^4_{L^4(\Omega)} = \|(\varepsilon_i \cdot \nabla) v_j\|^4_{L^4(\Omega)} \le \|\nabla v_j\|^4_{l^4(\Omega)} \le c_1^4 \|\nabla v_j\|^2_{l^2(\Omega)} \|\nabla v_j\|^2_{\mathcal{V}}. \tag{3.8}$$

The term on the right-hand side of (3.8) can be further simplified by

$$\|\nabla v_j\|^4_{l^4(\Omega)} \le c_1^4 \|\nabla^* v_j\|^2_{l^2(\Omega)} \|\nabla^* v_j\|^2_{\mathcal{V}} = c_1^4 \|v_j\|^2_{\mathcal{V}} \|\Delta^* v_j\|^2_{L^2(\Omega)}.$$

By using the auxiliary Lemma 3.10 we get

$$\|\nabla_i v_j\|^4_{L^4(\Omega)} \le c_1^4 \|v\|^2_{\mathcal{V}} \|\Delta^* v\|^2_{l^2(\Omega)} = c_1^4 \|v\|^2_{\mathcal{V}} \|Av\|^2_{l^2(\Omega)}.$$

Taking the fourth root of both sides of the equation yields

$$\|\nabla_i v_j\|_{L^4(\Omega)} \le c_1 \|v\|^{1/2}_{\mathcal{V}} \|Av\|^{1/2}_{l^2(\Omega)}.$$

By plugging the last result into (3.7) we find

$$|b(u,v,w)| \le 3\sqrt{3}c_1^2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}^{1/2} \|Av\|_{l^2(\Omega)}^{1/2} \|w\|_{l^2(\Omega)}.$$

Finally we can conclude the first estimate for the advection operator B:

$$\begin{aligned}\|B(u,v)\|_{l^2(\Omega)} &= \sup_{w\in H} \frac{\langle B(u,v),w\rangle_{l^2(\Omega)}}{\|w\|_{l^2(\Omega)}} = \sup_{w\in H} \frac{|b(u,v,w)|}{\|w\|_{l^2(\Omega)}} \\ &\le 3\sqrt{3}c_1^2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}^{1/2} \|Av\|_{l^2(\Omega)}^{1/2}.\end{aligned}$$

(ii) The second estimate follows from the first one by setting $v = u$.

(iii) To verify the third estimate we use the estimate from (3.2)

$$|b(u,v,w)| \le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} \|w\|_{l^2(\Omega)}^{1/2} \|w\|_{\mathcal{V}}^{1/2} \tag{3.9}$$

and use the antisymmetry relation for the trilinear form b from Lemma 3.1, i.e., $b(u,v,w) = -b(u,w,v)$, to deduce that

$$|b(u,v,w)| \le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|w\|_{\mathcal{V}} \|v\|_{l^2(\Omega)}^{1/2} \|v\|_{\mathcal{V}}^{1/2}.$$

By keeping in mind the Riesz–Fischer representation theorem we may regard for all $u \in \mathcal{V}$ and $v \in D(A)$ the value $B(u,v)$ as a representation of a linear functional in $\mathcal{V}'$. Thus we can estimate the norm of this functional by

$$\begin{aligned}\|B(u,v)\|_{\mathcal{V}'} &= \sup_{w\in\mathcal{V}} \frac{|\langle B(u,v),w\rangle_{l^2(\Omega)}|}{\|w\|_{\mathcal{V}}} = \sup_{w\in\mathcal{V}} \frac{|b(u,v,w)|}{\|w\|_{\mathcal{V}}} \\ &\le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{l^2(\Omega)}^{1/2} \|v\|_{\mathcal{V}}^{1/2},\end{aligned}$$

which finally proves the third estimate.

(iv) The proof of this estimate follows directly from relation (iii) and (3.9), i.e.,

$$\|B(u,v)\|_{\mathcal{V}'} = \sup_{\|w\|_{\mathcal{V}}=1} |b(u,v,w)| \le \sup_{\|w\|_{\mathcal{V}}=1} c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} \|w\|_{l^2(\Omega)}^{1/2} \|w\|_{\mathcal{V}}^{1/2}.$$

Since $\|w\|_{l^2(\Omega)} \le \|w\|_{\mathcal{V}}$ from the Poincaré inequality we finally arrive at

$$\|B(u,v)\|_{\mathcal{V}'} \le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}.$$

(v) The fifth estimate can easily be obtained from (iv) by applying again the Riesz–Fischer representation theorem. This theorem yields the identity $\|B(u,v)\|_{\mathcal{V}'} = \|B(u,v)\|_{\mathcal{V}}$. From $\|B(u,v)\|_{l^2(\Omega)} \le \|B(u,v)\|_{\mathcal{V}}$ we get the desired result.

(vi) The last estimate can be derived by using again (3.2). The Poincaré inequality yields

$$\begin{aligned}\|B(u,v)\|_{l^2(\Omega)} \le \|B(u,v)\|_{\mathcal{V}} &= \|B(u,v)\|_{\mathcal{V}'} \\ &\le \sup_{\|w\|_{\mathcal{V}}=1} c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} \|w\|_{l^2(\Omega)}^{1/2} \|w\|_{\mathcal{V}}^{1/2} \\ &\le \sup_{\|w\|_{\mathcal{V}}=1} c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} \|w\|_{\mathcal{V}} \\ &\le c_2 \|u\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}.\end{aligned}$$

Since we required $u \in D(A) \subset h^{(3)}(\{n(n+1)\};\Omega)$, we obtain the result by Corollary 3.4:

$$\|B(u,v)\|_{l^2(\Omega)} \le c_2 \|u\|_{h^{(3)}(\{n(n+1)\};\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}} = c_2 \|Au\|_{l^2(\Omega)}^{1/2} \|u\|_{\mathcal{V}}^{1/2} \|v\|_{\mathcal{V}}. \qquad \square$$

4. The nonlinear Galerkin method on the sphere. The implementation of the Galerkin method is based on the set of eigenvectors $\{y^{(3)}_{n,k}\}$, $n = 1, \ldots; k = -n, \ldots, n$, of the operator A with eigenvalues $0 < n(n+1), n = 1, 2, \ldots$. Further, for $m \in \mathbb{N}$, we consider the spaces $\mathrm{harm}^{(3)}_{1,\ldots,m}(\Omega)$ and $\mathrm{harm}^{(3)}_{m+1,\ldots,2m}(\Omega)$, respectively. Moreover, let $P_{1,\ldots,m}$ be the orthogonal projection operator $P_{1,\ldots,m} : l^2(\Omega) \to \mathrm{harm}^{(3)}_{1,\ldots,m}(\Omega)$, respectively, $P_{m+1,\ldots,2m}$ the projector on $\mathrm{harm}^{(3)}_{m+1,\ldots,2m}(\Omega)$.

Assume that u_0 is of class H, and let u be the solution of (2.12) with the initial condition $u(0) = u_0$. For every integer m, we are now looking for an approximation u_m of the form

$$u_m : \mathbb{R}^+_0 \to \mathrm{harm}^{(3)}_{1,\ldots,m}(\Omega), \qquad u_m(t) = \sum_{n=1}^{m} \sum_{k=-n}^{n} u^m_{n,k}(t) y^{(3)}_{n,k}. \tag{4.1}$$

The function u_m is determined by the resolution of a system involving another unknown function z_m given by

$$z_m : \mathbb{R}^+_0 \to \mathrm{harm}^{(3)}_{m+1,\ldots,2m}(\Omega), \qquad z_m(t) = \sum_{n=m+1}^{2m} \sum_{k=-n}^{n} z^m_{n,k}(t) y^{(3)}_{n,k}. \tag{4.2}$$

The pair (u_m, z_m) is expected to fulfill the equations

$$\begin{aligned} \frac{\partial}{\partial t}\langle u_m, v\rangle_{l^2(\Omega)} + \nu\langle Au_m, v\rangle_{l^2(\Omega)} + \langle Cu_m, v\rangle_{l^2(\Omega)} + b(u_m, u_m, v) \\ + b(z_m, u_m, v) + b(u_m, z_m, v) = \langle f, v\rangle_{l^2(\Omega)}, \end{aligned} \tag{4.3}$$

$$\nu\langle Az_m, \tilde{v}\rangle_{l^2(\Omega)} + \langle Cz_m, \tilde{v}\rangle_{l^2(\Omega)} + b(u_m, u_m, \tilde{v}) = \langle f, \tilde{v}\rangle_{l^2(\Omega)} \tag{4.4}$$

for all $v \in \mathrm{harm}^{(3)}_{1,\ldots,m}(\Omega)$ and $\tilde{v} \in \mathrm{harm}^{(3)}_{m+1,\ldots,2m}(\Omega)$ together with

$$u_m(0) = P_{1,\ldots,m} u_0. \tag{4.5}$$

Our purpose is to show that systems (4.3) and (4.4) are equivalent to an ordinary differential equation in terms of u_m.

Observing that $P_{m+1,\ldots,2m} A z_m = A z_m$, (4.4) is indeed linear in z_m and hence can be written as

$$(\nu A + P_{m+1,\ldots,2m} C) z_m = P_{m+1,\ldots,2m}(f - B(u_m, u_m)). \tag{4.6}$$

By virtue of Estimate 3.8, [17] guarantees the invertibility of the projected operator $P_{m+1,\ldots,2m}(\nu A + C)$ on $\mathrm{harm}^{(3)}_{m+1,\ldots,2m}(\Omega)$. Therefore, we have z_m explicitly given in terms of u_m by

$$z_m = (\nu A + P_{m+1,\ldots,2m} C)^{-1} P_{m+1,\ldots,2m}(f - B(u_m, u_m)). \tag{4.7}$$

Then, systems (4.3) and (4.4) are equivalent to the ordinary differential system

$$\frac{\partial u_m}{\partial t} + \nu A u_m + P_{1,\ldots,m}(Cu_m + B(u_m, u_m) + B(z_m, u_m) + B(u_m, z_m)) = P_{1,\ldots,m} f \tag{4.8}$$

with z_m given by (4.7).

Note that the system (4.8) with $z_m = 0$ yields exactly as a byproduct of the linear Galerkin method on the sphere.

Existence and uniqueness of a solution u_m to the system (4.8) equipped with the initial value from (4.5) defined on a maximal interval $[0, T_{\max})$ follows from standard theorems on ordinary differential equations. In fact, it is known that $T_{\max} = +\infty$ (see, for example, [17]). Based on all results stated above, we can study now the limit $m \to +\infty$.

THEOREM 4.1 (main convergence theorem). [illegible] 3.1 [illegible] 3.5 3.7 3.9 [illegible] u_m [illegible] (4.8) [illegible] (4.5) [illegible] $m \to +\infty$ [illegible] (2.12) [illegible] $u(0) = u_0$ [illegible] $u_0 \in H$ [illegible]

(4.9)
- $u_m \to u$ [illegible] $L^2(0,T;\mathcal{V})$ [illegible] $L^p(0,T;H)$ [illegible] $T > 0$ [illegible] $1 \le p < \infty$;
- $u_m \to u$ [illegible] $L^\infty(\mathbb{R}^+;H)$ [illegible]

The proof can be found in [17] for a general Hilbert space concept. It outlines the intuition that on the one hand z_m is small compared to u_m, but on the other hand that, at each given time, $z_m(t)$ is a minor correction to $u_m(t)$. Moreover, it turns out that on long intervals of times, z_m modifies u_m in a nonnegligible way.

5. Implementation of the nonlinear Galerkin scheme. Referring to (4.3) and (4.4) we formulate here an algorithm based on an Euler method for the sake of clarity. However, it should be outlined that we use in our numerical implementation an explicit Runge–Kutta-4(5) method.

ALGORITHM 5.1 (nonlinear Galerkin scheme (Euler variant)).
Purpose: Compute an approximation $u_m(t)$, $t \in [t_0, T]$, for some $T > t$ based on an Euler scheme. Let u_0 be the initial value, i.e., $u_0 = u(t_0) \in H$. Moreover, let $t_i = t_0 + ih$ be the time-step with some time-stepping size $h > 0$.

while $t_i \le T$
Compute $z_m = z_m(t_i)$ from $u_m = u_m(t_i)$ and $f = f(t_i)$ by solving the linear system

(5.1)
$$\nu \left\langle Az_m, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} + \left\langle Cz_m, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} + \left\langle B(u_m, u_m), y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} = \left\langle f, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)}$$

for $n = m+1, \dots, 2m; j = -n, \dots, n$. Compute $k = \sum_{n=1}^{m} \sum_{j=-n}^{n} k^{(3)}_{n,j} y^{(3)}_{n,j}$ by solving the linear system for the Fourier coefficients

$$\left\langle k, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} = \left\langle f, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} - \nu \left\langle Au_m, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} - \left\langle Cu_m, y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} \tag{5.2}$$

$$- \left\langle B(u_m, u_m), y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} - \left\langle B(z_m, u_m), y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} - \left\langle B(u_m, z_m), y^{(3)}_{n,j} \right\rangle_{l^2(\Omega)} \tag{5.3}$$

for $n = 1, \dots, m; j = -n, \dots, n$. Continue with the next step

$$u_m(t_{i+1}) = u_m(t_i) + hk$$

end while

The above algorithm requires the computation of coupling integrals, which are of special interest in the case of the Coriolis term and nonlinear advection. In [10] we find the following theorems, which are very cumbersome to establish by the use of Wigner-3j, -6j, and -9j symbols.

THEOREM 5.2 (calculation of the Coriolis term). [illegible] $k,l,n,j \in \mathbb{N}$ [illegible] $k>0$ $r>0$ $n>0$ [illegible] $\omega = |\omega|\varepsilon^3$ [illegible]

$$\int_\Omega \left(\omega \wedge y_{k,l}^{(3)}(\eta)\right) \cdot y_{n,j}^{(3)*}(\eta)\, dS(\eta) = |\omega| i \frac{-l}{n(n+1)} \delta_{k,n}\delta_{l,j}. \tag{5.4}$$

THEOREM 5.3 (calculation of the advection term). [illegible] $k,l,n,j,r,s \in \mathbb{N}$ [illegible] $k>0, r>0$ [illegible] $n>0$ [illegible]

$$\int_\Omega \left[\left(y_{k,l}^{(3)}(\eta)\cdot\nabla_\eta^*\right) y_{r,s}^{(3)}(\eta)\right] \cdot y_{n,j}^{(3)*}(\eta)\, dS(\eta) = T(n,j,r,s,k,l), \tag{5.5}$$

[illegible]

$$\begin{aligned} T(n,j,r,s,k,l) = (-1)^{j+1} i \frac{1}{\sqrt{4\pi}} & \frac{n(n+1)+r(r+1)-k(k+1)}{4\sqrt{r(r+1)n(n+1)k(k+1)}} \\ & \times \begin{pmatrix} r & n & k \\ s & -j & l \end{pmatrix} \sqrt{(2r+1)(2n+1)(2k+1)} \begin{pmatrix} r-1 & n & k \\ 0 & 0 & 0 \end{pmatrix} \\ & \times \sqrt{(n+k+r+1)(k+r-n)(n-k+r)(n+k-r+1)}. \end{aligned}$$

COROLLARY 5.4. [illegible] $3j$ [illegible] (5.5) [illegible]

$$\int_\Omega \left[\left(y_{k,l}^{(3)}(\eta)\cdot\nabla_\eta^*\right) y_{r,s}^{(3)}(\eta)\right] \cdot y_{n,j}^{(3)*}(\eta)\, dS(\eta) = 0,$$

[illegible]

(i) $k > r+n$ [illegible] $k < |r-n|$
(ii) $s+l-j \neq 0$
(iii) $r+n+k$ [illegible]

Now we consider the nonlinear part in (5.1) in more detail by dropping for a moment all other terms. Expanding u in terms of type 3 vector spherical harmonics, we are able to express the Fourier coefficients of z in connection with (5.5) by

$$\tilde{z}_{n,j} = \sum_{r=1}^{N} \sum_{s=-r}^{r} \sum_{k=1}^{N} \sum_{l=-k}^{k} u_{r,s}^{(3)} u_{k,l}^{(3)} T(n,j,r,s,k,l) \tag{5.6}$$

for $n = N+1,\dots,2N; |j| \le n$. Since our considered velocity field of the flow is real-valued, we can restrict the computation to all j with $0 \le j \le n$ and obtain all remaining coefficients for negative orders by antisymmetry. To estimate the number of operations required to compute the coefficients $\tilde{z}_{n,j}$, we assume that the quantity

T has been tabulated beforehand. By exploiting the Wigner-3j selection rule from Corollary 5.4(iii), we have a total of

$$\sum_{n=N+1}^{2N}(n+1)\sum_{r=1}^{N}(2r+1)^2 \approx 2N^5$$

complex × complex × real multiplications, each of them equivalent to six real multiplications. Selection rules for 3j-coefficients with zero bottom row require $r-1+n+k$ to be even, reducing the number of terms in (5.6) by a factor of 2. Thus, the number of real multiplications is about $6N^5$. For each six real multiplications, (5.6) involves two real additions; and thus a total of about $2N^5$ additions.

Since values of N greater than 100 occur in practice (for example, in meteorology, the European Center for Medium-Range Weather Forecast (ECMWF) uses $N=511$; see [22]), it is important to reduce the preceding estimate for the number of operations. Furthermore, the preceding estimate assumes that the coupling integrals have been tabulated in an a priori step. Although Wigner selection rules and symmetries greatly reduce the storage required, the number of storage locations scale like $O(N^5)$.

To improve the scaling for the evaluation of nonlinear terms, several authors [2, 6, 12, 20, 15, 25] introduced FFT techniques to evaluate these so-called vector-coupled sums. The first seems to be Orszag [20] in 1970, whereas James [15] extended Orszag's method to his tensor calculus in 1974. For a detailed overview of pseudospectral methods the reader is referred to Corey and Tromp [6] and the references therein.

5.1. Fast pseudospectral method (FPSM). The extension of Orszag's method to spherical vector and tensor fields can be found by James [15] for his tensor calculus. Unfortunately, his rank-k-tensor approach differs in its basis representation from our philosophy: We like to separate any vector and tensor field into physically meaningful parts, e.g., into tangential and radial components, or to be more specific, into surface gradient and surface curl (toroidal) contributions.

Thus, this section is dedicated to the rapid evaluation of the nonlinear advection term $(u\cdot\nabla^*)u = (\nabla^*\otimes u)^T u$ involving the framework of vectorial and tensorial harmonics. The main advantage of our methods proposed in the following is that they are based on [illegible] integration in the Gauss–Legendre transform (GLT) rather than on solving linear systems as proposed by James [15]. For a detailed description of the vectorial and tensorial variants, algorithms, and examples see [10]. The techniques introduced here allow us to evaluate and reconstruct the advection term in the order $O(N^3)$ instead of the estimate $O(N^5)$ explained above.

Idea. Before formulating the algorithm we sketch for brevity the idea of pseudospectral methods in the vectorial type 3 case. Obviously the evaluation and reconstruction of real type 3 vector fields $u_N^{(3)} \in \mathrm{harm}^{(3)}_{1,\dots,N}(\Omega)$ can be written out as follows:

$$u_N^{(3)} = \sum_{n=1}^{N}\sum_{k=-n}^{n} u_{n,k}^{(3)} y_{n,k}^{(3)}, \tag{5.7}$$

where

$$y_{n,k}^{(3)}(t,\phi) = \frac{(-1)^k}{\sqrt{n(n+1)}}\left[\frac{ik}{\sqrt{1-t^2}}c_{n,k}P_n^k(t)e^{ik\phi}\varepsilon^t - \sqrt{1-t^2}c_{n,k}P_n^{k'}(t)e^{ik\phi}\varepsilon^\phi\right]$$

for $n = 1, \ldots, N; k = 0, \ldots, N$. Thereby, $P_n^{k'}(t)$ denotes the first derivative of $P_n^k(t)$ with respect to t, and in what follows $P_n^{k''}(t)$ will denote the second derivative. Usually, the coefficients $c_{n,k}$ are chosen such that the spherical harmonics are orthonormal with respect to $\langle \cdot, \cdot \rangle_{L^2(\Omega)}$; see, for example, [8]. The negative orders can be obtained by

$$y_{n,-k}^{(3)}(t,\phi) = (-1)^k \left(y_{n,k}^{(3)}(t,\phi) \right)^*, \quad k = 1, \ldots, N.$$

Before entering details of our extended pseudospectral method, we should outline that the occurring singularities at the poles are of removable type [8]. Moreover, we do not expect computational problems, since we use in a latitudinal direction an open Gauss–Legendre quadrature (i.e., the poles do not belong to our integration nodes). The latter fact allows us to handle the terms belonging to ε^ϕ, respectively, ε^t, separately. Since both do not depend on the summations, the problem of reconstructing $u_N^{(3)}$ on the same tensor product grid as introduced above separates outside the poles into two subproblems such that

$$u_N^{(3)} = {}^{\varepsilon^t}u^{(3)}\varepsilon^t + {}^{\varepsilon^\phi}u^{(3)}\varepsilon^\phi.$$

Further, we introduce a vectorial expansion and a corresponding scalar reduction between backward and forward transform. Then, the [illegible] for type 3 vector fields works as follows.

Pseudospectral algorithm for type 3 vector fields. Pseudospectral transforms are essentially based on the [illegible] notation provided by (5.7) by interchanging the summation for n and k. Moreover, we set $y_{0,0}^{(3)} = 0$ and its "Fourier coefficient" $u_{0,0}^{(3)} = 0$ to benefit in a clear algorithmic structure. This introduced [illegible] saves us the analogy to the scalar pseudospectral algorithm (see, for example, [6]). To evaluate $u_N^{(3)} = {}^{\varepsilon^t}u^{(3)}\varepsilon^t + {}^{\varepsilon^\phi}u^{(3)}\varepsilon^\phi$ on the Gauss–Legendre grid [6], we have to compute the quantities

$${}^{\varepsilon^t}u_{*,k}^{(3)}(t_j) = \sum_{n=\max\{|k|,1\}}^{N} u_{n,k}^{(3)} \frac{(-1)^k}{\sqrt{n(n+1)}} \frac{ik}{\sqrt{1-t_j^2}} c_{n,k} P_n^k(t_j),$$

$j = 1, \ldots, N+1; k = 0, \ldots, N$, and

$${}^{\varepsilon^\phi}u_{*,k}^{(3)}(t_j) = \sum_{n=\max\{|k|,1\}}^{N} u_{n,k}^{(3)} \frac{-(-1)^k}{\sqrt{n(n+1)}} \sqrt{1-t_j^2}\, c_{n,k} P_n^{k'}(t_j),$$

$j = 1, \ldots, N+1; k = 0, \ldots, N$.

Since $u_N^{(3)}$ is real, we store ${}^{\varepsilon^t}u_{*,k}^{(3)}(t_j) = ({}^{\varepsilon^t}u_{*,k}^{(3)}(t_j))^*$ in decreasing order $k = N, \ldots, 1$ columnwise. To evaluate finally ${}^{\varepsilon^t}u^{(3)}$ on the Gauss–Legendre grid, we apply an inverse FFT in longitudinal direction (rowwise) and expand by ε^t. The same procedure for ${}^{\varepsilon^\phi}u_{*,k}^{(3)}$, which is associated with ε^ϕ, allows us to reconstruct $u_N^{(3)}(t_j, \phi_i) = {}^{\varepsilon^t}u^{(3)}(t_j, \phi_i)\varepsilon^t + {}^{\varepsilon^\phi}u^{(3)}(t_j, \phi_i)\varepsilon^\phi, j = 1, \ldots, N+1; i = 1, \ldots, 2N+1$.

Conversely, we can determine the Fourier coefficients of $u_N^{(3)}$ by integration: Assume $u_N^{(3)}$ to be real-valued in $\mathrm{harm}_{1,\ldots,N}^{(3)}(\Omega)$, to be given on the grid introduced above. By applying Fubini's theorem, we are able to deduce for the Fourier coefficients $u_{n,k}^{(3)}$

that

(5.8)

$$
\begin{aligned}
u_{n,k}^{(3)} &= \int_{\Omega} u^{(3)}(\eta)\cdot y_{n,k}^{(3)*}(\eta)\; dS(\eta)\\
&= \int_{-1}^{1}\Bigg[\underbrace{\frac{1}{2\pi}\int_0^{2\pi}\left(u^{(3)}(t,\phi)\cdot\varepsilon^{t}\right)e^{-ik\phi}d\phi}_{={}^{\varepsilon^t}u_{*,k}^{(3)}(t)}\Bigg]\frac{(-1)^k}{\sqrt{n(n+1)}}\frac{-ik}{\sqrt{1-t^2}}c_{n,k}P_n^k(t)\,dt\\
&\quad+\int_{-1}^{1}\Bigg[\underbrace{\frac{1}{2\pi}\int_0^{2\pi}\left(u^{(3)}(t,\phi)\cdot\varepsilon^{\phi}\right)e^{-ik\phi}d\phi}_{={}^{\varepsilon^\phi}u_{*,k}^{(3)}(t)}\Bigg]\frac{-(-1)^k}{\sqrt{n(n+1)}}\sqrt{1-t^2}c_{n,k}P_n^{k'}(t)\,dt\\
&= \int_{-1}^{1}{}^{\varepsilon^t}u_{*,k}^{(3)}(t)\frac{(-1)^k}{\sqrt{n(n+1)}}\frac{-ik}{\sqrt{1-t^2}}c_{n,k}P_n^k(t)\,dt\\
&\quad+\int_{-1}^{1}{}^{\varepsilon^\phi}u_{*,k}^{(3)}(t)\frac{-(-1)^k}{\sqrt{n(n+1)}}\sqrt{1-t^2}c_{n,k}P_n^{k'}(t)\,dt.
\end{aligned}
$$

Thus, we can apply an FFT in longitudinal direction to evaluate ${}^{\varepsilon^t}u_{*,k}^{(3)}(t_j), j=1,\dots,N+1$, and ${}^{\varepsilon^\phi}u_{*,k}^{(3)}(t_j), j=1,\dots,N+1$, and an exact integration to evaluate for $n=1,\dots,N; k=0,\dots,N$,

$$
{}^{\varepsilon^t}u_{n,k}^{(3)} = \sum_{j=1}^{N+1} w_j\,{}^{\varepsilon^t}u_{*,k}^{(3)}(t_j)\frac{(-1)^k}{\sqrt{n(n+1)}}\frac{-ik}{\sqrt{1-t_j^2}}c_{n,k}P_n^k(t_j) \tag{5.9}
$$

and

$$
{}^{\varepsilon^\phi}u_{n,k}^{(3)} = \sum_{j=1}^{N+1} w_j\,{}^{\varepsilon^\phi}u_{*,k}^{(3)}(t_j)\frac{-(-1)^k}{\sqrt{n(n+1)}}\sqrt{1-t_j^2}c_{n,k}P_n^{k'}(t_j)
$$

such that

$$
u_{n,k}^{(3)} = {}^{\varepsilon^t}u_{n,k}^{(3)} + {}^{\varepsilon^\phi}u_{n,k}^{(3)}.
$$

Since $u_N^{(3)}$ is real, $u_{n,-k}^{(3)}$ is given by

$$
u_{n,-k}^{(3)} = (-1)^k u_{n,k}^{(3)*}, \quad k=1,\dots,N.
$$

The same strategy can be performed analogously for tensor fields (see [10]). The idea for a fast evaluation of the nonlinear advection is to reconstruct $\nabla^*\otimes u$ and u on the exact integration grid and then to compute in the space domain $(\nabla^*\otimes u)^T u$. Consequently, we transform into the Fourier domain to obtain the corresponding coefficients. In addition, it should be noted that this method also allows the exact separation of vector fields of mixed types, since we apply a polynomially exact integration. This last point is of special importance for the advection.

Before entering into the details of our algorithm we introduce the following abbreviations: For $n=1,\dots,N; k=0,\dots,N$ we define

$$
{}^{\varepsilon^t}d_{n,k}^{(3)}(t) = \frac{(-1)^k}{\sqrt{n(n+1)}}\frac{ik}{\sqrt{1-t^2}}c_{n,k}P_n^k(t), \tag{5.10}
$$

$$
{}^{\varepsilon^\phi}d_{n,k}^{(3)}(t) = -\frac{(-1)^k}{\sqrt{n(n+1)}}\sqrt{1-t^2}c_{n,k}P_n^{k'}(t), \tag{5.11}
$$

where the upper left index reflects the association of these terms to ε^t, respectively, ε^ϕ. For structural simplicity we let ${}^{\varepsilon^t}d^{(3)}_{0,0}(t) = 0$ and ${}^{\varepsilon^\phi}d^{(3)}_{0,0}(t) = 0$ corresponding to the "Fourier coefficient" $u^{(3)}_{0,0} = 0$ operating in the following as a [illegible]. From the idea explained above, it is advisable to have a closer look at the reconstruction of $\nabla^* \otimes u^{(3)} = \mathbf{u}^{(2,3)}$ tensor fields. These tensor fields can be written as

$$\begin{aligned}\mathbf{u}^{(2,3)}(\eta) &= \sum_{n=1}^{N}\sum_{k=-n}^{n} u^{(2,3)}_{n,k}(\nabla^*_\eta \otimes y^{(3)}_{n,k})(\eta)\\ &= \sum_{k=-N}^{N}\sum_{n=\max\{|k|,1\}}^{N} u^{(2,3)}_{n,k}(\nabla^*_\eta \otimes y^{(3)}_{n,k})(\eta).\end{aligned}$$

From (2.1) we have $\mathbf{y}^{(2,3)}_{n,k}(\eta) = (\nabla^*_\eta \otimes y^{(3)}_{n,k})(\eta)$. By some minor calculation we derive a representation of $\mathbf{y}^{(2,3)}_{n,k}$ that allows us again to separate scalar latitudinal and longitudinal terms:

(5.12)

$$\begin{aligned}\mathbf{y}^{(2,3)}_{n,k}(t,\phi) = e^{ik\phi}\Bigg\{ &\\ &\left[\frac{-ik(-1)^k}{\sqrt{n(n+1)}}c_{n,k}P^{k'}_n(t) - \frac{t}{1-t^2}\frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^k_n(t)\right]\varepsilon^\phi\otimes\varepsilon^\phi\\ &+\left[\frac{-k^2}{1-t^2}\frac{1}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^k_n(t)\right]\varepsilon^\phi\otimes\varepsilon^t\\ &+\left[\frac{t}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^{k'}_n(t) - \frac{1-t^2}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^{k''}_n(t)\right]\varepsilon^t\otimes\varepsilon^\phi\\ &+\left[\frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^k_n(t)\right]\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t\\ &+\left[\frac{-1}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^{k'}_n(t)\right]\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi\\ &+\left[\frac{t}{1-t^2}\frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^k_n(t) + \frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P^{k'}_n(t)\right]\varepsilon^t\otimes\varepsilon^t\Bigg\}\end{aligned}$$

for $n = 1,\dots,N; k = 0,\dots,N$. The negative orders can be obtained by observing

$$\mathbf{y}^{(2,3)}_{n,-k} = (-1)^k\left(\mathbf{y}^{(2,3)}_{n,k}\right)^*.$$

It turns out that the problem of reconstructing $\mathbf{u}^{(2,3)}$ on the same tensor product grid as introduced above separates into six subproblems, namely,

$$\begin{aligned}\mathbf{u}^{(2,3)} = {}& {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u^{(2,3)}\varepsilon^\phi\otimes\varepsilon^\phi + {}^{\varepsilon^\phi\otimes\varepsilon^t}u^{(2,3)}\varepsilon^\phi\otimes\varepsilon^t + {}^{\varepsilon^t\otimes\varepsilon^\phi}u^{(2,3)}\varepsilon^t\otimes\varepsilon^\phi\\ &+ {}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}u^{(2,3)}\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t + {}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}u^{(2,3)}\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi + {}^{\varepsilon^t\otimes\varepsilon^t}u^{(2,3)}\varepsilon^t\otimes\varepsilon^t.\end{aligned}$$

According to this partition we introduce the following six functions (in analogy to the [illegible] known from quantum mechanics [5]):

(5.13)

$$
\begin{aligned}
{}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{n,k}^{(2,3)}(t) &= \frac{-ik(-1)^k}{\sqrt{n(n+1)}}c_{n,k}P_n^{k'}(t) - \frac{t}{1-t^2}\frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^k(t),\\
{}^{\varepsilon^\phi\otimes\varepsilon^t}d_{n,k}^{(2,3)}(t) &= \frac{-k^2}{1-t^2}\frac{1}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^k(t),\\
{}^{\varepsilon^t\otimes\varepsilon^\phi}d_{n,k}^{(2,3)}(t) &= \frac{t}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^{k'}(t) - \frac{1-t^2}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^{k''}(t),\\
{}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}d_{n,k}^{(2,3)}(t) &= \frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^k(t),\\
{}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}d_{n,k}^{(2,3)}(t) &= \frac{-1}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^{k'}(t),\\
{}^{\varepsilon^t\otimes\varepsilon^t}d_{n,k}^{(2,3)}(t) &= \frac{t}{1-t^2}\frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^k(t) + \frac{ik}{\sqrt{n(n+1)}}(-1)^k c_{n,k}P_n^{k'}(t)
\end{aligned}
$$

for $n = 1,\dots,N; k = 0,\dots,N$. Note that the upper left index reflects the association of these terms to the corresponding tensors.

To save us a clear matrix-vector notation, we again introduce dummy elements, viz.,

$$
\begin{aligned}
{}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{0,0}^{(2,3)}(t) &= 0, & {}^{\varepsilon^\phi\otimes\varepsilon^t}d_{0,0}^{(2,3)}(t) &= 0,\\
{}^{\varepsilon^t\otimes\varepsilon^\phi}d_{0,0}^{(2,3)}(t) &= 0, & {}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}d_{0,0}^{(2,3)}(t) &= 0,\\
{}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}d_{0,0}^{(2,3)}(t) &= 0, & {}^{\varepsilon^t\otimes\varepsilon^t}d_{0,0}^{(2,3)}(t) &= 0,\\
u_{0,0}^{(2,3)} &= 0.
\end{aligned}
$$

Then, except for the vectorial expansion, the backward transform into the space domain works in analogy to the type 3 vector spherical harmonic case. Accordingly, the vectorial expansion is substituted by the corresponding tensorial expansion.

ALGORITHM 5.5 (FSPM for the nonlinear advection term).
Purpose: The algorithm computes the Fourier coefficients $v_{n,k}^{(3)}$ corresponding to $v = (u\cdot\nabla^*)u = (\nabla^*\otimes u)^T u$. In the first part of this algorithm we evaluate the tensorial term $[(\nabla_\eta^*\otimes u)(\eta)]^T$ and then the vectorial term $u(\eta)$ of the integrand.

We use in the following the Gauss–Legendre integration grid: The grid points (t_j,ϕ_i) are given by $\phi_i = 2\pi(i-1)/(4N+1), i = 1,\dots,4N+1$, and $t_j, j = 1,\dots,2N+1$, where the latter denote the Gauss–Legendre integration nodes with corresponding weights $w_j, j = 1,\dots,2N+1$. This grid yields a cubature rule which allows polynomial exact integration up to degree $2(2N+1)-1 = 4N+1$.

1. [illegible] $(\nabla^*\otimes u)$ [illegible]

(i) Compute the quantities ${}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,k}^{(2,3)}(t_j) = \sum_{n=|k|}^N u_{n,k}^{(3)}\,{}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{n,k}^{(2,3)}(t_j)$ for $k = 0,\dots,N; j = 1,\dots,2N+1$. This can be achieved by applying an inverse GLT. Assume that the $N+1$ (rectangular) Gauss–Legendre matrices have been tabulated

beforehand. Equipped with them, perform matrix-vector products between

$$ {}^{\varepsilon^\phi\otimes\varepsilon^\phi}D_k^{(2,3)} = \begin{pmatrix} {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k,k}^{(2,3)}(t_1) & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k+1,k}^{(2,3)}(t_1) & \dots & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{N,k}^{(2,3)}(t_1) \\ {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k,k}^{(2,3)}(t_2) & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k+1,k}^{(2,3)}(t_2) & \dots & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{N,k}^{(2,3)}(t_2) \\ \vdots & \vdots & & \vdots \\ {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k,k}^{(2,3)}(t_{2N+1}) & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{k+1,k}^{(2,3)}(t_{2N+1}) & \dots & {}^{\varepsilon^\phi\otimes\varepsilon^\phi}d_{N,k}^{(2,3)}(t_{2N+1}) \end{pmatrix} \text{ and} $$

$$ u_{*,k} = \begin{pmatrix} u_{k,k}^{(3)} & u_{k+1,k}^{(3)} & \dots & u_{N,k}^{(3)} \end{pmatrix}^T \tag{5.14} $$

to obtain the vectors

$$ {}^{\varepsilon^\phi\otimes\varepsilon^\phi}D_k^{(2,3)}u_{*,k} = {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,k}^{(2,3)} \tag{5.15} $$

for $k = 0, \dots, N$. Assume that the column vectors ${}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,k}^{(2,3)}, k = 0, \dots, N$, are stored in the first $N+1$ columns of a $(2N+1) \times (4N+1)$ matrix. Compute ${}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,k}^{(2,3)}$ for the negative orders by

$$ {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,-k}^{(2,3)}(t_j) = \left({}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,k}^{(2,3)}(t_j) \right)^*, \quad k = 1, \dots, N; j = 1, \dots, 2N+1. $$

Then, store these vectors in descending order in the columns of this matrix starting from the rightmost with order -1. The remaining columns are filled up with zeros:

$$ \left({}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,0}^{(2,3)} \quad \dots \; {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,N}^{(2,3)} \quad 0 \quad \dots \quad 0 \quad {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,-N}^{(2,3)} \quad \dots \quad {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u_{*,-1}^{(2,3)} \right). $$

(ii) Apply rowwise an inverse FFT and obtain immediately the grid values ${}^{\varepsilon^\phi\otimes\varepsilon^\phi}u^{(2,3)}(t_j, \phi_i), j = 1, \dots, 2N+1; i = 1, \dots, 4N+1$.

2. Repeat the last step analogously for

$$ {}^{\varepsilon^\phi\otimes\varepsilon^t}d_{n,k}^{(2,3)}, \; {}^{\varepsilon^t\otimes\varepsilon^\phi}d_{n,k}^{(2,3)}, \; {}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}d_{n,k}^{(2,3)}, \; {}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}d_{n,k}^{(2,3)}, \; {}^{\varepsilon^t\otimes\varepsilon^t}d_{n,k}^{(2,3)} $$

to obtain the quantities ${}^{\varepsilon^\phi\otimes\varepsilon^t}u^{(2,3)}$, ${}^{\varepsilon^t\otimes\varepsilon^\phi}u^{(2,3)}$, ${}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}u^{(2,3)}$, ${}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}u^{(2,3)}$, and ${}^{\varepsilon^t\otimes\varepsilon^t}u^{(2,3)}$ evaluated on the grid $\{t_j, \phi_i\}, j = 1, \dots, 2N+1; i = 1, \dots, 4N+1$.

3. [illegible] Compute the desired tensor on the grid by

$$ \begin{aligned} \mathbf{u}^{(2,3)}(t_j, \phi_i) = {} & [\, {}^{\varepsilon^\phi\otimes\varepsilon^\phi}u^{(2,3)}\varepsilon^\phi \otimes \varepsilon^\phi](t_j, \phi_i) + [\, {}^{\varepsilon^\phi\otimes\varepsilon^t}u^{(2,3)}\varepsilon^\phi \otimes \varepsilon^t](t_j, \phi_i) \\ & + [\, {}^{\varepsilon^t\otimes\varepsilon^\phi}u^{(2,3)}\varepsilon^t \otimes \varepsilon^\phi](t_j, \phi_i) + [\, {}^{\varepsilon^t\otimes\frac{\partial}{\partial t}\varepsilon^t}u^{(2,3)}\varepsilon^t \otimes \varepsilon^{\frac{\partial}{\partial t}}] \\ & + [\, {}^{\varepsilon^\phi\otimes\frac{\partial}{\partial\phi}\varepsilon^\phi}u^{(2,3)}\varepsilon^\phi \otimes {}^{\frac{\partial}{\partial\phi}}\varepsilon^\phi](t_j, \phi_i) + [\, {}^{\varepsilon^t\otimes\varepsilon^t}u^{(2,3)}\varepsilon^t \otimes \varepsilon^t](t_j, \phi_i) \end{aligned} $$

for $j = 1, \dots, 2N+1; i = 1, \dots, 4N+1$.

4. [illegible]

(i) Compute the quantities ${}^{\varepsilon^t}u_{*,k}^{(3)}(t_j) = \sum_{n=|k|}^N u_{n,k}\, {}^{\varepsilon^t}d_{n,k}^{(3)}(t_j)$ for $k = 0, \dots, N$; $j = 1, \dots, 2N+1$. This can be achieved by applying an inverse GLT. Assume that the $N+1$ (rectangular) Gauss–Legendre matrices have been tabulated beforehand. Analogously to (5.15) we perform matrix-vector products between

$$ {}^{\varepsilon^t}D_k^{(3)} = \begin{pmatrix} {}^{\varepsilon^t}d_{k,k}^{(3)}(t_1) & {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_1) & \dots & {}^{\varepsilon^t}d_{N,k}^{(3)}(t_1) \\ {}^{\varepsilon^t}d_{k,k}^{(3)}(t_2) & {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_2) & \dots & {}^{\varepsilon^t}d_{N,k}^{(3)}(t_2) \\ \vdots & \vdots & & \vdots \\ {}^{\varepsilon^t}d_{k,k}^{(3)}(t_{2N+1}) & {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_{2N+1}) & \dots & {}^{\varepsilon^t}d_{N,k}^{(3)}(t_{2N+1}) \end{pmatrix} $$

and $u_{*,k}$ from (5.14) to obtain the vectors

$$ {}^{\varepsilon^t}D_k^{(3)} u_{*,k} = {}^{\varepsilon^t}u_{*,k}^{(3)} \tag{5.16} $$

for $k = 0, \ldots, N$. Assume that the column vectors ${}^{\varepsilon^t}u_{*,k}^{(3)}, k = 0, \ldots, N$ are stored in the first $N+1$ columns of a $(2N+1) \times (4N+1)$ matrix. Compute ${}^{\varepsilon^t}u_{*,k}^{(3)}$ for the negative orders by

$$ {}^{\varepsilon^t}u_{*,-k}^{(3)}(t_j) = \Big({}^{\varepsilon^t}u_{*,k}^{(3)}(t_j) \Big)^*, \quad k = 1, \ldots, N; j = 1, \ldots, 2N+1. $$

Then, store these vectors in descending order in the columns of this matrix starting from the rightmost with order -1. The remaining columns are filled up with zeros:

$$ \Big({}^{\varepsilon^t}u_{*,0}^{(3)} \quad \ldots \quad {}^{\varepsilon^t}u_{*,N}^{(3)} \quad 0 \quad \ldots \quad 0 \quad {}^{\varepsilon^t}u_{*,-N}^{(3)} \quad \ldots \quad {}^{\varepsilon^t}u_{*,-1}^{(3)} \Big). $$

(ii) Apply rowwise an inverse FFT and obtain immediately the grid values ${}^{\varepsilon^t}u^{(3)}(t_j, \phi_i), j = 1, \ldots, 2N+1; i = 1, \ldots, 4N+1$.

5. Repeat the last step analogously for ${}^{\varepsilon^\phi}d_{n,k}^{(3)}(t)$ to obtain the quantities ${}^{\varepsilon^\phi}u^{(3)}(t_j, \phi_i)$ evaluated on the grid $\{(t_j, \phi_i)\}, j = 1, \ldots, 2N+1; i = 1, \ldots, 4N+1$.

6. [illegible] Compute the desired vector field on the grid by

$$ u(t_j, \phi_i) = [\,{}^{\varepsilon^t}u^{(3)} \varepsilon^t](t_j, \phi_i) + [\,{}^{\varepsilon^\phi}u^{(3)} \varepsilon^\phi](t_j, \phi_i) $$

for $j = 1, \ldots, 2N+1; i = 1, \ldots, 4N+1$.

7. [illegible] $(\nabla \otimes u)^T u$ [illegible] Compute on the grid the vector field v:

$$ v(t_j, \phi_i) = \big[(\mathbf{u}^{(2,3)})^T u\big](t_j, \phi_i), j = 1, \ldots, 2N+1; i = 1, \ldots, 4N+1. $$

8. [illegible] Form the inner product with $\varepsilon^t(t_j, \phi_i), j = 1, \ldots, N+1; i = 1, \ldots, 2N+1$, which projects the vector field on its ${}^{\varepsilon^t}v$ content. Analogously, we obtain the projection on its ${}^{\varepsilon^\phi}v$ content. It should be remarked that v is a vector field of mixed type, namely, of types 2 and 3. By using implicitly a polynomial exact integration against $y_{n,k}^{(3)}$ we are able to separate the desired type 3 part.

9. [illegible] To transform into the Fourier domain, we apply rowwise a forward FFT to determine the quantities ${}^{\varepsilon^t}v_{*,k}(t_j), j = 1, \ldots, 2N+1; k = 0, \ldots, 2N$. This corresponds to the first $2N+1$ column vectors ${}^{\varepsilon^t}v_{*,k}$.

We conclude the algorithm by applying the forward GLT by computing the $2N+1$ matrix-vector products between

$$ {}^{\varepsilon^t}\widetilde{D}_k^{(3)} = \frac{2\pi}{4N+1} \begin{pmatrix} w_1\, {}^{\varepsilon^t}d_{k,k}^{(3)}(t_1) & w_2\, {}^{\varepsilon^t}d_{k,k}^{(3)}(t_2) & \ldots & w_n\, {}^{\varepsilon^t}d_{k,k}^{(3)}(t_{2N+1}) \\ w_1\, {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_1) & w_2\, {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_2) & \ldots & w_n\, {}^{\varepsilon^t}d_{k+1,k}^{(3)}(t_{2N+1}) \\ \vdots & \vdots & & \vdots \\ w_1\, {}^{\varepsilon^t}d_{2N,k}^{(3)}(t_1) & w_2\, {}^{\varepsilon^t}d_{2N,k}^{(3)}(t_2) & \ldots & w_n\, {}^{\varepsilon^t}d_{2N,k}^{(3)}(t_{2N+1}) \end{pmatrix} $$

and ${}^{\varepsilon^t}v_{*,k}$ to obtain the the Fourier coefficients corresponding to ε^t,

$$ {}^{\varepsilon^t}\widetilde{D}_k^{(3)} \begin{pmatrix} {}^{\varepsilon^t}v_{*,k}(t_1) \\ {}^{\varepsilon^t}v_{*,k}(t_2) \\ \vdots \\ {}^{\varepsilon^t}v_{*,k}(t_{2N+1}) \end{pmatrix} = \begin{pmatrix} {}^{\varepsilon^t}v_{k,k} \\ {}^{\varepsilon^t}v_{k+1,k} \\ \vdots \\ {}^{\varepsilon^t}v_{2N,k} \end{pmatrix} \tag{5.17} $$

for $k = 0, \ldots, 2N$. Analogously, we find ${}^{\varepsilon^\phi}v_{n,k}, n = 0, \ldots, 2N; k = 0, \ldots, n$, such that we finally get the type 3 Fourier coefficients of v by

$$v^{(3)}_{n,k} = {}^{\varepsilon^t}v_{n,k} + {}^{\varepsilon^\phi}v_{n,k}, \qquad n = 0, \ldots, 2N; k = 0, \ldots, n,$$

and with

$$v^{(3)}_{n,-k} = (-1)^k \left(v^{(3)}_{n,k}\right)^*, \qquad k = 1, \ldots, n,$$

the negative orders, since v is real.

Algorithmic effort. We have to perform six GLTs to reconstruct the tensor on the grid and two GLTs for the velocity field u. Beyond that, we have for the transform back into the Fourier domain two additional GLTs, which yields finally for the storage of the a priori calculated GLT matrices

$$(6+2)\sum_{k=0}^{N}(2N+1)(N-k+1) + 2\sum_{k=0}^{2N}(2N+1)(2N-k+1) \approx 16N^3. \tag{5.18}$$

The algorithmic effort scales analogously with $O(N^3)$. Namely, the $N+1$, respectively, $2N+1$, matrix-vector products in (5.15), (5.16), and (5.17) cost

$$(6+2)\sum_{k=0}^{N}(2N+1)(N-k+1) + 2\sum_{k=0}^{2N}(2N+1)(2N-k+1) \approx 16N^3$$

real $\times$ complex multiplications, each of which is equivalent to two real multiplications. Additionally, we have twice as many complex–complex additions, which are equivalent to two real additions. Thus, the GLT leads us to about $32N^3$ real multiplications, and about $64N^3$ real additions. In between, the complex FFTs (not necessarily complex: one could also use complex-to-real and real-to-complex FFTs) each require about $8N \log_2 N$ real multiplications, and $12N \log_2 N$ real additions. Thus all FFTs require $(6+2+2)(2N+1)8N \log_2 N \approx 160N^2 \log_2 N$ multiplications and $(6+2+2)(2N+1)12N \log_2 N \approx 240N^2 \log_2 N$ additions. Keeping in mind that we have $\approx 8N^2$ grid points, the vectorial and tensorial expansion, the remaining matrix-vector product in the space domain, and then the scalar reduction (see step 8 in Algorithm 5.5) requires approximately

$$8N^2(6 \cdot 9 + 2 \cdot 3 + 9 + 2 \cdot 3) \approx 600N^2$$

complex $\times$ complex multiplications, which is equivalent to $2400N^2$ real multiplications and $1200N^2$ real additions. Obviously, the algorithm profits from the semilinear scaling of the FFTs. Thus, the whole procedure is dominated by the GLT and scales with $O(N^3)$, while the break-even point is approximately at $N = 12$. Since we use in our simulations (also with respect to applications in meteorology) much larger basis sets, we obtain, for example, at $N = 100$ a speedup factor of 500!

6. Numerical results. In this section we report numerical results obtained by using the nonlinear Galerkin method. To demonstrate the efficiency of our method, we study an example that is motivated by [7] for a toroidal domain, i.e., the two-dimensional periodic case. However, we study here an extension of [7] to the rotating sphere by involving a Coriolis term and, moreover, time-dependent flow driving forces. Finally, we compare the nonlinear Galerkin method with the linear one, which can be obtained by setting $z_m = 0$.

Description of the computations. The initial condition $u_0 = u_0/\|u_0\|_{l^\Omega}$ with $u_0 = \sum_{n=1}^{20} \sum_{k=-n}^{n} u_{n,k}^{(3)} y_{n,k}^{(3)}$ is computed from a given spectrum given by

$$u_{n,k}^{(3)} = \frac{2}{n + (\sqrt{\nu} n)^5} e^{i\theta_k} \qquad \text{for} \quad n = 1, \dots, 20; k = 0, \dots, n,$$

where $\theta_0 = 0$, and $\theta_k, k = 1, \dots, n$, is generated by a random function. The negative orders are obtained by $u_{n,-k} = (-1)^k u_{n,k}^*$. The viscosity ν is set to $\nu = 10^{-4}$. This example is chosen in analogy to [7]. Obviously, the initial spectrum decays like n^{-1}. As Figure 6.1 shows, the flow has no organized structures and visible vortices. Moreover, the flow is driven by some time-dependent (decaying) external force f, given by

$$f(t, \eta) = \gamma(t) y_{3,0}^{(3)}(\eta), \qquad \eta \in \Omega,$$

with

$$\gamma(t) = \begin{cases} 1 & \text{for } t \in [0, 10] \\ \cos(\pi/5t) e^{-(t-10)/5} & \text{for } t \in (10, 60]. \end{cases}$$

In order to describe all the scales of motion, the spherical harmonics degree k_N must be chosen so that an associated (spherical grid) size $2\pi/N$ is smaller than the dissipative (Kraichnan) scale l_η; in terms of the spherical harmonic degree it means $k_N > k_\eta$. We recall that under the dissipative scales the motion is damped by viscosity. In fact, the total number of degrees of freedom needed to describe the motion, from dissipative scales to large scales containing eddies, can be estimated by the ratio $(k_\eta/k_N)^2$ (see, for example, [7]).

This quantity is related to the dimension of the attractor of the Navier–Stokes equations

$$\left(\frac{k_\eta}{k_N}\right)^2 \approx \mathrm{Re},$$

where Re is the integral scale Reynolds number. The Reynolds number can be defined by

$$\mathrm{Re} = \frac{LU}{\nu},$$

where U denotes the [illegible] velocity of the flow, and $L = 1/k_N$ the integral length scale defined by

$$L = \frac{U}{\sqrt{2}\eta^{1/3}}$$

with $\eta = \nu \|\Delta^* u\|_{l^2(\Omega)}^2 / |\Omega|$. Then $L \approx 0.7$ and $U \approx 0.3$, so that $\mathrm{Re} \approx 2100$. The dissipative wave number can be estimated by $k_\eta^2 = \mathrm{Re} \cdot k_N^2$ such that $k_\eta \approx 50$. Finally we take the angular velocity $|\omega| = 2.0$ such that the Rossby number relating the advection and Coriolis term reads as follows:

$$R_o = \frac{U}{L|\omega|} \approx 0.2.$$

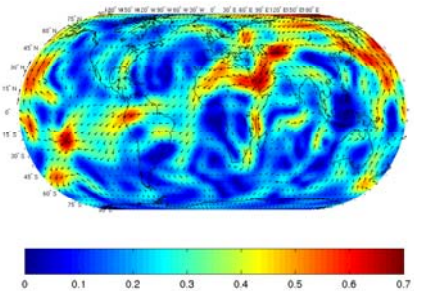


FIG. 6.1. *Initial flow velocity at* $t = 0$.

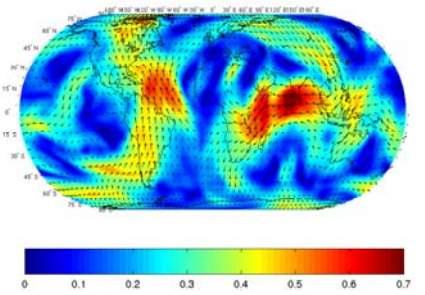


FIG. 6.2. *Flow at* $t = 5$.

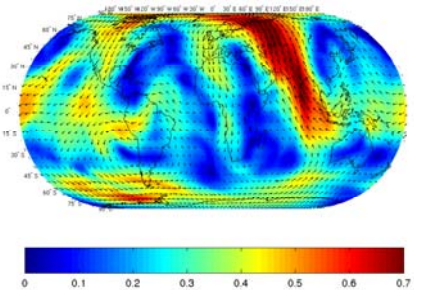


FIG. 6.3. *Flow velocity at* $t = 10$.

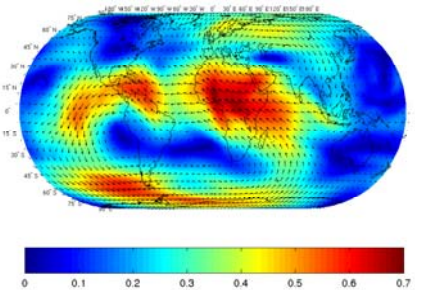


FIG. 6.4. *Flow velocity at* $t = 20$.

Obviously, this Rossby number is not much smaller than 1, which indicates that the Coriolis term is in the same order as the advection term. In order to be sure to resolve all the scales of the considered flow, we set the truncation level to a spherical harmonic degree of 100. According to (4.1) and (4.2) this truncation level corresponds to $m = 50$. In detail, the low-frequency approximation u_{50} is represented by spherical harmonics up to degree 50, whereas the high-frequency approximation z_{50} covers the terms up to degree 100. The results for the flow velocity of our simulation is plotted in Figures 6.1–6.8.

Now, it appears that the dissipation wave number k_η is of order 42 and varies slightly between 40 and 45 during the computation. Hence, the previous estimate based on the dimension of the attractor matches the computational results. We also remark that if, at $t = 0$, the small scales corresponding to a spherical harmonic degree larger than 20 are set to zero, then a dissipation range appears very quickly, as we can see in Figure 6.9 at $t = 5$. The enstrophy transfer from the large scales to the small ones acts on the smallest scales after a few iterations. Figure 6.10, illustrating $\|z_{50}\|_{l^2(\Omega)}$, indicates that this transition period is very short. After this period, the smaller scales are damped by viscosity until an equilibrium between viscous and nonlinear terms appears. Thenceforward, we can see that $\|z_{50}\|_{l^2(\Omega)}$ develops completely independently of $\|u_{50}\|_{l^2(\Omega)}$; see Figure 6.10. The $l^2(\Omega)$-norm of the Galerkin approximation u_{50} depends also on the forcing term f which varies over time. This dependency is illustrated in Figure 6.10, which shows also $\gamma(t)$. It is interesting to note that at $t \approx 30$ the viscous term dominates the external force f, and the energy decays.

As Figures 6.1–6.8 show the small random structures of the flow at the initial time disappear quickly; fusions of these very small structures lead to larger ones. So,

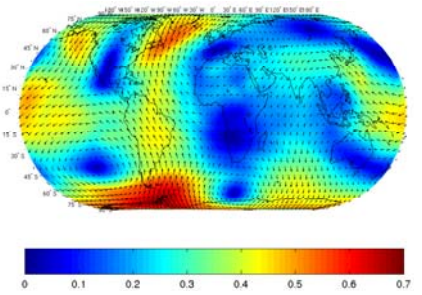


FIG. 6.5. *Flow velocity at* $t = 30$.

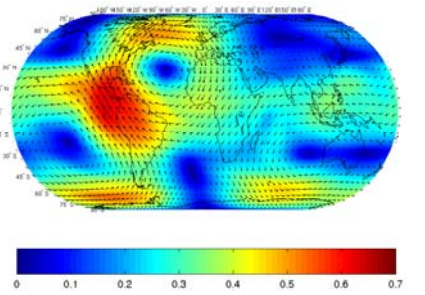


FIG. 6.6. *Flow velocity at* $t = 40$.

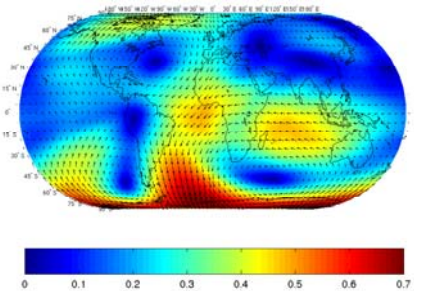


FIG. 6.7. *Flow velocity at* $t = 50$.

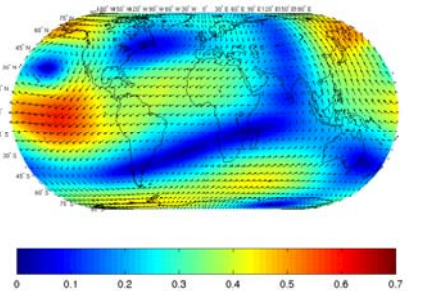


FIG. 6.8. *Flow velocity at* $t = 60$.

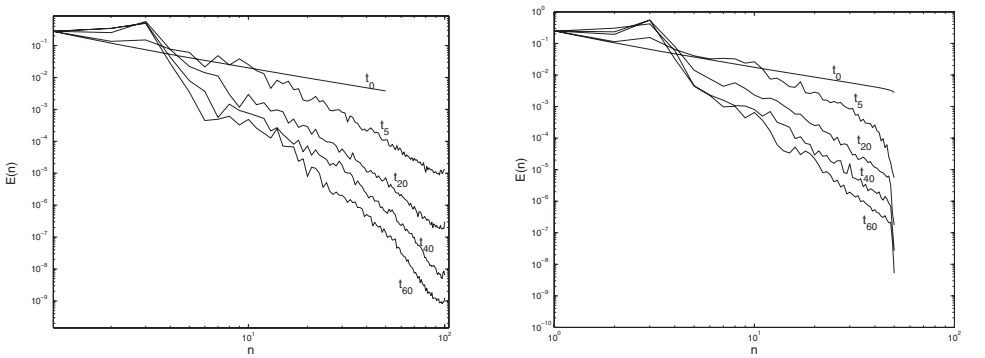


FIG. 6.9. *Comparison of the energy spectrum* $E(n)$ *for the linear (left) and for the nonlinear (right) Galerkin method with truncation level* $N = 100$ *at different time-steps.*

after a transient period, the flow is mainly constituted by large structures.

This can also be studied by looking at the energy spectrum; see Figure 6.9. The energy spectrum is given by

$$E(n) = \sum_{k=-n}^{n} |u_{n,k}^{(3)}|^2,$$

showing the energy contained in each degree averaged over all orders. It is interesting to note that after a short transient time an initial range is appearing. And we note that more and more energy is transferred into the low-frequency parts of the flow

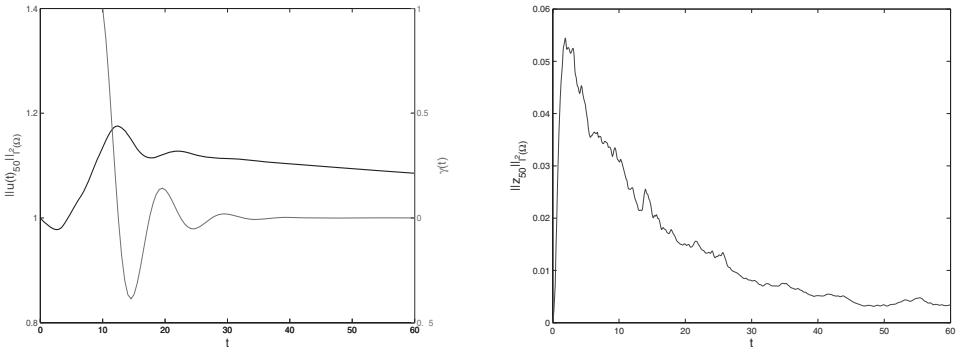


FIG. 6.10. *Time evolution of* $\|u_{50}(t)\|_{l^2(\Omega)}$ *and* $\gamma(t)$ *(left) and of* $\|z_{50}(t)\|_{l^2(\Omega)}$ *(right).*

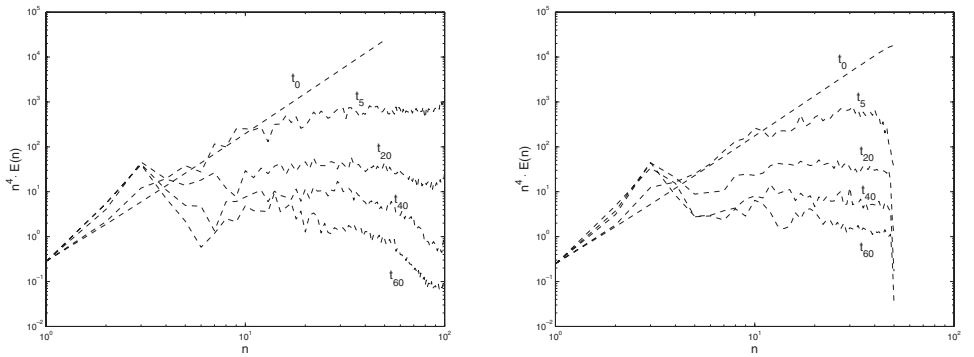


FIG. 6.11. *Comparison of the decay of the energy spectrum* $E(n)$ *of the linear Galerkin method (left) and the nonlinear Galerkin method (right) with truncation level* $N = 100$ *at different time-steps. We observe an* n^{-4}*-decay in both energy spectra after some eddy-turnover times.*

(large eddies). Moreover, let us study the energy decay of the Fourier modes from Figure 6.11. From this illustration we observe an approximate n^{-4}-decay that was also reported by Temam [7] for a flow in a two-dimensional, periodic geometry, i.e., on a torus. Evidently, the decay of energy is more rapid than the n^{-3}-decay predicted by Kraichnan's phenomenological theory of turbulence [16]. But our results are in agreement with the ones obtained by Brachet et al. [4] and Orszag [21], who show that an n^{-3}-energy spectrum can only be obtained when the Reynolds number is much larger, e.g., of the order of 2.5×10^4 (see [21]).

The time-step is chosen according to accuracy and stability considerations. To ensure the stability, the time-step h must satisfy a CFL condition like

$$hN\|u\|_{l^\infty(\Omega)} < 1. \tag{6.1}$$

Having tracked the $l^\infty(\Omega)$-norm of u on a grid, $\|u\|_{l^\infty(\Omega)}$ reached a maximum of 0.72 satisfying (6.1). Since we used a Runge–Kutta solver of fourth order for the time-integration, a time-step of 10^{-3} allows most of the spectrum to be recovered. Thus, the global accuracy is then of order 10^{-12}.

With the characteristic length scale L and velocity U, we define a characteristic

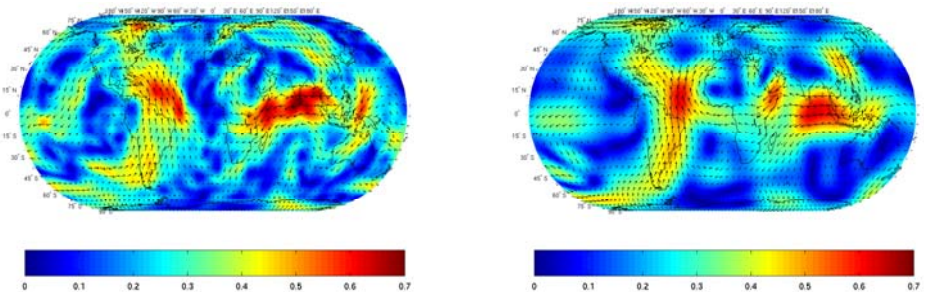


FIG. 6.12. *Comparison of the velocity structures obtained with the linear Galerkin method (left) and the nonlinear Galerkin method (right) at truncation level* $N = 30$ *and time* $t = 5$.

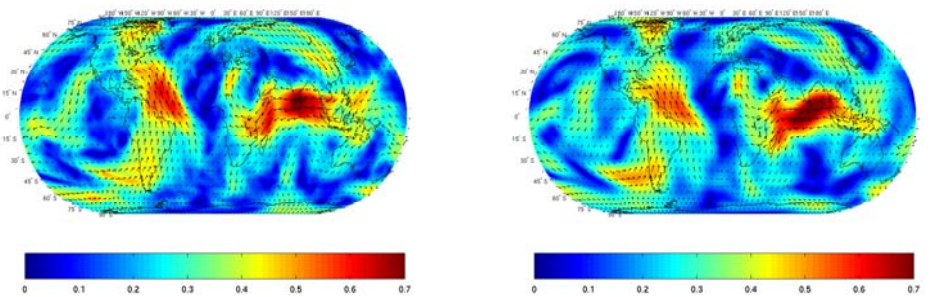


FIG. 6.13. *Comparison of the velocity structures obtained with the linear Galerkin method (left) and the nonlinear Galerkin method (right) at truncation level* $N = 50$ *and time* $t = 5$.

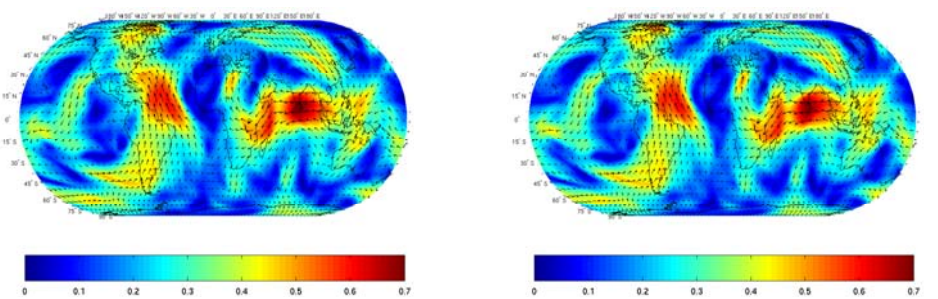


FIG. 6.14. *Comparison of the velocity structures obtained with the linear Galerkin method (left) and the nonlinear Galerkin method (right) at truncation level* $N = 100$ *and time* $t = 5$.

time unit by the ratio

$$\tau = \frac{L}{U},$$

which is called eddy-turnover time. This quantity is of order 2.33 for the flow considered. We have let the flow evolve over 6×10^4 iterations, i.e., on the time interval $[0, 60]$, which corresponds to more than 25 time units. The code has been implemented in Fortran 90 using Intel's Fortran compiler and the Math Kernel Library (MKL) to perform the FFTs and BLAS calls. The computation required more than 120 hours of CPU time on a Pentium 4 (3.2 GHz) processor, not counting all the preliminary tests needed to develop and improve the algorithm.

Mesh refinement and comparison to the linear Galerkin method. The strength of the nonlinear Galerkin method is to represent the considered flow with fewer degrees of freedoms than required by a linear Galerkin scheme. This is observed by obtaining similar results for the same problem in less time. In detail, we compare at a fixed time $t = 5$ and at different truncation levels $N = \{30, 50, 100\}$ the linear and nonlinear schemes in Figures 6.12–6.14. In the case of the linear Galerkin method the [illegible] N denotes that all scales of motion up to degree N are resolved. The mesh refinement indicates for both schemes that they are stable: In detail, we see in Figure 6.12 that the linear Galerkin method shows some high-frequency flicker that is induced by higher frequency scales of u_0 that are not resolved. This flicker becomes less for $N = 50$ and disappears when the truncation level is increased to $N = 100$; see Figures 6.13 and 6.14. Moreover, it should be outlined that the nonlinear method is about 20% faster than the linear Galerkin method (for the same resolution). Finally, a closer look at the tails of the energy spectrum in Figures 6.9 and 6.11 shows a small increase in the Fourier coefficients in the case of the linear method. This effect is known as [illegible] and is induced by aliasing effects; see Figure 6.13 and, for example, Boyd [3]. In Figure 6.14 we observe a rapid decrease in the tails of the energy spectrum. This effect is inherent to nonlinear Galerkin methods and is known as [illegible], which stabilizes the whole solution procedure (see [3, 7]).

Concluding remarks. In this work we introduced a nonlinear Galerkin method which was applied to the inhomogenous, incompressible Navier–Stokes equation on the sphere. The arising coupling terms were expressed by Wigner-3j symbols known from quantum mechanics. To increase the efficiency, we extended the idea of FFT-based pseudospectral algorithms to the rapid evaluation of the considered type 3 vector and tensor fields. This allowed us to compute in each time-step the coupling integrals of the nonlinear advection term at once, yielding finally an $O(N^3)$ method. Finally, we demonstrated the algorithm in an extensive numerical example including the Coriolis term. At last, we like to point the reader's attention to the segmentation of the energy spectrum in Figure 6.9. While the inertial range shows a power-law spectrum, the dissipative range shows an exponential decay. This result fits excellently into the phenomenological theory of turbulence by Kraichnan [16] and differs from the linear Galerkin method. In the latter case the energy spectrum does not show a distinct segmentation into an inertial and dissipative range (for more illustrations, see [7, 10]). A challenge for the future is to extend the algorithm to the full set of equations noted in (2.6)–(2.9) in the compressible case.

REFERENCES

[1] F. Baer and G. Platzman, *A procedure for numerical integration of the spectral vorticity equation,* J. Met., 18 (1961), p. 393.

[2] W. Bourke, *An efficient, one-level, primitive-equation spectral model,* Mon. Weather Rev., 100 (1972), pp. 683–689.

[3] J. P. Boyd, *Chebyshev and Fourier Spectral Methods,* 2nd ed. (revised), Dover, New York, 2001.

[4] M. E. Brachet, M. Meneguzzi, H. Politano, and P. L. Sulem, *The dynamics of freely decaying two-dimensional turbulence,* J. Fluid Mech., 194 (1988), pp. 333–349.

[5] D. M. Brink and G. R. Satchler, *Angular Momentum,* Oxford Library of the Physical Sciences, Oxford, UK, 1968.

[6] G. C. Corey and J. W. Tromp, *Fast pseudospectral algorithm in curvilinear coordinates,* in Numerical Grid Methods and Their Application to Schrödinger's Equation, C. Cerjan, ed., Kluwer Academic, Dordrecht, The Netherlands, 1993, pp. 1–23.

[7] A. Debussche, T. Dubois, and R. Temam, *The nonlinear Galerkin method: A multiscale method applied to the simulation of homogeneous turbulent flows,* Theoret. Comput. Fluid Dyn., 7 (1995), pp. 279–315.

[8] A. R. Edmonds, *Drehimpulse in der Quantenmechanik,* Bibliographisches Institut, Mannheim, Germany, 1964.

[9] H. W. Elsaesser, *Evaluation of spectral versus grid methods of hemispheric numerical weather prediction,* J. Appl. Meteorol., 5 (1966), p. 246.

[10] M. J. Fengler, *Vector Spherical Harmonic and Vector Wavelet Based Non-Linear Galerkin Schemes for Solving the Incompressible Navier–Stokes Equation on the Sphere,* Ph.D. thesis, University of Kaiserslautern, Department of Mathematics, Geomathematics Group, Kaiserslautern, Germany, 2005.

[11] W. Freeden, T. Gervens, and M. Schreiner, *Constructive Approximation on the Sphere (with Applications to Geomathematics),* Oxford Science Publications, Clarendon, Oxford, UK, 1998.

[12] B. J. Hoskins and A. J. Simmons, *A multi-layer spectral model and the semi-implicit method,* Quart. J. Roy. Meteor. Soc., 101 (1975), pp. 637–655.

[13] A. A. Il'in, *The Navier–Stokes and Euler equations on two-dimensional closed manifolds,* Math. USSR Sb., 69 (1991), pp. 559–579.

[14] A. A. Il'in and A. N. Filatov, *On unique solvability of the Navier–Stokes equations on the two-dimensional sphere,* Soviet Math. Dokl., 38 (1989), pp. 9–13.

[15] R. W. James, *New tensor spherical harmonics, for application to the partial differential equations of mathematical physics,* Philos. Trans. Roy. Soc. London Ser. A, 281 (1976), pp. 195–221.

[16] R. H. Kraichnan, *Inertial ranges in two-dimensional turbulence,* Phys. Fluids, 10 (1967), pp. 1417–1423.

[17] M. Marion and R. Temam, *Nonlinear Galerkin methods,* SIAM J. Numer. Anal., 26 (1989), pp. 1139–1157.

[18] J. Norbury and I. Roulstone, *Large-Scale Atmosphere-Ocean Dynamics* I. *Analytic Methods and Numerical Models,* Cambridge University Press, Cambridge, UK, 2002.

[19] J. Norbury and I. Roulstone, *Large-Scale Atmosphere-Ocean Dynamics* II. *Geometric Methods and Models,* Cambridge University Press, Cambridge, UK, 2002.

[20] S. A. Orszag, *Transform method for the calculation of vector-coupled sums: Application to the spectral form of the vorticity equation,* J. Atmos. Sci., 27 (1970), pp. 890–895.

[21] S. A. Orszag, *Turbulence and transition: A progress report,* in Proceedings of 5th International Conference on Numerical Methods in Fluids Dynamics, Lecture Notes in Phys. 59, Springer-Verlag, Berlin, 1977, pp. 32–51.

[22] A. Persson, *User Guide to ECMWF Forecast Products,* Technical report, European Center for Medium-Range Weather Forecast, Reading, UK, 2000.

[23] J. Shen and R. Temam, *Nonlinear Galerkin method using Legendre polynomials,* in Proceedings of the International Conference on Nonlinear Evolution Partial Differential Equations, Beijing, China, 1993, AMS/IP Stud. Adv. Math. 3, AMS, Providence, RI, 1997, pp. 363–376.

[24] R. Temam, *Navier–Stokes Equations. Theory and Numerical Analysis,* North–Holland, Amsterdam, 1979.

[25] C. Temperton, *On scalar and vector transform methods for global spectral models,* Mon. Weather Rev., 119 (1991), pp. 1303–1307.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 995–1013
© 2005 Society for Industrial and Applied Mathematics

A COMPARISON OF TROUBLED-CELL INDICATORS FOR RUNGE–KUTTA DISCONTINUOUS GALERKIN METHODS USING WEIGHTED ESSENTIALLY NONOSCILLATORY LIMITERS*

JIANXIAN QIU† AND CHI-WANG SHU‡

Abstract. In [*SIAM J. Sci. Comput.*, 26 (2005), pp. 907–929], we initiated the study of using WENO (weighted essentially nonoscillatory) methodology as limiters for the RKDG (Runge–Kutta discontinuous Galerkin) methods. The idea is to first identify "troubled cells," namely, those cells where limiting might be needed, then to abandon all moments in those cells except the cell averages and reconstruct those moments from the information of neighboring cells using a WENO methodology. This technique works quite well in our one- and two-dimensional test problems [*SIAM J. Sci. Comput.*, 26 (2005), pp. 907–929] and in the follow-up work where more compact Hermite WENO methodology is used in the troubled cells. In these works we used the classical minmod-type TVB (total variation bounded) limiters to identify the troubled cells; that is, whenever the minmod limiter attempts to change the slope, the cell is declared to be a troubled cell. This troubled-cell indicator has a TVB parameter M to tune and may identify more cells than necessary as troubled cells when M is not chosen adequately, making the method costlier than necessary. In this paper we systematically investigate and compare a few different limiter strategies as troubled-cell indicators with an objective of obtaining the most efficient and reliable troubled-cell indicators to save computational cost.

Key words. Runge–Kutta discontinuous Galerkin method, limiters, weighted essentially nonoscillatory finite volume scheme, high-order accuracy

AMS subject classifications. 65M60, 65M99, 35L65

DOI. 10.1137/04061372X

1. Introduction.

In [13], we initiated the study of using weighted essentially nonoscillatory (WENO) methodology as limiters for the Runge–Kutta discontinuous Galerkin (RKDG) methods. The idea is to first identify "troubled cells," namely, those cells where limiting might be needed, then to abandon all moments in those cells except the cell averages and reconstruct those moments from the information of neighboring cells using a WENO methodology. This technique works quite well in our one- and two-dimensional test problems [13] and in the follow-up work [14] and [15], where more compact Hermite WENO methodology was used in the troubled cells. In these earlier works we adopted the classical minmod-type total variation bounded (TVB) limiters as in [5, 3, 7] to identify the troubled cells; that is, whenever the minmod limiter attempts to change the slope, the cell is declared to be a troubled cell. This troubled-cell indicator has a TVB parameter M to tune. When M is taken too small, more cells than necessary may be identified as troubled cells. Even though

*Received by the editors August 20, 2004; accepted for publication (in revised form) April 19, 2005; published electronically December 8, 2005.
http://www.siam.org/journals/sisc/27-3/61372.html

†Department of Mathematics, Nanjing University, Nanjing, Jiangsu 210093, People's Republic of China, and Department of Mechanical Engineering, National University of Singapore, Singapore 119260 (jxqiu@nju.edu.cn). This author's research was partially supported by NNSFC grant 10371118 and by NUS research project R-265-000-118-112.

‡Division of Applied Mathematics, Brown University, Providence, RI 02912 (shu@dam.brown.edu). This author's research was partially supported by the Chinese Academy of Sciences while the author was in residence at the University of Science and Technology of China (grant 2004-1-8) and at the Institute of Computational Mathematics and Scientific/Engineering Computing. Additional support was provided by ARO grant W911NF-04-1-0291, NSF grant DMS-0207451, and AFOSR grant F49620-02-1-0113.

the WENO reconstruction procedure used in these cells will not degrade the order of accuracy of the original RKDG scheme, the extra work done for these WENO reconstructions in such wrongly identified troubled cells increases the cost of the algorithm. On the other hand, when M is taken too large, some of the real troubled cells may not be identified, resulting in oscillations or even instability for the method. In this paper we systematically investigate and compare a few different limiter strategies as troubled-cell indicators with an objective of trying to obtain the most efficient and reliable troubled-cell indicators to save the computational cost.

The first discontinuous Galerkin (DG) method was introduced in 1973 by Reed and Hill [16] in the framework of neutron transport (steady-state linear hyperbolic equations). A major development of the DG method was carried out by Cockburn et al. in a series of papers [6, 5, 4, 3, 7], in which they established a framework to easily solve , , , time-dependent hyperbolic conservation laws,

$$\begin{cases} u_t + \nabla \cdot f(u) = 0, \\ u(x,0) = u_0(x), \end{cases} \tag{1.1}$$

using explicit, nonlinearly stable high-order Runge–Kutta time discretizations [20] and DG discretization in space with exact or approximate Riemann solvers as interface fluxes and TVB limiter [18] to achieve nonoscillatory properties for strong shocks. These schemes are termed RKDG methods.

An important component of RKDG methods for solving conservation laws (1.1) with strong shocks in the solutions is a nonlinear limiter, which is applied to detect discontinuities and control spurious oscillations near such discontinuities. Many such limiters have been used in the literature on RKDG methods. For example, we mention the minmod-type TVB limiter [6, 5, 4, 3, 7], which is a slope limiter using a technique borrowed from the finite volume methodology; the moment-based limiter [1] and an improved moment limiter [2], which are specifically designed for DG methods and work on the moments of the numerical solution. These limiters tend to degrade accuracy when mistakenly used in smooth regions of the solution. More recently, a strategy was designed in [12], based on a strong superconvergence at the outflow boundary of each element in smooth regions for the discontinuous Galerkin method, to detect discontinuities and to lower the order of accuracy in the approximation there to avoid spurious oscillations near such discontinuities when solving hyperbolic systems of conservation laws. There are also many limiters developed in the finite volume and finite difference literature, such as the various flux limiters [23], the monotonicity-preserving (MP) limiters [22], modifications of MP limiters [17], and a discontinuity detecting method [9] used to sharpen contact discontinuities in ENO schemes.

In [13], we initiated a study of using WENO methodology as limiters for RKDG methods. We have adopted the following framework.

, , 1. First we identify the "troubled cells," namely, those cells which might need the limiting procedure.

, , 2. Then we replace the solution polynomials in those troubled cells by reconstructed polynomials using WENO methodology which maintain the original cell averages (conservation), have the same orders of accuracy as before, but are less oscillatory.

This technique works quite well in our one- and two-dimensional test problems in [13] and in the follow-up work [14] and [15], where a more compact Hermite WENO methodology was used in the troubled cells. The emphasis of all these works is on Step 2, where different WENO reconstruction strategies are considered. Step 1 is

dealt with by the classical minmod-type TVB limiters as in [5, 3, 7] to identify the troubled cells. It is realized that the TVB parameter M must be chosen carefully in order to reach a nonoscillatory result with the minimal number of troubled cells. However, Step 1 is not investigated in detail in [13, 14, 15].

We do want to point out a fact, which was also highlighted in [13, 14, 15], that since the WENO reconstruction procedure in Step 2 maintains the high-order accuracy of the original RKDG scheme, it is less crucial for Step 1 to identify exactly the troubled cells. If more troubled cells are identified than necessary, the order of accuracy will not degrade, although the computational cost will increase.

In this paper we concentrate on Step 1 in the procedure above, and investigate systematically a few discontinuity detecting methods as troubled-cell indicators. We also test a new discontinuity detecting method based on the subcell resolution idea of Harten [9]. For Step 2 in the procedure above, we simply use the regular WENO reconstruction procedure developed in [13]. We review and describe the details of a few discontinuity detecting methods in section 2, and present extensive numerical results in section 3 to compare their performance as troubled-cell indicators. Concluding remarks are given in section 4.

2. Review and implementation of discontinuity detecting methods. In this section we review a few discontinuity detecting methods to identify troubled cells. We also develop a new discontinuity detecting method based on the subcell resolution method of Harten [9].

We start with the description in the one-dimensional case and use the notation in [5]; however, we emphasize that the procedure described below does not depend on the specific basis chosen for the polynomials and works also in multidimensions. We would like to solve the one-dimensional scalar conservation law:

$$\begin{cases} u_t + f(u)_x = 0, \\ u(x,0) = u_0(x). \end{cases} \tag{2.1}$$

The computational domain is divided into N cells with boundary points $0 = x_{\frac{1}{2}} < x_{\frac{3}{2}} < \cdots < x_{N+\frac{1}{2}} = L$. The points x_i are the centers of the cells $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$, and we denote the cell sizes by $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ and the maximum cell size by $h = \max_i \Delta x_i$. The solution as well as the test function space are given by $V_h^k = \{p : p|_{I_i} \in P^k(I_i)\}$, where $P^k(I_i)$ is the space of polynomials of degree $\le k$ on the cell I_i. We adopt a local orthogonal basis over I_i $\{v_l^{(i)}(x),\ l = 0, 1, \ldots, k\}$, namely, the scaled Legendre polynomials:

$$v_0^{(i)}(x) = 1, \qquad v_1^{(i)}(x) = \frac{x - x_i}{\Delta x_i/2}, \qquad v_2^{(i)}(x) = \left(\frac{x - x_i}{\Delta x_i/2}\right)^2 - \frac{1}{3}, \ldots .$$

Then the numerical solution $u^h(x,t)$ in the space V_h^k can be written as

$$u^h(x,t) = \sum_{l=0}^{k} u_i^{(l)}(t) v_l^{(i)}(x) \qquad \text{for } x \in I_i, \tag{2.2}$$

and the degrees of freedom $u_i^{(l)}(t)$ are the moments defined by

$$u_i^{(l)}(t) = \frac{1}{a_l} \int_{I_i} u^h(x,t) v_l^{(i)}(x)\, dx, \qquad l = 0, 1, \ldots, k,$$

where $a_l = \int_{I_i} (v_l^{(i)}(x))^2 dx$ are the normalization constants since the basis is not orthonormal. In order to determine the approximate solution, we evolve the degrees of freedom $u_i^{(l)}$:

$$\frac{d}{dt}u_i^{(l)} + \frac{1}{a_l}\left(- \int_{I_i} f(u^h(x,t)) \frac{d}{dx} v_l^{(i)}(x)\, dx + \hat{f}(u^-_{i+1/2}, u^+_{i+1/2}) v_l^{(i)}(x_{i+1/2}) \right. \tag{2.3}$$
$$\left. - \hat{f}(u^-_{i-1/2}, u^+_{i-1/2}) v_l^{(i)}(x_{i-1/2}) \right) = 0, \qquad l = 0, 1, \ldots, k,$$

where $u^{\pm}_{i+1/2} = u^h(x^{\pm}_{i+1/2}, t)$ are the left and right limits of the discontinuous solution u^h at the cell interface $x_{i+1/2}$, and $\hat{f}(u^-, u^+)$ is a monotone flux (nondecreasing in the first argument and nonincreasing in the second argument) for the scalar case and an exact or approximate Riemann solver for the system case. The semidiscrete scheme (2.3) is discretized in time by a nonlinearly stable Runge–Kutta time discretization, e.g., the third-order version [20]. The integral term in (2.3) can be computed either exactly or by a suitable numerical quadrature accurate to at least $O(h^{k+l+2})$.

We will now review a few discontinuity detecting methods to identify troubled cells. Numerical examples to compare them will be given in the next section.

1. [illegible] [5]. Denote

$$u^-_{i+1/2} = u_i^{(0)} + \tilde{u}_i, \qquad u^+_{i-1/2} = u_i^{(0)} - \tilde{\tilde{u}}_i.$$

From (2.2) we can see that

$$\tilde{u}_i = \sum_{l=1}^{k} u_i^{(l)} v_l^{(i)}(x_{i+1/2}), \qquad \tilde{\tilde{u}}_i = -\sum_{l=1}^{k} u_i^{(l)} v_l^{(i)}(x_{i-1/2}).$$

These are modified either by the standard minmod limiter [8]

$$\tilde{u}_i^{(mod)} = m(\tilde{u}_i, \Delta_+ u_i^{(0)}, \Delta_- u_i^{(0)}), \qquad \tilde{\tilde{u}}_i^{(mod)} = m(\tilde{\tilde{u}}_i, \Delta_+ u_i^{(0)}, \Delta_- u_i^{(0)}),$$

where the minmod function m is given by

(2.4)
$$m(a_1, a_2, \ldots, a_n) = \begin{cases} s \cdot \min_{1 \le j \le n} |a_j| & \text{if } \operatorname{sign}(a_1) = \operatorname{sign}(a_2) = \cdots = \operatorname{sign}(a_n) = s, \\ 0 & \text{otherwise,} \end{cases}$$

or by the TVB-modified minmod function [18]

$$\tilde{m}(a_1, a_2, \ldots, a_n) = \begin{cases} a_1 & \text{if } |a_1| \le Mh^2, \\ m(a_1, a_2, \ldots, a_n) & \text{otherwise,} \end{cases} \tag{2.5}$$

where $M > 0$ is a constant. The choice of M depends on the solution of the problem. For scalar problems it is possible to estimate M by the initial condition as in [5] (M is proportional to the second derivative of the initial condition at smooth extrema); however, it is more difficult to estimate M for the system case. If M is chosen too small, more cells than necessary will be identified as troubled cells, thereby increasing the computational cost; however, if M is chosen too large, spurious oscillations may appear.

2. [illegible] [1]. We will denote this limiter as the BDF limiter in this paper. The moment-based limiter in [1] is given by

$$u_i^{(l),mod} = \frac{1}{2l-1}\, m\left((2l-1)u_i^{(l)}, u_{i+1}^{(l-1)} - u_i^{(l-1)}, u_i^{(l-1)} - u_{i-1}^{(l-1)}\right), \tag{2.6}$$

where m is again the minmod function (2.4). This limiter is applied adaptively. First, the highest-order moment $u^{(k)}$ is limited. Then the limiter is applied to successively lower-order moments when the next higher-order moment on the interval has been changed by the limiting. For our purpose, when the BDF limiter (2.6) is enacted (returns other than the first argument) for the highest-order moment, the cell is declared as a troubled cell and marked for further reconstruction.

3. [illegible] [2]. We will denote this modified moment limiter as the BSB limiter in this paper. For our purpose as a troubled-cell indicator, if both (2.6) and

$$\hat{u}_i^{(l),mod} = \frac{1}{2l-1} m\Big((2l-1)u_i^{(l)}, u_{i+1/2}^{(l-1)+} - u_i^{(l-1)}, u_i^{(l-1)} - u_{i-1/2}^{(l-1)-}\Big) \tag{2.7}$$

are enacted for the highest-order moment $u^{(k)}$, where

$$u_{i+1/2}^{(l-1)+} = u_{i+1}^{(l-1)} - (2l-1)u_{i+1}^{(l)}, \qquad u_{i-1/2}^{(l-1)-} = u_{i-1}^{(l-1)} + (2l-1)u_{i-1}^{(l)},$$

that is, if both $u_i^{(k),mod} \neq u_i^{(k)}$ and $\hat{u}_i^{(k),mod} \neq u_i^{(k)}$, then the cell is identified as a troubled cell, marked for further reconstruction.

4. [illegible] [22]. In [22], Suresh and Huynh designed a limiter to preserve accuracy near smooth extrema, which works well with Runge–Kutta time stepping for a class of high-order MP schemes. The interface values in these schemes are obtained by limiting a higher-order polynomial reconstruction. The key idea in that work is to distinguish between a smooth local extremum and a genuine $O(1)$ discontinuity.

For our purpose as a troubled-cell indicator, the MP limiter can be described as follows. First a median function is defined as

$$\text{median}(x, y, z) = x + m(y - x, z - x), \tag{2.8}$$

where m is again the minmod function (2.4). If

$$u_{i+1/2}^{-} \neq \text{median}(u_{i+1/2}^{-}, u_{i+1/2}^{\min}, u_{i+1/2}^{\max}), \tag{2.9}$$

where

$$u_{i+1/2}^{\min} = \max\Big[\min(u_i^{(0)}, u_{i+1}^{(0)}, u_{i+1/2}^{MD}), \min(u_i^{(0)}, u_{i+1/2}^{UL}, u_{i+1/2}^{LC})\Big],$$
$$u_{i+1/2}^{\max} = \min\Big[\max(u_i^{(0)}, u_{i+1}^{(0)}, u_{i+1/2}^{MD}), \max(u_i^{(0)}, u_{i+1/2}^{UL}, u_{i+1/2}^{LC})\Big],$$

and

$$d_i = u_{i+1}^{(0)} - 2u_i^{(0)} + u_{i-1}^{(0)}, \quad d_{i+1/2}^{M4X} = m(4d_i - d_{i+1}, 4d_{i+1} - d_i, d_i, d_{i+1}, d_{i-1}, d_{i+2}),$$
$$u_{i+1/2}^{MD} = \frac{1}{2}\left(u_i^{(0)} + u_{i+1}^{(0)} - d_{i+1/2}^{M4X}\right), \quad u_{i+1/2}^{UL} = u_i^{(0)} + \alpha\Big(u_i^{(0)} - u_{i-1}^{(0)}\Big),$$
$$u_{i+1/2}^{LC} = u_i^{(0)} + \frac{1}{2}\left(u_i^{(0)} - u_{i-1}^{(0)}\right) + \frac{\beta}{3} d_{i-1/2}^{M4X},$$

or if $u^+_{i-1/2}$ satisfies a similar (symmetric) condition, then the cell is identified as a troubled cell, marked for further reconstruction. We take the parameters $\alpha = 2$ and $\beta = 4$ in the numerical tests in the next section, as suggested in [22].

5. [illegible] [17]. We will denote this modified MP limiter as the MMP limiter in this paper. In [17], Rider and Margolin presented a simple modification of the standard MP limiter in [22]. These modified MP limiters relax the relatively stringent condition of preserving monotonicity, while enforcing less-restrictive conditions. For our purpose as a troubled-cell indicator, the MMP limiter can be described as follows:

$$\phi = \min(1, \Delta\overline{u}^{\min}/\Delta_{\min}u), \tag{2.10}$$

where

$$\Delta\overline{u}^{\min} = u_i^{(0)} - \min\Big(u_{i-1}^{(0)}, u_i^{(0)}, u_{i+1}^{(0)}\Big), \quad \Delta_{\min}u = u_i^{(0)} - \min\big(u^+_{i-1/2}, u^-_{i+1/2}\big).$$

When $\phi \neq 1$, the limiter enacts and the cell is identified as a troubled cell, marked for further reconstruction.

6. [illegible] [12]. We will denote the troubled-cell indicator based on this technique as the KXRCF indicator in this paper.

The strategy in [12] is based on a strong superconvergence at the outflow boundary of each element in smooth regions for the DG method to detect discontinuities and to lower the order of accuracy in the approximation there to avoid spurious oscillations near such discontinuities when solving hyperbolic systems of conservation laws.

For our purpose as a troubled-cell indicator, the KXRCF indicator can be described as follows. Partition the boundary of a cell I_i into two portions ∂I_i^- and ∂I_i^+, where the flow is into ($\vec{v}\cdot\vec{n} < 0$) and out of ($\vec{v}\cdot\vec{n} > 0$) I_i, respectively. The discontinuity detector in [12] is defined as

$$\mathcal{I}_i = \frac{\left|\displaystyle\int_{\partial I_i^-} (u^h|_{I_i} - u^h|_{I_{n_i}})\, ds\right|}{h^{(k+1)/2}\left|\partial I_i^-\right| \|u^h|_{I_i}\|}. \tag{2.11}$$

Here we choose h as the radius of the circumscribed circle in the element I_i, I_{n_i} is the neighbor of I_i on the side of ∂I_i^-, and the norm is based on the maximum norm taken at the integration quadrature points in two dimensions and based on an element average in one dimension. If $\mathcal{I}_i > 1$, the cell I_i is identified as a troubled cell, marked for further reconstruction.

7. [illegible] [9]. We will denote this indicator as the Harten indicator in this paper.

In [9], Harten introduced the notion of subcell resolution, which is based on the observation that, unlike point values, cell averages of a discontinuous piecewise-smooth function contain information about the exact location of a discontinuity within the cell. For our purpose as a troubled-cell indicator, the Harten indicator can be described as follows.

Let

$$F_i(z) = \frac{1}{\Delta x}\left\{\int_{x_{i-1/2}}^{z} u^h(x,t)|_{I_{i-1}}dx + \int_{z}^{x_{i+1/2}} u^h(x,t)|_{I_{i+1}}dx\right\} - u_i^{(0)},$$

where $u^h(x,t)|_{I_{i-1}}$ denotes the approximate solution u^h defined in the cell I_{i-1}, extended into cell I_i, and likewise $u^h(x,t)|_{I_{i+1}}$ is the approximate solution u^h defined in the cell I_{i+1}, extended into cell I_i. When

$$F_i(x_{i-1/2}) \cdot F_i(x_{i+1/2}) \le 0,$$

I_i is suspected of having a discontinuity of $u^h(x,t)$ in its interior. However, this could also be a smooth extremum of the solution. To exclude the latter case, Harten [9] has a criterion comparing a minmod function of the first divided differences. We modify this criterion in the context of the RKDG method as follows. We compare the kth moment $u_i^{(k)}$, which has the same magnitude of the kth derivative of $u^h(x,t)$ modulo constant with that of the neighbors. Thus, if

(2.12)
$$F_i(x_{i-1/2}) \cdot F_i(x_{i+1/2}) \le 0 \quad \text{and} \quad |u_i^{(k)}| > \alpha |u_{i-1}^{(k)}|, \quad |u_i^{(k)}| > \alpha |u_{i+1}^{(k)}|,$$

then the cell I_i is identified as a troubled cell, marked for further reconstruction. We take the constant $\alpha = 1.5$ in the numerical tests in the next section.

Once the troubled cells are identified with one of the previous seven methods, we would reconstruct the polynomial solution in this cell, while retaining its cell average using the WENO methodology [13]. We refer to [13] for the details of this reconstruction and will not repeat them here. For the case of hyperbolic systems, to identify the troubled cells, we could use either a componentwise indicator or a characteristic one. The former works on each component of the solution and identifies a troubled cell when any component of the solution is flagging this cell as a troubled cell. The latter works in the local characteristic direction to do this identification. The componentwise indicator is less expensive to implement but may identify more cells as troubled cells, thereby increasing the cost on the reconstruction in these cells. In this paper we will only show results using the componentwise indicator. We have also implemented the characteristic indicator and have obtained qualitatively similar comparison results. For both the one-dimensional and two-dimensional Euler equations, we use only the components of density and energy as indicator variables. We emphasize that the componentwise strategy is used only to identity troubled cells. Once these cells are identified, the WENO reconstructions in them are performed in local characteristic directions. We again refer to [13] and [19] for more details of the reconstruction.

3. Numerical results. In this section we perform extensive numerical experiments to compare the seven different troubled-cell indicators outlined in the previous section when used on the RKDG methods with the WENO reconstruction limiters. The comparison is concentrated on whether the results are oscillatory and on the percentage of cells flagged as troubled cells. We only plot cell averages of the solution in the graphs. The notation TVB-1, TVB-2, and TVB-3 refer to the minmod TVB limiter with the TVB constants $M = 0.01$, $M = 10$, and $M = 100$, respectively.

' . ,' 3.1 We solve the following one-dimensional nonlinear system of Euler equations:

$$u_t + f(u)_x = 0 \tag{3.1}$$

with

$$u = (\rho, \rho v, E)^T, \qquad f(u) = (\rho v, \rho v^2 + p, v(E+p))^T.$$

TABLE 3.1
Average and maximum percentages of cells flagged as troubled cells subject to different troubled-cell indicators for the Sod problem, and the quality of the solutions.

N	Schemes indicator	P^1			P^2			P^3		
		Ave	Max	Osc	Ave	Max	Osc	Ave	Max	Osc
200	TVB-1	10.12	23.00		11.41	23.00		13.70	23.50	
	TVB-2	0.94	2.50		1.25	3.00		1.53	3.00	
	BDF	6.24	12.00		15.89	24.50		13.15	21.50	
	BSB	5.91	11.00		13.56	19.00		14.08	21.00	
	MP	29.34	41.00		25.54	40.00		30.46	46.00	
	MMP	12.28	20.00	L	9.31	18.50		9.72	20.50	
	KXRCF	1.15	2.50		2.05	3.00		2.34	5.50	
	Harten	4.16	10.50		1.00	3.50		2.10	8.50	
400	TVB-1	8.52	17.00		10.16	20.50		11.24	20.75	
	TVB-2	0.79	1.75		0.95	2.25		1.55	2.75	
	BDF	5.46	10.00		13.40	21.25		10.64	19.50	
	BSB	5.27	8.75		10.43	14.75		10.87	16.00	
	MP	18.62	28.75		17.70	28.75		21.65	36.25	
	MMP	9.17	15.75	L	7.48	14.50		8.38	16.50	
	KXRCF	0.89	1.50		1.31	2.25		1.51	3.00	
	Harten	3.31	7.50		0.51	2.00		1.54	5.25	

Here ρ is the density, v is the velocity, E is the total energy, p is the pressure, related to the total energy by $E = \frac{p}{\gamma-1} + \frac{1}{2}\rho v^2$ with $\gamma = 1.4$.

We use the following Riemann initial condition for the Sod problem:

$$(\rho, v, p) = (1, 0, 1) \text{ for } x \le 0, \qquad (\rho, v, p) = (0.125, 0, 0.1) \text{ for } x > 0.$$

Average (over time steps) and maximum percentages of cells being flagged as troubled cells and the quality of the approximation with respect to spurious oscillations, for the different troubled-cell indicators, are summarized in Table 3.1 for $t = 2.0$ using $N = 200$ and $N = 400$ uniform cells. The quality of the approximation is marked in the column "Osc" with "Y" indicating the result is oscillatory (there exist overshoots or undershoots with a size more than 2% of the range of the exact or reference solution), "L" indicating the result is slightly oscillatory (there exist overshoots or undershoots with a size between 0.5% and 2% of the range of the exact or reference solution), or blank indicating the result is nonoscillatory (there are no overshoots or undershoots with a size more than 0.5% of the range of the exact or reference solution). We observe from Table 3.1 that the winners for this example, in terms of having smaller percentages of troubled cells, are the TVB-2, KXRCF, and Harten indicators. We can also observe that the percentage of troubled cells decreases with a mesh refinement in almost all the cases, which is a very desirable property for troubled-cell indicators.

In Figures 3.1–3.3, we plot the computed densities ρ at $t = 2.0$ against the exact solution by the RKDG method with WENO limiters using the TVB-2, KXRCF, and Harten troubled-cell indicators with $N = 200$ uniform cells for $k = 1$, 2, and 3 (second-, third-, and fourth-order RKDG). We can observe that the results for all the cases keep sharp discontinuities and are mostly oscillation-free.

' . ,' 3.2 We repeat the numerical experiments of the previous example using the following Riemann initial condition for the Lax problem:

$$(\rho, v, p) = (0.445, 0.698, 3.528) \text{ for } x \le 0, \qquad (\rho, v, p) = (0.5, 0, 0.571) \text{ for } x > 0.$$

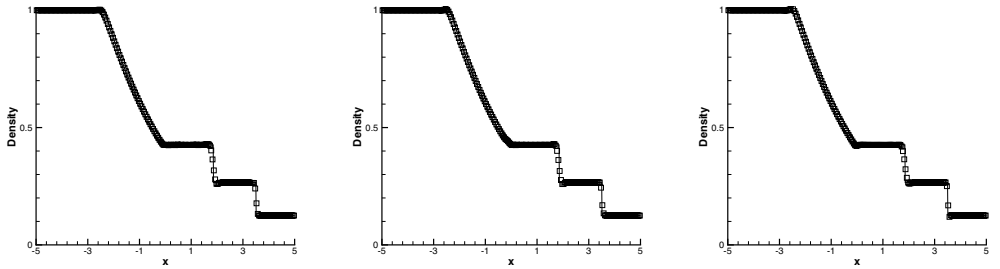


FIG. 3.1. *Sod problem, $t = 2.0$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 1$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

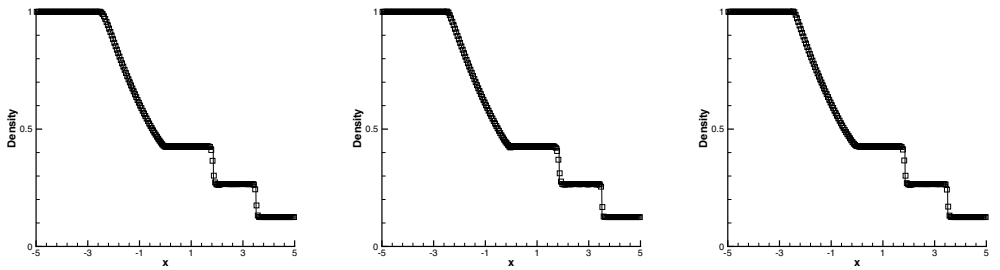


FIG. 3.2. *Sod problem, $t = 2.0$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 2$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

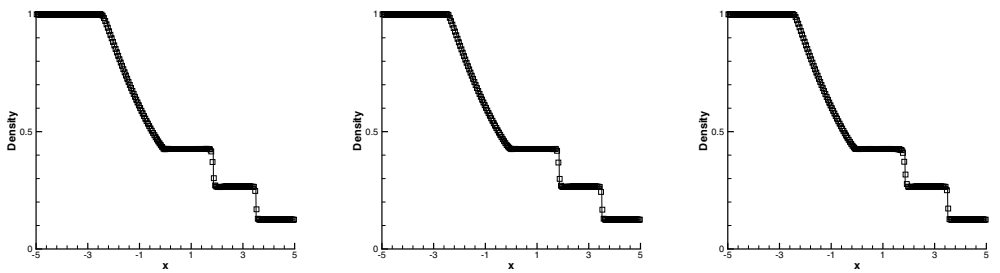


FIG. 3.3. *Sod problem, $t = 2.0$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 3$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

This is a more demanding test case in terms of controlling spurious oscillations. Results similar to those in the previous example, at $t = 1.3$, are given in Table 3.2 and in Figures 3.4–3.6. We can observe a similar pattern as in the previous example, except that more cases involving the TVB-2, MMP, KXRCF, and Harten troubled-cell indicators are marked as "slightly oscillatory" or even "oscillatory."

TABLE 3.2

Average and maximum percentages of cells flagged as troubled cells subject to different troubled-cell indicators for the Lax problem, and the quality of the solutions.

N	Schemes indicator	P^1			P^2			P^3		
		Ave	Max	Osc	Ave	Max	Osc	Ave	Max	Osc
200	TVB-1	10.68	22.00		12.70	24.00		16.31	29.00	
	TVB-2	1.87	4.50		2.56	5.50		3.87	5.50	L
	BDF	8.75	17.00		21.52	39.00		19.53	33.00	
	BSB	8.22	17.00		20.73	36.00		21.62	33.00	
	MP	28.61	36.50		27.95	41.00		32.54	41.50	
	MMP	13.24	22.00	Y	11.05	22.50	L	11.97	21.50	L
	KXRCF	1.33	3.00	L	2.10	3.50		2.46	4.50	
	Harten	3.85	9.00	L	1.00	5.00	L	2.07	6.50	L
400	TVB-1	8.55	17.50		11.02	21.75		13.66	26.00	
	TVB-2	1.37	2.75		1.82	3.50		3.29	4.50	
	BDF	7.86	15.75		19.29	36.75		17.09	34.00	
	BSB	6.57	12.50		18.96	35.50		16.23	24.75	
	MP	16.94	25.50		17.71	28.25		18.94	26.00	
	MMP	9.87	19.00	Y	9.06	18.25	L	9.23	17.00	L
	KXRCF	0.98	1.75	L	1.36	2.75		1.70	3.25	
	Harten	2.51	6.00		0.59	2.75		1.47	6.00	

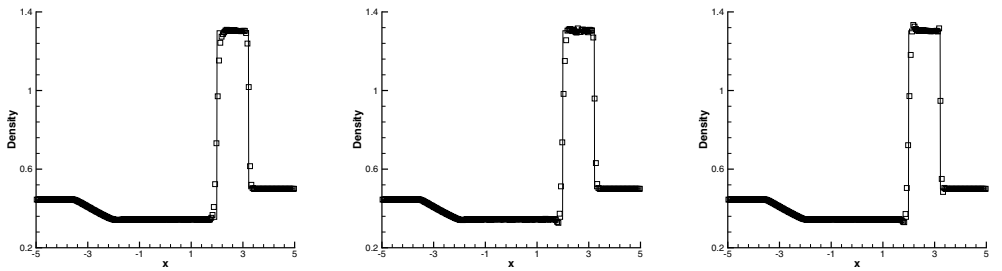


FIG. 3.4. *Lax problem, $t = 1.3$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 1$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

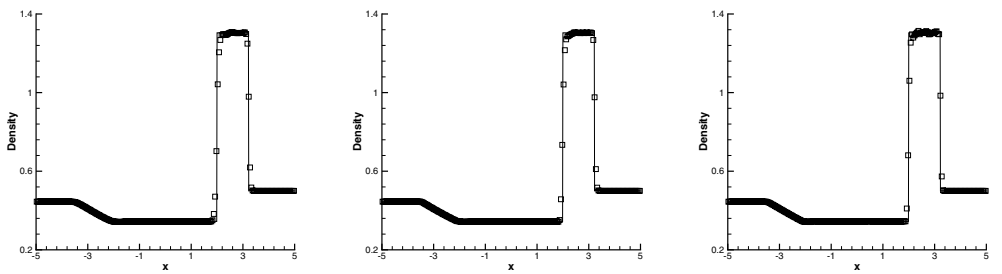


FIG. 3.5. *Lax problem, $t = 1.3$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 2$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

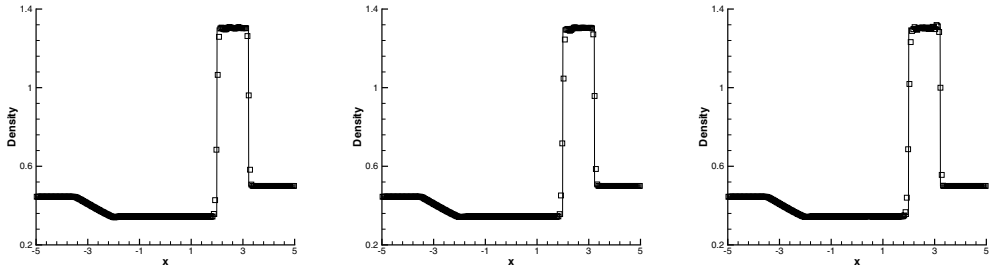


FIG. 3.6. *Lax problem, $t = 1.3$. RKDG with WENO limiters using different troubled-cell indicators: TVB-2 (left), KXRCF (middle), and Harten (right); $k = 3$, 200 cells. Density: solid line, the exact solution; squares, numerical solution.*

TABLE 3.3

Average and maximum percentages of cells flagged as troubled cells subject to different troubled-cell indicators for the shock density wave interaction problem, and the quality of the solutions.

N	Schemes indicator	P^1			P^2			P^3		
		Ave	Max	Osc	Ave	Max	Osc	Ave	Max	Osc
200	TVB-1	14.25	22.50		17.39	25.50		18.52	28.00	
	TVB-3	2.15	5.50		4.20	10.00		5.00	11.50	
	BDF	12.91	22.50		34.13	53.50		36.62	55.00	
	BSB	10.36	15.50		29.43	47.00		37.04	61.50	
	MP	25.09	35.00		22.29	35.00		27.79	39.50	
	MMP	16.82	22.00	L	14.33	21.00		14.17	19.00	L
	KXRCF	2.20	3.50		2.50	4.50		3.26	6.50	
	Harten	3.47	8.50		1.50	6.00		3.39	11.00	
400	TVB-1	9.41	16.50		10.34	15.00		10.61	15.25	
	TVB-3	2.44	6.25		3.61	9.00		3.81	8.50	
	BDF	8.81	15.75		22.31	37.75		19.15	42.00	
	BSB	7.71	11.50		18.62	36.00		21.66	45.75	
	MP	14.68	20.00		13.70	17.50		17.36	21.50	
	MMP	9.98	15.00	L	7.53	11.50		7.28	10.25	L
	KXRCF	1.31	2.25		1.51	3.25		2.31	5.00	
	Harten	1.44	3.75		0.72	3.25		2.37	8.75	

3.3 A higher-order scheme would show its advantage when the solution contains both shocks and complex smooth regions. A typical example for this is the problem of shock interaction with entropy waves [21]. We solve the Euler equations (3.1) with a moving Mach = 3 shock interacting with sine waves in density, i.e., initially

$$(\rho, v, p) = (3.857143, 2.629369, 10.333333) \text{ for } x < -4,$$
$$(\rho, v, p) = (1 + \varepsilon \sin(5x), 0, 1) \text{ for } x \geq -4.$$

Here we take $\varepsilon = 0.2$. The computed density ρ is plotted at $t = 1.8$ against the reference "exact" solution, computed using a fifth-order WENO scheme [11] using 2000 grid points.

Results similar to the previous two examples, at $t = 1.8$, are given in Table 3.3 and in Figures 3.7–3.9. We can observe a similar pattern as in the previous examples; namely, the winners for this example, in terms of having smaller percentages of troubled cells, are the TVB-3, KXRCF, and Harten indicators. Notice that some cases with the MMP troubled-cell indicator are marked as "slightly oscillatory." Because

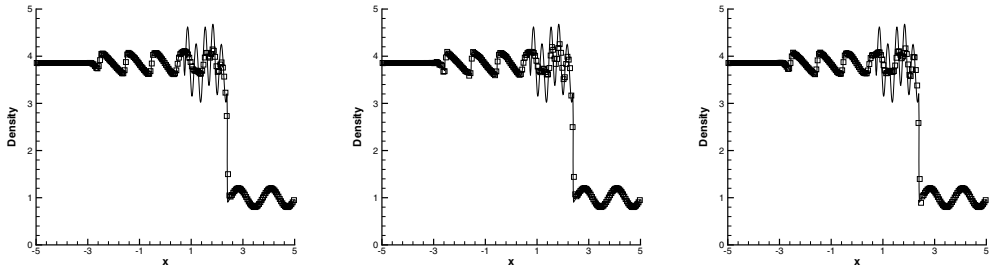


FIG. 3.7. *The shock density wave interaction problem, $t = 1.8$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 1$, 200 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

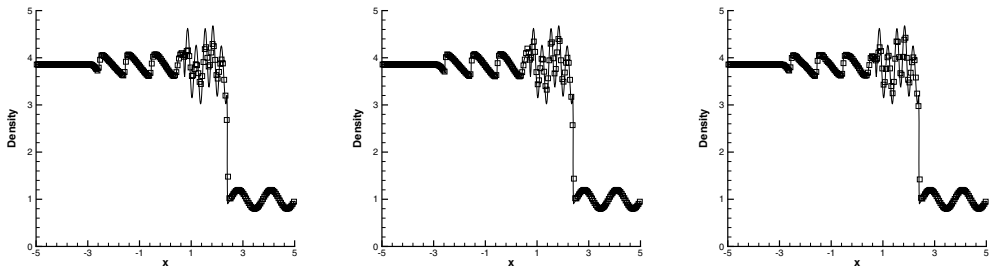


FIG. 3.8. *The shock density wave interaction problem, $t = 1.8$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 2$, 200 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

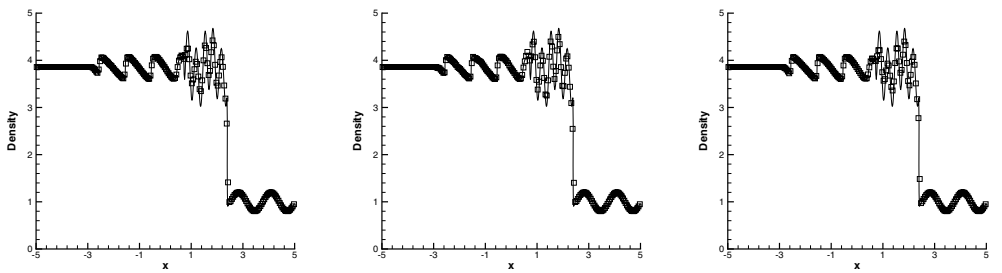


FIG. 3.9. *The shock density wave interaction problem, $t = 1.8$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 3$, 200 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

the TVB-3 and KXRCF troubled-cell indicators identify a smaller percentage of troubled cells than the Harten indicator for this problem, the resolution of the detailed structures in the solution behind the shock is better for the results using the TVB-3 and KXRCF indicators than that using the Harten indicator. This also indicates that the original RKDG method does a better job than the WENO reconstruction in

TABLE 3.4

Average and maximum percentages of cells flagged as troubled cells subject to different troubled-cell indicators for the blast wave problem, and the quality of the solutions.

N	Schemes indicator	P^1			P^2			P^3		
		Ave	Max	Osc	Ave	Max	Osc	Ave	Max	Osc
200	TVB-1	16.56	27.00		15.83	28.50		19.10	34.00	
	TVB-3	9.50	19.50		10.90	19.00		13.87	24.00	
	BDF	15.21	26.50		40.83	63.00		48.70	71.00	
	BSB	13.95	23.00		34.43	48.50		50.33	72.00	
	MP	22.90	38.50		21.89	41.50		24.35	41.50	
	MMP	12.98	22.50		11.46	22.00		12.76	23.50	
	KXRCF	13.66	23.50		15.29	22.50		20.17	29.00	
	Harten	3.59	9.50		1.67	7.00		3.66	11.00	
400	TVB-1	11.07	19.50		11.47	24.25		13.53	26.00	
	TVB-3	6.90	11.75		8.30	15.25		10.04	17.25	
	BDF	10.24	20.25		35.10	51.75		43.08	69.25	
	BSB	9.88	15.50		28.30	41.25		41.74	61.50	
	MP	14.22	25.25		15.41	26.75		17.87	31.75	
	MMP	8.69	14.00		9.08	16.50	L	8.68	18.25	L
	KXRCF	8.45	14.00		10.16	13.75		14.24	20.50	
	Harten	2.11	5.50		0.97	3.50		3.03	8.75	

keeping the resolution for such structures, so we should keep the number of troubled cells, hence the reconstruction, to a minimum under the nonoscillatory constraint. This will be beneficial both in saving computational costs for the reconstruction and in obtaining better resolutions from the original RKDG method.

EXAMPLE 3.4 We consider the interaction of blast waves of the Euler equation (3.1) with the initial condition:

$$(\rho, v, p) = (1, 0, 1000) \text{ for } 0 \le x < 0.1,$$
$$(\rho, v, p) = (1, 0, 0.01) \text{ for } 0.1 \le x < 0.9,$$
$$(\rho, v, p) = (1, 0, 100) \text{ for } 0.9 \le x.$$

A reflecting boundary condition is applied to both ends; see [24, 10]. The computed density ρ is plotted at $t = 0.038$ against the reference "exact" solution, which is a converged solution computed by the fifth-order finite difference WENO scheme [11] with 2000 grid points.

Results similar to the previous examples, at $t = 0.038$, are given in Table 3.4 and in Figures 3.10–3.12. We can observe a similar pattern as in the previous examples; namely, the winners for this example, in terms of having smaller percentages of troubled cells, are still the TVB-3, KXRCF, and Harten indicators. For this problem, only a few cases with the MMP troubled-cell indicators are marked as "slightly oscillatory." Unlike in the previous example, in this example the Harten indicator identifies a smaller percentage of troubled cells than the TVB-3 and KXRCF indicators. As a result, the resolution is better for the results using the Harten indicator than that using the TVB-3 and KXRCF indicators, especially for the $k = 1$ (second-order) case.

EXAMPLE 3.5 Double Mach reflection. This problem is originally from [24]. The computational domain for this problem is $[0, 4] \times [0, 1]$. The reflecting wall lies at the bottom, starting from $x = \frac{1}{6}$. Initially a right-moving Mach 10 shock is positioned at $x = \frac{1}{6}, y = 0$ and makes a 60° angle with the x-axis. For the bottom boundary, the exact postshock condition is imposed for the part from $x = 0$ to $x = \frac{1}{6}$ and a reflective

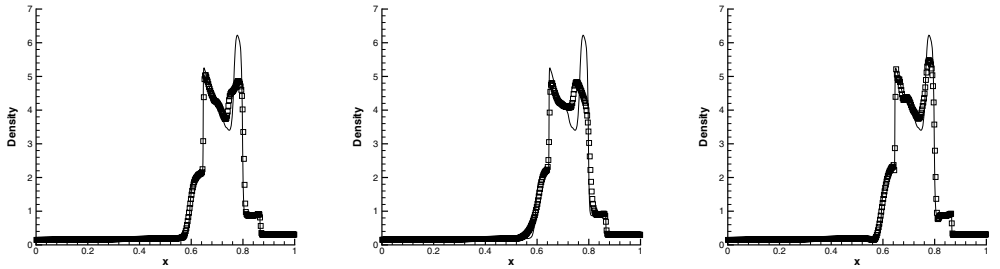


FIG. 3.10. *The blast wave problem, $t = 0.038$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 1$, 400 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

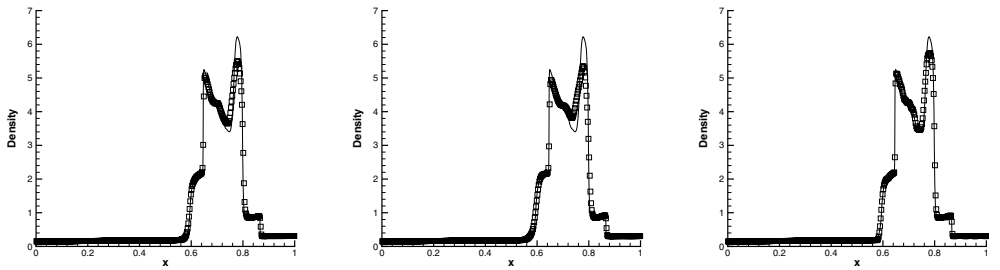


FIG. 3.11. *The blast wave problem, $t = 0.038$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 2$, 400 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

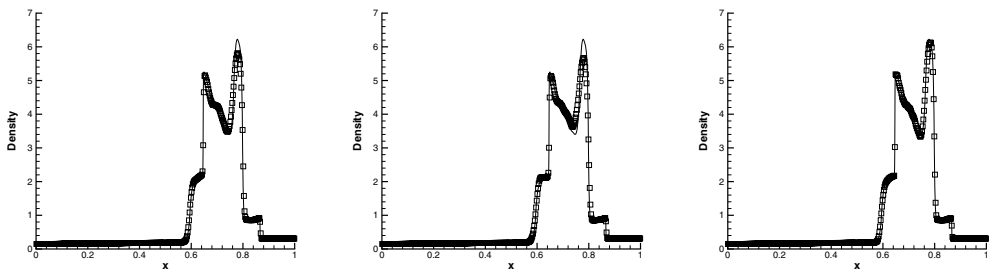


FIG. 3.12. *The blast wave problem, $t = 0.038$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (left), KXRCF (middle), and Harten (right); $k = 3$, 400 cells. Density: solid line, the reference "exact" solution; squares, numerical solution.*

boundary condition is used for the rest. At the top boundary, the flow values are set to describe the exact motion of a Mach 10 shock. We compute the solution up to $t = 0.2$. In Table 3.5 we document the percentage of cells declared to be troubled cells for different orders of accuracy with the TVB-3, KXRCF, and Harten indicators to identify troubled cells. We can see that in most cases only a small percentage

TABLE 3.5
Average and maximum percentages of cells flagged as troubled cells subject to different troubled-cell indicators for the double Mach reflection problem.

$N_x \times N_y$	Schemes indicator	P^1		P^2		P^3	
		Average	Maximal	Average	Maximal	Average	Maximal
120×30	TVB-3	2.53	5.19	2.84	7.00	2.88	5.94
	KXRCF	6.92	9.47	11.28	17.64	12.32	19.58
	Harten	8.65	15.08	6.98	13.94	16.91	33.64
240×60	TVB-3	2.04	3.95	2.72	5.95	3.58	6.78
	KXRCF	4.65	6.51	7.49	11.99	8.45	14.65
	Harten	4.69	8.25	4.08	8.53	13.33	27.51
480×120	TVB-3	1.49	3.05	2.40	4.93	3.96	6.95
	KXRCF	2.99	4.31	4.93	7.86	7.65	12.69
	Harten	2.60	4.87	2.42	5.25	11.19	23.45
960×240	TVB-3	1.06	2.11	2.74	4.76	5.44	8.82
	KXRCF	1.85	2.73	3.44	5.40	7.80	13.10
	Harten	1.52	2.82	1.47	3.43	10.27	21.62

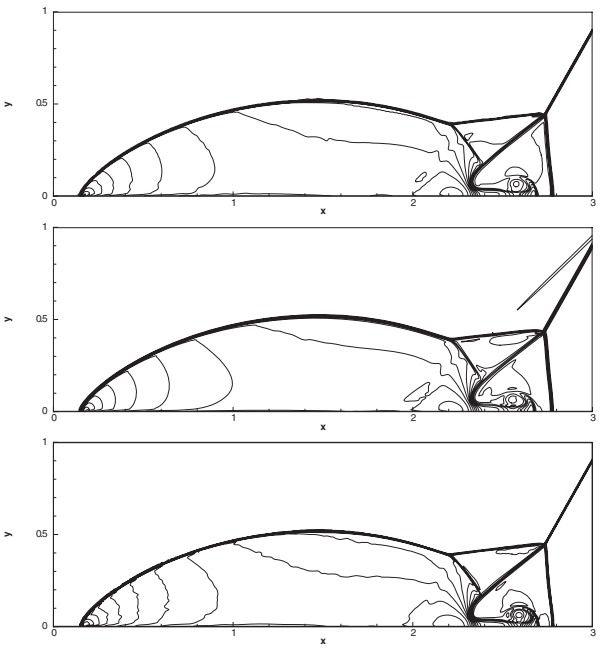


FIG. 3.13. *Double Mach reflection problem, $t = 0.2$. RKDG with WENO limiters using different troubled-cell indicators: TVB-3 (top), KXRCF (middle), and Harten (bottom); $k = 1$, 960×240 cells. Thirty equally spaced density contours from 1.5 to 22.7.*

of cells are declared as troubled cells. For this problem, the TVB-3 indicator seems to be better (in terms of producing a smaller percentage of troubled cells) than the KXRCF indicator, which in turn is better than the Harten indicator. Four uniform meshes, with 120×30, 240×60, 480×120, and 960×240 uniform cells, and three different orders of accuracy for the RKDG with WENO limiters, from $k = 1$ to $k = 3$ (second- to fourth-order), are used in the numerical experiments. To save space, we plot only the simulation results on the most refined mesh with 960×240 cells with the TVB-3, KXRCF, and Harten indicators to identify troubled cells in Figures 3.13,

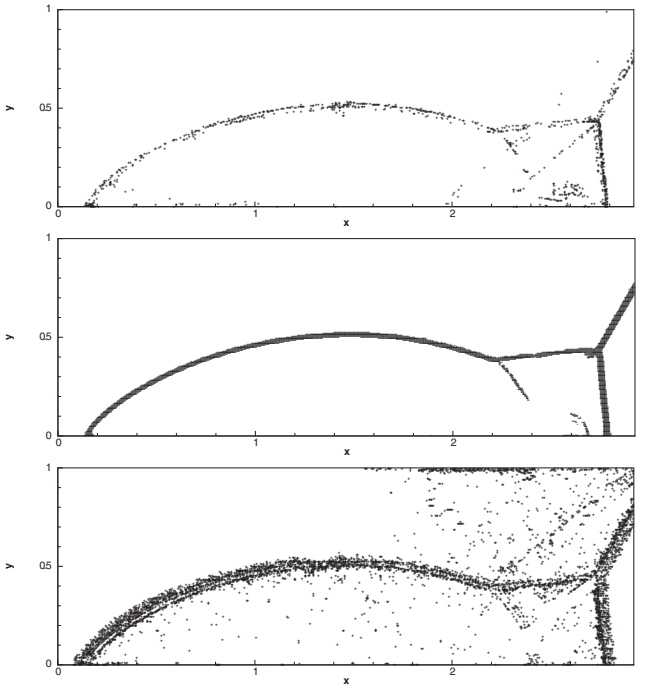


FIG. 3.14. *Double Mach reflection problem. Cells which are identified as troubled cells at* $t = 0.2$. *RKDG with WENO limiters using different troubled-cell indicators: TVB-3* (*top*), *KXRCF* (*middle*), *and Harten* (*bottom*); $k = 1$, 960×240 *cells.*

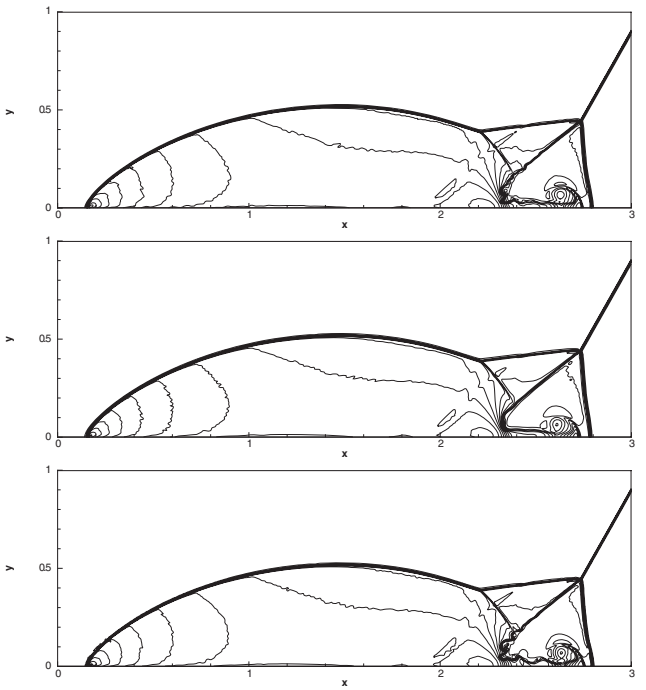


FIG. 3.15. *Double Mach reflection problem,* $t = 0.2$. *RKDG with WENO limiters using different troubled-cell indicators: TVB-3* (*top*), *KXRCF* (*middle*), *and Harten* (*bottom*); $k = 2$, 960×240 *cells. Thirty equally spaced density contours from* 1.5 *to* 22.7.

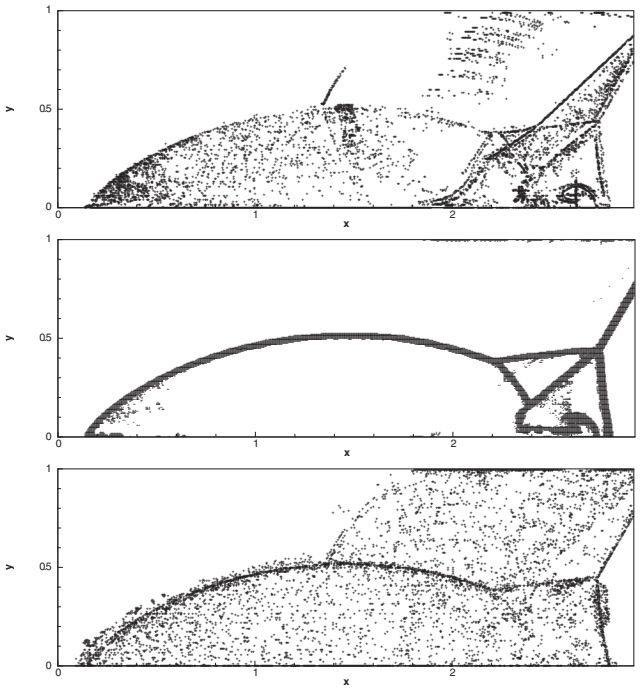


FIG. 3.16. *Double Mach reflection problem. Cells which are identified as troubled cells at* $t = 0.2$. *RKDG with WENO limiters using different troubled-cell indicators: TVB-*3 (*top*), *KXRCF* (*middle*), *and Harten* (*bottom*); $k = 2$, 960×240 *cells.*

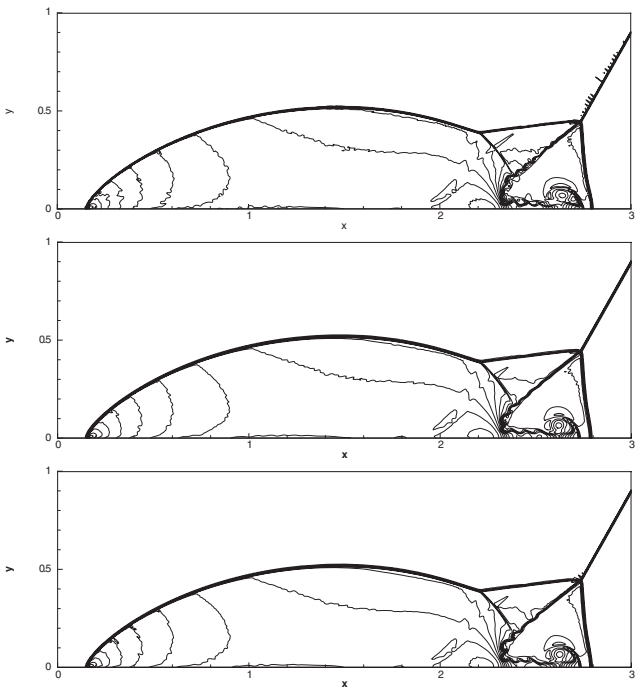


FIG. 3.17. *Double Mach reflection problem,* $t = 0.2$. *RKDG with WENO limiters using different troubled-cell indicators: TVB-*3 (*top*), *KXRCF* (*middle*), *and Harten (bottom);* $k = 3$, 960×240 *cells. Thirty equally spaced density contours from* 1.5 *to* 22.7.

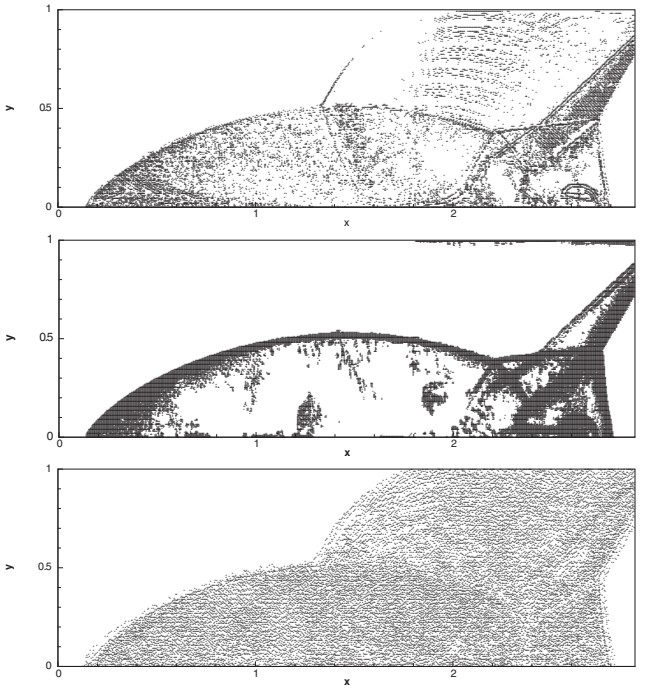


FIG. 3.18. *Double Mach reflection problem. Cells which are identified as troubled cells at* $t = 0.2$. *RKDG with WENO limiters using different troubled-cell indicators: TVB-*3 (*top*), *KXRCF* (*middle*), *and Harten* (*bottom*); $k = 3$, 960×240 *cells.*

3.15, and 3.17. All the figures are showing 30 equally spaced density contours from 1.5 to 22.7. The troubled cells identified by the TVB-3, KXRCF, and Harten indicators at the last time step are shown in Figures 3.14, 3.16, and 3.18, respectively. It is intriguing to see, in Figures 3.13–3.18, that raising the order of approximation from second- to fourth-order (from $k = 1$ to $k = 3$) leads to an increase in the number of troubled cells. Also, the extent to which the three troubled-cell indicators give the correct identification of the troubled cells affects not only the number but also the distribution of the troubled cells within the computational domain, as k increases from 1 to 3.

4. Concluding remarks. In this paper we have systematically studied and compared a few different troubled-cell indicators for the RKDG methods using WENO methodology as limiters. Extensive one- and two-dimensional simulations on the hyperbolic systems of Euler equations indicate that the minmod-based TVB indicator (when the TVB constant M is suitably chosen), the KXRCF indicator by Krivodonova et al. [12], and an indicator based on Harten's subcell resolution idea [9] are better than other choices in all the test cases. Among these three there is no clear winner: any one of them would work better in some examples but not in all examples. All three of them should be suitable candidates for applications of the RKDG methods using WENO reconstructions.

Acknowledgments. The first author would like to thank Professor B. C. Khoo and Professor G. W. Wei for their support and help.

REFERENCES

[1] R. Biswas, K.D. Devine, and J. Flaherty, *Parallel, adaptive finite element methods for conservation laws*, Appl. Numer. Math., 14 (1994), pp. 255–283.

[2] A. Burbeau, P. Sagaut, and C.H. Bruneau, *A problem-independent limiter for high-order Runge–Kutta discontinuous Galerkin methods*, J. Comput. Phys., 169 (2001), pp. 111–150.

[3] B. Cockburn, S. Hou, and C.-W. Shu, *The Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws* IV: *The multidimensional case*, Math. Comp., 54 (1990), pp. 545–581.

[4] B. Cockburn, S.-Y. Lin, and C.-W. Shu, *TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws* III: *One dimensional systems*, J. Comput. Phys., 84 (1989), pp. 90–113.

[5] B. Cockburn and C.-W. Shu, *TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws* II: *General framework*, Math. Comp., 52 (1989), pp. 411–435.

[6] B. Cockburn and C.-W. Shu, *The Runge–Kutta local projection $P1$-discontinuous Galerkin finite element method for scalar conservation laws*, RAIRO Modél. Math. Anal. Numér., 25 (1991), pp. 337–361.

[7] B. Cockburn and C.-W. Shu, *The Runge–Kutta discontinuous Galerkin method for conservation laws* V: *Multidimensional systems*, J. Comput. Phys., 141 (1998), pp. 199–224.

[8] A. Harten, *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., 49 (1983), pp. 357–393.

[9] A. Harten, *ENO schemes with subcell resolution*, J. Comput. Phys., 83 (1989), pp. 148–184.

[10] A. Harten, B. Engquist, S. Osher, and S. Chakravathy, *Uniformly high order accurate essentially non-oscillatory schemes,* III, J. Comput. Phys., 71 (1987), pp. 231–303.

[11] G. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes*, J. Comput. Phys., 126 (1996), pp. 202–228.

[12] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, and J.E. Flaherty, *Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws*, Appl. Numer. Math., 48 (2004), pp. 323–338.

[13] J. Qiu and C.-W. Shu, *Runge–Kutta discontinuous Galerkin method using WENO limiters*, SIAM J. Sci. Comput., 26 (2005), pp. 907–929.

[14] J. Qiu and C.-W. Shu, *Hermite WENO schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method: One dimensional case*, J. Comput. Phys., 193 (2003), pp. 115–135.

[15] J. Qiu and C.-W. Shu, *Hermite WENO schemes and their application as limiters for Runge–Kutta discontinuous Galerkin method* II: *Two dimensional case*, Comput. Fluids, 34 (2005), pp. 642–663.

[16] W.H. Reed and T.R. Hill, *Triangular Mesh Methods for Neutron Transport Equation*, Technical report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.

[17] W.J. Rider and L.G. Margolin, *Simple modifications of monotonicity-preserving limiters*, J. Comput. Phys., 174 (2001), pp. 473–488.

[18] C.-W. Shu, *TVB uniformly high-order schemes for conservation laws*, Math. Comp., 49 (1987), pp. 105–121.

[19] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Math. 1697, A. Quarteroni, ed., Springer, New York, 1998, pp. 325–432.

[20] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, J. Comput. Phys., 77 (1988), pp. 439–471.

[21] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes* II, J. Comput. Phys., 83 (1989), pp. 32–78.

[22] A. Suresh and H.T. Huynh, *Accurate monotonicity-preserving schemes with Runge–Kutta time stepping*, J. Comput. Phys., 136 (1997), pp. 83–99.

[23] P.K. Sweby, *High resolution schemes using flux limiters for hyperbolic conservation laws*, SIAM J. Numer. Anal., 21 (1984), pp. 995–1011.

[24] P. Woodward and P. Colella, *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., 54 (1984), pp. 115–173.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1014–1031
© 2005 Society for Industrial and Applied Mathematics

DETERMINING ANALYTICITY FOR PARAMETER OPTIMIZATION OF THE GEGENBAUER RECONSTRUCTION METHOD*

A. GELB† AND Z. JACKIEWICZ†

Abstract. The Gegenbauer reconstruction method effectively eliminates the Gibbs phenomenon and restores exponential accuracy to the approximations of piecewise smooth functions. Recent investigations show that its success depends upon choosing parameters in such a way that the regularization and the truncation error estimates are equally considered. This paper shows that the underlying analyticity of the function in smooth regions plays a critical role in the regularization error estimate. Hence we develop a technique that first analyzes the behavior of the function in its regions of smoothness and then applies this knowledge to refine the regularization error estimate. Such refinement yields better parameter choices for the Gegenbauer reconstruction method, and is confirmed both by better accuracy and more robustness in the approximation of piecewise smooth functions.

Key words. Fourier pseudospectral approximation, Gegenbauer reconstruction, exponential convergence

AMS subject classifications. 42A25, 42A30, 65D10

DOI. 10.1137/040603814

1. Introduction.

The Gegenbauer reconstruction method, originally developed in [14], effectively eliminates the Gibbs phenomenon from the spectral approximation of a piecewise smooth function while maintaining its exponential convergence properties, even up to the discontinuities of the function. Since its conception, investigators have considered various aspects of the Gegenbauer reconstruction method, including construction from Fourier and other spectral expansion coefficients (e.g., Chebyshev and Legendre) in one and two dimensions as well as on spheres. Recent advancements of the Gegenbauer reconstruction method include an adaptation to noisy environments [2], avoidance of round-off error [7], parameter optimization techniques [7] and [15], and an alternative implementation by the inverse method [17]. All have contributed to making the Gegenbauer reconstruction method a viable option in both high and low resolution environments for multidimensional and multiscale problems, even in the presence of noise. Additionally, the Gegenbauer reconstruction method has been successfully employed in numerous scientific applications, including medical image reconstruction (e.g., [1], [10]) and as a postprocessing technique for conservation laws (e.g., [8], [16]). A good overview of the development and application of the Gegenbauer reconstruction method can be found in [11], [12], and [13], and the references therein.

The success of the Gegenbauer reconstruction method hinges on the underlying analytic behavior of the piecewise smooth function $f(x)$ in the regions between known discontinuities. In particular, the approximation of the regularization error of the

*Received by the editors February 4, 2004; accepted for publication (in revised form) November 29, 2004; published electronically December 30, 2005. This work was partially supported by NSF grants DMS 0324957, DMS–0107428, and EAR–0222327.

http://www.siam.org/journals/sisc/27-3/60381.html

†Department of Mathematics, Arizona State University, Tempe, AZ 85287 (ag@math.la.asu.edu, jackiewi@math.la.asu.edu). The work of the first author was supported by grant NIH–EB02533–01. The work of the second author was supported by NSF grant DMS–9971164.

Gegenbauer reconstruction method has been guided by an assumption of smoothness, first described in [14] as follows.

ASSUMPTION 1.1. Let $f(x)$ be an analytic but nonperiodic function on $[-1,1]$. There exist constants $\rho \geq 1$ and $C(\rho)$ such that for every $k \geq 0$,

$$\max_{-1\leq x\leq 1} \left| \frac{d^k f(x)}{dx^k} \right| \leq C(\rho) \frac{k!}{\rho^k}.$$

The authors proceed to explain that this is a standard assumption for analytic functions where ρ is the distance from $[-1,1]$ to the nearest singularity of $f(x)$ in the complex plane. Note that for $x \in [a,b]$ the assumption can be easily modified.

Later in [12] the regularization error is validated for functions under the following assumption.

ASSUMPTION 1.2. There exist a constant $0 \leq r_0 < 1$ and an analytic extension of $f(x)$ onto the elliptic domain

$$D = \left\{ z : z = \frac{1}{2}\left(r\, e^{i\theta} + \frac{1}{r}\, e^{-i\theta} \right), \quad 0 \leq \theta \leq 2\pi, \quad r_0 \leq r < 1 \right\}. \tag{1.1}$$

Once again this assumption is satisfied by all analytic functions defined on $[-1,1]$ and can easily be modified for $f(x)$ on general $[a,b]$.

Despite the fact that the smoothness of $f(x)$ is generally unknown, the Gegenbauer reconstruction method was shown to yield exponential convergence in most cases [14]. However, the accuracy of the method was seemingly function dependent, and varied with the particular parameters used in its construction.

A parameter optimization technique was first suggested for the Fourier-based Gegenbauer reconstruction method in [7], where an "average case scenario" $\rho = 1$ in Assumption 1.1 was used to estimate the regularization error, since the underlying analyticity of the function in smooth regions was unknown. Later in [15] the procedure was refined for the Chebyshev case, where Assumption 1.2 was also considered. Here it was assumed that the smoothness of $f(x)$ was known explicitly, or equivalently that the values ρ and r_0 were somehow predetermined. As discussed in [15], such knowledge yields significant improvement to the sharpness of the regularization error estimate, which in turn produces a more refined technique for choosing optimal parameters for the reconstruction. It was shown that in some cases the reconstruction was dramatically improved, in particular when $\rho \neq 1$. It is therefore of critical interest to assess the analyticity of the underlying function for reconstruction, since such knowledge will serve to both refine and automate the parameter optimization techniques introduced in [7] and [15]. Specifically, we seek to accurately determine the smoothness parameters ρ and r_0 in Assumptions 1.1 and 1.2. Once this is accomplished, the parameters for the Gegenbauer reconstruction method can be determined accordingly. The Gegenbauer reconstruction method can then be directly applied in any region of smoothness without further consideration of the behavior of the function.

Our paper is organized in the following way. In section 2, the Gegenbauer reconstruction method based on Fourier coefficients is briefly reviewed. A general parameter optimization technique is described in section 3, and a new method for evaluating the analyticity of $f(x)$ is introduced in section 4. Numerical examples for the completely automated procedure are provided in section 5. For ease of presentation, we restrict our discussion throughout this paper to the reconstruction of a smooth nonperiodic function $f(x)$ on $[-1,1]$ from its Fourier coefficients, and note that although some

additional complications involving edge detection will arise for piecewise smooth functions, the techniques discussed here for the specific purpose of reconstruction remain the same.

2. Review of the Gegenbauer reconstruction method. Consider an analytic but nonperiodic function $f(x)$ defined on the interval $[-1,1]$. The Fourier partial sum,

$$f_N(x) = \sum_{n=-N}^{N} \widehat{f}_n e^{in\pi x}, \tag{2.1}$$

with Fourier coefficients $\widehat{f}_n$ defined by

$$\widehat{f}_n = \frac{1}{2}\int_{-1}^{1} f(x)e^{-in\pi x}dx, \tag{2.2}$$

shows $O(1)$ spurious oscillations near the boundaries ± 1. Additionally, the convergence of $f_N(x)$ to $f(x)$ on $[-1,1]$ is reduced to first order. This behavior is the well-known Gibbs phenomenon, and many techniques have been developed to reduce or eliminate its effects. The Gegenbauer reconstruction method, first introduced in [14], uses information from the first $2N+1$ Fourier coefficients (2.2) to completely resolve the Gibbs phenomenon and yields exponential convergence to $f(x)$ in the uniform norm on $[-1,1]$. Here we briefly describe the method and refer the reader to [14], [11], [12], and [13] for more complete details.

Denote by C_l^λ the Gegenbauer polynomials [3], [5], i.e., the polynomials which are orthonormal with respect to the inner product,

$$\langle C_k^\lambda, C_l^\lambda \rangle = \frac{1}{h_l^\lambda}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}} C_k^\lambda(x) C_l^\lambda(x)\,dx,$$

where for $\lambda > 0$ the scaling factor h_l^λ is defined by

$$h_l^\lambda = \sqrt{\pi}\, C_l^\lambda(1)\frac{\Gamma(\lambda+\frac{1}{2})}{\Gamma(\lambda)(l+\lambda)}, \quad C_l^\lambda(1) = \frac{\Gamma(l+2\lambda)}{l!\,\Gamma(2\lambda)}.$$

The Gegenbauer reconstruction method consists of approximating the function $f(x)$ by computing the truncated approximate Gegenbauer series

$$f_g^{m,\lambda}(x) = \sum_{l=0}^{m} \widehat{g}^\lambda(l) C_l^\lambda(x), \tag{2.3}$$

where the coefficients $\widehat{g}^\lambda(l)$ are given by

$$\widehat{g}^\lambda(l) = \frac{1}{h_l^\lambda}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}} f_N(x) C_l^\lambda(x)dx. \tag{2.4}$$

As first shown in [14], the coefficients (2.4) provide an exponentially accurate approximation to the exact Gegenbauer coefficients, defined by

$$\widehat{f}^\lambda(l) = \frac{1}{h_l^\lambda}\int_{-1}^{1}(1-x^2)^{\lambda-\frac{1}{2}} f(x) C_l^\lambda(x)dx, \tag{2.5}$$

provided that $\lambda, m \sim N$; that is, they scale like N. In fact, if λ and m are proportional to N, i.e.,

$$\lambda = \alpha N, \quad m = \beta N, \tag{2.6}$$

then

$$\max_{-1\le x\le 1} \left| f(x) - f_g^{m,\lambda}(x) \right| \le A\Big(q_T^N + q_R^N\Big), \tag{2.7}$$

where A is a polynomial in N, q_T corresponds to the truncation error

$$TE = \max_{-1\le x\le 1} \left| \sum_{l=0}^{m} \Big(\widehat{f}^{\lambda}(l) - \widehat{g}^{\lambda}(l)\Big) C_l^{\lambda}(x) \right|, \tag{2.8}$$

and q_R corresponds to the regularization error

$$RE = \max_{-1\le x\le 1} \left| f(x) - \sum_{l=0}^{m} \widehat{f}^{\lambda}(l) C_l^{\lambda}(x) \right|. \tag{2.9}$$

Both q_T and q_R are constants less than one. Although it was shown that in general $\lambda, m \sim N$ will yield exponential convergence [14], the parameter choices (2.6) lead to the convenient analysis of (2.8) and (2.9) and are furthermore shown in [7] to produce greater theoretical accuracy than other parameter alternatives.

It was shown in [14] that the constant q_T for (2.6) takes the form

$$q_T = \frac{(\beta + 2\,\alpha)^{\beta+2\,\alpha}}{(2\pi e)^{\alpha}\alpha^{\alpha}\beta^{\beta}}. \tag{2.10}$$

As discussed in the introduction, the constant q_R, representing the regularization error, can be defined in different ways depending on the smoothness of $f(x)$ on $[-1, 1]$. If we consider the latter case given in Assumption 1.2, then q_R is defined as [12]

$$q_R = q_R^1 = \left(\frac{(1+2\gamma)^{\frac{1+2\gamma}{2}}}{(2\gamma)^{\gamma}}\, r_0 \right)^{\beta}, \qquad \gamma = \frac{\lambda}{m}. \tag{2.11}$$

The parameters defined in (2.6) yield

$$q_R^1 = \frac{(\beta + 2\alpha)^{\frac{\beta+2\alpha}{2}} r_0^{\beta}}{(2\alpha)^{\alpha}\beta^{\frac{\beta}{2}}}. \tag{2.12}$$

On the other hand, Assumption 1.1 gives the ratio q_R as [14]

$$q_R = q_R^2 = \left(\frac{(1+2\gamma)^{1+2\gamma}}{\rho\, 2^{1+2\gamma}\, \gamma^{\gamma}\, (1+\gamma)^{1+\gamma}} \right)^{\beta}, \qquad \gamma = \frac{\lambda}{m}, \tag{2.13}$$

which, assuming (2.6), reduces to

$$q_R^2 = \frac{(\beta + 2\alpha)^{\beta+2\alpha}}{2^{\beta+2\alpha}\alpha^{\alpha}(\alpha+\beta)^{\alpha+\beta}\rho^{\beta}}. \tag{2.14}$$

In section 3 we propose a parameter optimization technique based on the minimization of the ratio (2.10) corresponding to the truncation error assuming that it is equal to the respective ratios (2.12) and (2.14) corresponding to the regularization errors. We note that although the original theoretical discussion [14] is based on the continuous Fourier coefficients (2.2), all of our numerical experiments are conducted utilizing the corresponding pseudospectral Fourier coefficients

$$\widetilde{f}_n = \frac{1}{2Nc_n} \sum_{j=0}^{2N} f(x_j) e^{-in\pi x_j}, \tag{2.15}$$

based on equally spaced grid points x_j, $j = 0, 1, \ldots, 2N$. Here $c_n = \frac{1}{2}$ for $n = -N$ or $n = N$, and $c_n = 1$ otherwise.

3. Determining optimal parameters. Although it was shown in [14] that the parameter choices $\lambda = \alpha N$ and $m = \beta N$ appearing in (2.6) yield exponential convergence of $f_g^{m,\lambda}(x)$ to $f(x)$ for a large class of functions, the accuracy of the approximation, and even the convergence itself, can depend critically on the choices of these parameters. For instance, numerical experiments in [14] for the reconstruction of analytic nonperiodic functions from Fourier coefficients indicated that choosing the parameters $\alpha = \beta = 0.4$ typically led to more accurate results than for $\alpha = \beta = 0.25$, provided that no round-off error was present. It was additionally shown that for fixed $\beta \le \frac{\pi}{4}$, the ratio q_T defined by (2.10) attains its minimum for

$$\alpha = \frac{1}{4}\Big(\pi - 2\beta + \sqrt{\pi(\pi - 4\beta)}\Big). \tag{3.1}$$

The resulting experiments conducted with $\beta = 0.25$ and α chosen according to (3.1) demonstrated very fast convergence for functions having no regularization error. However, for more general functions in which both the truncation and regularization errors are present, the convergence rate was not improved. This was further analyzed in [7] where a systematic approach was designed to select m and λ to yield exponential convergence for both the truncation and regularization errors. A similar technique was later developed for the Chebyshev–Gegenbauer reconstruction method in [15], where it was additionally assumed that the underlying "smoothness" behavior of $f(x)$ in regions between the discontinuities was known, i.e., the values of ρ and r_0 in Assumptions 1.1 and 1.2 were given. It was also shown that if the parameters m and λ were characterized by (2.6), then the contribution of A to the error bound on the right-hand side of (2.7) was negligible since A is only a polynomial in N. Hence the optimization technique presented in [15] is a function only of the parameters α and β, and is independent of N. We will follow this approach here.

The strategy for choosing α and β in (2.6) is simple. As in [7] and [15], it is based on balancing the truncation (2.10) and regularization errors (2.12) or (2.14). Specifically, we set $q_T = q_R^1$ or $q_T = q_R^2$, depending on the underlying smoothness of the function. The optimal parameters are then determined by the minimization of q_T.

We realize that this optimization is valid only for relatively small N or sparse data sets. Round-off error greatly impacts the ability of the computed Gegenbauer coefficients to decay at the rate needed to balance the large growth of the Gegenbauer polynomials. Here we proceed with the discussion without consideration of round-off error, and note that some numerical results may be more affected by round-off error than by lack of analytical convergence. In such cases it was shown in [7] that

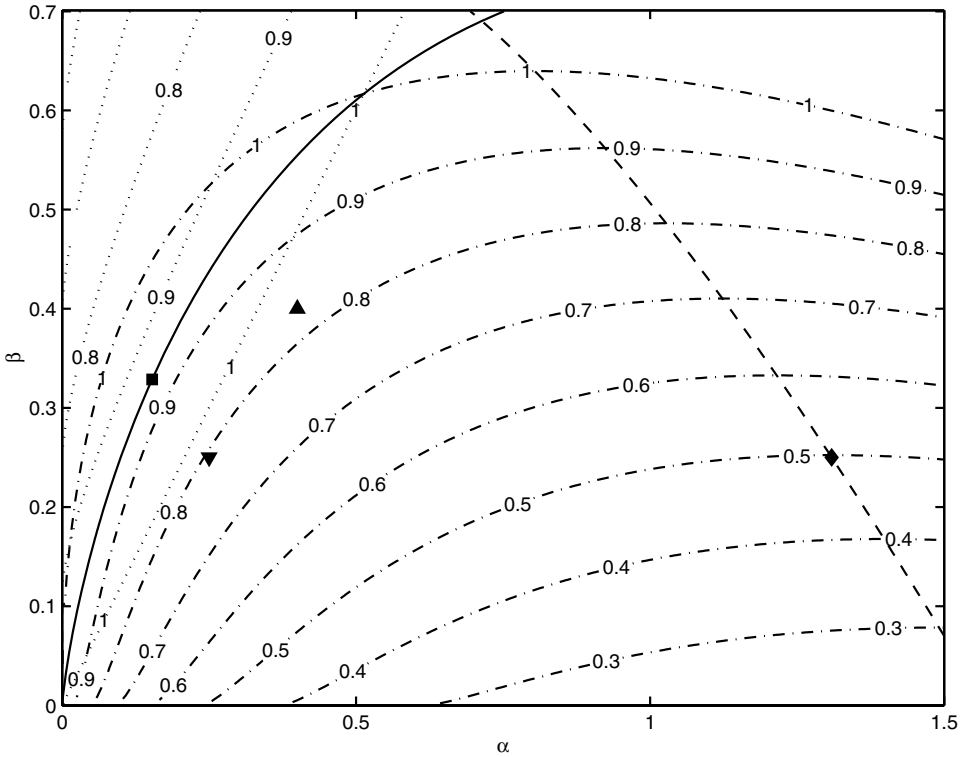


FIG. 3.1. *Contour plots of q_T (dash-dotted lines), q_R^1 (dotted lines), the curve $q_T = q_R^1$ for $r_0 = \sqrt{2}-1$ (solid line), and the curve defined by* (3.1) *(dashed line). Specific point values for α and β include $\alpha = \beta = 0.25$ (downward triangle), $\alpha = \beta = 0.4$ (upward triangle), $\beta = 0.25$ and $\alpha = 1.31$ computed by* (3.1) *(diamond), and the points obtained using our strategy based on minimization of* (3.2) *(square).*

rather than forcing the relationship (2.6) of m and λ with N, it is better to determine maximum admissible values for m and λ such that the Gegenbauer reconstruction is only minimally affected by round-off error. All admissible combinations m and λ are then considered in the computation of (2.7), and the smallest value of (2.7) yields the "optimal" admissible pair of m and λ. Details are discussed in [7]. However, the underlying smoothness of a function still plays a critical role in determining suitable parameters even when round-off error becomes relevant in the reconstruction.

To illustrate the parameter optimization technique derived here, Figure 3.1 displays the contours of q_T (dash-dotted lines), q_R^1 (dotted lines), and the curve $q_T = q_R^1$ for the given value $r_0 = \sqrt{2} - 1$ (solid line). For comparison with the previous parameter selection strategies, Figure 3.1 also displays the curve defined by (3.1) (dashed line).

Figure 3.2 displays the corresponding contour plots of q_R^2 for the "average case scenario," i.e., with $\rho = 1$ which, as discussed in [12] and [15], corresponds to $r_0 = \sqrt{2}-1$ assuming that the singularity of $f(x)$ closest to the interval $[-1, 1]$ is located on the imaginary axis. Both figures display the values for α and β originally suggested in [14] to illustrate the relationship of the parameters to the truncation and regularization errors. Here we point out that the strategy proposed in this paper takes into account both the regularization and truncation errors obtained from the Gegenbauer recon-

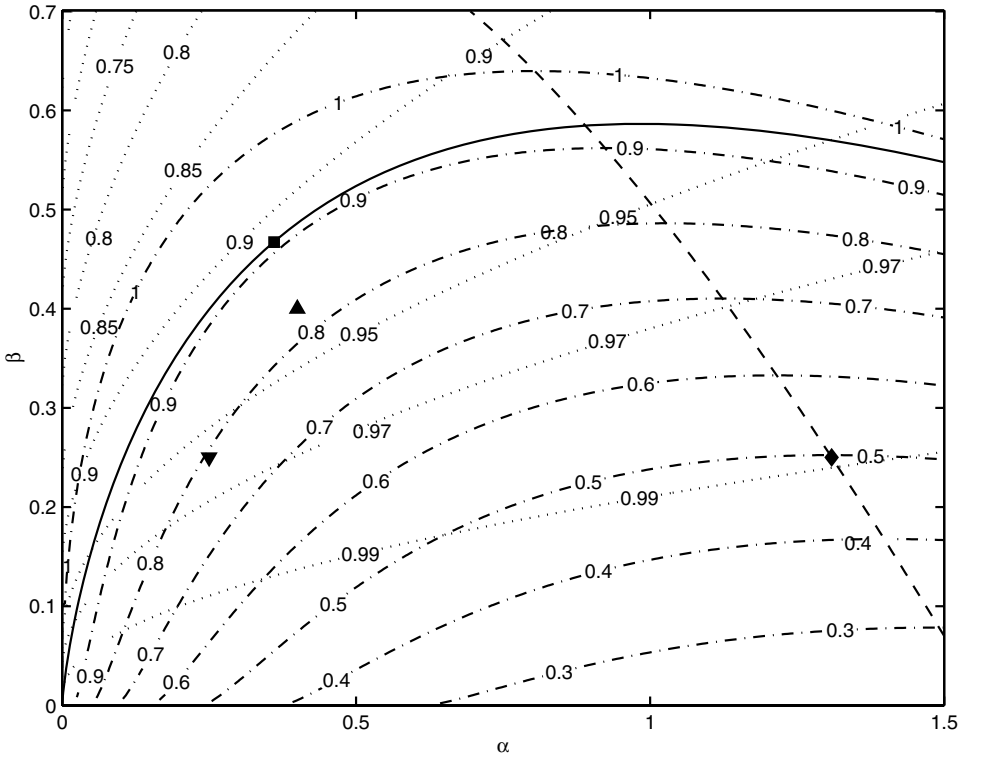


FIG. 3.2. *Contour plots of q_T (dash–dotted lines), q_R^2 (dotted lines), the curve $q_T = q_R^2$ for $\rho = 1$ (solid line), and the curve defined by* (3.1) *(dashed line). Specific point values for α and β include $\alpha = \beta = 0.25$ (downward triangle), $\alpha = \beta = 0.4$ (upward triangle), $\beta = 0.25$ and $\alpha = 1.31$ computed by* (3.1) *(diamond), and the points obtained using our strategy based on minimization of* (3.2) *(square).*

struction method. These values are obtained numerically by minimizing the objective functions

$$\phi^l(\alpha, \beta) = q_T + C\Big(q_T - q_R^l\Big)^2, \quad l = 1, 2, \tag{3.2}$$

where C is a positive penalty constant which was chosen as $C = 1000$. The resulting values for $q_T = q_R^1$ with $r_0 = \sqrt{2} - 1$ are $\alpha = 0.176$, $\beta = 0.343$, while the same algorithm for $q_T = q_R^2$ with $\rho = 1$ yields $\alpha = 0.370$, $\beta = 0.401$.

Figures 3.3 and 3.4 depict the optimal choices for α and β obtained by minimizing (3.2) for $0 \le r_0 \le 1$ and $\rho \ge 1$ (upper graphs) together with the corresponding values of $q_T = q_R^l$, $l = 1, 2$ (lower graphs). The thin lines plotted in Figure 3.4 illustrate the values of α and β obtained by using (2.14) for $\rho < 1$. Once ρ is less than about 0.5, the resulting estimates are meaningless since $q_T = q_R^2$ is complex.

There are several things to note here:

1. The optimization strategy (3.2) used here assumes knowledge of the smoothness parameters ρ and r_0. This information is not inherently known, and hence in section 4 we develop a technique to approximate these values to allow a more refined strategy for determining optimal parameters α and β.

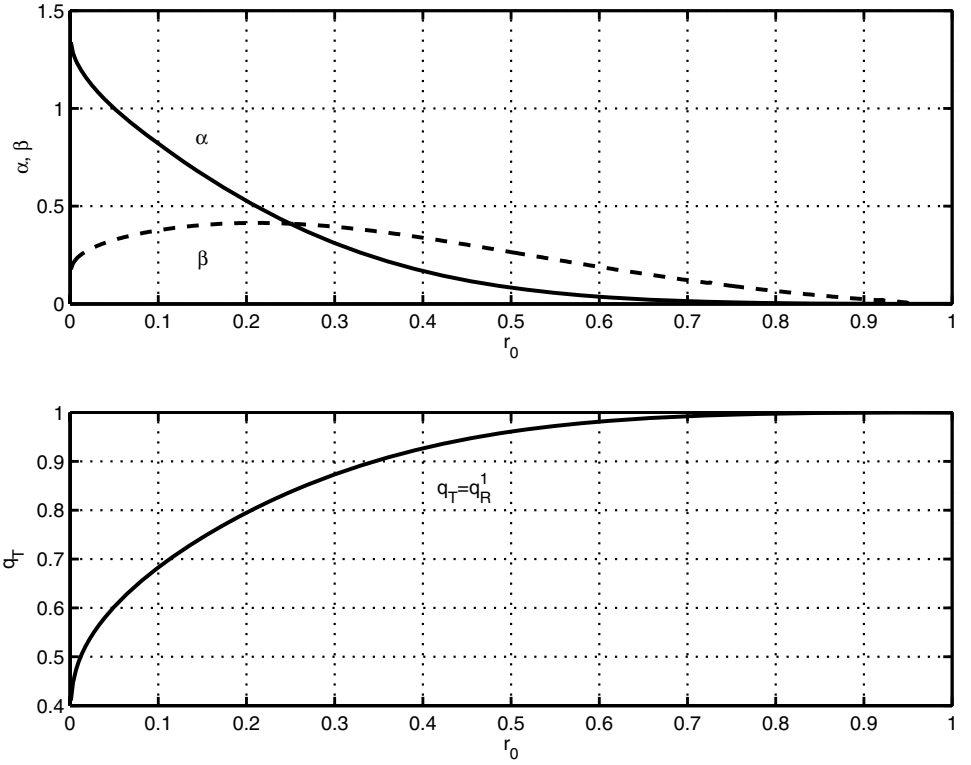


FIG. 3.3. *Parameters α and β versus r_0 (upper graph) and $q_T = q_R^1$ versus r_0 (lower graph).*

2. We reiterate that the optimal strategy (3.2) does not consider round-off errors in its minimization. As discussed previously in [7], the Gegenbauer reconstruction method is numerically inhibited by large values of $m = \beta N$ and $\lambda = \alpha N$, due to the lack of numerical convergence of the Gegenbauer coefficients (2.4). For instance, the values for α shown in Figure 3.4 would be unreasonable for $N > 32$ (conservatively), and in general, the numerically computed Gegenbauer coefficients cease to decay at the required exponential rate for even moderate values of α and β.
3. Although the algorithm is derived for smooth functions in $[-1, 1]$, it can be easily modified to optimize parameter choices for any smooth region $[a, b]$.

To compare the strategy based on minimizing $\phi^1(\alpha, \beta)$ or $\phi^2(\alpha, \beta)$, Figure 3.5 exhibits the equations $q_T = q_R^1$ and $q_T = q_R^2$ versus r_0 under the assumption that the closest singularity of $f(x)$ to $[-1, 1]$ is on the imaginary axis. As explained in [15] (see also [12]), the relationship between r_0 and ρ is then given by

$$\rho = \frac{1}{2}\Big(\frac{1}{r_0} - r_0\Big) \quad \text{or} \quad r_0 = \sqrt{1+\rho^2} - \rho. \tag{3.3}$$

Figure 3.5 indicates that if both strategies for determining α and β are applicable, then minimizing $\phi^2(\alpha, \beta)$ will yield smaller analytical truncation and regularization

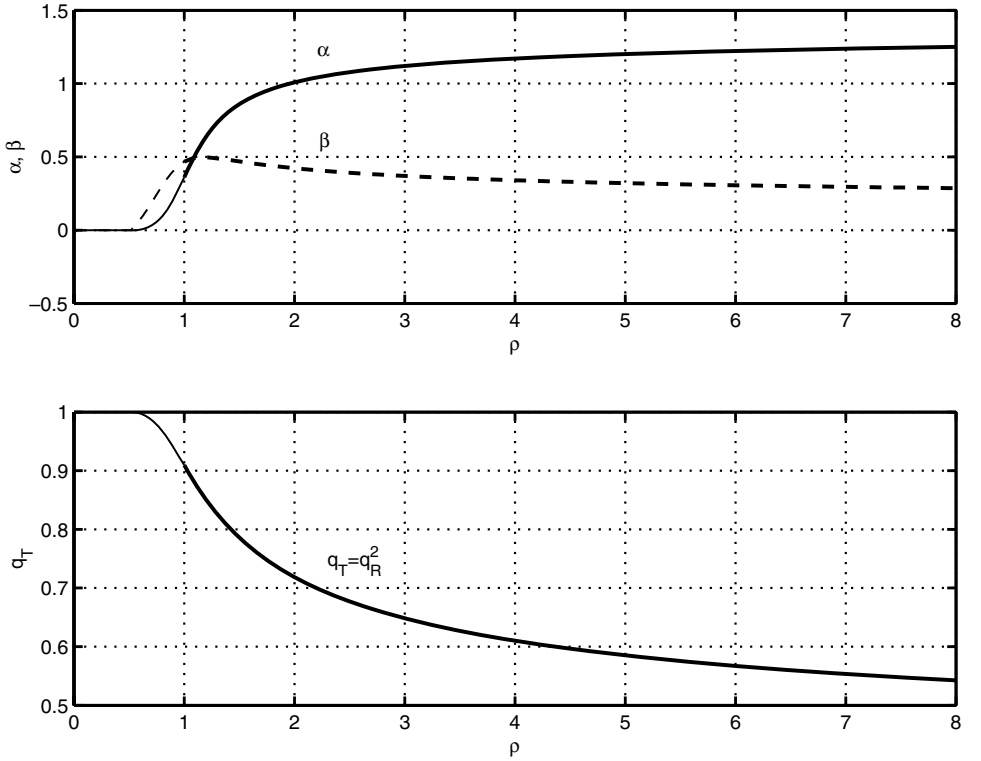


FIG. 3.4. *Parameters α and β versus ρ (upper graph) and $q_T = q_R^2$ versus ρ (lower graph).*

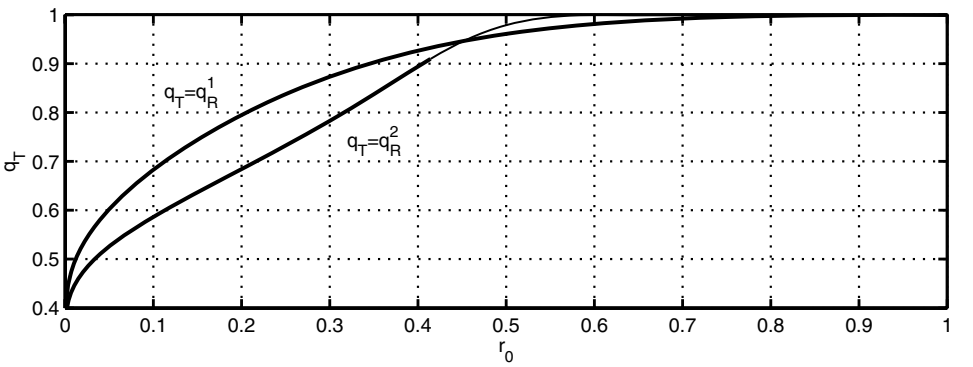


FIG. 3.5. *Ratios $q_T = q_R^1$ and $q_T = q_R^2$ versus r_0.*

errors than minimizing $\phi^1(\alpha, \beta)$, concurring with previous results described for the Chebyshev case in [15]. Figures 3.1 and 3.2 suggest the same conclusion. However, as apparent from the top of Figure 3.4 and prior remarks, minimizing $\phi^2(\alpha, \beta)$ for $\rho \geq 2$ $(0 \leq r_0 \leq \sqrt{5} - 2)$ can potentially yield large round-off errors in the Gegenbauer reconstruction case. In fact, the strategy can be inhibited by round-off error even when $\rho < 2$ $(r_0 > \sqrt{5} - 2)$ for $\lambda = \alpha N$ as N increases. In this case, the top of Figure 3.3 suggests that the strategy based on minimizing $\phi^1(\alpha, \beta)$, although not

theoretically optimal, will yield better results as N increases. Additionally, note the thin line in Figure 3.5 corresponding to $\rho < 1$ or equivalently $r_0 > \sqrt{2} - 1$. In this case, $\rho < 0.5$ ($r_0 > 0.618$) produces complex values for $q_T = q_R^2$ and subsequently yields useless estimates.

Due to the various factors discussed above, we will employ the parameters α and β determined by minimizing $\phi^1(\alpha, \beta)$ in our numerical experiments discussed in section 5.

4. Determination of smoothness parameters. As discussed in section 3, the notion of smoothness plays an important role in determining the optimal parameters α and β for the Gegenbauer reconstruction method. Specifically we wish to determine the analyticity of the underlying function based on the parameters ρ and r_0 described in Assumptions 1.1 and 1.2. These values are unknown in advance and must be approximated from the $2N+1$ Fourier coefficients (2.2) or pseudospectral coefficients (2.15). This is an inherently difficult problem, since the Fourier approximation (2.1) is contaminated by Gibbs oscillations and cannot be expected to produce reliable results for determining factors ρ or r_0.

As shown in [9], another way for assessing smoothness of a function is from its Chebyshev coefficients, namely

$$\limsup_{k\to\infty} |a_k|^{1/k} = r_0, \tag{4.1}$$

where a_k, $k = 0, 1, \ldots$, are the Chebyshev coefficients of the function $f(x)$. This suggests that determining the smoothness parameters ρ and r_0 is equivalent to evaluating the decay rate of the Chebyshev coefficients a_k as $k \to \infty$. Clearly (4.1) cannot be used directly since the exact function $f(x)$ is not known. However, it is possible to compute an approximation to the Chebyshev coefficients $a_k^{(N)}$ based on the partial sum $f_N(x)$ in (2.1). Unfortunately, due to the poor convergence properties of $f_N(\mathrm{x})$, we cannot expect that

$$r_0^{(N)} := \limsup_{k\to\infty} \left| a_k^{(N)} \right|^{1/k} \tag{4.2}$$

will give a reliable estimate of r_0. Hence we seek a function approximation $\widetilde{f}_N(x)$ whose Chebyshev coefficients characterize the original function $f(x)$, such that

$$\widetilde{r}_0^{(N)} := \limsup_{k\to\infty} \left| \widetilde{a}_k^{(N)} \right|^{1/k} \to r_0, \tag{4.3}$$

where $\widetilde{a}_k^{(N)}$, $k = 0, 1, \ldots, K$, are the computed Chebyshev coefficients of $\widetilde{f}_N(x)$.

One possibility is to compute $\widetilde{f}_N(x) = f_g^{m,\lambda}(x)$ in (2.3) based on the assumed value of $\rho = 1$, as in [7]. This approximation converges exponentially to $f(x)$ when $\rho \neq 1$ (although not optimally), yet the computed Chebyshev coefficients $\widetilde{a}_k^{(N)}$ will not yield the correct r_0 in (4.3). This is due to the fact that $f_g^{m,\lambda}$ is a polynomial yielding $r_0 = \infty$ irrespective of m and λ.

A second alternative is to exploit the Bernoulli reconstruction method, [4] and [6], to approximate a function that is characterized by the same analyticity of $f(x)$ once the discontinuities have been removed. To be more specific, in this approach we define

$$\widetilde{f}_N(x) := f_N(x) - \frac{\sigma_N}{2}\Big(s_N(x) - s(x)\Big), \tag{4.4}$$

where $s_N(x)$ is the Fourier partial sum of the function

$$s(x) = x, \quad -1 \le x \le 1,$$

and σ_N is an approximation to the exact jump $\sigma = f(1) - f(-1)$ of the function $f(x)$. This approximation can be computed from the formula [18]

$$\sigma_N = \frac{\pi}{2\mu_1}\Big(f_N(\xi_N) - f_N(\bar{\xi}_N)\Big), \tag{4.5}$$

where

$$\mu_1 = \int_0^\pi \frac{\sin t}{t}\, dt \approx 1.852, \quad \xi_N = \frac{2N-1}{2N}, \quad \text{and} \quad \bar{\xi}_N = \frac{1-2N}{2N}.$$

It can be verified that

$$\widetilde{f}_N(x) - f(x) = g_N(x) - g(x) + \frac{\sigma - \sigma_N}{2}\Big(s_N(x) - s(x)\Big), \tag{4.6}$$

where

$$g(x) = f(x) - \frac{\sigma}{2}\, s(x) \quad \text{and} \quad g_N(x) = f_N(x) - \frac{\sigma}{2}\, s_N(x).$$

The function $g(x)$ is periodic and continuous on $[-1, 1]$ and as a result $g_N(x)$ is linearly convergent to $g(x)$ in the uniform norm. Since σ_N is also linearly convergent to σ, [18], it follows from (4.6) that $\widetilde{f}_N(x)$ is linearly convergent to $f(x)$. Therefore, (4.4) can be viewed as a preliminary resolution of the Gibbs phenomenon. More precisely, since $\widetilde{f}_N(x)$ was obtained by subtracting the oscillatory behavior of $f_N(x)$ we can expect that $\widetilde{f}_N(x)$ and $f(x)$ share similar smoothness characteristics on the interval $[-1, 1]$ described by the parameters ρ or r_0. Hence we expect that the Chebyshev coefficients $\widetilde{a}_k^{(N)}$, $k = 0, 1, \ldots, K$, for $K \approx N$, of the function $\widetilde{f}_N(x)$ will provide an accurate and reliable estimate of r_0 in (4.1). To illustrate that this is indeed the case, Figure 4.1 exhibits the approximations of r_0, $r_0^{(N)}$, and $\widetilde{r}_0^{(N)}$ versus ρ for the family of functions

$$f(x) = \frac{e^x}{x^2 + \rho^2}, \quad -1 \le x \le 1. \tag{4.7}$$

In this case, $f(x)$ has singularities on the imaginary axis at $x = \pm i\rho$ and the relationship between r_0 and ρ is given by (3.3). The approximations to r_0 are computed from the respective formulae

$$r_0 \approx \min\Big\{|a_k|^{1/k} : \ k_0 \le k \le K\Big\}, \tag{4.8}$$

$$r_0^{(N)} \approx \min\Big\{\Big|a_k^{(N)}\Big|^{1/k} : \ k_0 \le k \le K\Big\}, \tag{4.9}$$

and

$$\widetilde{r}_0^{(N)} \approx \min\Big\{\Big|\widetilde{a}_k^{(N)}\Big|^{1/k} : \ k_0 \le k \le K\Big\}, \tag{4.10}$$

which are less susceptible to round-off errors than the approximations based on (4.1), (4.2), and (4.3). Here, k_0 is a small positive integer chosen to eliminate the influence

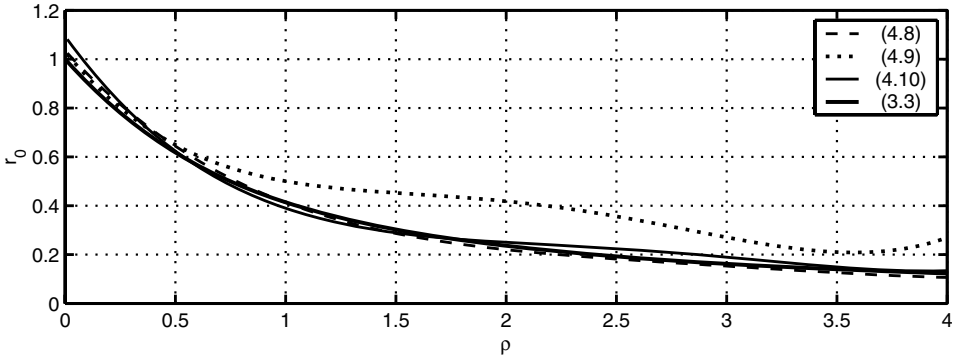


FIG. 4.1. r_0, $r_0^{(N)}$, *and* $\widetilde{r}_0^{(N)}$ *given by* (4.8), (4.9), *and* (4.10) *for* $K = N = 20$ *and* r_0 *given by* (3.3) *versus* ρ *for the function* (4.7).

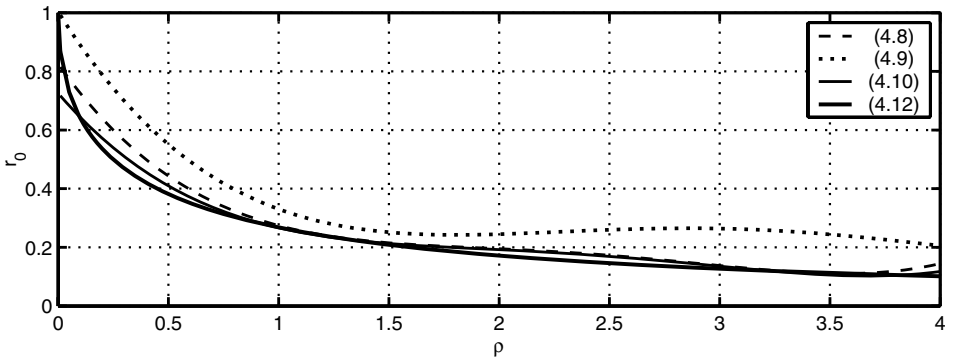


FIG. 4.2. r_0, $r_0^{(N)}$, *and* $\widetilde{r}_0^{(N)}$ *given by* (4.8), (4.9), *and* (4.10) *for* $K = N = 20$ *and* r_0 *given by* (4.12) *versus* ρ *for the function* (4.11).

of the first few Chebyshev coefficients on the estimate of r_0 and $K < N$. The exact value of r_0 given in (3.3) for $f(x)$ given in (4.7) is also plotted in Figure 4.1 by a thick solid line. It is evident that the approximation for $\widetilde{r}_0^N$, (4.10), converges to the desired value r_0. Figure 4.2 displays the approximations r_0, $r_0^{(N)}$, and $\widetilde{r}_0^{(N)}$ versus ρ for the family of functions

$$f(x) = \frac{e^x}{x^2 - (1+\rho)^2}, \quad -1 \le x \le 1, \tag{4.11}$$

which have singularities on the real axis at $x = \pm(1+\rho)$. In this case the relationship of r_0 and ρ is given by (see [12] and [15])

$$\rho = \frac{(r_0 - 1)^2}{2\, r_0} \quad \text{or} \quad r_0 = 1 + \rho - \sqrt{\rho^2 + 2\rho}. \tag{4.12}$$

As exhibited in Figure 4.2, the value of $\widetilde{r}_0^{(N)}$ computed by (4.10) yields an accurate comparison to (4.12), plotted by the thick solid line. The success of the smoothness predictor (4.10) will be further illustrated in section 5.

It is important to consider the following:

1. Recall that when the closest singularity of $f(x)$ to $[-1,1]$ is on the imaginary axis, then the relationship between r_0 and ρ is given by (3.3). This is further indication that parameter optimization based on Assumption 1.2, or equivalently the minimization of $\phi^1(\alpha,\beta)$ in (3.2), is more convenient since the computation of r_0 is possible without a priori knowledge of the position of the closest singularities. To determine ρ for $\phi^2(\alpha,\beta)$, it would first be necessary to determine where these singularities are in the complex plane.
2. Observe in Figures 4.1 and 4.2 that r_0 is approximated for $K = 20$. Although theoretically the estimate of r_0 should improve as $K \to \infty$, numerical implementation indicates that round-off error will affect the results for large K. Furthermore, reasonable estimates are already obtained for $K = 20$ yielding better computational efficiency for the evaluation of the approximation to the parameter r_0.
3. As explained in [6], the approximation (4.4) to $g(x)$ can be of higher order. It is likely that the smoothness parameter $\widetilde{r}_0^{(N)}$ resulting from a higher-order approximation of $g(x)$ will better match r_0 determined from (4.10). However, numerical experiments suggest that the first-order approximation is good enough in resolving (4.3), and we will therefore confine ourselves to this approximation.
4. The method described is developed exclusively for one-dimensional problems. Future investigations will address smoothness assessment for higher-dimensional functions.

5. Numerical examples. In this section we demonstrate the effectiveness of using the optimal parameters α and β which is determined by minimizing $\phi^1(\alpha,\beta)$ in (3.2). Recall that this optimization is inherently dependent on the smoothness parameter, r_0, (4.8), which is approximated by (4.10). As will become apparent, the strategy for selecting the Gegenbauer reconstruction parameters α and β is ideal for sparse data sets, that is, small N. For higher resolution, round-off error prohibits the Gegenbauer reconstruction method from converging for large m and λ, so other parameter alternatives must be considered. We experimented with the functions defined by (4.7) and (4.11), which have singularities on imaginary and real axis. For these respective functions, Tables 5.1 and 5.2 compare the exact values of r_0 given by (3.3) and (4.12) to the values $\widetilde{r}_0^{(N)}$, (4.10). Also displayed are the optimized parameters α and β corresponding to $\widetilde{r}_0^{(N)}$, which are computed by the least squares minimization method described in section 3. Finally, the last column shows the minimum balanced values of $q_T = q_R^1$ which indicate the speed of theoretical exponential convergence of the resulting Gegenbauer reconstruction.

TABLE 5.1
Parameters r_0, $\widetilde{r}_0^{(N)}$ for $K = N = 20$, α, β, and $q_T = q_R^1$ for $\rho = 0.25$, 0.5, 1, 2, and 4 corresponding to the function (4.7).

ρ	r_0	$\widetilde{r}_0^{(N)}$	α	β	$q_T = q_R^1$
0.25	0.781	0.809	0.0035	0.0625	0.998
0.5	0.618	0.618	0.0313	0.177	0.984
1	0.414	0.393	0.176	0.343	0.923
2	0.236	0.249	0.412	0.410	0.837
4	0.123	0.130	0.724	0.395	0.721

TABLE 5.2
Parameters r_0, $\widetilde{r}_0^{(N)}$ for $K = N = 20$, α, β, and $q_T = q_R^1$ for $\rho = 0.25$, 0.5, 1, 2, and 4 corresponding to the function (4.11).

ρ	r_0	$\widetilde{r}_0^{(N)}$	α	β	$q_T = q_R^1$
0.25	0.500	0.540	0.061	0.234	0.970
0.5	0.382	0.405	0.163	0.335	0.929
1	0.268	0.269	0.370	0.401	0.852
2	0.172	0.185	0.566	0.412	0.780
4	0.101	0.113	0.777	0.385	0.700

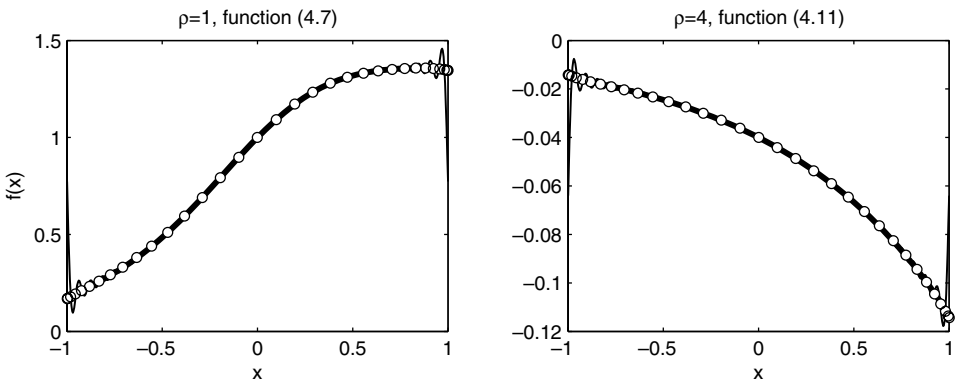


FIG. 5.1. *A comparison of the Gegenbauer (denoted by circles) and Fourier partial sum reconstructions (thin lines) for* (a) (4.7) *with $\rho = 1$ and* (b) (4.11) *with $\rho = 4$. Original functions are plotted by thick lines. Here $N = 32$, with α and β chosen from option* 4.

In the numerical experiments that follow, we consider the following four options for choosing the Gegenbauer parameters α and β:

- Option 1: $\beta = 0.25$ and $\alpha = 1.31$ computed from (3.1).
- Option 2: $\alpha = \beta = 0.25$, as suggested in [14].
- Option 3: $\alpha = \beta = 0.4$, also suggested in [14].
- Option 4: α and β computed by the minimization of $\phi^1(\alpha, \beta)$ in (3.2).

These values (corresponding to option 4) are listed in Table 5.1 for the function (4.7) and in Table 5.2 for the function (4.11) for specific values of the parameter ρ.

Figure 5.1 compares the Gegenbauer and Fourier partial sum reconstruction of (4.7) for $\rho = 1$ and (4.11) for $\rho = 4$ based on the parameters chosen by option 4. In each case we used $N = 32$. The results for the Gegenbauer reconstruction, using the parameters suggested by our strategy are clearly Gibbs-free and convergent.

The truncation (2.10) and regularization (2.12) errors can be viewed as competitive. Specifically, we observe that while the overall error converges for $\lambda, m \sim N$, the results of q_T get worse as m increases, and similarly q_R^1 (and q_R^2) will not decay as quickly for large λ. This behavior, illustrated in Figures 5.2 and 5.3, demonstrates why parameter optimization based on the analyticity of the underlying function is critical for the spectral convergence of the Gegenbauer reconstruction method, and why option 4 sometimes yields the only convergent approximation. The values of α and β corresponding to option 4 and specific values of the parameter ρ are indicated on Figures 5.2 and 5.3 by white diamonds. This is further punctuated by the example

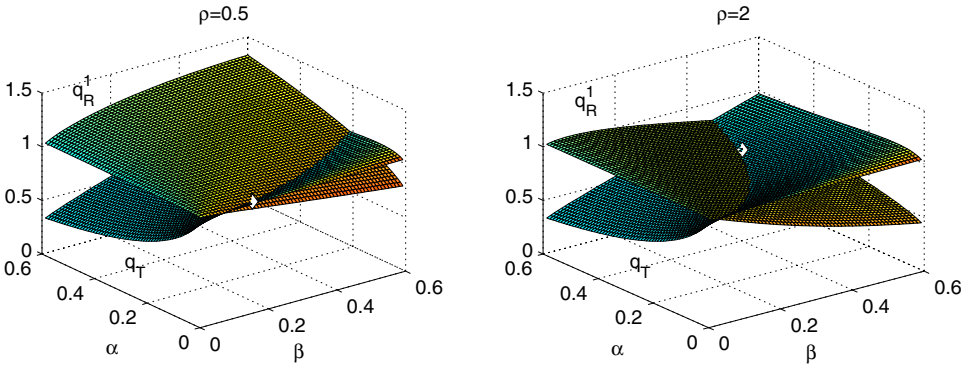


FIG. 5.2. *A comparison of the "competing" regularization and truncation errors for the function* (4.7) *with* (a) $\rho = 0.5$ *and* (b) $\rho = 2$.

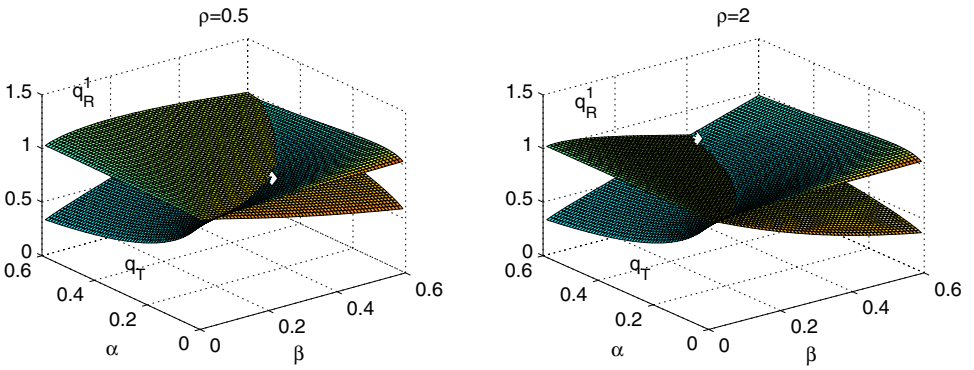


FIG. 5.3. *A comparison of the "competing" regularization and truncation errors for the function* (4.11) *with* (a) $\rho = 0.5$ *and* (b) $\rho = 2$.

given in (4.7) for $\rho \leq 0.5$. The Gegenbauer method is not convergent when α and β are chosen according to option 1, option 2, or option 3. Convergence is only observed when option 4 is used to pick the Gegenbauer parameters. Even in this case, as indicated by the ratio $q_T = q_R^1$ in Table 5.1, the convergence is quite slow and manifests itself only for large values of N. For example, for both $\rho = 0.25$ and $\rho = 0.5$, we require $N \geq 100$ for reasonable convergence and for $\rho = 0.5$, $q_T = q_R^1$ only for values close to 1. On the other hand, when $\rho = 1$, 2, or 4, all of the parameter choosing options lead to exponential convergence. The fastest rate corresponds to option 4. Figure 5.4 (left) illustrates a comparison of the convergence rates of the Gegenbauer reconstruction method utilizing different strategies for determining α and β for the example in (4.7) with $\rho = 1$. A least squares fit of these errors is also provided. As displayed in Table 5.1, when $\rho = 2$ the values of α and β determined by option 2 and by option 4 are quite similar, leading to similar convergence rates for these two choices of α and β.

Similar conclusions are also valid for the example given in (4.11), although in this case we observe exponential convergence for all choices of α and β and all values of ρ listed in Table 5.2. The parameters chosen by option 4 yield the fastest rate of

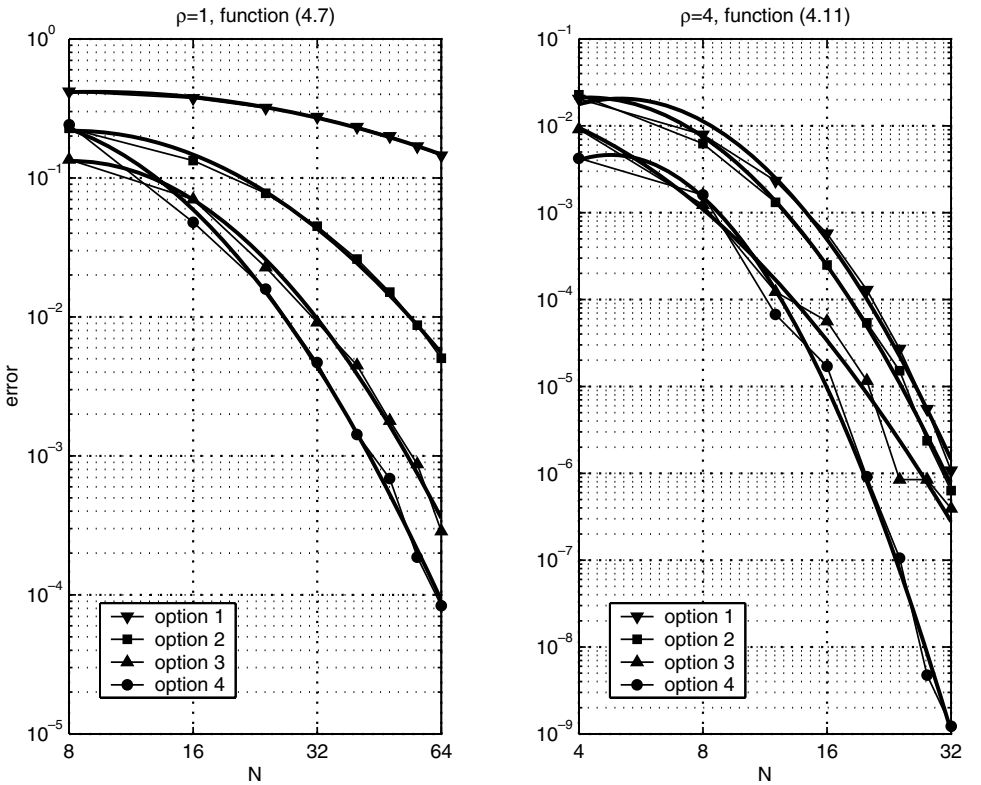


FIG. 5.4. *Errors of the Gegenbauer reconstruction methods for $\rho = 1$ and function* (4.7) *(left graph) and for $\rho = 4$ and the function* (4.11) *(right graph) versus N for different values of the parameters α and β. Least squares fits of these errors are also plotted by thick solid lines.*

convergence with the most pronounced improvement occurring for large values of ρ. For illustration, the errors of the resulting Gegenbauer reconstruction corresponding to $\rho = 4$ and α and β chosen according to option 1, option 2, option 3, and option 4 are displayed in Figure 5.4 (right) along with the least squares fits of these errors.

We note that the results are shown for relatively sparse data sets, or small N. As N and subsequently $\lambda = \alpha N$ and $m = \beta N$ increase, round-off error prohibits the convergence of the numerical Gegenbauer coefficients (2.4), and the overall convergence of the Gegenbauer reconstruction method will start to deteriorate. In fact it is possible to observe the beginning of round-off error effects in Figure 5.4 (right), where option 3 for $N = 28$ and $N = 32$ yields errors that do not signify exponential convergence. The continuation of these numerical experiments for larger N would yield round-off error for all options, and eventually the Gegenbauer reconstruction method would fail to converge at all. This problem can be alleviated by restricting the parameters m and λ by some upper bound where the effects of round-off errors can be controlled. These topics are discussed further in the recent paper [7].

6. Concluding remarks. The efficient resolution of a discontinuous function f from its first $2N + 1$ Fourier coefficients by the Gegenbauer reconstruction method requires the determination of the parameters $\lambda = \alpha N$ and $m = \beta N$ of the Gegenbauer polynomials C_l^λ, and coefficients $\widehat{g}^\lambda(l)$, $l = 0, 1, \ldots, m$. These parameters deter-

mine the speed of exponential convergence of the resulting reconstruction. However, they depend on the smoothness characteristics of the function f and are usually not known in advance. In this paper we have developed a new technique to estimate these smoothness characteristics of the function f from the Fourier partial sum $f_N(x)$. The smoothness of the function f can be characterized by a parameter r_0 which corresponds to the "size" of the ellipse for which the function f has analytic continuation. Our technique of assessing r_0 is based on using the information provided by the Chebyshev coefficients $a_k^{(N)}$ corresponding to the partial Fourier sum $f_N(x)$ and utilizes the fundamental formula $\limsup_{k\to\infty}|a_k|^{1/k} = r_0$, where a_k are Chebyshev coefficients of f, as discussed in [9]. The approximation method for the smoothness parameter r_0 described in this paper is suitable for one-dimensional problems, but due to its construction, we anticipate additional complexities in higher dimensions. Future work will address these issues. Once r_0 is computed, the optimal parameters α and β are then determined by minimizing the objective function (3.2) which assumes that the convergence rates corresponding to the regularization and truncation errors of the Gegenbauer reconstruction are asymptotically equal. The effectiveness of this technique for sparse data sets is demonstrated by numerical experiments for discontinuous functions which have singularities on the real or imaginary axis. In the case of large data sets, the smoothness indicator can be combined with other error estimates to alleviate the effects of round-off errors.

The results given here and other recent investigations suggest that the Gegenbauer reconstruction method may be beneficial for a wide variety of problems, particularly in imaging applications, especially if it continues to produce evidence of satisfactory performance in noisy environments. This will be the topic of future investigations, as will smoothness parameter approximation in higher dimensions.

Acknowledgement. The authors wish to express their gratitude to the anonymous referees for their useful comments.

REFERENCES

[1] R. Archibald and A. Gelb, *A method to reduce the Gibbs ringing artifact in MRI scans while keeping tissue boundary integrity*, IEEE Trans. Medical Imaging, 21 (2002), pp. 305–319.

[2] R. Archibald and A. Gelb, *Reducing the effects of noise in image reconstruction*, J. Sci. Comput., 17 (2002), pp. 167–180.

[3] M. Abramowitz and I. S. Stegun, *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, National Bureau of Standards, Washington, 1972.

[4] W. Cai, D. Gottlieb, and C. W. Shu, *Essentially non-oscillatory spectral Fourier methods for shock wave calculations*, Math. Comp., 52 (1989), pp. 389–410.

[5] P. J. Davis, *Methods of Numerical Integration*, Academic Press, Ontario, San Diego, New York, 1984.

[6] K. S. Eckhoff, *Accurate reconstructions of functions of finite regularity from truncated series expansions*, Math. Comp., 64 (1995), pp. 671–690.

[7] A. Gelb, *On the reduction of round–off error for the Gegenbauer reconstruction method*, J. Sci. Comp., 20 (2004), pp. 433–459.

[8] A. Gelb and E. Tadmor, *Enhanced spectral viscosity approximations for conservation laws*, Appl. Numer. Math., 33 (2000), pp. 3–21.

[9] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1986.

[10] D. Gottlieb, B. Gustafsson, and P. Forssen, *On the direct Fourier method for computer tomography*, IEEE Trans. Medical Imaging, 19 (2000), pp. 223–233.

[11] D. Gottlieb and C. W. Shu, *On the Gibbs phenomenon* IV: *Recovering exponential accuracy in a subinterval from a Gegenbauer partial sum of a piecewise analytic function*, Math. Comp., 64 (1995), pp. 1081–1095.

[12] D. Gottlieb and C. W. Shu, *On the Gibbs phenomenon and its resolution*, SIAM Rev., 39 (1997), pp. 644–668.
[13] D. Gottlieb and C. W. Shu, *A general theory for the resolution of the Gibbs phenomenon*, Atti Convegni Lincei, 147 (1998), pp. 39–48.
[14] D. Gottlieb, C. W. Shu, A. Solomonoff, and H. Vandeven, *On the Gibbs phenomenon* I: *Recovering exponential accuracy from the Fourier partial sum of a nonperiodic analytic function*, J. Comput. Appl. Math., 43 (1992), pp. 81–98.
[15] Z. Jackiewicz, *Determination of optimal parameters for the Chebyshev–Gegenbauer reconstruction method*, SIAM J. Sci. Comput., 25 (2003), pp. 1187–1198.
[16] Z. Jackiewicz and B. Zubik-Kowal, *Spectral collocation and waveform relaxation methods with Gegenbauer reconstruction for nonlinear conservation laws*, Comput. Methods, Appl. Math, 5 (2005), pp. 51–71.
[17] B. Shizgal and J. H. Jung, *Towards the resolution of the Gibbs phenomena*, J. Comput. Appl. Math., 161 (2003), pp. 41–65.
[18] A. Zygmund, *Trigonometrical Series*, Dover, New York, 1955.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1032–1057
© 2005 Society for Industrial and Applied Mathematics

MULTILEVEL ILU WITH REORDERINGS FOR DIAGONAL DOMINANCE*

YOUSEF SAAD†

Abstract. This paper presents a preconditioning method based on combining two-sided permutations with a multilevel approach. The nonsymmetric permutation exploits a greedy strategy to put large entries of the matrix in the diagonal of the upper leading submatrix. The method can be regarded as a complete pivoting version of the incomplete LU factorization. This leads to an effective incomplete factorization preconditioner for general nonsymmetric, irregularly structured, sparse linear systems.

Key words. incomplete LU factorization, ILUT, multilevel ILU preconditioner, Krylov subspace methods, multielimination, complete pivoting

AMS subject classifications. 65F10, 65N06

DOI. 10.1137/030602733

1. Introduction.

In recent years, preconditioned Krylov subspace methods have made good progress toward their acceptance as general purpose techniques for solving irregularly structured sparse linear systems. One may distinguish between two categories of linear systems that are tackled by such methods. First is the class of linear systems which originate from the discretization of partial differential equations. Initially, preconditioned Krylov methods were only attempted on problems of elliptic type, for which a good body of theory existed. Because of their success, these methods have increasingly been employed on systems arising from various flow problems, Stokes and Navier–Stokes equations, Maxwell's equations, Helmholtz equations, and other types of models which lead to indefinite linear systems. The second category of linear systems includes problems which arise from applications which are not governed by partial differential equations (PDEs). An archetype of this class of problems is one that originated from power networks (see, e.g., [21, 18]) and which can be considered in some ways to be at the origin of irregularly structured sparse matrix problems as we know them today. Nowadays, these problems are considered small and are often best handled by sparse direct solvers. However, there are somewhat related problems in circuit simulation [13, 38], which cause difficulties to linear solvers. One may ask whether the distinction between these two classes of problems is important for a given system solver. For direct solvers, the origin of the problem has no direct impact on the solution algorithm, except that its "geometric" nature can influence complexity. The geometry of the underlying graph may matter more than whether or not the problem is likely to be highly indefinite, as it may have a major impact on fill-in. In general, however, the prevailing consensus so far has been that iterative methods are not as successful for this second class of methods as they have been for the PDE-based problems. As the sizes of the linear systems are becoming larger and more challenging, this issue has resurfaced recently and a few methods have been developed to increase the range of applicability of iterative methods. The work of Olschowka and Neumaier

*Received by the editors December 23, 2003; accepted for publication (in revised form) December 2, 2004; published electronically December 30, 2005. This work was supported by NSF grants ACI-0305120 and INT-0003274 and by the Minnesota Supercomputer Institute.
http://www.siam.org/journals/sisc/27-3/60273.html
†Department of Computer Science and Engineering, University of Minnesota, 4-192 EE/CS Building, 200 Union Street S.E., Minneapolis, MN 55455 (saad@cs.umn.edu).

[33] in particular showed that it is possible to essentially perform "static" pivoting prior to Gaussian elimination. Otherwise stated, it is possible to permute the rows of a matrix in such a way that in many cases pivoting becomes unnecessary during the elimination process. Olschowka and Neumaier indicated that this could be helpful in an incomplete LU factorization as well. The work by Duff and Koster described in [23, 24] followed this idea further by developing efficient codes and showing that the method can be quite effective in some cases. Other articles have since tested the idea on various problems and reported excellent success; see, for example, [3, 38].

This paper proposes a reordering method that is similar in spirit to the one described in [23, 24]. While both methods are based on improving diagonal dominance the one presented in this paper is essentially "dynamic" instead of "static." It proceeds by levels whereby each level corresponds to a block which has been extracted from the original matrix by permuting its rows and columns in a nonsymmetric way.

2. The quest for robust ILUs. Several distinct paths have been taken to improve the robustness of preconditioners for irregularly structured linear systems. The first [1] is to add partial pivoting to the incomplete LU factorization with threshold (ILUT); see [35]. The standard ILUT technique often fails for irregularly structured matrices. Partial column pivoting can be added to ILUT, and this yields the ILUTP algorithm [35]. Pivoting does help but there is no guarantee that a factorization will even be produced in some cases. One of the most common cases of failure results from the appearance of zero rows during factorization caused by the combination of permutations and dropping. However, ILUTP can work rather well if substantial fill-in is allowed—but the cost of the factorization may become prohibitive.

The second class of methods, which became popular in recent years, avoids ILU factorizations altogether and utilizes instead one of several methods to compute an approximation to the inverse of A directly; see, e.g., [5, 7, 8, 15, 27, 29]. Initially, these "approximate inverse methods" have been primarily developed for their parallelism. However, they are often viewed as robust alternatives to ILU factorization preconditioners; see, e.g., comments in [4, 6]. Though the factors produced by these methods do not suffer from the instability which plagues incomplete factorizations, they tend to require more memory and to cost more to build.

A third approach is to use a form of preordering whose goal is to move large entries of the matrix to the diagonal. In this method, the matrix is ordered on one side (e.g., rows only). Then pivoting is obviated during Gaussian elimination [24, 23]. The combination of ILU-type preconditioners with such a reordering strategy has been shown to be quite effective [3, 24, 23, 38].

The last class of methods we will mention has been developed following the work by Bollhöfer [9] which addresses the problem of rigorous dropping in ILU-type methods. These methods are founded on the intimate relationships between ILU factorizations and approximate inverse techniques [10]. Small terms are usually dropped in ILU either by considering their location (positional dropping) or their magnitude (threshold dropping) in some relative sense. The first approach works well for regularly structured problems arising from PDEs, but it is not designed for the more general case. The second is generally applicable but the greedy criterion on which it is based focuses on making LU close to A instead of making the preconditioned matrix close to the identity. It is often observed that the resulting factors are very ill-conditioned, and this leads to poor performance. A better strategy was defined in

[1] The order is not chronological.

[9] which exploits condition number estimators for dropping terms whose removal is least likely to be detrimental to the preconditioner. This strategy was combined with a Crout implementation of ILU in [31].

In contrast with similar existing strategies, the method discussed in this paper does not attempt to fill the whole diagonal of the original matrix with large entries. Instead, a more progressive, multilevel approach is taken whereby the unknowns associated with the upper leading submatrix are eliminated and the process is repeated on the reduced system. An alternative view of the process is that of a block complete pivoting ILU. Viewed from this angle, the process consists of finding two permutations P and Q such that

$$PAQ^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

in which the B matrix has large diagonal entries. The unknowns associated with the B block can now be "safely" eliminated and the process repeated on the reduced system. Preconditioning techniques developed in this paper obtain approximate factorizations derived from this strategy. The method can also be regarded as a nonsymmetric version of the algebraic recursive multilevel solver (ARMS), which uses symmetric permutations (i.e., $P = Q$). Numerical experiments indicate a marked improvement in robustness over using symmetric permutations.

Note that in the ideal case, B would be of the same size as A. One can think, for example, of an algorithm that would yield a B matrix that is diagonal and contains most of the large entries of the matrix A. However, forcing B to be diagonal could severely limit its size in general, resulting in an expensive option.

We now briefly discuss practical issues surrounding pivoting for incomplete factorization. When a row version of delayed-update Gaussian elimination (the so-called IKJ version; see [26, p. 99]) is used, it is relatively easy to incorporate column pivoting. Since the pivoting is local, i.e., it is oblivious to the rows below the working (kth) row, dropping may lead to undesired results in later steps during the ILU factorization. For example, we already mentioned the common occurrence of zero rows during the process, especially when low levels of fill are used. When it works, the resulting LU factorization is capable of solving more problems in general, but the loss of structure due to pivoting results in a much more expensive algorithm, especially in terms of memory usage.

Incorporating partial pivoting in the more common rank-one update variant (the so-called KIJ variant) or other forms of incomplete LU factorizations does not seem to have been undertaken. However, the more interesting question one might ask is whether or not a form of complete pivoting is practical in the context of ILU. Searching for the largest entry in the working submatrix at each step seems to be exceedingly expensive and impractical. On the other hand, if this were possible, it would potentially enable the dropping of more terms. This is because the resulting factors are more likely to be stable (the norms of their inverses are not too large), and this will cause a milder effect on the terms dropped; see, for example, [9, 11]. A simplified argument of this fact is as follows. Let $A = LU + E$ where E represents the matrix of the terms that have been dropped from the factorization. Then the split-preconditioned matrix is such that

$$L^{-1}AU^{-1} = I + L^{-1}EU^{-1} .$$

This shows that the larger $\|L^{-1}\|$ and $\|U^{-1}\|$, the smaller the dropped terms must be in order to obtain a preconditioned matrix that is not too far from the identity.

To cope with the issue of cost, the method presented in this paper essentially resorts to a block pivoting approach that performs only an approximate version of complete pivoting.

3. Multilevel ILU preconditioners. A common method used to obtain a preconditioner is via a block ILU factorization, which involves an approximate Gaussian elimination process based on separating the original unknowns into a "coarse" and a "fine" set. These terms are borrowed from the algebraic multigrid (AMG) literature. In the class of preconditioners developed in [14, 37, 1, 36] the idea of independent sets or "group-independent" sets [25, 32, 30, 17, 34] is exploited to define this partition. Block independent set orderings permute the original linear system $Ax = b$ into the form

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1}$$

in which the submatrix B is block diagonal. The ILUM [34] factorization utilizes standard independent set orderings which result in a matrix B that is diagonal. In this situation, it is easy to eliminate the u variable to obtain a system with only the y variable. The coefficient matrix for this "reduced system" is the Schur complement $S = C - EB^{-1}F$ which is still sparse, in general.[2] This idea was used recursively, applying dropping to S to limit fill-in and reordering the resulting reduced system into the above form via independent sets. This is repeated for a few levels until the system is small enough or a maximum number of levels is reached. Then the system is solved by some other standard means, such as a direct solver or an ILUT-GMRES combination.

3.1. Two-sided permutations. The algorithms described in this paper do not require B to have any particular structure. Instead, the rows and columns of A are permuted using two different permutations P (rows) and Q (columns), which will transform A into the form

$$PAQ^T = \begin{pmatrix} B & F \\ E & C \end{pmatrix}, \tag{2}$$

in which the matrix B has the property of being [illegible], in a way that is yet to be specified.

At the lth level of the procedure, the coefficient matrix is reordered as in (2) and the following block factorization is computed "approximately" (subscripts corresponding to level numbers are introduced):

$$P_l A_l Q_l^T = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \approx \begin{pmatrix} L_l & 0 \\ E_l U_l^{-1} & I \end{pmatrix} \times \begin{pmatrix} U_l & L_l^{-1} F_l \\ 0 & A_{l+1} \end{pmatrix}, \tag{3}$$

where

$$B_l \approx L_l U_l, \tag{4}$$

$$A_{l+1} \approx C_l - (E_l U_l^{-1})(L_l^{-1} F_l)\ . \tag{5}$$

[2] For example, when B is diagonal, E has no more than nz_E nonzero elements per row, and F has no more than nz_F nonzero elements per row, then $EB^{-1}F$ will have no more than $nz_E \times nz_F$ nonzero entries per row.

When described with the help of recursivity, the procedure consists essentially of three steps. First, an ordering of the matrix in the form (2) is obtained and applied. Then, an ILU factorization $B_l \approx L_l U_l$ for B_l is computed. Finally, approximations to the matrices $L_l^{-1}F_l$, $E_l U_l^{-1}$, and then A_{l+1} are obtained. The process is repeated recursively on the matrix A_{l+1} until a selected number of levels is reached. At the last level, a simple ILUT factorization, typically with pivoting and very high fill-in levels, can be applied.

The A_is will no longer remain sparse in general, since B is no longer assumed to be diagonal or block-diagonal. Sparsity is regained by dropping small terms in various ways in the construction of L_l, U_l, $L_l^{-1}F_l, E_l U_l^{-1}$, and A_{l+1}. Note that the matrices $E_l U_l^{-1}, L_l^{-1}F_l$ or their approximations

$$G_l \approx E_l U_l^{-1}\ , \qquad W_l \approx L_l^{-1}F_l, \tag{6}$$

which are used to obtain the Schur complement via (5), need not be saved. They are computed only for the purpose of obtaining an approximation to A_{l+1}. Once this is accomplished, they are discarded to save storage. Subsequent operations with $L_l^{-1}F_l$ and $E_l U_l^{-1}$ are performed using U_l, L_l, and the blocks E_l and F_l.

The rationale of the orderings that will be proposed next is that it is critical to have an accurate and well-conditioned B block in the multilevel factorization discussed above. Alternatively, we can also think in terms of complete pivoting. Assume for a moment that B is a 1×1 block. Then, clearly one can think of selecting the permutations P and Q in such a way as to move the best possible pivot from the matrix to location $(1,1)$. The first row can then be eliminated and the process could be repeated for the remaining $(n-1) \times (n-1)$ matrix. Unfortunately, this procedure is too costly because of the search required to find the best pivot at each step. A more effective alternative is to select the best k pivots at the same time, so B now becomes a block of size k. The next section discusses a few heuristics which use diagonal dominance as a criterion for selecting the B block.

3.2. The block complete pivoting viewpoint. We begin by introducing the notation and terminology used in describing the algorithms. A pair of permutations (P, Q) is completely defined from the corresponding two permutations $\{p_1, p_2, \ldots, p_n\}$ $\{q_1, q_2, \ldots, q_n\}$ of the set $\{1, 2, \ldots, n\}$. Since we use partial permutations (which define the block B), we define a matching set $\mathcal{M}$ as a set of n_M pairs (p_i, q_i) where $n_M \leq n$ and

$$1 \leq p_i, q_i \leq n \text{ for } i = 1, \ldots, n_M \quad \text{and} \quad p_i \neq p_j \text{ for } i \neq j, \quad q_i \neq q_j \text{ for } i \neq j. \tag{7}$$

Note that the case when $n_M = n$ corresponds precisely to a (full) permutation pair (P, Q). A partial matching set can be completed into a complete matching set by simply scanning the set of p_is and adding entries from the set $\{1, 2, \ldots, n\}$ which are missing, and then repeating the process for the q_is. The function returning the number of nonzero entries in a row or column is denoted by $nz(.)$. Thus, for example, $nz(a_{i,:})$ is the number of nonzero elements in the ith row of A and $nz(a_{:,j})$ is the number of nonzero elements in the jth column of A.

For lack of a better name we will refer to the class of reordering algorithms to be described in this section as "ddPQ" orderings (P and Q stand for the permutations used and "dd" indicates that the ideal permutation would yield a diagonally dominant B block). These are heuristic methods which consist of two stages. In a first stage, candidate entries a_{ij} are "preselected" as possible diagonal entries. This procedure will

also order the candidate entries that are selected by using a certain measure which will attempt to use a criterion of diagonal dominance and the number of nonzero entries. The second stage scans these candidate entries in the order given by the preselection stage and decides on accepting or rejecting the entry into the $\mathcal{M}$ set.

The simplest preselection algorithm is based on using the ratios

$$\frac{|a_{ij}|}{\|a_{i,:}\|_1}.$$

A criterion using ratios of this type for the diagonal entries ($i = j$) was also used in an early version of ARMS [36] for filtering equations that are least diagonal dominant in a symmetric permutation setting. Let $j(i)$ be a column index of the largest (in absolute value) entry of row i ($|a_{i,j(i)}| \geq |a_{ij}|$ for all j). The ratio

$$\frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1} \tag{8}$$

can be viewed as a measure of rowwise diagonal dominance of row i if the term $(i, j(i))$ is permuted as a diagonal entry. The selection of $a_{i,j(i)}$ as a diagonal term would result in a (weakly) diagonally dominant row if the above ratio is $\geq 1/2$. In practice, we can screen out all entries for which these ratios are no greater than a certain threshold τ.

In a second step, the preselection algorithm will rank all the pairs $(i, j(i))$ in preparation for the next phase of matching, where the permutations P and Q are actually built. The ranking will simply yield a good order in which to scan the list of pairs when determining which ones are valid or best to keep. The simplest possibility is to sort the selected pairs $(i, j(i))$ by decreasing value of the ratios (8). This would favor diagonally dominant rows to be scanned first. However, this ranking does not take sparsity into account. For example, a dense row may be selected because of a high diagonal dominance ratio, and this will yield a full Schur complement matrix. To remedy this, we multiply the above ratios by a quantity that will encourage sparser rows to be selected first. Several such measures were tested and they lead to similar results. The measure

$$\frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1} \times \frac{1}{nz(a_{i,:})} \tag{9}$$

gives higher priority to rows with good diagonal dominance characteristics which are at the same time sparse. Other possibilities tested incorporated the value $nz(a_{:,j(i)})$ or both the values $nz(a_{:,j(i)})$ and $nz(a_{i,:})$. The following algorithm uses the ratios (9).

ALGORITHM 3.1. PRESELECTION

1 $\mathcal{D} = \emptyset$
2 For $i = 1, \ldots, n$
3 Compute $t_i = \|a_{i,:}\|_1$ and $j(i) = argmax_k |a_{ik}|$
4 If $|a_{i,j(i)}| > \tau\, t_i$ then $(i, j(i)) \in \mathcal{D}$
5
6 For each $(i, j(i)) \in \mathcal{D}$
7 Compute $w_i = \frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1} \times \frac{1}{nz(a_{i,:})}$
8
9 Sort $\mathcal{D}$ according to decreasing values of w_i. Thus, $\mathcal{D} = \{(i_1, j_1), (i_2, j_2), \ldots, (i_{n_D}, j_{n_D})\}$ where $w_{i_1} \geq w_{i_2} \geq \cdots \geq w_{i_{n_D}}$

In order to avoid situations in which the preselection algorithm returns with an empty set, we scale the threshold τ relative to the largest of these ratios. Thus, given an input parameter τ_0, the actual threshold τ used in the algorithm is

$$\tau = \tau_0 \times \max_i \left[\frac{|a_{i,j(i)}|}{\|a_{i,:}\|_1} \right] . \tag{10}$$

This ensures that the worst rows on a relative basis will be rejected. The parameter τ_0 should be between 0 and 1 and typical values are in the range 0.1–0.3. If τ_0 is strictly less than one, then at least one row will be preselected.

The algorithm begins by computing the required norms, and then in line 4 it prescreens the rows by keeping only those that do show enough diagonal dominance (relative to other rows). This initial rejection of poor candidate rows will reduce the cost of the next phase, which involves a sort (line 9). As discussed above, in order to reduce fill-in the diagonal dominance ratios are modified. They are divided by the number of nonzero elements in row i (see line 7). The last stage of the algorithm sorts the entries in $\mathcal{D}$ in descending order of the resulting ratios.

The above algorithm selects good candidates for diagonal entries according to a simple diagonal dominance criterion. To complete the process we now need to select pairs to keep in $\mathcal{M}$. We could, for example, scan all pairs returned by Algorithm 3.1 and keep all the pairs as long as they do not violate the requirement in the definition (7). Clearly, the list should be traversed in the same order in which it is provided by Algorithm 3.1. Therefore, the simplest strategy is a greedy technique which scans the set $\mathcal{D}$ and accepts any pair (p_i, q_i) such that neither p_i nor q_i have already been included in a pair of $\mathcal{M}$.

ALGORITHM 3.2. GREEDY MATCHING SET SELECTION

0 $\mathcal{D}$ $count = 0$
1 $\mathcal{M} = \emptyset$ $P(i) = Q(i) = -1$ $i = 1, \ldots, n$
2 $k = 1, \ldots, n_D$
3 (i_k, j_k) k $\mathcal{D}$
4 $(Q(j_k) == -1$
5 (i_k, j_k) $\mathcal{M}$ $count = count + 1$
6 $P(i_k) = count$; $Q(j_k) = count$.
7
8

Note that normally we would also require that $P(i_k) == -1$ in line 4, but this is unnecessary since there is only one (i_k, j_k) selected for each row and therefore $P(i_k)$ can be assigned a positive value only once. At the completion of the algorithm, the matching set is the set of all pairs (i, j) such that $P(i)$ and $Q(j)$ are both nonnegative. Note also that $P(:)$ represents an incomplete reverse permutation array. If completed it would simply be the "old-to-new" usual mapping for the row permutations. Similarly, the Q array represents an incomplete "old-to-new" mapping for the columns. The only remaining step now is to complete these two permutations. This is achieved by a simple greedy procedure such as the following, applied here to P.

```
1. For i = 1 : n Do:
2.       if (P(i) < 0)
3.                 count = count + 1
4.                 P(i) = count;
5.       EndIf
6. EndFor
```

Note that, initially, *count* must have the value returned by the matching algorithm. The same procedure as above is applied to complete the permutation Q. We will refer to this procedure as the "greedy completion" algorithm. It should be mentioned that this simple procedure will allow zero entries to be introduced in the diagonal of the C block. This is not a problem, for two reasons. First, it is the Schur complement matrix that counts and when it is built, fill-ins are likely to be introduced into its diagonal. Second, this block will be further permuted anyway, either by the recursive application of the process to the new level created or, if the last level is reached, by column pivoting in the ILUTP procedure or the full Gaussian elimination, when solving the system.

This process is illustrated in Figure 1. In the 8×8 matrix shown on the left of the figure, the circled entries are the ones preselected by the algorithm, with the numbers in the circles indicating the ranks attributed by the sorting step in Line 9 of the algorithm. The entries shown in shaded squares are the other nonzero elements of the matrix.

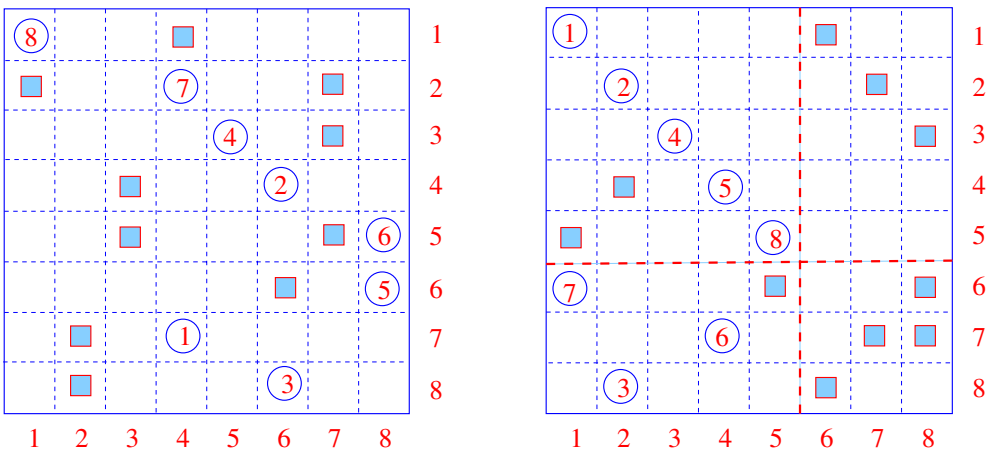


FIG. 1. *An* 8×8 *matrix after the preselection algorithm (left) and the result of the nonsymmetric permutation obtained from the greedy matching procedure. Circled numbers are selected entries and the numbers are the ranks attributed by the preselection algorithm (right).*

Observe that, by its nature, the algorithm clearly selects at most one entry per row. However, there may be more than one selected entry per column. If the greedy matching algorithm is applied, it will end up accepting 5 pairs, and the status of the arrays P, Q at the end is as follows:

P	5	-1	3	2	-1	4	1	-1
Q	5	-1	-1	1	3	2	-1	4

These are then completed by "greedy completion" into

P	5	6	3	2	7	4	1	8
Q	5	6	7	1	3	2	8	4

Once the permutation is found, the matrix is reordered into the form (2). The thick dashed lines in the matrix on the right side of the figure correspond to the partitioning in the right-hand side of (2). Note that there is no particular structure associated with the block B. In the next block-elimination step, equations from 1

to 5 (first level) of the permuted matrix are eliminated and dropping is applied to the resulting 3×3 Schur complement matrix. This means that we compute the factorization (3). This entails computing the ILU factorization of B (4), and then computing an approximation to the Schur complement, as shown in (5). If desired, the process can now be repeated on the next level, i.e., on the Schur complement matrix A_{l+1}.

The outer (Gaussian) elimination procedure stops whenever either a maximum number of levels is exceeded or the Schur complement becomes small enough. Assume that for the above example it is decided that only one level is performed. Then it only remains to solve this last (3×3) linear system. In our code, this is done with a pivoting ILUTP procedure, typically with very little dropping (as dictated by a parameter). For all purposes, one can think of this as an accurate solve with Gaussian elimination with column pivoting. ILUTP is selected for the cases when the last block is not small enough, in order to provide more options to the procedure. Note that one can easily obtain a singular Schur complement matrix S after dropping is applied. If no dropping is applied, then it is well known that S will be nonsingular as long as the original matrix and B are both nonsingular. An important point here is that the permutations P, Q that are sought at a given level do not have to take into consideration the quality of the resulting Schur complement A_{l+1}. This matrix will be dealt with at the next level.

Of course, if the original matrix is strongly diagonally dominant, then the process will retain many rows in the first one or two levels, so the algorithm will require very few levels. In general, the algorithm requires a small number of levels, unless τ_0 is close to one.

If B were to be a diagonal matrix, then the elimination process could be viewed as a multielimination algorithm with complete pivoting. Instead of selecting one pivot at a time, the algorithm selects the largest possible number of pivots allowed by a greedy approach, in which the selection criterion attempts to compromise between a diagonal dominance requirement and the number of nonzero entries in the rows.

Note that one alternative to Algorithm 3.2 is to use the MC21 procedure, which performs a maximal transversal for bipartite matching; see, e.g., [23, 21, 20, 19]. This algorithm, whose goal is to find a maximal set of matching pairs, is not expensive and is easy to implement. One major difference from Algorithm 3.2 is that MC21 does not take into account numerical values or any weights assigned to entries. It only finds a row (or column) permutation P such that PA has nonzero entries on the diagonal.

Another observation is that the procedure will fail when a row or column of the Schur complement is zero. It turns out that this rarely happens in practice. Recall that if B is nonsingular and A is nonsingular, then S is nonsingular, so the situation will not happen in this ideal case. The only risk is to make S singular because of dropping, but this is rare with standard threshold-based dropping procedures.

3.3. Diagonal scaling. Scaling the rows and columns of a general sparse matrix prior to attempting to compute its ILU factorization is critical to the success of the preconditioner. The effect of row and column scaling on the quality of the preconditioner is not too well understood, however. When scaling, e.g., the rows of a nondiagonally dominant matrix by, say, their infinity norms, the effect may not necessarily be beneficial to the accelerator. This may be due to the fact that scaling affects the degree of nonnormality. Diagonally or nearly diagonally dominant matrices do not seem to be affected as much by row/column scaling.

Our research code ARMS-C (C for complete pivoting) offers some options for

scaling the matrices obtained at each level and the last level. If requested, rows of the interlevel matrices A_i are first scaled by their 1-norms. Then, if requested, columns of the resulting matrix are also scaled by their 1-norms. In particular, vectors must be allocated to save the scaling factors for the left and right scaling at each level. Though this strategy works well in many cases, experiments reveal that it is not always the best. As was already mentioned, we have found that the use of scaling may yield a poorer preconditioner than without any scaling at all (see the numerical tests section), though this situation is somewhat uncommon. There are also cases when it is best to start scaling the columns first and then the rows, or scaling rows only or columns only. The issue of scaling in the context of preconditioners is a topic of great importance, and one that has not been given enough attention in the literature. Scaling will be revisited in the numerical experiments section.

4. Matching heuristics. We now examine a few alternatives to the greedy approach of Algorithm 3.2. It may be desirable, for example, to seek to enforce some form of diagonal dominance in the B part of (1). Having a B matrix in a form that leads to easy solutions may also be appealing. We have tested a good number of strategies but show only three here in addition to the greedy technique of Algorithm 3.2.

It is worthwhile to first examine carefully the problem of enforcing diagonal dominance in B. The strategy of Algorithm 3.2 is not guaranteed to produce a B matrix that is diagonally dominant, in general. The ideal situation would be to find permutations P, Q so that the B block in (2) is exactly diagonally dominant, as well as of large size. Note that we are restricting the diagonal dominance requirement to the rows of B, not those of A; i.e., entries in F do not matter. We can state that the problem is to find a matching set satisfying (7) such that the corresponding B matrix, which is defined as

$$\{a_{p(i),q(j)}\}_{i,j=1,\ldots,n_M},$$

is diagonally dominant. As can be easily grasped, this is a difficult problem since its requirement is a global one: the diagonal dominance requirement involves the matching set as a whole.

One possibility for obtaining a diagonally dominant B is simply not to accept any rows that are not diagonally dominant as rows of A in the preselection algorithm. Taking $\tau = 0.5$ in Algorithm 3.1 forces candidate rows scanned in line 4 of the algorithm to be diagonally dominant. Since the corresponding B rows have fewer (nondiagonal) entries than the same row of A, then a fortiori the corresponding row in B will be diagonally dominant. However, this does not work in general simply because in many cases, A may not even have one single row that is diagonally dominant. The sufficient condition used in this strategy is too restrictive. It should be mentioned that in our codes, the scalar τ is determined on a relative basis by a formula such as (10).

Enforcing diagonal dominance in B is easily achieved by restricting B to be triangular. This strategy is presented first in spite of its main limitation, which is that it tends to produce a B block that is small. It will also serve as an introduction to the other matching strategies, which relax the triangularity restriction.

4.1. Matching for a triangular diagonally dominant B. To obtain a lower triangular matrix, it suffices to mark all column indices of a selected row i so that they will become ineligible for inclusion in the later steps. In addition, before adding $(i, j(i))$ to the set $\mathcal{M}$, we need to verify that the row is diagonally dominant. This gives the following modification of Algorithm 3.2.

ALGORITHM 4.1. REORDERING FOR A TRIANGULAR B BLOCK

1 $\mathcal{M} = \emptyset$. $P(i) = Q(i) = -1$, $i = 1, \ldots, n$. $count = 0$.
2 For $i = 1, \ldots, n_D$
3 If $Q(j(i)) != -1$
4 Compute $t_B = \sum_{k \in S_B(i)} |a_{ik}|$ where $S_B(i) = \{k |\ k \in adj(A,i).$
$Q(k) \geq 0\}$
5 If $t_B \leq |a_{i,j(i)}|$
6 Add $(i, j(i))$ to $\mathcal{M}$ (a) $count = count + 1$. (b) $P(i) = Q(j(i)) = count$.
7 For $k = 1, \ldots, ks$ if $adj(A,i)$ and $Q(k) == -1$
8 Set $Q(k) = -2$
9 EndIf
10 EndFor

The set $S_B(i)$ introduced in line 4 represents the set of column indices that have already been accepted into B prior to step i. These will represent the indices of the nonzero entries of row $P(i)$ of the strict lower triangular matrix after permutation. Thus, step 4 computes the sum of the absolute values of all entries in the strict lower part of the matrix that would become B after reordering. The next line tests if the corresponding row is (weakly) diagonally dominant. If it is, then the candidate pair $(i, j(i))$ is accepted (line 6) and all column indices in the row which are not yet assigned are assigned the value -2, so that these columns will not be selected in future steps. This ensures that B will be lower triangular. A negative value that is different from -1 is used to allow proper completion by the greedy completion algorithm.

4.2. Matching for an augmented triangular B. Forcing B to be triangular may be restrictive. It leads to a B block that is typically much smaller than with the greedy approach of Algorithm 3.2, resulting in a higher number of levels and a more expensive factorization in the end. The algorithm described next can be viewed as a less restrictive version of Algorithm 4.1 whereby some entries are allowed in the upper triangular part of B so long as diagonal dominance is preserved.

At each ith step, the modified algorithm begins by computing the quantity

$$t = |a_{i,j(i)}| - \sum_{k \in S_B(i)} |a_{ik}| .$$

Recall that $S_B(i)$ is simply the set of indices that correspond to the lower triangular part of row $P(i)$ of the matrix B. Similarly, we denote by $S_F(i)$ the set of column indices that have been rejected prior to step i. We set $n_B(i) = |S_B(i)|$ and, similarly, $n_F(i) = |S_F(i)|$. Thus, there are at most $nz(a_{i,:}) - n_B(i) - n_F(i)$ entries which have not yet been tagged as members of B or F. If t is negative, then this row cannot be diagonally dominant since the lower part of B is already not diagonally dominant. In this case the algorithm will skip to the next candidate pair. If, on the other hand, t is positive, then we may accept entries to the right of the diagonal entry of B. In this case, the algorithm will proceed to accept entries that are not larger than

$$\frac{t}{nz(a_{i,:}) - n_B(i) - n_F(i))},$$

where $n_B(i)$ is the number of entries of the row being examined that are already in B, and, similarly, $n_F(i)$ is the number of entries that are already in F. All other column entries are rejected; i.e., they are tagged as part of F.

ALGORITHM 4.2. AUGMENTED TRIANGULAR B

1 $\mathcal{M} = \emptyset$. $P(i) = Q(i) = -1$, $i = 1, \ldots, n$. $count = 0$
2 $i = 1, \ldots, n_D$
3 $Q(j(i)) \; != \; -1$
4 $S_B(i) = \{k |\; k \in adj(A, i); Q(k) \geq 0\}$
5 $S_F(i) = \{k |\; k \in adj(A, i); Q(k) == -2\}$.
6 $t_B = \sum_{k \in S_B(i)} |a_{ik}| \quad n_B = |S_B(i)|$ $n_F = |S_F(i)|$
7 $t_B \leq |a_{i,j(i)}|$
8 $(i, j(i))$ $\mathcal{M}$ (a) $count = count + 1$.
(b) $P(i) = Q(j(i)) = count$.
9 $\gamma = (|a_{i,j(i)}| - t_B)/(nz(a_{i,:}) - nB - nF)$
10 k, $adj(A, i)$ $Q(k) == -1$
11 $|a_{ik}| > \gamma$ $Q(k) = -2$
12
13
14
15 $\mathcal{M}$

The resulting rows of B will be row diagonally dominant.

PROPOSITION 4.1. P, Q 4.2 B (2)

Recall that $S_B(i)$ is the subset of column indices that have been accepted into B prior to step i and $S_F(i)$ is the set of rejected columns prior to step i. Define $S'_B(i)$ to be the set of all column indices that have been accepted into B of step i and $S_a(i)$ to be the set of entries that have been accepted at step i. So $S'_B(i) = S_B(i) \cup S_a(i)$. Let $n_a = |S_a(i)|$ and note $n_a + n_B(i) + n_F(i) \leq nz(a_{i,:})$, i.e., $n_a \leq nz(a_{i,:}) - n_B(i) - n_F(i)$. Then

$$\begin{aligned} |a_{i,j(i)}| - \sum_{k \in S'_B(i)} |a_{ik}| &= |a_{i,j(i)}| - \sum_{k \in S_B(i)} |a_{ik}| - \sum_{k \in S_a(i)} |a_{ik}| \\ &= t - \sum_{k \in S_a} |a_{ik}| \\ &\geq t - n_a \times \frac{t}{nz(a_{i,:}) - n_B(i) - n_F(i)} \\ &\geq 0. \qquad \square \end{aligned}$$

A variation on this procedure consists essentially of adjusting the average γ computed in line 9 as new entries are accepted. The sum t_B as well as the number n_B are changed each time a new entry is accepted into B. This allows the acceptance of more entries. An algorithm of this type is described next.

4.3. Matching based on dynamic averages. This technique assigns a "dynamic" measure of diagonal dominance of a row when considered as a row of B at a given step. Each step of the algorithm examines the selected pairs $i, j(i)$ from $\mathcal{D}$. It begins by computing $\rho_i = |a_{i,j(i)}|$, which is the entry of the row with largest magnitude. The row is then scanned to determine how much room is left for accepting new entries into B. This is done as follows. Prior to step i the algorithm has accepted a number of entries already. These will show up in the lower triangular part after reordering. The indicator ρ_i will be adjusted by subtracting from it the absolute values of all these entries. It will be updated similarly every time a column is accepted into

the B part. Some columns have also been rejected and these will show up in the F part after reordering. The columns that are left are potential candidates for being accepted in the B part. The algorithm will reject columns among these based on a criterion similar to the one in the previous algorithm.

When a row is scanned, an entry $|a_{ik}|$ is tested against the current value of ρ_i divided by the number of entries that may still potentially be candidates for being accepted in B. If it is less, then it is accepted into B and ρ_i is decreased accordingly. If the test is not satisfied the column k is marked as not accepted. The counter of the number of entries in the row that are not in B is also decremented.

ALGORITHM 4.3. AUGMENTED TRIANGULAR B WITH DYNAMIC AVERAGES

1 $\mathcal{M} = \emptyset$ $P(i) = Q(i) = -1$ $i = 1, \ldots, n$ $count = 0$
2 $i = 1, \ldots, n_D$
3 $Q(j(i)) \; != \; -1$ $(i, j(i)$
4 $S_B(i) = \{k |\; k \in adj(A, i); Q(k) \geq 0\}$
5 $S_F(i) = \{k |\; k \in adj(A, i); Q(k) == -2\}.$
6 $\rho = |a_{i,j(i)}| - \sum_{k \in S_B(i)} |a_{ik}|$
7 $nz = |adj(A, i)| - |S_B(i)| - |S_F(i)|$
8 $(\rho < 0)$ $(i, j(i))$
9 $(i, j(i))$ $\mathcal{M}$ (a) $count = count + 1$. (b) $P(i) = Q(j(i)) = count$.
10 $j \in Adj(i)$ $Q(j) == -1$
11 $(nz * |a_{i,j}| > \rho)$
12 $Q(j) = -2$ j
13 j
14 $\rho = \rho - |a_{i,j}|$ ρ
15
16 $nz = nz - 1$ nz
17
18
19 $\mathcal{M}$

We now briefly discuss the main steps of the algorithm. The variable ρ can be viewed as a diagonal dominance indicator for the running row, as a row of B. As long as it is nonnegative then the current row remains diagonally dominant. Lines 4 to 7 compute the value of ρ at the start, which is the max value $|a_{i,j(i)}|$ minus the absolute values of the entries $a_{i,j}$ for which the column j is already a row of B. The number of columns that are still "undecided," i.e., that are neither in B nor in F, is also computed (line 7) and it is called nz. If the computed initial value of ρ is negative then clearly row i must be rejected as is done in line 8. Otherwise the pair $(i, j(i))$ is added to $\mathcal{M}$ by updating P, Q, and the counter *count* as is done in line 9. The loop in lines 10–17 scans row i a second time. Each column j that is still undecided ($Q(j) == -1$) is examined. If $|a_{ij}| > \rho/nz$, then the column is tagged as rejected ($Q(j) == -2$). Otherwise it is accepted (so far), and the B-diagonal dominance indicator ρ is updated accordingly. In either case, the number of undecided columns is reduced by one. It is clear that acceptance into B is only tentative since a column that is accepted at this step may be rejected later.

As can be easily established, the algorithm produces a matrix B that is row (weakly) diagonally dominant.

PROPOSITION 4.2. P, Q 4.3 B (2)

. Throughout step i of the algorithm, the variable ρ represents the absolute

value of the difference between $a_{i,j(i)}$ and the entries a_{ij} that are accepted into the B part, i.e., those for which $Q(j)$ will wind up positive after step $i-1$. After line 8, the B part of row i is diagonally dominant since $\rho \geq 0$. When a column is not tagged with -2, then it can potentially become part of B. Consider now the row at the completion of the next loop (lines 10–17). At each step when a column is not rejected (lines 13–15), a new ρ is computed, $\rho_{new} = \rho - |a_{ij}|$, and the new total potential entries in the B part of the row is also updated as $nz_{new} = nz - 1$. We have

$$\rho_{new} = \rho - |a_{ij}| \geq nz|a_{ij}| - |a_{ij}| = nz_{new}|a_{ij}|,$$

which shows that ρ will remain nonnegative throughout the step. In the end there will be fewer entries that are finally accepted (as some columns will be rejected in later steps). Therefore the resulting row of B will remain diagonally dominant. ∎

5. Cost analysis of multilevel nonsymmetric orderings. The first concern when considering the use of the algorithms described above is likely to be in regards to cost, specifically the cost of obtaining the pair of permutations. Obtaining the permutations may be expensive with a poor implementation or a poor choice of parameters.

Consider the preselection algorithm first and let nnz_l be the number of nonzero elements in A_l. At each level there are nnz_l potential candidates to become diagonal entries. The parameter τ allows the elimination of a large fraction of them with the goal of reducing the cost. The algorithm starts by obtaining the largest entry in each row at the cost of nnz_l. In the second phase of the algorithm, weight factors are computed for each of the n_D entries of $\mathcal{D}$ at the cost of $O(n_D)$. Finally, the entries in $\mathcal{D}$ are sorted according to these weights, at the cost of $O(n_D \log n_D)$. Since we do not have an idea of the size of n_D, we need to make an assumption which will enable us to obtain an estimate. We will assume that from one level to the next the size n_l of the matrix A_l is reduced by a constant ratio K. In other words, referring to the splitting (3), we assume that

$$size(A_l)/size(A_{l+1}) \approx K > 1 \ .$$

On average this is a fairly representative model. It is clear that K will depend on τ as well as on the structure of the matrix, its number of nonzero elements per row, etc. At each level the size of $\mathcal{D}$ is a fraction of n_l, the matrix size at level l. Since we limit the fill-in in each row (see the parameters used in the experiments), the number nnz_l of nonzero entries in A_l is a multiple of n_l (in worse situations it can possibly be as high as $n_l \log n_l$, but this does not affect the end result). With this the cost at each level becomes

$$O(nnz_l) + O(n_D \log n_D) \leq \beta n_l + \gamma n_l \log n_l \sim \gamma n_l \log n_l \ .$$

If we have p levels the total cost will be of the form

$$\begin{aligned} T(n) &= \gamma \left[n \log n + \frac{n}{K} \log \frac{n}{K} + \frac{n}{K^2} \log \frac{n}{K^2} + \cdots + \frac{n}{K^p} \log \frac{n}{K^p} \right] \\ &= \gamma \sum_{i=0}^{p} \left[\frac{n}{K^i} \log \frac{n}{K^i} \right] \\ &= \gamma \sum_{i=0}^{p} \left[\frac{n \log n}{K^i} - i \frac{n \log K}{K^i} \right] \end{aligned}$$

$$= \gamma \left[\frac{K - K^{-p}}{K-1} \; n \log n - n \log(K) \; \sum_{i=0}^{p} \frac{i}{K^i} \right] .$$

A close look at the second term in the final expression reveals that it is of order n, since the sum of i/K^i is bounded by a constant which depends only on K (and not p). In the end,

$$T(n) \approx \gamma \frac{K}{K-1} \; n \log n \; .$$

As can be seen, when K is close to one, the algorithm will do poorly, reflecting the fact that the diagonal dominance criterion does not reject enough rows at each level. The analysis indicates that the overall cost for obtaining the permutation is dominated by the sorting step in the preselection phase. In fact, it is clear that an exact sort is not necessary and could easily be replaced by an inexact and less expensive sort. For example, it is possible to reduce the cost of the sorting step by grouping the ratios into a small number of nearby representatives and resorting to a bucket sort. This would lead to a cost of $O(n)$.

The cost of the matching algorithm (for example, Algorithm 4.2) is easier to estimate using a similar model. Once again, the main assumption is that nnz_l is a small multiple, say α, of n_l—due to the dropping strategy. Thus, on average each row of A_l has α nonzero elements. In this case the cost of the ith step of Algorithm 4.2 is $O(n_D) \times \alpha$, with $n_D \approx n_l/K$. Adding these costs for each level will yield

$$n\alpha \left[\frac{1}{K} + \frac{1}{K^2} + \cdots + \frac{1}{K^p} \right] \approx \frac{n\alpha}{K-1},$$

a cost which is linear in n and which shows a similar degradation as for the preselection algorithm, when K is close to one.

6. Numerical tests. We have implemented a version of the algorithm described in earlier sections by using the ARMS framework [36]. We refer to the resulting algorithm as ARMS-C. The goal of the experiments in this section is to illustrate the performance of the ARMS-C strategy as well as to give an idea of its cost and the effect of some parameters. The experiments in section 6.2 and 6.3 were performed on one processor of an IBM/SP computer. The processor is a 375 MHz Power3 (Winterhawk) node with 4 GB of memory. All other experiments were performed on a Linux platform with two 1.7 Xeon GHz processors, a 256 KB cache, and 1 GB of main memory.

6.1. Comparing matching strategies. We begin this section with a comparison of a few matching strategies. In the following experiments, we use only one preselection technique, namely, the one described by Algorithm 3.1. We run a number of strategies and attempt to solve a sequence of linear systems from the Harwell-Boeing collection [22]. The linear systems are obtained from all those matrices from the RUA (real unsymmetric assembled) collection whose size is 500 or higher. There are 58 matrices that satisfy this criterion and their names are listed in Table 1 along with their sizes. If one (or several) right-hand side(s) is (are) provided, then the system with this (the first) right-hand side is solved. Otherwise an artificial right-hand side is taken, which is formed by taking $b = A * e$, where e is the vector of all ones. The iteration is stopped whenever the residual norm has been reduced by 8 orders of magnitude. This does not necessarily mean that the solution that is computed is accurate, because the

TABLE 1
The 58 matrices from the Harwell-Boeing collection with sizes (shown to their right) ≥ 500.

BP1000	822	BP1200	822	BP1400	822	BP1600	822
BP200	822	BP400	822	BP600	822	BP800	822
FS5411	541	FS5412	541	FS5413	541	FS5414	541
FS6801	680	FS6802	680	FS6803	680	FS7601	760
FS7602	760	FS7603	760	GEMAT11	4929	GEMAT12	4929
GRE1107	1107	GRE512	512	JPWH991	991	LNS3937	3937
LNS511	511	LNSP3937	3937	LNSP511	511	MAHINDAS	1258
HBMCFE	765	NNC1374	1374	NNC666	666	ORANI678	2529
ORSIRR1	1030	ORSIRR2	886	ORSREG1	2205	PDE9511	961
PORES2	1224	PORES3	532	PSMIGR1	3140	PSMIGR2	3140
PSMIGR3	3140	SAYLR3	1000	SAYLR4	3564	SHERMAN1	1000
SHERMAN2	1080	SHERMAN3	5005	SHERMAN4	1104	SHERMAN5	3312
HBSHL0	663	SHL200	663	SHL400	663	STEAM2	600
WATT1	1856	WATT2	1856	WEST0655	655	WEST0989	989
WEST1505	1505	WEST2021	2021				

matrices involved may be highly ill-conditioned. Since right-preconditioning is used, the tests are on the actual residuals not the preconditioned ones.

In spite of their rather small sizes, some of the matrices in this list cause difficulties to iterative solvers. It is, for example, not easy to find an iterative procedure which solves all the systems with the same input parameters. Most of the matrices are irregularly structured, and some have a very irregular pattern. The next plots show a comparison of the performance of GMRES preconditioned with ARMS-C with various matching strategies. We compare four methods:

- the greedy strategy of Algorithm 3.2,
- the triangular B algorithm described by Algorithm 4.1,
- the augmented triangular B technique described by Algorithm 4.2,
- the algorithm of dynamic averages described by Algorithm 4.3.

The parameters used for the experiment are listed below.

		Drop tolerance				Fill-max			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
100	0.1	0.001	0.01	0.001	0.01	10	10	10	5

In the table, $nlev_{max}$ is the maximum number of levels allowed and tol_{DD} is the tolerance τ_0 used in the preselection algorithm; see (10). The next 4 parameters define the dropping factors for the factorization algorithms. The drop tolerance for LU-B is the one used in the ILUT factorization [35] of the B block, while the one denoted by GW is used when computing the matrices G_l and W_l defined by (6) at each level. The drop tolerance used to compute S from G_l and W_l is given next. Finally, the drop tolerance for the ILU factorization at the last level is given. The incomplete factorization with column pivoting (ILUTP) is used for this. The next set of columns define the upper limit for fill-ins for each of these computations in the same order. Note that these numbers are the multiples of the average number of nonzero elements, i.e., at the lth level the actual amount of nonzero entries allowed in each row is $fill_{max} * nnz(A_l)/n$. Another parameter not shown here is the cut-off dimension for the last level, which is set to 100. This means that no further levels are sought when the dimension of the Schur complement is ≤ 100 or when the maximum number of levels allowed is reached. We ran ARMS-C with different matching strategies with the [illegible] 58 [illegible].

The incomplete factorizations were computable in all cases. However, the iteration

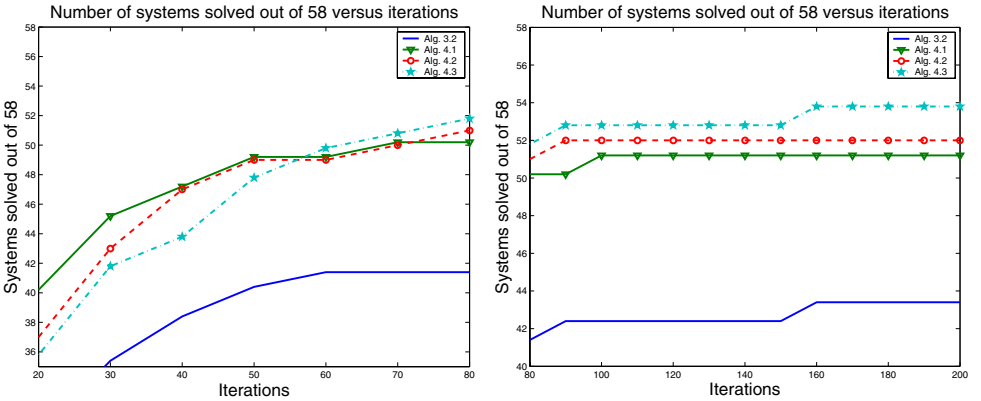


FIG. 2. *Experiment with* 58 *HB matrices showing the number of converged systems among* 58*, versus iteration count.*

TABLE 2
Some statistics on the tests with the 58 *HB matrices.*

Method	Average fill-fact	Average iter. count	Successful iterations
ILUTP	2.71	12.12	43
Alg. 3.2	1.61	20.58	43
Alg. 4.1	1.87	15.00	51
Alg. 4.2	1.63	17.23	52
Alg. 4.3	1.65	22.11	54

phase was not successful in all cases. An iteration was labeled as a failure when the residual norm was not reduced by 8 orders of magnitude in fewer than 200 GMRES steps. We used GMRES with a restart of 100. This was done in order to measure the quality of the preconditioner alone as distinct from that of the accelerator.

The results are divided into two plots for better clarity. The left side of Figure 2 shows the number of successful solves requiring iteration counts of 80 or less and the right side those requiring more than 80 steps. For better visibility, the curves have been slightly shifted in order to avoid superposition. As can be observed, the strategy of Algorithm 4.3 handles harder problems quite well (right side of figure) while Algorithm 4.1 yields faster convergence on easier problems (left side of figure).

A standard preconditioner which is often used for comparison purposes is ILUTP [35]. When a good amount of fill-in is allowed, the preconditioner can be fairly robust. However, it does not perform well for matrices with highly irregular patterns and when not enough fill-in is allowed. For comparison purposes, we ran ILUTP-GMRES under similar conditions on the 58 problems. Specifically, we selected a drop tolerance of 0.01 and a fill-in limit of $3 * nnz(A)/n$, which means that each row of the factors L and U has at most 3 times the number of nonzero elements per row of A. To give an idea of the overall performance, we show in Table 2 the average fill-factor and average iteration count for each [illegible] for all methods, including ILUTP. The fill-factor reflects the memory requirement of the preconditioner and is defined as the ratio of the number of nonzero elements of the resulting preconditioner over that of the original matrix.

It is interesting to note that ILUTP required far more fill-in to solve the same number of problems (43) as the worst performer among the ARMS-C algorithms. The

average iteration counts are not indicative of robustness. Since the fill-in is rather high, the iteration counts for the easy problems are usually very small numbers for the (fewer) successful runs of ILUTP, which leads to a low average among the 43 successful cases. This is in contrast with the ARMS-C strategies which solve more of the harder problems but, as can be expected, at the cost of a larger number of GMRES steps on average.

ILUTP was able to solve some of the problems for which it failed in the above test. However, this was achieved at the cost of much more memory.

6.2. Effect of the filtering parameter τ_0. We vary the parameter τ_0 used in the preselection algorithm and compare the iteration number and the times to compute the factorization for a system associated with the matrix `twotone` from the University of Florida Sparse Matrix Collection [16]. This matrix is known to cause difficulties to standard ILUT and ILUTP preconditioners. The parameters used for the test are given in the following table.

		Drop tolerance				Fill-max			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
200	varies	0.001	0.0001	0.001	0.01	10	10	10	5

Algorithm 4.3 was used to obtain the P, Q permutations.

For this matrix the ARMS-C procedure requires unusually small values for the drop tolerances before the preconditioner starts yielding a converging iteration. On the other hand, and somewhat surprisingly, the resulting fill-factor is not too large, and convergence can be obtained for fill-factors below 2. This phenomenon is not observed too often and is likely related to the structure of the problem. ILUTP did not yield convergence for this problem, even for a fairly large fill-factor.

As τ_0 increases from 0.0 to 0.4, one can observe a steady improvement in the convergence of the algorithm while the fill-factor remains almost the same. Therefore a better quality factorization is being computed with about the same amount of fill. Then the iteration count sees a sizable decrease, reaching the smallest number of 27 when $\tau_0 = 0.7$. At this point the maximum number of levels, which has been set to 200, is reached and the fill-factor increases. Eventually, the iteration starts to deteriorate slowly, reaching 38 when $\tau_0 = 0.9$. For a small value of τ_0 more entries are initially preselected. For example, when $\tau_0 = 0$ all the largest entries in each row are candidates. They are then sorted using a measure which mixes diagonal dominance with the number of nonzero entries. The resulting algorithm is more tolerant of nondiagonal dominance. Selecting a larger τ_0 sets a stricter condition for accepting an entry as a candidate. Excluding more rows in this way results in more robust, often faster, convergence, but this increases the number of levels as can be seen in Table 3. In fact the number of levels shown in the table for $\tau_0 \geq 0.7$ is the limit allowed.

One may wonder why the iteration time is not proportional to the amount of fill of the factorization and the iteration count. For example, when between $\tau_0 = 0.8$ and $\tau_0 = 0.9$, the iteration time jumps 78% while the iteration count increases only 20%. This is likely due to the fact that the last Schur complement is much larger for the latter case.

6.3. Effect of the number of levels. One important consideration when using ARMS-C is the number of levels. At one extreme when only one level is used, ARMS-C will yield the ILUTP factorization. Indeed, the procedure considers the matrix at the top level to be the Schur complement matrix and uses ILUTP to factor it. Large Schur complements are likely to lead to expensive factorizations. The current

TABLE 3
Performance of ARMS-C for various values of τ_0 for the `twotone` *matrix.*

τ_0	filf	lev	iter	setup sec	iters sec
0.00	1.75	91	81	5.96e+01	6.84e+01
0.10	1.79	90	74	6.05e+01	6.24e+01
0.20	1.75	97	87	5.90e+01	7.45e+01
0.30	1.78	117	71	7.01e+01	6.19e+01
0.40	1.80	145	47	8.70e+01	4.29e+01
0.50	1.89	193	34	1.18e+02	3.60e+01
0.60	1.91	200	28	1.46e+02	3.41e+01
0.70	1.95	200	27	1.64e+02	4.14e+01
0.80	1.95	200	29	1.72e+02	5.45e+01
0.90	1.91	200	35	2.34e+02	9.73e+01

code uses as a parameter the maximum number of levels and the smallest allowable Schur complement, which is usually set to 100, i.e., as soon as a matrix A_{l+1} in (3) reaches a size of 100 or smaller the procedure stops generating more levels. In the next experiment we explore this with a goal of demonstrating the beneficial effect of a multilevel approach in contrast with one that is based on one or very few levels.

The matrix used in this test arises from the simulation of a two-dimensional flow in a driven cavity and is available from the Matrix Market[3] under the name `E40R0100` in the collection DRIVCAV. Its size is $n = 17,922$ and it has $nnz = 567,467$ nonzero entries. The Reynolds number for the example is 100. The parameters used for the experiment are listed below.

		Drop tolerance				Fill-max			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
varies	0.2	0.01	0.05	0.05	0.01	6	6	6	6

The stopping criterion is as before, namely the norm of the residual should be reduced by 8 orders of magnitude. The case $nlev_{max} = 0$ corresponds to ILUTP—with the drop tolerance 0.01 and a max fill-in of $6 \times nz/n$. Figure 3 shows the number of GMRES iterations as well as the resulting fill-factor when the number of levels varies from zero to 30. Note that for this case only 27 levels were required before the Schur complement matrix reached the smallest allowable size of 100, so the case $nlev_{max} = 30$ was never reached. The plot shows a fairly regular decrease in the fill-factor as the number of levels increases. At the same time, the iteration count increases and then levels off at around 75.

A different perspective on this experiment would be to vary the parameter τ_0 as was done in the previous section and let the number of levels be as large as required. With the set of parameters used in the previous experiment, the maximum number of levels for the matrix `E40R0100` was 27. In order to obtain larger numbers of levels, we tightened the drop tolerance slightly by halving the two drop tolerance values of 0.05 used for the G, W matrices and the Schur complement to 0.025. The parameter τ_0 is varied from 0.0 to 0.7 in increments of 0.05. The plot on the right of Figure 3 shows the result. Essentially, the fill-factor remains in a rather narrow band (1.93 2.23) while the iteration count changes, but not substantially. The additional axis shows the number of levels required for each case. These numbers are much bigger than the ones obtained with a drop tolerance of 0.05. This last experiment, as well as the one in section 6.2 and others not reported here, suggests that it is best not

[3] `http://math.nist.gov/MatrixMarket/`

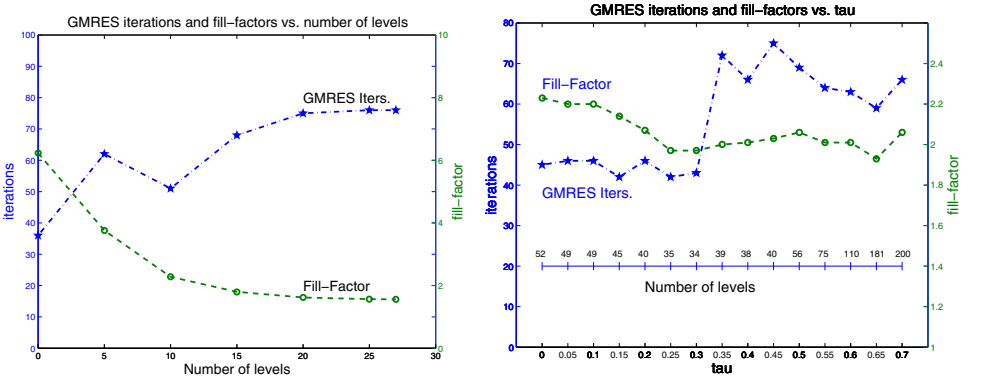


FIG. 3. *Iteration and fill-factor for the driven cavity problem, versus the number of levels (left side) and τ_0 parameter (right side). The right plot uses a slightly different set of parameters.*

to limit the number of levels to a too small number. Limiting the number of levels in this way may lead to a large last Schur complement matrix which is then poorly handled by the ILUTP approach.

6.4. Solution of KKT problems and a comparison. The ARMS-C procedure can be applied to solve saddle-point problems such as those arising from the incompressible Navier–Stokes equations or from Karush–Kuhn–Tucker (KKT) optimality conditions in nonlinear programming. In [28] Haws and Meyer focus on KKT problems and compare a few methods for solving them. Such problems are symmetric and highly indefinite, but one of the methods tested in [28] is to ignore symmetry and use ILUT as a preconditioner on a reordered system. The reordering used is MC64, the one-sided reordering developed in [23, 24, 33] whose goal is to put large entries on the diagonal. Two sets of problems are tested in [28]. We only consider the first set corresponding to the results reported in Table I of that paper.

TABLE 4
Results for solving five KKT problems with ARMS-C.

Matrix	n	nnz	nnz_{pre}	lev	Iters.	Setup sec.	Iter sec.
stiff4	8496	41318	73332	4	49	0.09	0.39
stiff5	33410	177256	341363	6	112	0.48	6.73
mass04	8496	56818	61799	2	7	0.07	0.04
mass05	33410	241012	259026	5	41	0.35	1.53
mass06	33794	257220	302009	6	20	0.47	0.73

Table 4 shows the same statistics as those of Table I in [28] plus the number of levels (lev) required for each case for a similar approach which ignores symmetry and solves the problems by an ARMS-C/GMRES combination. In the table, n and nnz are the size and number of nonzero entries of the matrix, nnz_{pre} is the number of nonzero entries required for the preconditioner, and the rest of the data is self-explanatory. The parameters used for the experiment are listed below.

		Drop tolerance				Fill-max			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
100	0.25	0.05	0.01	0.01	0.01	4	4	4	4

The right-hand side, initial guess, and convergence criterion are identical with those

of [28]. As with earlier tests we do not attempt in any way to tune the parameters for each separate case. These were selected to yield a comparable number of nonzero elements for the preconditioners as in [28]. In fact, all five problems tested here are relatively easy to solve with the ARMS-C procedure in the sense that almost any parameter selection would yield convergence. This is not the case for some matrices (see, e.g., section 6.2).

TABLE 5
Results for solving the five KKT systems of Table 4 *with ILUT-MC*64*, as reported in* [28]. *Note that the times are not directly comparable with those of Table* 4.

Matrix	nnz_{pre}	Iters.	Setup sec.	Iter. sec
stiff4	77 118	28	0.129	0.786
stiff5	341417	135	0.575	15.48
mass04	68875	3	0.126	0.186
mass05	323299	32	0.705	3.902
mass06	394821	30	0.806	4.038

As a comparison we reproduce in Table 5 the statistics for the MC64-ILUT combination from [28]. As can be seen for similar fill-in, the number of iterations obtained in each case are comparable. Note that the accelerator used in [28] is BCGSTAB instead of GMRES. Execution times are not easily comparable as they depend on many factors in addition to the raw speed of the processors used. The processor used in [28] is an AMD Duron 800 MHz processor, which is slower than the Xeon 1.7 GHz on which this experiment has been conducted. On the other hand the codes in [28] were optimized with the use of ATLAS BLAS. In addition, the algorithms in [28] are coded in FORTRAN whereas the ARMS-C code is coded mostly in C. The ARMS-C codes have not been optimized in any way. All that can be said is that by taking into account the speeds of the processors, the times obtained in these experiments are within a comparable range.

Note that the MC64-ILUT procedure was not the best overall method reported in [28]. However, the point of this experiment is not to show that the method discussed in this paper can compete with specialized solvers tailored to a given application or structure. It is only to show that the method can perform reasonably well as a general-purpose solver, in that it can handle KKT problems as easily as a problem arising from the discretization of PDEs.

6.5. Tests with larger matrices. Apart from the test in section 6.2 with the matrix `twotone`, the linear systems of the earlier numerical tests are all relatively small. In this section we report on some tests performed with larger, more realistic matrices arising from applications which are known to cause difficulties to iterative solvers. We selected 10 matrices, most of which have already been used as test problems in the literature. The first 5 matrices were taken from [38]. They are the largest linear systems in size taken from [38] if the matrix `pre2` therein is ignored. For the system `pre2` no method among those reported in [38] worked, and a similar conclusion was reached with ARMS-C. These 5 matrices, which originate from device simulation, are available from the `Schenk` sets of the University of Florida collection [16]. Along with these matrices, we also selected the two largest matrices from [2], namely the matrix `circ2a` and the matrix `scircuit`. The matrix `scircuit` can also be obtained from the set `Hamm` of the University of Florida collection, while the larger matrix `circ2a` was sent to us by Achim Basermann [2]. Finally, we added the 2 largest matrices (`wang1, wang2`) from the `Wang` set of the University of Florida collection, and

TABLE 6
General information on 10 *test matrices.*

Matrix	Order	Nonzeros	Application (Origin)
barrier2-9	115,625	3,897,557	Semiconductor Device (UFL, Schenk set)
matrix_9	103,430	2,121,550	Semiconductor Device (UFL, Schenk set)
matrix-new_3	125,329	2,678,750	Semiconductor Device (UFL, Schenk set)
ohne2	181,343	11,063,545	Semiconductor Device (UFL, Schenk set)
para-4	153,226	5,326,228	Semiconductor Device (UFL, Schenk set)
cir2a	482,969	3,912,413	circuit simulation
scircuit	170998	958936	Digital circuit simulation (UFL, Hamm set)
circuit_4	80209	307604	Circuit simulation (UFL, Bomhof set)
wang3.rua	26064	177168	Device simulation (UFL, Wang set)
wang4.rua	26068	177196	Device simulation (UFL, Wang set)

the largest matrix (`circuit_4`) from the set `Bomhoff` of the same collection. These examples are all from circuit simulation. Some general information on all 10 matrices is shown in Table 6.

We tested ARMS-C, based on the ddPQ ordering given by Algorithm 4.3 on all 10 matrices with the following parameter settings.

		Drop tolerance				Fill-max			
$nlev_{max}$	tol_{DD}	LU-B	GW	S	LU-S	LU-B	GW	S	LU-S
40	0.1	0.01	0.01	0.01	1.e-05	3	3	3	20

The parameters for the ILUTP preconditioner for the last level essentially lead to a direct solve. Another parameter not shown here is the cut-off dimension for the last level, which is set to 300. No further levels are sought when the dimension of the Schur complement is ≤ 300 or when the maximum number of levels allowed (40) is reached.

We ran two tests with the GMRES accelerator, one with a restart dimension of 60, the other with a restart dimension of 100. Note that memory is allocated dynamically for the basis vectors required by GMRES(k). In both cases, the iteration was stopped whenever either a maximum number of iterations of 300 was reached or the initial residual was reduced by 10^{-8}. The times are in seconds on the 1.7 GHz Xeon processor mentioned earlier. The iteration times are repeated for the case when GMRES(60) and GMRES(100) take the same number of iterations (these correspond to the exact same operations and should be the same under ideal conditions). Results are shown in Table 7.

TABLE 7
Solution of the 10 *systems in Table* 6 *with* $GMRES(60)$ *and* $GMRES(100)$ *preconditioned with ARMS-C.*

	Fill		Set-up	GMRES(60)		GMRES(100)	
Matrix	factor	Levels	time	Its.	Time	Its.	Time
barrier2-9	0.62	5	4.01e+00	113	3.29e+01	93	3.02e+01
matrix-new_3	0.89	8	7.53e+00	40	1.02e+01	40	1.00e+01
matrix_9	1.77	9	5.53e+00	160	4.94e+01	82	2.70e+01
ohne2	0.62	6	4.34e+01	99	6.35e+01	80	5.43e+01
para-4	0.62	5	5.70e+00	49	1.94e+01	49	1.93e+01
wang3	2.33	4	8.90e-01	45	2.09e+00	45	1.95e+00
wang4	1.86	4	5.10e-01	31	1.25e+00	31	1.20e+00
scircuit	0.90	4	1.86e+00	Fail	7.08e+01	Fail	8.80e+01
circuit_4	0.75	2	1.60e+00	199	1.69e+01	96	1.07e+01
circ2a	0.76	3	2.19e+02	18	1.08e+01	18	1.03e+01

TABLE 8
Solution of the system `scircuit` *when no scaling is used, using two different sets of parameters.*

	Fill factor	Levels	Set-up time	GMRES(60) Its.	GMRES(60) Time	GMRES(100) Its.	GMRES(100) Time
Same parameters	0.89	4	1.81e+00	400	9.13e+01	297	8.79e+01
Drop-tol = 0.001	1.00	5	1.89e+00	98	2.23e+01	82	2.27e+01

Of all the matrices tested, only `scircuit` failed to reduce the residual by 8 orders of magnitude in 300 steps with the set of parameters selected above. In 300 steps the residual norm was reduced by $\approx 1.8 \times 10^{-07}$ by GMRES(100) and by 3.13×10^{-7} by GMRES(60). This particular matrix is a good representative of those situations mentioned earlier in which row/column scaling does not necessarily help. If the exact same parameters are used but no row or column scaling is performed (set by parameters in ARMS-C), then, as is shown in Table 8, GMRES(100) does succeed in reducing the residual norm by 8 orders of magnitude in 297 steps, while GMRES(60) also achieves the same result in 400 steps. Table 8 also shows an additional test for the same matrix when no scaling is used. Here the drop tolerance parameters labeled "LU-B," "GW," and "S," which were set to 0.01 in this experiment, are changed to 0.001. The resulting convergence is much faster. It should be noted that convergence is also achieved with these values of the drop tolerance for the situation when row/column scaling is used, although it is much slower.

Observe that the fill-factors, the memory required for the preconditioner normalized by that required for the original matrix A, are generally quite modest. It is less than 1 in 7 of the 10 cases. Not seen in the reported results is a peculiar behavior of the solver with the matrix `circuit_4` as well as with the matrix `circ2a`. In contrast with most other systems, results with `circuit_4` were exceedingly sensitive to parameters and even to the initial guess (which is always a random vector).

Regarding the matrix `circ2a`, it can be observed in the table that the time to compute the preconditioner appears to be abnormally high relative to the other test cases. This matrix is the largest in size among the 10 tested, but it is not the largest in terms of the number of nonzero entries. The first reaction to this might be that the algorithms have some hidden cost associated with the size n rather than the nonzero entries. A test with the matrix `pre2` mentioned earlier disproved this. This matrix is of size $n = 659,033$ and has $nz = 5,959,282$ nonzero entries. Yet the same parameters produced the factorization in almost an order of magnitude less time (24.2 seconds to be exact) on the same platform. The fill-factor was 0.99, which was even higher than that for `circ2a`. We examined the sizes and numbers of nonzero entries of the matrices obtained at each level. The method stopped prematurely with a C block of size zero yielding the previous level (third one, in this case) as the last level. This level had a size of 4,679, which is somewhat large. Indeed, no fill-minimizing ordering is performed prior to factoring the last Schur complement, because in most cases the last Schur complement is a small matrix of size a few hundreds at most. However, further examination revealed that, quite remarkably, this last Schur complement is a diagonal matrix. This explains why the algorithm stopped prematurely. In the end, we reached the only plausible explanation that some intermediate matrices that were generated were quite dense before they were sparsified by dropping. Selecting a different filter parameter τ_0 can force a larger number of levels and reduce the set-up substantially in this case. For example, with $\tau_0 = 0.7$, 18 levels were used, with a fill-factor of 0.77, quite close to the previous case. The set-up time was reduced to

43 seconds. The number of GMRES steps was also reduced from 18 to 12 (but the iteration time increased slightly to 12.1 seconds due to the larger number of levels).

6.6. Available software and other tests. The algorithms described in this paper are part of a recent package, called ILUPACK, developed primarily by Matthias Bollhöfer [11]. The package is available from Bollhöfer's web-site; see [12]. It offers a large selection of preconditioners, many of which utilize the inverse-based dropping technique mentioned in the introduction; see also [9]. Along with software, the site offers a repository of statistics on a sizable number of numerical experiments with various methods. These experiments show, in particular, a number of comparisons with the similar MC64 strategy.

7. Conclusion. We conclude with a few remarks and observations regarding the performance of ARMS-C and on issues remaining to be examined. Our first observation is that the method seems to perform well as a general purpose alternative to a direct solver. Though the algorithms do not compete with the robustness of direct solvers, they can often tackle problems that were inaccessible to iterative solvers so far at a small fraction of the memory and computational costs required by direct solvers. Note also that the methods perform in a similar way for problems with a very poor structure as for systems which arise from discretized PDEs. There are, however, a number of questions that remain to be answered and issues to be examined for a better understanding of the general approach proposed in this paper.

One of the primary concerns with a strategy such as the one described in this paper is the large number of tunable parameters available. With such variety it may indeed be difficult to find the best fit for a given linear system. However, in a software package, one may simply fix most of these parameters without causing serious harm to performance. The primary candidates for this "hard-wiring" are the maximum number of levels allowed (along with the minimum size of the Schur complement before switching to ILUTP) and the diagonal dominance filtration parameter τ_0. Indeed, when the better matching algorithms are used (Algorithm 4.3 or 4.2) performance does not vary too much with these parameters for [illegible] problems. The other parameters left are those that control fill-in, i.e., the accuracy of the factorization. Although our research code provides separate fill-in parameters for various parts of the factorization at a given level, these can be taken to be the same as was done in most of our experiments. The drop tolerance and fill-in parameters can be fixed to provide a very accurate factorization for the last level Schur complement matrix. In the end all that remains to select are the drop tolerances and the max fill-in allowed for the interlevel matrices (those for LU-B and GW in the various tables). This would not be too different from the parameters required by ILUT [35].

We observed that fill-in does not necessarily improve the quality of the factorization. In fact the contrary is often seen. While the same observation can be made for ILUT, we note that it is not as common. This behavior underscores the difficulty in predicting the effect of dropping small terms in the factors on the conditioning of the preconditioned system. The dropping strategies which were tested utilize a simple threshold criterion, but it is clear that one based on estimating the condition number of the factors [9] would be ideal in this situation.

A diagonal dominance criterion was selected as a means of building a good B block because it is easy to apply and ensures large diagonal entries in a relative sense to the 1-norm of the row. However, one can ask if it is possible to find a more elaborate criterion—for example, one that takes into account the "stability" of the resulting factors. A related subquestion is the problem of selecting the parameter

τ. The choice made in this paper uses a relative measure which keeps the best rows according to a rule based on a parameter input by the user. There remains to find a rigorous way to select the parameter τ.

Another important issue that deserves further investigation is that of scaling. As was reported, the performance of the preconditioners is significantly affected by scaling. Yet it still remains difficult to devise scaling strategies whose effects on the preconditioner are well understood and predictable.

Finally, our current codes do not attempt to reduce fill-in by means other than dropping. A simple strategy which can be employed is to use fill-reducing (symmetric) orderings on the B matrices after they have been generated.

Acknowledgments. The code ARMS-C discussed in the experiments has been adapted from ARMS [36], a C code which was developed jointly with Brian Suchomel. I am grateful to Matthias Bollhöfer for numerous insightful discussions on the topic of this paper. The numerical experiments would not have been possible without the availability of a sizable collection of test matrices. I would like to thank John Haws for providing the matrices used in section 6.4, Achim Basermann for sending us his circuit simulation matrices, and Olaf Schenk for making available the large device and circuit matrices discussed in section 6.5. Special thanks to Tim Davis for making these and other linear systems available in his excellent repository of test matrices. Finally, I am grateful to the referees for their numerous suggestions which helped improve the quality of this paper.

REFERENCES

[1] R. E. Bank and C. Wagner, *Multilevel ILU decomposition*, Numer. Math., 82 (1999), pp. 543–576.

[2] A. Basermann, U. Jaekel, and K. Hachiya, *Preconditioning parallel sparse iterative solvers for circuit simulation*, in SIAM Conference on Applied Linear Algebra, July 2003, Williamsburg, VA, CP–2.

[3] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmtric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[4] M. Benzi, R. Kouhia, and M. Tůma, *Stabilized and block approximate inverse preconditioners for problems in solid and structural mechanics*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 6533–6554.

[5] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[6] M. Benzi and M. Tůma, *Orderings for factorized sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1851–1868.

[7] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[8] M. Benzi and M. Tůma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl., 10 (2003), pp. 385–400.

[9] M. Bollhöfer, *A robust ILU with pivoting based on monitoring the growth of the inverse factors*, Linear Algebra Appl., 338 (2001), pp. 201–213.

[10] M. Bollhöfer and Y. Saad, *On the relations between ILUs and factored approximate inverses*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 219–237.

[11] M. Bollhöfer and Y. Saad, *Multilevel Preconditioners Constructed from Inverse-based ILUs*, Technical Report umsi-2004-75, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 2004.

[12] M. Bollhöfer and Y. Saad, *ILUPACK—Preconditioning Software Package*, release v1.0, http://www.tu-berlin.de/ilupack/ (14 May 2004).

[13] C. W. Bomhof and H. A. Van der Vorst, *A parallel linear system solver for circuit simulation problems*, Numer. Linear Algebra Appl., 7 (2000), pp. 649–665.

[14] E. F. F. Botta and F. W. Wubs, *Matrix renumbering ILU: An effective algebraic multilevel ILU preconditioner for sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1007–1026.

[15] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.
[16] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse (1997).
[17] T. A. Davis, *A Parallel Algorithm for Sparse Unsymmetric LU Factorizations*, Ph.D. thesis, University of Illinois at Urbana Champaign, Urbana, IL., 1989.
[18] I. S. Duff, *A survey of sparse matrix research*, in Proceedings of the IEEE 65, New York, 1977, pp. 500–535.
[19] I. S. Duff, *Algorithm* 575: *Permutations for a zero-free diagonal*, ACM Trans. Math. Software, 7 (1981), pp. 387–390.
[20] I. S. Duff, *On algorithms for obtaining a maximal transversal*, ACM Trans. Math. Software, 7 (1981), pp. 315–330.
[21] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, 1986.
[22] I. S. Duff, R. G. Grimes, and J. G. Lewis, *User's Guide for the Harwell-Boeing Sparse Matrix Collection*, Technical Report TR/PA/92/86, CERFACS, Toulouse, France, 1992.
[23] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.
[24] I. S. Duff and J. Koster, *On algorithms for permuting large entries to the diagonal of a sparse matrix*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 973–996.
[25] J. A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[26] G. H. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[27] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.
[28] J. C. Haws and C. D. Meyer, *Preconditioning KKT Systems*, Technical Report M&CT-TECH-02-002, Boeing, Seattle, WA, 2002.
[29] L. Yu. Kolotilina and A. Y. Yeremin, *On a family of two-level preconditionings of the incomplete block factorization type*, Soviet J. Numer. Anal. Math. Modeling, 1 (1986), pp. 293–320.
[30] R. Leuze, *Independent set orderings for parallel matrix factorizations by Gaussian elimination*, Parallel Comput., 10 (1989), pp. 177–191.
[31] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Sci. Comput., 25 (2003), pp. 716–728.
[32] M. Luby, *A simple parallel algorithm for the maximal independent set problem*, SIAM J. Comput., 15 (1986), pp. 1036–1053.
[33] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.
[34] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
[35] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelpha, PA, 2003.
[36] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.
[37] Y. Saad and J. Zhang, *BILUTM: A domain-based multilevel block ILUT preconditioner for general sparse matrices*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 279–299.
[38] O. Schenk, S. Röllin, and A. Gupta, *The effects of nonsymmetric matrix permutations and scalings in semiconductor device and circuit simulation*, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, 23 (2004), pp. 400–411.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1058–1070
© 2005 Society for Industrial and Applied Mathematics

MULTIDIMENSIONAL INTEGRAL INVERSION, WITH APPLICATIONS IN SHAPE RECONSTRUCTION*

ANNIE CUYT†, GENE GOLUB‡, PEYMAN MILANFAR§, AND BRIGITTE VERDONK†

Abstract. In shape reconstruction, the celebrated Fourier slice theorem plays an essential role. It allows one to reconstruct the shape of a quite general object from the knowledge of its Radon transform [S. Helgason, *The Radon Transform*, Birkhäuser Boston, Boston, 1980]—in other words from the knowledge of projections of the object. In case the object is a polygon [G. H. Golub, P. Milanfar, and J. Varah, *SIAM J. Sci. Comput.*, 21 (1999), pp. 1222–1243], or when it defines a quadrature domain in the complex plane [B. Gustafsson, C. He, P. Milanfar, and M. Putinar, *Inverse Problems*, 16 (2000), pp. 1053–1070], its shape can also be reconstructed from the knowledge of its moments. Essential tools in the solution of the latter inverse problem are quadrature rules and formal orthogonal polynomials.

In this paper we show how shape reconstruction from the knowledge of moments can also be realized in the case of general compact objects, not only in two but also in higher dimensions. To this end we use a less-known homogeneous Padé slice property. Again integral transforms—in our case the multivariate Stieltjes transform and univariate Markov transform—formal orthogonal polynomials in the form of Padé denominators, and multidimensional integration formulas or cubature rules play an essential role.

We emphasize that the new technique is applicable in all higher dimensions and illustrate it through the reconstruction of several two- and three-dimensional objects.

Key words. shape, multidimensional, inverse problem, moment problem

AMS subject classifications. 65D32, 65F20, 41A21, 44A60

DOI. 10.1137/030601703

1. Problem statement.

The problem of reconstructing a function and/or its domain given its moments is encountered in many areas. Several applications from diverse areas such as probability and statistics [10], signal processing [18], computed tomography [16, 17], and inverse potential theory [4, 19] (magnetic and gravitational anomaly detection) can be cited, to name just a few. We can expound on some of these applications in a bit more detail. Consider the following diverse set of examples:

- A region of the plane can be regarded as the domain of a probability density function. In this case, the problem is that of reconstructing this density function and/or approximating its domain from measurements of its moments [10].
- Tomographic (line integral) measurements of a body can be converted into moments from which an approximation to its density and boundary can be extracted [17].
- Measurements of exterior gravitational field induced by a body of uniform mass can be turned into moment measurement, from which the shape of the region may be reconstructed [19].
- Measurements of an exterior magnetic field induced by a body of uniform

*Received by the editors November 5, 2003; accepted for publication (in revised form) March 29, 2005; published electronically December 30, 2005.
http://www.siam.org/journals/sisc/27-3/60170.html
†Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, Antwerp, B2020, Belgium (annie.cuytua.ac.be, brigitte.verdonk@ua.ac.be).
‡Department of Computer Science, Stanford University, Stanford, CA 94305 (golub@sccm.stanford.edu).
§Electrical Engineering Department, University of California at Santa Cruz, Santa Cruz, CA 95064 (milanfar@ee.ucsc.edu).

magnetization can yield measurement of the moments of the region from which the shape of the region may be determined [19].

- Measurements of thermal radiation made outside a uniformly hot region can yield moment information, which can subsequently be inverted to give the shape of the region [19].

In fact, aside from the general case where the density inside the body may not be uniform, the set of inverse problems for uniform density regions related to general elliptical equations can all be cast as moment problems which fall within the scope of application of the results of this paper.

Although the reconstruction of a shape from its Radon transform is well understood, the reconstruction of a shape from its moments is a problem that has only partially been solved. For instance, when the object is a polygon [11], or when it defines a quadrature domain in the complex plane [12], it has been proved that its shape can be exactly reconstructed from the knowledge of its moments. Both results deal with particular two-dimensional shapes. For general n-dimensional shapes no inversion algorithm departing from the moments is known. In order to explain the type of result we are looking for, we briefly repeat the inversion formula based on a shape's projections provided by the Radon transform.

The Radon transform $R_{\vec{\xi}}(f)$ of a square-integrable n-variate function $f(\vec{x})$ with $\vec{x} = (x_1, \ldots, x_n)$ is defined as (for ease of notation we drop the dependence of f in the notation)

$$R_{\vec{\xi}}(u) = \int_{\mathbb{R}^n} f(\vec{x})\ \delta(\vec{\xi}\vec{x} - u)\ d\vec{x}$$

with $||\vec{\xi}|| = 1$ and $\vec{\xi} \cdot \vec{x} = u$ an $(n-1)$-dimensional manifold orthogonal to $\vec{\xi}$. When $n = 2$, $\vec{\xi}$ is fully determined by an angle θ and is given by

$$R_\theta(u) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(t,s)\, \delta(t\cos\theta + s\sin\theta - u)\, dt\, ds.$$

For $n = 3$, $\vec{\xi}$ is determined by angles θ and ϕ and

$$R_{\theta,\phi}(u) = \int_{\mathbb{R}^3} f(t,s,v)\, \delta(t\cos\phi\cos\theta + s\cos\phi\sin\theta + v\sin\phi - u)\, dt\, ds\, dv.$$

Making use of the celebrated Fourier slice theorem, one obtains, for instance, that the one-dimensional Fourier transform of $R_\theta(u)$,

$$\mathcal{F}_1(R_\theta)(z) = \int_{-\infty}^{+\infty} R_\theta(u) \exp(-2\pi \mathrm{i} z u)\, du,$$

equals the two-dimensional Fourier transform of the function f restricted to the straight line $(z\cos\theta, z\sin\theta)$:

$$\begin{aligned} \mathcal{F}_2(f)(z\cos\theta, z\sin\theta) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(t,s) \exp\left(-2\pi \mathrm{i} z(t\cos\theta + s\sin\theta)\right)\, dt\ ds \\ &= \mathcal{F}_1(R_\theta)(z). \end{aligned} \tag{1}$$

When $f(t,s)$ is the characteristic function of a compact set A in the complex plane, then (1) allows one to reconstruct A, departing from the Radon transform $R_\theta(u)$, by

taking the inverse two-dimensional Fourier transform of $\mathcal{F}_1(R_\theta)$. In higher dimensions the procedure is completely analogous [14].

Our aim is to establish a similar type of relationship, making use of moment information instead of projections. To this end we need to introduce a few tools.

2. Univariate Markov transform and Padé approximant. A Markov function $g(z)$ is defined to be a function with an integral representation of the form

(2)
$$g(z) = \int_a^b \frac{f(u)}{1+zu}\, du, \quad -\infty < a \le 0 \le b < +\infty, \quad z \notin]-\infty, -1/b] \cup [-1/a, +\infty[,$$

where $f(u)$ is nontrivial and positive and the moments

$$c_i = \int_a^b u^i f(u)\, du, \qquad i = 0, 1, \ldots, \tag{3}$$

are finite. The function g is called the Markov transform of f and is also denoted by $g = \mathcal{M}_1(f)$. A Markov series is defined to be a series of the form

$$\sum_{i=0}^{\infty} (-1)^i c_i z^i \tag{4}$$

which is derived by a formal expansion of (2). It is well known that the one-dimensional Markov moment problem is determinate.

Given a series of the form (4), one can construct Padé approximants of this series as follows. With the moments c_i introduced in (3), one computes coefficients $a_0, \ldots, a_{m+k}$ and $b_0, \ldots, b_m$ such that for

$$p_{m+k,m}(z) = \sum_{i=0}^{m+k} a_i z^i,$$
$$q_{m+k,m}(z) = \sum_{i=0}^{m} b_i z^i,$$

the series expansion of $(gq_{m+k,m} - p_{m+k,m})(z)$ satisfies

$$\sum_{i=0}^{\infty} d_i z^i = \left(\sum_{i=0}^{\infty} (-1)^i c_i z^i \right) q_{m+k,m}(z) - p_{m+k,m}(z) = O(z^{2m+k+1}). \tag{5}$$

In other words, the $2m+k+2$ coefficients $a_0, \ldots, a_{m+k}$ and $b_0, \ldots, b_m$ are determined from the $2m+k+1$ conditions $d_0 = 0, \ldots, d_{2m+k} = 0$ and an additional normalization condition. The irreducible form of $p_{m+k,m}(z)/q_{m+k,m}(z)$ is denoted by $r_{m+k,m}(z)$ and is called the $(m+k, m)$ Padé approximant. It is usually normalized by putting the constant term in the denominator equal to 1.

The following property plays a crucial role in our novel shape reconstruction technique.

THEOREM 1 (see [1, p. 228]). [illegible] (2) [illegible] $\{r_{m+k,m}(z)\}_{m \in \mathbb{N}}$ [illegible] $k \ge -1$ [illegible] (2) [illegible] $z \notin]-\infty, -1/b] \cup [-1/a, +\infty[$ [illegible]

$$\limsup_{m \to \infty} |g(z) - r_{m+k,m}(z)|^{1/m} \le \left| \frac{\sqrt{1/z+b} - \sqrt{1/z+a}}{\sqrt{1/z+b} + \sqrt{1/z+a}} \right|.$$

3. Multivariate Stieltjes transform and homogeneous Padé approximant. Padé approximants have been generalized to higher dimensions by several authors in different ways. For an overview and comparison of these definitions the reader is referred to [8]. For our purpose the definition given in [7, 6] is most useful. Without loss of generality, we repeat it only for bivariate functions, but it can be defined in any number of variables.

A bivariate Stieltjes function $g(v,w)$ is defined by the integral representation

$$g(v,w) = \int_0^\infty \int_0^\infty \frac{f(t,s)}{1+(vt+ws)}\, dt\, ds \tag{6}$$

with finite real-valued moments

$$c_{ij} = \int_0^\infty \int_0^\infty t^i s^j f(t,s)\, dt\, ds.$$

A formal expansion of (6) provides a bivariate Stieltjes series

$$\sum_{i,j=0}^{\infty} \binom{i+j}{i} (-1)^{i+j} c_{ij} v^i w^j. \tag{7}$$

The function g is also called the bivariate Stieltjes transform of f and is denoted by $g = \mathcal{S}_2(f)$.

Given the moments c_{ij}, one can compute the $(m+k,m)$ homogeneous bivariate Padé approximant of (7) as follows. First we introduce the homogeneous expressions

$$\begin{aligned} A_\ell(v,w) &= \sum_{i+j=\ell} a_{ij} v^i w^j, \\ B_\ell(v,w) &= \sum_{i+j=\ell} b_{ij} v^i w^j \end{aligned}$$

to define the polynomials

$$\begin{aligned} p_{m+k,m}(v,w) &= \sum_{\ell=(m+k)m}^{(m+k)(m+1)} A_\ell(v,w), \\ q_{m+k,m}(v,w) &= \sum_{\ell=(m+k)m}^{(m+k+1)m} B_\ell(v,w). \end{aligned}$$

Second we write down the homogeneous accuracy-through-order conditions

$$\begin{aligned} C_\ell(v,w) &= \sum_{i+j=\ell} \binom{\ell}{i} c_{ij} v^i w^j, \\ \sum_{i,j=0}^{\infty} d_{ij} v^i w^j &= \left(\sum_{\ell=0}^{\infty} (-1)^\ell C_\ell(v,w) \right) q_{m+k,m}(v,w) - p_{m+k,m}(v,w) \\ &= O\left(v^i w^j, i+j \geq (m+k+2)m + k + 1\right). \end{aligned} \tag{8}$$

It has been shown [7, pp. 60–61] that a nontrivial solution of these conditions can always be computed. Moreover, all solutions $p_{m+k,m}(v,w)/q_{m+k,m}(v,w)$ deliver the

same unique irreducible form $r_{m+k,m}(v,w)$, which is called the homogeneous Padé approximant of (7). A proper normalization of $r_{m+k,m}(v,w)$ can still be chosen but differs most of the time from the univariate normalization $q_{m+k,m}(0)=1$ since the denominator of $r_{m+k,m}(v,w)$ need not start with a constant term. It starts with a homogeneous expression in v and w of as low degree as possible. This homogeneous generalization of the Padé approximant is the only one to satisfy the following powerful slice theorem. Although it was pointed out soon after the introduction of homogeneous Padé approximants, its full impact was only understood recently [9, 3]. For the sake of the reader we also repeat the short proof.

Let us define the slice function:

$$g_\theta(z) = g(z\cos\theta, z\sin\theta), \quad -\pi/2 < \theta \le \pi/2, \tag{9}$$
$$= \int_0^\infty \int_0^\infty \frac{f(t,s)}{1+(t\cos\theta + s\sin\theta)z}\, dt\, ds.$$

We denote the univariate $(m+k,m)$ Padé approximant of $g_\theta(z)$ as defined in (5) by $r^{(g_\theta)}_{m+k,m}(z)$.

THEOREM 2 (see [15, 5]). [illegible] $r_{m+k,m}(v,w)$ [illegible] $g(v,w)$ [illegible]

$$r_{m+k,m}(z\cos\theta, z\sin\theta) = r^{(g_\theta)}_{m+k,m}(z), \qquad -\pi/2 < \theta \le \pi/2.$$

[illegible]. The univariate rational function of the variable z, parameterized by θ,

$$\begin{aligned} r_{m+k,m}(z\cos\theta, z\sin\theta) &= \frac{\sum_{\ell=(m+k)m}^{(m+k)(m+1)} \left(\sum_{i+j=\ell} a_{ij}\cos^i\theta\sin^j\theta\right) z^\ell}{\sum_{\ell=(m+k+1)m}^{(m+k+1)m} \left(\sum_{i+j=\ell} b_{ij}\cos^i\theta\sin^j\theta\right) z^\ell} \\ &= \frac{\sum_{\ell=0}^{m+k} \left(\sum_{i+j=(m+k)m+\ell} a_{ij}\cos^i\theta\sin^j\theta\right) z^\ell}{\sum_{\ell=0}^{m} \left(\sum_{i+j=(m+k)m+\ell} b_{ij}\cos^i\theta\sin^j\theta\right) z^\ell} \\ &=: \frac{p^{(\theta)}_{m+k,m}(z)}{q^{(\theta)}_{m+k,m}(z)}, \end{aligned}$$

satisfies

$$\left(\sum_{\ell=0}^{\infty} (-1)^\ell C_\ell(\cos\theta,\sin\theta) z^\ell\right) q^{(\theta)}_{m+k,m}(z) - p^{(\theta)}_{m+k,m}(z)$$
$$= g_\theta(z) q^{(\theta)}_{m+k,m}(z) - p^{(\theta)}_{m+k,m}(z) = O(z^{2m+k+1})$$

with d_{2m+k+1} in (5) being a homogeneous expression in $\cos\theta$ and $\sin\theta$. Because of the unicity of the Padé approximant, $r^{(g_\theta)}_{m+k,m}(z)$ must equal the irreducible form of

$$\frac{p^{(\theta)}_{m+k,m}(z)}{q^{(\theta)}_{m+k,m}(z)}, \tag{10}$$

which completes the proof. □

In other words, restricting the homogeneous Padé approximant to the slice

$$S_\theta = \{(z\cos\theta, z\sin\theta) \mid z \in \mathbb{R}\} \tag{11}$$

is equivalent to computing the univariate Padé approximant of the slice function $g_\theta(z)$. After analyzing the behavior of the homogeneous Padé approximant on the slices S_θ, let us have another look at the slice function $g_\theta(z)$ itself.

Let the square-integrable function $f(t,s)$ be defined in a compact region A of the first quadrant $t \geq 0,\ s \geq 0$ of the plane. According to a fundamental property of the Radon transform $R_\theta(u)$ of $f(t,s)$ [14], the following relation holds for any square-integrable function $F(u)$:

$$\int_{-\infty}^{+\infty} R_\theta(u)F(u)\,du = \int_0^\infty \int_0^\infty f(t,s)F(t\cos\theta + s\sin\theta)\,dt\,ds. \tag{12}$$

If we take $F(u) = 1/(1+zu)$, then

$$\int_{-\infty}^{+\infty} \frac{R_\theta(u)}{1+zu}\,du = \int_0^\infty \int_0^\infty \frac{f(t,s)}{1+(t\cos\theta + s\sin\theta)z}\,dt\,ds = g_\theta(z). \tag{13}$$

Consequently, if $f(t,s)$ is zero outside a compact subset A of the first quadrant, then $g_\theta(z)$ is a Markov function, because $R_\theta(u)$ is zero outside a compact support. In addition Theorem 1 applies.

4. Connection and new results. Making use of the homogeneous Padé slice property and the fact that the slice function $g_\theta(z)$ is a Markov function with f in (2) equal to the Radon transform of $f(t,s)$, it is now easy to obtain the following result for the bivariate Stieltjes transform $g(v,w)$ defined by (6).

THEOREM 3. [illegible] $f(t,s)$ [illegible] (6) [illegible] $A \subset \{(t,s) : 0 \leq t^2+s^2 \leq 1\} \cap \{(t,s) : t \geq 0, s \geq 0\}$ [illegible] S_θ [illegible] (11) [illegible] $k \geq -1$ [illegible] $-\pi/2 < \theta \leq \pi/2$ [illegible] $\{r_{m+k,m}(z\cos\theta, z\sin\theta)\}_{m\in\mathbb{N}}$ [illegible] $g(z\cos\theta, z\sin\theta)$ [illegible] $|z| < 1$ [illegible] (6) [illegible]

$$\limsup_{m\to\infty} |g(z\cos\theta, z\sin\theta) - r_{m+k,m}(z\cos\theta, z\sin\theta)|^{1/m} \leq \left|\frac{\sqrt{1/z+1}-\sqrt{1/z-1}}{\sqrt{1/z+1}+\sqrt{1/z-1}}\right|. \tag{14}$$

[illegible]. Since $f(t,s)$ is zero outside A and by means of the celebrated Fourier slice formula (12), we know that

$$\begin{aligned}\mathcal{S}_2(f)(z\cos\theta, z\sin\theta) = g(z\cos\theta, z\sin\theta) &= \int\int_A \frac{f(t,s)}{1+(t\cos\theta+s\sin\theta)z}\,dt\,ds \\ &= \int_{a(\theta)}^{b(\theta)} \frac{R_\theta(u)}{1+zu}\,du = \mathcal{M}_1(R_\theta)(z)\end{aligned}$$

with $-1 \leq a(\theta) \leq b(\theta) \leq 1$. Combining Theorems 1 and 2 delivers

$$\int_{a(\theta)}^{b(\theta)} \frac{R_\theta(u)}{1+zu}\,du = \lim_{m\to\infty} r^{(g_\theta)}_{m+k,m}(z) = \lim_{m\to\infty} r_{m+k,m}(z\cos\theta, z\sin\theta)$$

for $-1/b(\theta) < z < -1/a(\theta)$ with $|1/b(\theta)| \geq 1$ and $|1/a(\theta)| \geq 1$. The rate of convergence easily follows from Theorem 2. □

When the compact support A intersects all four quadrants, the theorem still holds. The current formulation is just one way of scaling the problem, without loss of generality.

To the authors' knowledge, at the time of this writing, no inverse Stieltjes transform exists for the transform defined by (6). Hence the function $f(t,s)$ must be solved for from the relationship

$$\mathcal{S}_2(f)(z\cos\theta, z\sin\theta) = \mathcal{M}_1(R_\theta)(z) = \lim_{m\to\infty} r_{m+k,m}(z\cos\theta, z\sin\theta), \tag{15}$$

which is an identity of the same type as (1). Note that the switch from Cartesian to polar coordinates is only required to make the transition from the Stieltjes transform to the Markov and Radon transform. More generally,

$$\mathcal{S}_2(f)(v,w) = \lim_{m\to\infty} r_{m+k,m}(v,w),$$

where $(v,w) = (z\cos\theta, z\sin\theta)$ with $-\pi/2 < \theta \leq \pi/2$. In the new reconstruction algorithm, this Markov and Radon transform shall not be computed explicitly. We only want to make use of moment information. We summarize the computations described so far and complete the reconstruction of $f(t,s)$ by solving (15).

5. Shape reconstruction algorithm. From the preceding theory we now summarize the bivariate reconstruction algorithm, with the higher-dimensional case being completely similar. Given the moments

$$c_{ij} = \int_0^\infty \int_0^\infty f(t,s)\, t^i s^j \, dt\, ds \tag{16}$$

we compute, for some $k \geq -1$ and consecutive m, the homogeneous bivariate Padé approximant $r_{m+k,m}(v,w)$ as a function of the Cartesian coordinates v and w. A fast algorithm for the computation of the multivariate Padé approximant is given in [2]. The computation of $r_{m+k,m}(v,w)$ requires knowledge of the moments c_{ij} appearing in the expressions $C_\ell(v,w)$ given in (8) for $\ell = 0,\ldots,2m+k$, or in other words the first $(2m+k+1)(2m+k+2)/2$ moments c_{ij}.

In the reconstructions, shape often means compact set A in $\mathbb{R}^2$ (or $\mathbb{R}^3$) and then $f(t,s)$ equals the characteristic function δ_A. In that case

$$c_{ij} = \int\int_A t^i s^j \, dt\, ds.$$

Theorem 3 guarantees that the approximants $r_{m+k,m}(v,w)$ converge rapidly on each slice S_θ to $g(v,w)$ restricted to that slice. A typical value for m is between 3 and 7, and hence one needs with $k=-1$ on average between 21 and 105 moments. Moreover, the moments are not required very accurately. For a rough estimate of a shape as in Figure 2, moments with 2 to 3 significant digits (a relative error bounded by 5×10^{-2} or 5×10^{-3}) are sufficient.

The Padé approximant is then evaluated in a discrete number of points (v_j,w_j) inside the unit disc to approximate

$$g(v_j,w_j) \approx r_{m+k,m}(v_j,w_j).$$

The latter constitutes the right-hand side of (17). To speed up the convergence of the Padé approximants $r_{m+k,m}$ the points (v_j,w_j) can be taken in a disc of radius $r<1$: Padé approximants converge more rapidly in the neighborhood of the origin. At the same time, for each point (v_j,w_j) the value $g(v_j,w_j)$ can be approximated to high accuracy by a cubature formula,

$$\sum_{i=1}^{L} \frac{\omega_i}{1+t_i v_j + s_i w_j} f(t_i,s_i), \quad j=0,1,\ldots,$$

with weights ω_i and nodes (t_i, s_i). The inverse problem of computing $f(t_i, s_i)$ from

$$\sum_{i=1}^{L} \frac{\omega_i}{1 + t_i v_j + s_i w_j} f(t_i, s_i) \approx g(v_j, w_j) = \lim_{m \to \infty} r_{m+k,m}(v_j, w_j) \tag{17}$$

is a (structured) system of linear equations. When applying a quadrature method directly to (16), it is clear that a few dozen moments are not sufficient to retrieve the value of $f(t, s)$ with the accuracy shown in the illustrations below ($h = k = 2^{-5}$ gives a resolution of about 3200 pixels in the unit disk and $h = k = 2^{-6}$ a resolution of almost 13,000 pixels)! The current technique allows one to write down as many equations as required, by adding evaluations of the Padé approximant in points (v_j, w_j), without increasing the number of required moments c_{ij}.

The linear problem (17) is in general ill-conditioned, and therefore a regularization technique must be applied. In all of the following examples we have found the technique known as truncated SVD [13] to do an excellent job. After regularization, we solve (17) for $f(t_i, s_i)$ and identify an approximation for the shape A with

$$A \approx \{(t_i, s_i) \mid f(t_i, s_i) \geq 0.5\}.$$

The threshold 0.5 is chosen because for the original shape $f(t, s) = 1$ inside A and $f(t, s) = 0$ outside A.

Since the homogeneous Padé approximant can be defined analogously in higher dimensions, the procedure for three-dimensional shape reconstruction is entirely similar. In this case, we are given the moments

$$c_{ijk} = \int_0^\infty \int_0^\infty \int_0^\infty f(t, s, v)\, t^i s^j v^k \, dt \, ds \, dv. \tag{18}$$

Here Theorems 2 and 3 hold on the slices

$$S_{\theta,\phi} = \{(z \cos\phi \cos\theta, z \cos\phi \sin\theta, z \sin\phi) \mid z \in \mathbb{R}\},$$

and the three-dimensional version of (13) is

$$\begin{aligned} \int_{-\infty}^{+\infty} \frac{R_{\theta,\phi}(u)}{1 + zu} du &= \int_0^\infty \int_0^\infty \int_0^\infty \frac{f(t, s, v)}{1 + (t \cos\phi \cos\theta + s \cos\phi \sin\theta + v \sin\phi) z} \, dt \, ds \, dv \\ &= \lim_{m \to \infty} r_{m+k,m}(z \cos\phi \cos\theta, z \cos\phi \sin\theta, z \sin\phi). \end{aligned} \tag{19}$$

The homogeneous Padé approximant $r_{m+k,m}(w, x, y)$ of the Stieltjes transform

$$\int_0^\infty \int_0^\infty \int_0^\infty \frac{f(t, s, v)}{1 + (tw + sx + vy)} \, dt \, ds \, dv \tag{20}$$

is now constructed from the trivariate Stieltjes series representation

$$\sum_{i,j,k=0}^{\infty} (-1)^{i+j+k} \binom{i+j}{i} \binom{i+j+k}{i+j} c_{ijk} w^i x^j y^k$$

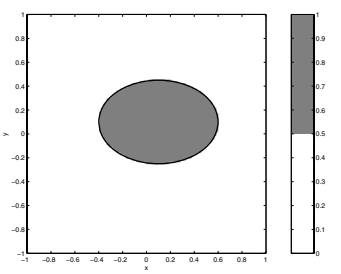
for (20). Note that the vector $(\cos\phi \cos\theta, \cos\phi \sin\theta, \sin\phi)$ generating the one-dimensional slice $S_{\theta,\phi}$ belongs to the three-dimensional unit sphere.

6. Numerical examples. For each example we give the cubature parameters h and k used in formula (21) or h, k, and ℓ used in (22) and the distance Δ between the coordinates of the points (v_j, w_j) used in the formulation of (17). For these evaluation points we construct a uniform grid (the distance Δ in the x- and y-directions is kept equal) that we intersect with the disc of radius r. All points in the intersection are selected. We also list the denominator degree m of the Padé approximant (the numerator degree is $m-1$), the radius r of the disc in which the points (v_j, w_j) are chosen, and the relative accuracy ϵ of the moments (here $2^{-4}, 2^{-11}, 2^{-14}$, and 2^{-24} stand for, respectively, 2, 4, 5, and 8 significant digits). Even when $r < 1$, the shape A is allowed to cover the entire unit disk. Note that the resolution of the reconstruction is expressed by h and k or h, k, and ℓ, not by Δ.

6.1. Reconstruction of two-dimensional shapes. We take for $f(t,s)$ in Theorem 3 the characteristic function of a compact set A contained in the unit disc and use the simple compound 4-point Gauss-Legendre product rule [20, pp. 230–231] for the approximation of $g(v_j, w_j)$ in (17):

$$
\begin{aligned}
\int_a^{a+h}\int_b^{b+k} \frac{f(t,s)}{1+tv_j+sw_j}dt\,ds &\approx \frac{hk}{4}\sum_{i=1}^{4}\frac{f(t_i,s_i)}{1+t_iv_j+s_iw_j},\\
(t_1,s_1) &= \left(a+\frac{3-\sqrt{3}}{6}h,\ b+\frac{3-\sqrt{3}}{6}k\right),\\
(t_2,s_2) &= \left(a+\frac{3-\sqrt{3}}{6}h,\ b+\frac{3+\sqrt{3}}{6}k\right),\\
(t_3,s_3) &= \left(a+\frac{3+\sqrt{3}}{6}h,\ b+\frac{3-\sqrt{3}}{6}k\right),\\
(t_4,s_4) &= \left(a+\frac{3+\sqrt{3}}{6}h,\ b+\frac{3+\sqrt{3}}{6}k\right).
\end{aligned}
\tag{21}
$$

We reconstruct several shapes, such as a simple convex shape like the ellipse in Figures 1 and 2, the more difficult nonconvex lemniscates in Figures 3 and 4, and the bone-like Figure 5 containing a hole. In each of the illustrations we delimit the original shape in black and show the reconstructed area in gray. The black contour is given only for comparison. Note that the shape's boundary is unknown in real-life situations where only the shape's moments are known up to some order and accuracy. For the reconstruction of the two-dimensional shapes we choose h and k in (21) equal to $h = k = 2^{-5}$.



$A = \{(t,s) \mid ((t-0.1)/0.5)^2 + ((s-0.1)/0.35)^2 = 1\}$

FIG. 1.
$m = 7, r = 1.00, \epsilon = 2^{-24}, \Delta = 2^{-4}$.

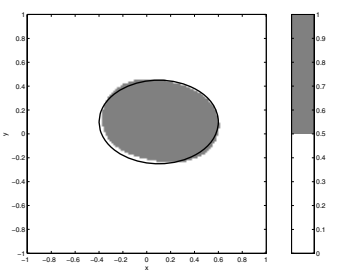
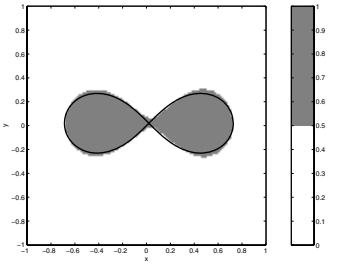


FIG. 2.
$m = 5, r = 0.15, \epsilon = 2^{-4}, \Delta = 2^{-6}$.



$$A = \{(t,s) \mid \left((t-t_0)^2 + (s-s_0)^2 + \alpha^2\right)^2 - 4\alpha^2(t-0.1)^2 = \beta^4\}$$

FIG. 3. $t_0 = s_0 = 0.02, \alpha = \beta = 0.5,$ $m = 9, r = 0.12, \epsilon = 2^{-11}, \Delta = 2^{-6}.$

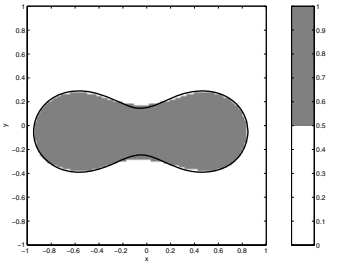
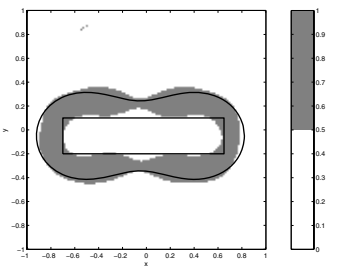


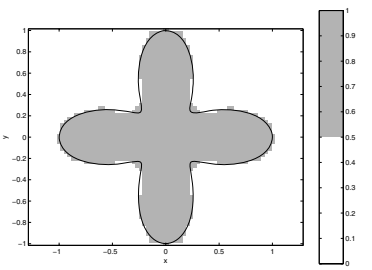
FIG. 4. $t_0 = s_0 = -0.05, \alpha = 0.62, \beta = 0.65,$ $m = 9, r = 0.30, \epsilon = 2^{-14}, \Delta = 2^{-5}.$



$$A = \{(t,s) \mid \left(t^2 + s^2 + 0.4225\right)^2 - 1.69t^2 = 0.2401\} \setminus \{(t,s) \mid |t| < 0.8, |s| < 0.1\}$$

FIG. 5. $m = 9, r = 0.50, \epsilon = 2^{-24}, \Delta = 2^{-4}.$

Finally, a difficult two-dimensional shape is presented in Figure 6. Here, for a change, $h = k = 2^{-4}$ and the evaluation points (v_j, w_j) are placed on a radial grid, while the moments are computed to almost full double precision.

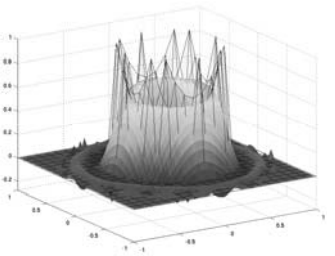


$$A = \{(t,s) \mid t = (\cos(4\tau) + 2)\cos(\tau)/3, s = (\cos(4\tau) + 2)\sin(\tau)/3\}$$

FIG. 6. $m = 10, r = 0.5, \Delta \approx 1/12.$

6.2. Reconstruction of particular three-dimensional shapes. Since Theorem 3 also applies to more general functions $f(t,s)$ than characteristic functions, particular three-dimensional shapes as in Figure 7, namely, with one flat side, can also be reconstructed by means of the two-dimensional integral inversion technique. Here the positive function $f(t,s)$ defines the top surface of the three-dimensional shape while the bottom surface is the domain of f in the (t,s)-plane. Further, the object is cylindrical. The reconstruction of the crater-like object is obtained by plotting the reconstruction of $f(t,s)$ as a function of t and s. The plot is overlaid with a mesh depicting the exact function $f(t,s)$.

In this example we have also increased the sizes of h and k to 2^{-4}, while keeping the number of points (v_j, w_j) rather moderate.

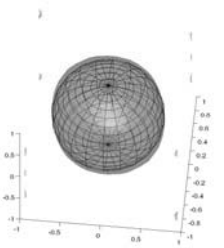


Three-dimensional object with surface $f(t,s) = t^2 + s^2$ *for* $t^2 + s^2 \leq 0.25$.
FIG. 7. $m = 8, r = 0.5, \epsilon = 2^{-11}, \Delta = 2^{-4}$.

6.3. Reconstruction of three-dimensional shapes. A more general three-dimensional shape, such as the ball in Figure 8, can be reconstructed through the solution of the three-dimensional analogue of (17) for its characteristic function $f(t,s,v)$. Here we use the compound 8-point Gauss–Legendre product cubature formula given in [20, pp. 230–231]:

$$\int_a^{a+h}\int_b^{b+k}\int_c^{c+\ell} \frac{f(t,s,v)}{1+tw+sx+vy}dt\,ds\,dv \approx \frac{hk\ell}{8}\sum_{i=1}^{8}\frac{f(t_i,s_i,v_i)}{1+t_iw_j+s_ix_j+v_iy_j},$$

$$(t_i,s_i,v_i) = \left(a+\frac{3\pm\sqrt{3}}{6}h, b+\frac{3\pm\sqrt{3}}{6}k, c+\frac{3\pm\sqrt{3}}{6}\ell\right). \tag{22}$$



Reconstruction of the ball $t^2 + s^2 + v^2 \leq 0.49$.
FIG. 8. $m = 6, h = k = \ell = 2^{-2}, \Delta = 2^{-4}$.

7. Conclusion. The new technique is able to deal with very general shapes: nonconvex such as in Figures 3 and 4, shapes with nonconnected boundary such as in Figure 5, and last but not least higher-dimensional objects such as in Figures 7 and 8. In Figure 7 we illustrate that the technique can also be used for more general functions $f(t,s)$ than characteristic functions.

By carrying out a lot of numerical experiments, we have come to the conclusion that the numerical quality of the output delivered by an implementation of the mathematical property formulated in Theorem 3 seems to depend on a number of elements. Let us formulate a conclusion and some numerical advice.

When the accuracy in the moment information decreases, then it is recommended to reduce the radius r of the ball from which the points (v_j, w_j) (in two dimensions) or (w_j, x_j, y_j) (in three dimensions) are selected. This shrinking of the region for the evaluation points improves the quality of the Padé approximant. We refer the reader to Figures 1 and 2 for a clear illustration. With moments known up to 7

significant digits, a perfect reconstruction is possible with $r = 1$, with a moderate number of moments. With moments known only up to 2 significant digits, a good quality reconstruction is possible when selecting the (v_j, w_j) inside a disc with far smaller radius. Fortunately, the lack of quality of the moments does not have to be compensated for by their quantity. Note that in Figure 2 even fewer moments are used than in Figure 1: $m = 5$ versus $m = 7$.

The quality of the reconstruction improves when the shape's center of gravity is positioned near the origin. To illustrate this we increase the values of t_0 and s_0 in Figure 3 to $t_0 = s_0 = 0.1$, without altering the other parameters. The reconstruction technique then delivers Figure 9. The same observation is made in [11], the reason for this being that the conditioning of the problem improves.

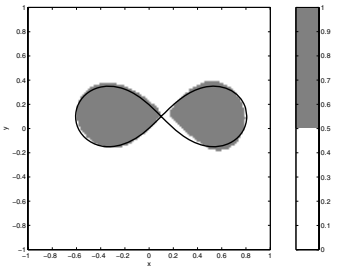


FIG. 9. $m = 9, r = 0.12, \epsilon = 2^{-11}, \Delta = 2^{-6}$.

Increasing the degree of the Padé denominator has little or no influence on the reconstruction. This is due to the fact that the Padé approximants converge quite rapidly. Their relative error drops quickly below the relative error on the moments. As a rule of thumb one can bound the relative error of the Padé approximant $r_{m-1,m}$ in the proposed reconstruction technique by 10^{-m-1}.

REFERENCES

[1] G. A. BAKER, JR., AND P. GRAVES-MORRIS, *Padé approximants*, 2nd ed., Cambridge University Press, Cambridge, UK, 1996.

[2] S. BECUWE AND A. CUYT, *On the Fast Solution of Toeplitz-Block Linear Systems Arising in Multivariate Approximation Theory*, Technical report, University of Antwerp, 2002.

[3] B. BENOUAHMANE AND A. CUYT, *Properties of multivariate homogeneous orthogonal polynomials*, J. Approx. Theory, 113 (2001), pp. 1–20.

[4] M. BRODSKY AND E. PANAKHOV, *Concerning a priori estimates of the solution of the inverse logarithmic potential problem*, Inverse Problems, 6 (1990), pp. 321–330.

[5] C. CHAFFY, *Interpolation polynomiale et rationnelle d'une fonction de plusieurs variables complexes*, Thèse, Institut Polytechnique Grenoble, Grenoble, France, 1984.

[6] A. A. M. CUYT, *A comparison of some multivariate Padé-approximants*, SIAM J. Math. Anal., 14 (1983), pp. 195–202.

[7] A. CUYT, *Padé Approximants for Operators: Theory and Applications*, Lecture Notes in Math. 1065, Springer-Verlag, Berlin, 1984.

[8] A. CUYT, *How well can the concept of Padé approximant be generalized to the multivariate case?*, J. Comput. Appl. Math., 105 (1999), pp. 25–50.

[9] A. CUYT AND D. LUBINSKY, *A de Montessus theorem for multivariate homogeneous Padé approximants*, Ann. Numer. Math., 4 (1997), pp. 217–228.

[10] P. DIACONIS, *Application of the method of moments in probability and statistics*, in Moments in Mathematics (San Antonio, TX, 1987), Proc. Sympos. Appl. Math. 37, AMS, Providence RI, 1987, pp. 125–142.

[11] G. H. GOLUB, P. MILANFAR, AND J. VARAH, *A stable numerical method for inverting shape from moments*, SIAM J. Sci. Comput., 21 (1999), pp. 1222–1243.

[12] B. Gustafsson, C. He, P. Milanfar, and M. Putinar, *Reconstructing planar domains from their moments*, Inverse Problems, 16 (2000), pp. 1053–1070.
[13] P. C. Hansen, *The truncated SVD as a method for regularization*, BIT, 27 (1987), pp. 543–553.
[14] S. Helgason, *The Radon Transform*, Birkhäuser Boston, Boston, 1980.
[15] J. Karlsson, *private communication*, 1981.
[16] P. Milanfar, W. C. Karl, and A. S. Willsky, *A moment-based variational approach to tomographic reconstruction*, IEEE Trans. Image Proc., 5 (1996), pp. 459–470.
[17] P. Milanfar, G. C. Verghese, W. Karl, and A. S. Willsky, *Reconstructing polygons from moments with connections to array processing*, IEEE Trans. Signal Process., 43 (1995), pp. 432–443.
[18] M. I. Sezan and H. Stark, *Incorporation of a priori moment information into signal recovery and synthesis problems*, J. Math. Anal. Appl., 122 (1987), pp. 172–186.
[19] V. N. Strakhov and M. A. Brodsky, *On the uniqueness of the inverse logarithmic potential problem*, SIAM J. Appl. Math., 46 (1986), pp. 324–344.
[20] A. H. Stroud, *Approximate Calculation of Multiple Integrals*, Prentice–Hall, Englewood Cliffs, NJ, 1971.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1071–1091
© 2005 Society for Industrial and Applied Mathematics

A WEIGHTED ESSENTIALLY NONOSCILLATORY, LARGE TIME-STEP SCHEME FOR HAMILTON–JACOBI EQUATIONS*

E. CARLINI†, R. FERRETTI‡, AND G. RUSSO§

Abstract. We investigate the application of weighted essentially nonoscillatory (WENO) reconstructions to a class of semi-Lagrangian schemes for first order time-dependent Hamilton–Jacobi equations. In particular, we derive a general form of the scheme, study sufficient conditions for its convergence with high-order reconstructions, and perform numerical tests to study its efficiency. In addition, we prove that the weights of the WENO interpolants are positive for any order.

Key words. semi-Lagrangian schemes, Hamilton–Jacobi equations, WENO methods

AMS subject classifications. Primary, 65N12, 65M10; Secondary, 49L25

DOI. 10.1137/040608787

1. Introduction.

In this paper we treat a class of high-order, large time-step Godunov schemes for time-dependent Hamilton–Jacobi equation of first order. Although several extensions are possible, we will focus on the model problem

$$\begin{cases} v_t + H(\nabla v) = 0 & \text{in } \mathbb{R}^N \times \mathbb{R}, \\ v(x,0) = v_0(x) & \text{in } \mathbb{R}^N \end{cases} \tag{1.1}$$

and assume that the function H is smooth and strictly convex.

Many approximation schemes have been proposed for (1.1), but only a relatively recent part of the existing literature is devoted to high-order schemes. Within this rapidly growing research line, we quote the pioneering works by Osher and Sethian [23], Osher and Shu [24], Abgrall [1], [2] and, among more recent contributions, Lin and Tadmor [20], and Bryson and Levy [6]. Weighted essentially nonoscillatory (WENO) schemes were introduced by Liu, Osher, and Chan in [22], extensively studied and improved in [18], [16], and applied to Hamilton–Jacobi (HJ) equations in Jiang and Peng [17], Zhang and Shu [29], and Bryson and Levy [7].

On the other hand, large time-step, characteristics-based schemes for hyperbolic PDEs have been first proposed by Courant, Isaacson, and Rees in [8] (see also [19]). At the moment, they have gained a considerable popularity in the Numerical Weather Prediction community (where they are also known as [illegible]; see the review paper [27]). In the field of HJ equations, such schemes have been first applied to stationary Bellman equations related to optimal control problems (see [10] and the review [9]) and later to time-dependent equations for which the schemes under consideration might be seen (see [11], [12], [28]) as discrete counterparts of the Lax–Hopf–Oleinik representation formula.

*Received by the editors November 12, 2004; accepted for publication (in revised form) March 29, 2005; published electronically December 30, 2005. This work has been partly supported by MURST, Progetto Nazionale "Metologie Numeriche Avanzate per il Calcolo Scientifico."
http://www.siam.org/journals/sisc/27-3/60878.html
†Dipartimento di Matematica, Università di Roma "La Sapienza," p.le Aldo Moro, 2, 00185 Roma (carlini@mat.uniroma1.it).
‡Dipartimento di Matematica, Università di Roma Tre, L.go S. L. Murialdo, 1, 00146 Roma (ferretti@mat.uniroma3.it).
§Dipartimento di Matematica ed Informatica, Università di Catania, v.le A. Doria, 6, 95125 Catania (russo@dmi.unict.it).

It is well known that solutions of (1.1) should be understood in the viscosity sense (see [21], [4], [5]), which does not require v to be everywhere differentiable. Moreover, singularities in the gradient ∇v usually bring important information in HJ equations (as an example, in dynamic programming equations, singularities are typically associated to switching curves for the optimal control). Therefore, it would be crucial for the scheme to be accurate in smooth regions, while retaining low numerical dissipation and a nonoscillatory behavior in the neighborhood of singularities.

A further argument leads us to consider the use of WENO reconstructions in SL schemes. In fact, the scheme performs at any node and any time step a minimization which involves several computations of the numerical reconstruction. Its efficiency is, therefore, critical. Piecewise smooth functions can be effectively reconstructed using ENO or WENO methods (see [14], [15], [25]). Both techniques tend to choose interpolation points on the smooth side of the function; however, ENO schemes compute twice the required number of divided differences: at each step (that is, at any increase of the polynomial degree) the choice between the two new interpolation points is made according to the magnitude of the two divided differences obtained with the two candidate stencils (see [25]). On the other hand, in WENO schemes this process is performed by weighting the different polynomials, and this allows information from both sides to be used whenever possible.

The paper is organized as follows. Section 2 reviews the construction and basic convergence theory of SL schemes for (1.1), section 3 presents a general framework for the implementation of WENO interpolation as required by the SL scheme, with the explicit form of the schemes of order 2/3 and 3/5. In section 4 we prove positivity of the linear and nonlinear weights for any order of the polynomial and give their explicit form for evenly spaced nodes. In section 5 we derive from the positivity of the weights a sufficient condition for the convergence of the scheme, whereas section 6 is devoted to numerical tests.

2. Construction of the scheme and basic convergence theory. As it has been already pointed out, the SL schemes for HJ equations might be interpreted (see [11]) as discrete versions of the Lax–Hopf–Oleinik representation formula for the solution, which reads

$$v(x,t) = \inf_{\alpha\in\mathbb{R}^N} \{tH^*(\alpha) + v_0(x+\alpha t)\} \tag{2.1}$$

(where $H^*(\alpha)$ is the Legendre transform of $H(p)$). The general form of the scheme is a discretization of (2.1) on a single time step, namely,

$$v_j^{n+1} = \min_{a\in\mathbb{R}^N} \left(\Delta t H^*(a) + I[V^n](x_j + a\Delta t)\right), \tag{2.2}$$

where $V^n = \{v_j^n\}_{j\in\mathbb{Z}}$, $I[V^n](x)$ is a numerical reconstruction of the approximate solution at time $t_n = n\Delta t$ for any $x \in \mathbb{R}^N$, and v_j^{n+1} approximates $v(x_j, t_{n+1})$. If the reconstruction $I[V^n]$ is of degree r, then the consistency error of the scheme is estimated by

$$E_{\Delta t,\Delta x} \le C\frac{\Delta x^r}{\Delta t} \tag{2.3}$$

(note that in this situation, the best accuracy of the scheme would, in principle, be achieved by performing a single time step). If the Hamiltonian H depends also on x, then the representation formula (2.1) takes a more general form (see [4]) and in the

consistency analysis, there arises a further error term related to the approximation of characteristics (see [11] for a complete analysis). More precisely, if characteristics are approximated with order p, the full consistency estimate reads

$$E_{\Delta t,\Delta x} \le C_1 \Delta t^p + C_2 \frac{\Delta x^r}{\Delta t}. \tag{2.4}$$

In this case, both the term Δt^p and the need for a global minimum search in (2.2) make it unfeasible to perform a single time step when approximating the solution at a fixed time T. Note that according to (2.4), the scheme is consistent provided $\Delta x^r = o(\Delta t)$ and that the highest consistency rate is achieved for

$$\Delta t \sim \Delta x^{\frac{r}{p+1}}.$$

We briefly review here the convergence theory developed in [13] for the scheme (2.2). Assume that (1.1) is posed on $\mathbb{R}$ and discretized on an infinite uniform grid with nodes $x_j = j\Delta x$ ($j = 0, \pm 1, \pm 2, \ldots$). Also assume that the Hamiltonian function H is a $W^{2,\infty}$ function and that

$$H''(p) \ge m_H > 0. \tag{2.5}$$

The key assumption is that for any Lipschitz continuous function $v(x)$, once defined the sequence $V = \{v_j\}_j = \{v(x_j)\}_j$, the interpolation operator $I[V]$ satisfies, for some constant $C < 1$,

$$I[V](x_k) = v(x_k)\ , \quad |I[V](x) - I_1[V](x)| \le C \max_{x_k \in U(x)} |v_{k+1} - 2v_k + v_{k-1}|, \tag{2.6}$$

where by $I_1[V]$ we denote the P_1, i.e., piecewise linear, interpolation on the sequence V, and by $U(x) = (x - h_-\Delta x, x + h_+\Delta x)$ the stencil of the reconstruction $I[V](x)$. For example, a quadratic Lagrange reconstruction can be performed taking one node on the left and two nodes on the right of the point x (in this case, $h_- = 1$, $h_+ = 2$), or two nodes on the left and one on the right (and in this case, $h_- = 2$, $h_+ = 1$). In a second order ENO reconstruction (see [25]), both cases are possible depending on the solution (and thus, $h_- = h_+ = 2$). A third order Lagrange reconstruction is typically performed using two nodes on the left and two on the right, so that $h_- = h_+ = 2$. In the third order ENO case, $h_- = h_+ = 3$ and so forth. It is proved in [13] that condition (2.6) is satisfied for Lagrange and ENO reconstructions up to the fifth order if the reconstruction stencil includes the interval $[x_j, x_{j+1}]$.

Under such assumptions it is possible (see [13]) to prove the following.

THEOREM 2.1. [illegible] (2.2) [illegible] (1.1) [illegible] $N = 1$ [illegible] (2.5), (2.6) [illegible] $\Delta x = O(\Delta t^2)$ [illegible] v_0 [illegible] $V^n = \{v_j^n\}_j$ [illegible] v_j^n [illegible] (2.2)) [illegible]

$$\|I[V^n] - v(n\Delta t)\|_\infty \to 0$$

[illegible] v [illegible] (1.1) [illegible] $0 \le n \le T/\Delta t$ [illegible] $\Delta t \to 0$

3. A general form of WENO interpolation. Although the construction of a WENO interpolation is a well-established matter, we will give here some details of the procedure, since it is usually treated and used in a different framework. In this section, we will [illegible] assume a uniform mesh spacing.

To construct a WENO interpolation of degree $2n-1$ on the interval $[x_j, x_{j+1}]$, we start from the Lagrange polynomial built on the stencil $S = \{x_{j-n+1}, \dots, x_{j+n}\}$ and written in the form

$$Q(x) = \sum_{k=1}^{n} C_k(x) P_k(x), \tag{3.1}$$

where the "linear weights" C_k are polynomials of degree $n-1$ and the P_k are polynomials of degree n interpolating V on the stencil $S_k = \{x_{j-n+k}, \dots, x_{j+k}\}$, $k = 1, \dots, n$ (note that all the stencils S_k overlap on the interval $[x_j, x_{j+1}]$, and that the dependence on j has been dropped in this general expression). The nonlinear weights are then constructed so as to obtain an approximation of the highest degree if suitable smoothness indicators β_k give the same result on all stencils. This leads us to define

$$\alpha_k(x) = \frac{C_k(x)}{(\beta_k + \varepsilon)^2} \tag{3.2}$$

(with ε a properly small parameter, usually of the order of 10^{-6}), and then the nonlinear weights as

$$w_k(x) = \frac{\alpha_k(x)}{\sum_l \alpha_l(x)}, \tag{3.3}$$

where, following [25], the smoothness indicators are typically computed as

$$\beta_k = \sum_{l=1}^{n} \int_{x_j}^{x_{j+1}} \Delta x^{2l-1} (P_k^{(l)})^2 \, dx. \tag{3.4}$$

In the section on numerical tests we shall consider some different expressions for the smoothness indicators, such as the one proposed in [17], which does not depend on the first derivative of the function.

Last, the WENO interpolation reads

$$I[V](x) = \sum_{k=1}^{n} w_k(x) P_k(x). \tag{3.5}$$

Note that usually one is interested in computing the reconstruction in specific points in the cell. In this case the polynomials C_k take some specific value.

Let us now turn back to the linear weights C_k, which have not been given a precise expression. Due to (3.1) and to the definition of the polynomials P_k, the C_k are characterized by the fact that $Q(x)$ should be the interpolating polynomial of $V(x)$ on the stencil S. Since P_k interpolates V on the stencil S_k, it would be natural to require that C_k should vanish at the nodes outside S_k, and that in the nodes of S the nonzero weights should have unit sum. In this way, if P_k is the polynomial that interpolates the function on S_k, then (3.1) is the polynomial that interpolates V on S.

Taking into account that the number of nodes belonging to S but not to S_k is precisely n, we thus infer that the linear weights must necessarily have the form

$$C_k(x) = \gamma_k \prod_{x_l \in S \setminus S_k} (x - x_l) = \gamma_k \tilde{C}_k(x) \tag{3.6}$$

with γ_k to be determined in order to have unit sum. Note that the conditions

$$\sum_{k=1}^{n} C_k(x_i) = 1 \tag{3.7}$$

for each $x_i \in S$ apparently constitute a set of $2n$ equations in n unknowns. Actually, the correct perspective to look at these conditions is the following. The left-hand side of (3.7) is a polynomial of degree $n-1$ (computed at x_i), and on the right-hand side we have the polynomial $p(x) \equiv 1$. Therefore, imposing that the two polynomials coincide on more than $n-1$ points is equivalent to impose that they are identical. Now, it is easy to show that the monic polynomials $\tilde{C}_k$, $k = 1, \ldots, n$, are independent and form a basis of the space $\mathbb{P}_{n-1}$ of polynomials of degree $n-1$. Condition

$$\sum_{k=1}^{n} C_k(x) = 1 \quad \forall x \in \mathbb{R} \tag{3.8}$$

uniquely determines the constants γ_k as a polynomial identity.

Condition (3.8) can be conveniently satisfied by imposing it on a suitable set of points. Consider first the node x_{j-n+1}. Then, using (3.6) into (3.1), we note that, among all the linear weights computed at x_{j-n+1}, the only nonzero weight is C_1, so that

$$Q(x_{j-n+1}) = C_1(x_{j-n+1})P_1(x_{j-n+1}) = C_1(x_{j-n+1})v_{j-n+1} \tag{3.9}$$

and this implies that $C_1(x_{j-n+1}) = 1$ and allows us to compute γ_1. On the other hand, in the following node x_{j-n+2}, the nonzero weights are C_1 and C_2, and with a similar argument we obtain the condition

$$\begin{aligned} Q(x_{j-n+2}) &= C_1(x_{j-n+2})P_1(x_{j-n+2}) + C_2(x_{j-n+2})P_2(x_{j-n+2}) \\ &= [C_1(x_{j-n+2}) + C_2(x_{j-n+2})]v_{j-n+2}, \end{aligned} \tag{3.10}$$

which implies that $C_1(x_{j-n+2}) + C_2(x_{j-n+2}) = 1$, whence γ_2 can be computed. Following this guideline, we finally obtain the set of conditions

$$\sum_{i=1}^{k} C_i(x_{j-n+k}) = 1 \quad (k = 1, \ldots, n), \tag{3.11}$$

that is, more explicitly,

$$\sum_{i=1}^{k} \gamma_i \prod_{x_l \in S \setminus S_i} (x_{j-n+k} - x_l) = 1 \quad (k = 1, \ldots, n), \tag{3.12}$$

which is a linear triangular system in the unknowns γ_i.

Although the construction outlined in this section does not require a constant space step, the problem of giving an explicit expression to the coefficients γ_i is considerably easier in the case of constant Δx. This case will be treated in detail in section 4, along with the study of sign for the linear weights. In the next subsections we give two examples of this construction (on a uniform mesh), including the expressions for the smoothness indicators (3.4). The resulting schemes will be used in the section on numerical tests.

3.1. Second–third order WENO interpolation. To construct a third order interpolation we start from two polynomials of second degree, so that

$$I[V^n](x) = w_L P_L(x) + w_R P_R(x), \tag{3.13}$$

where $P_L(x)$ and $P_R(x)$ are second order polynomials constructed, respectively, on the nodes x_{j-1}, x_j, x_{j+1} and on the nodes x_j, x_{j+1}, x_{j+2}. The two linear weights C_L and C_R are first degree polynomials in x, and according to the general theory outlined so far, they read

$$C_L = \frac{x_{j+2} - x}{3\Delta x}, \quad C_R = \frac{x - x_{j-1}}{3\Delta x}, \tag{3.14}$$

and the expressions of α_L, α_R, w_L, and w_R may be easily recovered from the general form.

According to (3.4), the smoothness indicators have the explicit expressions

$$\beta_L = \frac{13}{12}v_{j-1}^2 + \frac{16}{3}v_j^2 + \frac{25}{12}v_{j+1}^2 - \frac{13}{3}v_{j-1}v_j + \frac{7}{6}v_{j-1}v_{j+1} - \frac{19}{3}v_j v_{j+1}, \tag{3.15}$$

$$\beta_R = \frac{13}{12}v_{j+2}^2 + \frac{16}{3}v_{j+1}^2 + \frac{25}{12}v_j^2 - \frac{13}{3}v_{j+2}v_{j+1} + \frac{7}{6}v_{j+2}v_j - \frac{19}{3}v_j v_{j+1}. \tag{3.16}$$

3.2. Third–fifth order WENO interpolation. To construct a fifth order interpolation we start from three polynomials of third degree:

$$I[V^n](x) = w_L P_L(x) + w_C P_C(x) + w_R P_R(x), \tag{3.17}$$

where the third order polynomials $P_L(x)$, $P_C(x)$, and $P_R(x)$ are constructed, respectively, on $x_{j-2}, x_{j-1}, x_j, x_{j+1}$, on $x_{j-1}, x_j, x_{j+1}, x_{j+2}$, and on $x_j, x_{j+1}, x_{j+2}, x_{j+3}$. The weights C_L, C_C, and C_R are second degree polynomials in x, and have the form

$$\begin{aligned} C_L &= \frac{(x - x_{j+2})(x - x_{j+3})}{20\Delta x^2}, \quad C_C = \frac{(x - x_{j-2})(x - x_{j+3})}{10\Delta x^2}, \\ C_R &= \frac{(x - x_{j-2})(x - x_{j-1})}{20\Delta x^2}, \end{aligned} \tag{3.18}$$

while the smoothness indicators β_C and β_R have the expressions

$$\begin{aligned} \beta_C = {} & \frac{61}{45}v_{j-1}^2 + \frac{331}{30}v_j^2 + \frac{331}{30}v_{j+1}^2 + \frac{61}{45}v_{j+2}^2 - \frac{141}{20}v_{j-1}v_j + \frac{179}{30}v_{j-1}v_{j+1} \\ & + \frac{293}{180}v_{j-1}v_{j+2} - \frac{1259}{60}v_j v_{j+1} + \frac{179}{30}v_j v_{j+2} - \frac{141}{20}v_{j+1}v_{j+2}, \end{aligned} \tag{3.19}$$

$$\begin{aligned} \beta_R = {} & \frac{407}{90}v_j^2 + \frac{721}{30}v_{j+1}^2 + \frac{248}{15}v_{j+2}^2 + \frac{61}{45}v_{j+3}^2 - \frac{1193}{60}v_j v_{j+1} + \frac{439}{30}v_j v_{j+2} \\ & - \frac{683}{180}v_j v_{j+3} - \frac{2309}{60}v_{j+1}v_{j+2} + \frac{309}{30}v_{j+1}v_{j+3} - \frac{553}{60}v_{j+2}v_{j+3}, \end{aligned} \tag{3.20}$$

and β_L can be obtained using the same set of coefficients of β_R in a symmetric way (that is, replacing the indices $j-2, \dots, j+3$ with $j+3, \dots, j-2$).

It is interesting to check that, due to their structure, the linear weights are always positive in the interval $[x_j, x_{j+1}]$ for $n = 2, 3$. This also means that pointwise the linear combination (3.5) is in fact a convex combination, and this has important consequences at the level of convergence analysis. We will prove in the next section that this property holds for linear weights up to an arbitrary order.

4. On the positivity of the weights in WENO reconstruction. We prove in this section that the positivity result for the weights in WENO interpolation, obtained in the previous section by direct computation up to third–fifth order, actually holds in general. In addition, we give an explicit expression for the linear weights in the case of evenly spaced nodes.

It is worth pointing out that this result of positive sign for the linear and nonlinear weights is not in contrast with the well-known situation in which the weights may change sign (see [26]). In fact, this latter situation may occur in the case of reconstructions based on cell averages instead of pointwise values such as the interpolation procedure above.

We first give a general proof of positivity for the linear weights $C_k(x)$.

THEOREM 4.1. $\{x_i\}$ $\mathbb{R}$ $Q(x)$ $f(x)$ $S = \{x_{j-n+1}, \dots, x_{j+n}\}$ (3.1) $C_k(x)$ $(k = 1, \dots, n)$ $x \in (x_j, x_{j+1})$ $\sum_k C_k(x) \equiv 1$

Let us denote by $Q^{(l,m)}(x)$ (with $j - n + 1 \le l \le j < j + 1 \le m \le j + n$ and $m - l \ge n$) the interpolating polynomial constructed on the stencil $\{x_l, \dots, x_m\}$, so that

$$Q(x) = Q^{(j-n+1,j+n)}(x) \tag{4.1}$$

and, moreover,

$$P_k(x) = Q^{(j-n+k,j+k)}(x). \tag{4.2}$$

We will also write a generic $Q^{(l,m)}(x)$ as

$$Q^{(l,m)}(x) = \sum_k C_k^{(l,m)}(x) P_k(x), \tag{4.3}$$

where the summation may be extended to all $k = 1, \dots, n$ by setting $C_k^{(l,m)} = 0$ whenever the stencil of P_k is not included in $\{x_l, \dots, x_m\}$. Therefore, the final linear weights will be

$$C_k(x) = C_k^{(j-n+1,j+n)}(x). \tag{4.4}$$

We proceed by induction on the degree of $Q^{(l,m)}$, given by $i = \deg Q^{(l,m)} = m - l$, starting with $i = n$ up to $i = 2n - 1$. First we prove that the claim of the theorem extends to all polynomials $Q^{(l,m)}(x)$ defined in (4.3). The claim is obviously true for $Q^{(j-n+k,j+k)}(x)$ by (4.2). In this case the coefficients $C_k^{(l,m)} = 1$, $C_s^{(l,m)} \equiv 0$, $s \ne k$. Then, we assume the claim is true for the set of linear weights $C_k^{(l,m)}(x)$ of a generic $Q^{(l,m)}(x)$ such that $m - l = i$ and add a node to the right (adding a node to the left leads to an analogous computation). By inductive assumption we have, for any l and m such that $l \le j < j + 1 \le m$ and $m - l = i$, that (4.3) holds with $C_k^{(l,m)}(x) \ge 0$. On the other hand, by elementary interpolation theory arguments (Neville's recursive form of the interpolating polynomial),

$$Q^{(l,m+1)}(x) = \frac{x_{m+1} - x}{x_{m+1} - x_l} Q^{(l,m)}(x) + \frac{x - x_l}{x_{m+1} - x_l} Q^{(l+1,m+1)}(x). \tag{4.5}$$

Note that $\deg Q^{(l,m)} = \deg Q^{(l+1,m+1)} = i$, and that both fractions which multiply $Q^{(l,m)}(x)$ and $Q^{(l+1,m+1)}(x)$ are positive as long as $x \in (x_j, x_{j+1})$. Therefore, using

(4.3) into (4.5), we get

$$Q^{(l,m+1)}(x) = \sum_k \left[\frac{x_{m+1} - x}{x_{m+1} - x_l} C_k^{(l,m)}(x) + \frac{x - x_l}{x_{m+1} - x_l} C_k^{(l+1,m+1)}(x) \right] P_k(x). \tag{4.6}$$

The term in square brackets is $C_k^{(l,m+1)}(x)$ and it is immediate to check that it is nonnegative. Iterating this argument up to $Q^{(j-n+1,j+n)}(x)$ completes the proof of nonnegativity for all the linear weights $C_k(x)$.

Last, setting $f(x) \equiv 1$, we also have $Q(x) \equiv 1$ and, for any k, $P_k(x) \equiv 1$. Plugging these identities into (3.1) we obtain $\sum_k C_k(x) \equiv 1$. □

We turn now to the problem of giving an explicit expression to the linear weights in the situation of evenly spaced nodes. The expression of the weights for the linear scheme is

$$C_k = \gamma_k \prod_{x_l \in S \setminus S_k} (x - x_l).$$

First observe that the polynomial $C_k(x)$ does not change sign in the interval $[x_j, x_{j+1}]$. More precisely, the sign of it, in this interval, is given by $\text{sign}(\gamma_k)(-1)^{n+k}$. Therefore,

$$C_k(x) > 0 \quad \forall x \in [x_j, x_{j+1}] \quad \forall k, n \quad \Leftrightarrow \quad \text{sign}\gamma_k = (-1)^{n+k}. \tag{4.7}$$

For simplicity, set $j = n$ in the definition of the stencil S, so that $S = \{x_1, \ldots, x_{2n}\}$. Let us introduce the integer stencils

$$\tilde{S} = S/\Delta x, \quad \tilde{S}_k = S_k/\Delta x.$$

Then the polynomials C_k can be written as

$$C_k(\Delta x \xi) = \tilde{\gamma}_k \prod_{l \in \tilde{S} \setminus \tilde{S}_k} (\xi - l),$$

where $\tilde{\gamma}_k = \gamma_k \Delta x^{n-1}$. The nondimensional constants $\tilde{\gamma}_k$ satisfy the condition

$$\sum_{i=1}^{k} \tilde{\gamma}_i \prod_{\substack{l=1 \\ l \notin (i,\ldots,i+n)}}^{2n} (i - l) = 1.$$

Introducing now the matrix

$$a_{ki} = \frac{\prod_{l=1}^{2n}(k-l)}{\prod_{l=i}^{i+n}(k-l)} = \frac{(k-1)!(2n-k)!}{(n+i-k)!(k-i)!}(-1)^{n+i}, \tag{4.8}$$

the system can be written as

$$\sum_{i=1}^{k} a_{ki} \tilde{\gamma}_i = 1.$$

Let $\mu_i \equiv (-1)^{n+i}\tilde{\gamma}_i$. Then, such constants satisfy the triangular system

$$\sum_{i=1}^{k} |a_{ki}| \mu_i = 1, \tag{4.9}$$

and according to (4.7) $C_i(x) > 0$, $x \in [x_n, x_{n+1}]$ if and only if $\mu_i > 0$.

The solution to system (4.9) can be explicitly given. In fact, we will prove now that

$$\mu_i = \frac{n!(n-1)!}{(n-i)!(i-1)!(2n-1)!} \quad (i = 1, \dots, n). \tag{4.10}$$

Actually, plugging (4.10) and (4.8) into (4.9) we obtain the set of conditions that we want to check:

$$\sum_{i=1}^{k} \frac{(k-1)!(2n-k)!n!(n-1)!}{(n+i-k)!(k-i)!(n-i)!(i-1)!(2n-1)!} = 1.$$

These relations can be rearranged in the form

$$\frac{(k-1)!(2n-k)!}{(2n-1)!} \sum_{i=1}^{k} \binom{n}{k-i} \binom{n-1}{i-1} = 1,$$

and, hence, rewriting the first term and shifting the summation index as $j = i - 1$,

$$\sum_{j=0}^{k-1} \binom{n}{k-1-j} \binom{n-1}{j} = \binom{2n-1}{k-1},$$

which is in turn a special case of the identity[1] (see [3])

$$\sum_{j=0}^{N} \binom{n}{N-j} \binom{m}{j} = \binom{n+m}{N}$$

with $N = k - 1$ and $m = n - 1$.

We also prove here that the μ_i are inverse of integers. In fact, note first that by (4.10), $\mu_i = \mu_{n-i+1}$, so that it suffices to prove the claim for $i \le (n+1)/2$. Then, we rewrite the μ_i as

$$\mu_i = \begin{cases} \dfrac{n!}{(2n-1)!} & \text{if } i = 1, \\ \dfrac{(n-i+1)\cdots(n-1)n(n-1)!}{(i-1)!(2n-1)!} & \text{if } i > 1. \end{cases} \tag{4.11}$$

While the claim is obvious if $i = 1$, if $i > 1$, we derive from (4.11)

$$\mu_i = \frac{(2n-2i+2)\cdots(2n-2)}{2^{i-1}(i-1)!(n+1)\cdots(2n-1)}.$$

Now, if $i \le (n+1)/2$, then $2n-2i+2 \ge n+1$ and, therefore, any term of the product $(2n-2i+2)\cdots(2n-2)$ can be simplified with a term of the product $(n+1)\cdots(2n-1)$. This completes the proof of the claim.

We report in Table 1 the value of $1/\mu_i$ for values of n up to $n = 8$.

[1] We thank professor Corrado Falcolini for bringing this result to our attention.

TABLE 1
Values of $1/\mu_i$ for the first 8 values of n.

n	$1/\mu_i$
1	1
2	3, 3
3	20, 10, 20
4	210, 70, 70, 210
5	3024, 756, 504, 756, 3024
6	55440, 11088, 5544, 5544, 11088, 55440
7	1235520, 205920, 82368, 61776, 82368, 205920, 1235520
8	32432400, 4633200, 1544400, 926640, 926640, 1544400, 4633200, 32432400

5. A sufficient condition for convergence. We briefly check here that the convergence theory developed in [13] may be applied to the scheme under consideration. To this end, we work in the framework of Theorem 2.1 and prove the following.

THEOREM 5.1. [illegible] $V = \{v_j\}_j = \{v(x_j)\}_j$ [illegible] $v(x)$ [illegible] $I[V]$ [illegible] (3.5) [illegible] (2.6) [illegible] $n \le 5$

[illegible] First, the condition

$$I[V](x_k) = v(x_k) \tag{5.1}$$

is clearly satisfied since (3.5) is built as an interpolation. Concerning the second part of (2.6), namely,

$$|I[V](x) - I_1[V](x)| \le C \max_{x_k \in U(x)} |v_{k+1} - 2v_k + v_{k-1}| \tag{5.2}$$

with $C < 1$, this condition is satisfied for all the polynomials $P_k(x)$ if $n \le 5$, as proved in [13], since all the stencils S_k include the interval $[x_j, x_{j+1}]$. Although the structure under consideration for $I[V]$ is not globally polynomial, it is easy to see from (3.2), (3.3) that once x is given, since all the linear weights C_k are nonnegative by Theorem 4.1, then the right-hand side of (3.5) is in fact a convex combination of the polynomials P_k. Therefore,

$$\min_k P_k(x) \le I[V](x) \le \max_k P_k(x) \tag{5.3}$$

and, therefore, the constant C appearing in (2.6) is the same as obtained in [13] for polynomials of degree n. That is, (2.6) is satisfied with $C < 1$ up to WENO interpolations of fifth–ninth order. □

[illegible] The positivity of the linear weights, obtained in the framework of WENO reconstructions, also allows to prove a further result related to Lagrange reconstructions. Actually, the bound (5.3) does not require $I[V]$ to be in the form (3.5), but it also applies to the form (3.1). Therefore, (2.6) is also satisfied for Lagrange reconstructions up to the ninth order, at least if the reconstruction stencil is symmetrical, which corresponds to odd orders of interpolation. Moreover, in the case of even (up to the eighth) orders, an analogous result can be obtained. More precisely, if a sixth order interpolation is performed using the nodes $x_{j-3}, \dots, x_{j+3}$, the linear weights have the expressions

$$C_1(x) = \frac{(x - x_{j+2})(x - x_{j+3})}{30\Delta x^2},$$

$$C_2(x) = -\frac{(x - x_{j-3})(x - x_{j+3})}{15\Delta x^2},$$

$$C_3(x) = \frac{(x - x_{j-3})(x - x_{j-2})}{30\Delta x^2},$$

whereas if an eighth order interpolation is performed using the nodes $x_{j-4}, \ldots, x_{j+4}$, the linear weights read

$$C_1(x) = -\frac{(x - x_{j+2})(x - x_{j+3})(x - x_{j+4})}{336\Delta x^3},$$

$$C_2(x) = \frac{(x - x_{j-4})(x - x_{j+3})(x - x_{j+4})}{112\Delta x^3},$$

$$C_3(x) = -\frac{(x - x_{j-4})(x - x_{j-3})(x - x_{j+4})}{112\Delta x^3},$$

$$C_4(x) = \frac{(x - x_{j-4})(x - x_{j-3})(x - x_{j-2})}{336\Delta x^3}.$$

Again, a direct computation shows that $C_k(x) > 0$ for $x \in [x_j, \ldots, x_{j+1}]$, so that (5.3) (and hence (2.6)) still holds.

6. Numerical tests. In this section we present the results of several numerical tests in which we compare the WENO scheme described with P_1, P_2, cubic Lagrange, and third order ENO schemes. The WENO schemes considered are of second–third order and third–fifth order, and the two-dimensional reconstruction has been implemented by a tensor product of one-dimensional interpolations. The minimization step has been performed by bisection in the one-dimensional case, and by the Powell-like routine `PRAXIS` (taken from the NETLIB repository at ftp.netlib.org) in the two-dimensional examples. In order to evaluate the accuracy of the reconstruction, the tolerance in the minimization phase has been set to an order of magnitude far lower than the truncation error, although this may have somewhat slowed down the scheme. The upwinding along characteristics has been performed by means of a plain Euler method as in (2.2) (in the situation of H depending on the gradient alone, this choice has the best accuracy anyway).

Errors have been expressed in terms of the relative discrete L^∞ norm, defined as

$$\epsilon_{\infty,rel} = \frac{\max_j |v_j - v(x_j)|}{\max_j |v_j|}.$$

The computation of relative discrete L^1 norm of the errors has given similar indications and is therefore omitted.

Unless otherwise stated, figures showing approximate solutions have been obtained with the WENO 2/3 scheme.

In addition to the usual choice (3.4) as proposed by Shu, which has been denoted by S in the tables, some different possibility has been considered for the smoothness indicator. An indicator based on the second derivative alone, that is,

$$\beta_k = \int_{x_j}^{x_{j+1}} \Delta x^3 (P_k'')^2 dx, \tag{6.1}$$

has been denoted by D2 in the tables, whereas an indicator which only takes into account the highest order derivative, namely,

$$\beta_k = \int_{x_j}^{x_{j+1}} \Delta x^{2n-1} (P_k^{(n)})^2 dx, \tag{6.2}$$

has been implemented—it clearly coincides with the indicator D2 for the second–third order scheme, and has been denoted by D3 for the third–fifth order scheme. Last, as proposed by Jiang and Peng (see [17]), a smoothness indicator which takes into account all derivatives but the first, that is,

$$\beta_k = \sum_{l=2}^{n} \int_{x_j}^{x_{j+1}} \Delta x^{2l-1} (P_k^{(l)})^2 dx, \tag{6.3}$$

has been implemented for the third-fifth order scheme and denoted by D2+D3. Although the creation of singularities in HJ equations is at the level of the gradient, and therefore it could seem natural to drop first derivatives in the smoothness indicator, nevertheless, we will show that the use of the full expression (3.4) seems to be the most robust choice, although it might not be the most accurate one in any situation. We have summarized in Table 7 a comparison of the WENO schemes (with various smoothness indicators) with P_1, P_2, cubic, and third order ENO implementations of (2.2) on all numerical tests, in the standard setting of a 50×50 grid (50 nodes for test 1), and after the onset of the singularity for tests 1, 2, and 4.

Test 1: One-dimensional periodic solution. This test (also considered in [6], [20], [17]) deals with the HJ equation

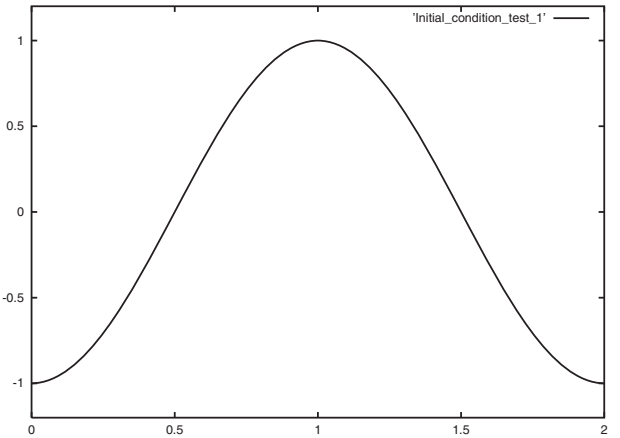
$$\begin{cases} v_t(x,t) + \frac{1}{2}(v_x(x,t)+1)^2 = 0, \\ v(x,0) = v_0(x) = -\cos(\pi x), \end{cases} \tag{6.4}$$

considered in $[0,2]$, with periodic boundary conditions. The approximate solution is computed before the singularity, at $T = 0.8/\pi^2$, with 4 time steps, and after the singularity, at $T = 1.5/\pi^2$, with 5 time steps. Plots of exact versus approximate solutions are shown in Figure 1 and relative errors obtained with the Shu smoothness indicator are compared in Tables 2–3.

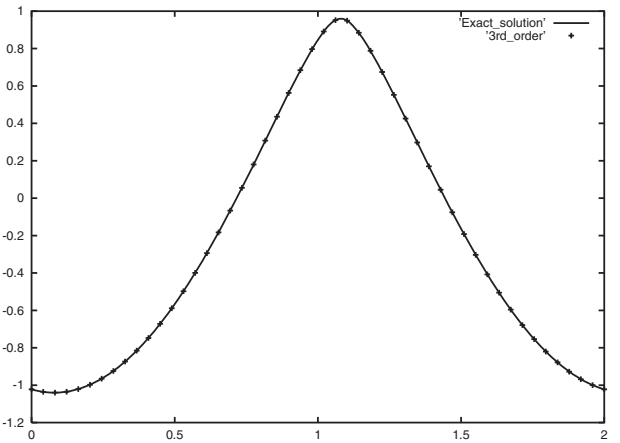
Note that according to the consistency estimate (2.3), the convergence rates should be $p = 4$ for the WENO3 scheme and $p = 6$ for the WENO5 scheme, since the time step has been kept constant. Roughly speaking, this behavior is confirmed by numerical tests, although with a slight decrease in the convergence rate after the onset of the singularity. However, note that even in this latter case the computation of the approximate solution is highly accurate. If the exact solution v is semiconcave (roughly speaking, this means that it can be written as the sum of a concave and a $W^{2,\infty}$ function), this fact can be heuristically explained as follows. At a time step n and for a given node x_j, the reconstruction is performed at an "upwind" point which in the Lax–Hopf formula would be a minimum of the function

$$F_{n,j}(a) = \Delta t H^*(a) + v(t_{n-1}, x_j + a\Delta t).$$

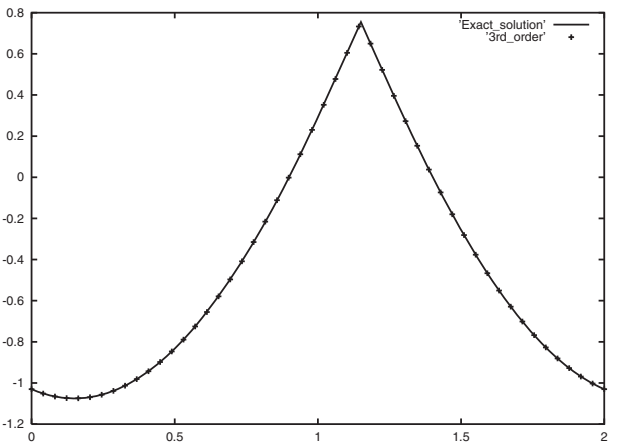
Now, the function to be minimized contains two terms. The first is smooth, due to the assumptions on H, whereas the second is semiconcave. Their sum is therefore semiconcave and it is known (see [4]) that minima of semiconcave functions can occur only at points where the function is at least differentiable. Although the scheme in



(a)



(b)



(c)

FIG. 1. *Approximate versus exact solution for test* 1. $N = 50$ *points have been used.* (a) *Initial condition.* (b) *Solution before singularity.* (c) *Solution after singularity.*

TABLE 2
Errors before and after the singularity for test 1*, WENO*3 *scheme.*

WENO 3rd order				
	Before singularity		After singularity	
N	Relative L^∞-error	L^∞ order	Relative L^∞-error	L^∞ order
25	2.52×10^{-3}		2.88×10^{-3}	
50	8.77×10^{-5}	4.1	5.12×10^{-5}	5.8
100	1.53×10^{-5}	2.5	2.19×10^{-6}	4.5
200	9.63×10^{-7}	3.9	2.39×10^{-7}	3.1

TABLE 3
Errors before and after the singularity for test 1*, WENO*5 *scheme.*

WENO 5th order				
	Before singularity		After singularity	
N	Relative L^∞-error	L^∞ order	Relative L^∞-error	L^∞ order
25	1.29×10^{-3}		3.05×10^{-3}	
50	1.87×10^{-5}	6.1	5.83×10^{-6}	9.0
100	9.13×10^{-7}	4.3	7.25×10^{-8}	6.3
200	2.01×10^{-8}	5.5	1.89×10^{-9}	5.2

fact uses a discretization of the function $F_{n,j}$, this analysis seems to give a convincing explanation of numerical results.

In the summarizing Table 7 we note that the performance of the third order schemes is similar and that there is an apparent loss of accuracy in the D2 version of the WENO3 scheme, as well as in the D2+D3 version of the WENO5 scheme.

Test 2: Generation of a two-dimensional singularity. This test refers to the HJ equation

$$\begin{cases} v_t(x,t) + \frac{1}{2}|\nabla v(x,t)|^2 = 0, \\ v(x,0) = v_0(x) = \max(0, 1 - |x|^2). \end{cases} \tag{6.5}$$

The problem is considered in $[-2,2]^2$. For any positive time, the solution of this test problem has discontinuous second derivatives along the circle $|x| = 1$ and generates a further singularity in the gradient at the origin for $t \geq 1/2$.

The exact solution of (6.5) can be explicitly computed and for $t \geq 1/2$ reads

$$v(t,x) = \begin{cases} \frac{(|x|-1)^2}{2t} & \text{if } |x| \leq 1, \\ 0 & \text{if } |x| \geq 1. \end{cases}$$

The solution, computed with $\Delta t = 0.1$, is shown in Figure 2 at $t = 0$ and $t = 0.5$. Table 4 shows execution times (on a SUN E 420 R platform) at $t = T = 0.5$ for different reconstructions and different smoothness indicators, on a 50×50 grid. The errors reported in Table 7 show that the solution is computed with nearly the same accuracy by all the schemes, and the use of nonoscillatory interpolation does not seem to give any advantage (this is related to the lack of uniform semiconcavity of the solution; see the above discussion of this point). We point out, in any case, that the singularity in the gradient is well resolved, and that the scheme does not introduce instabilities. However, looking at Figure 3, in which we zoom the solution close to the x_1-x_2 plane for the third order linear, ENO, and WENO schemes, we may see

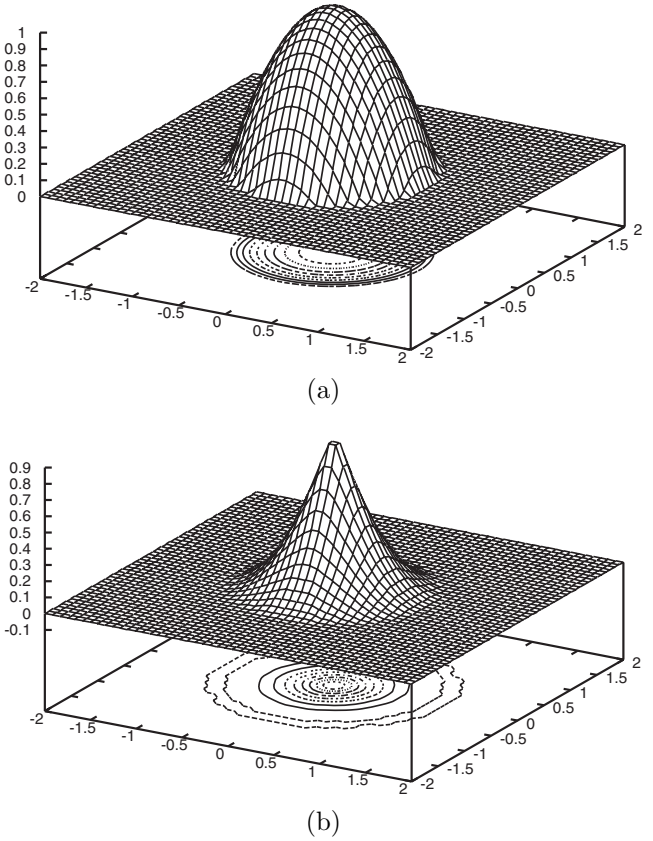
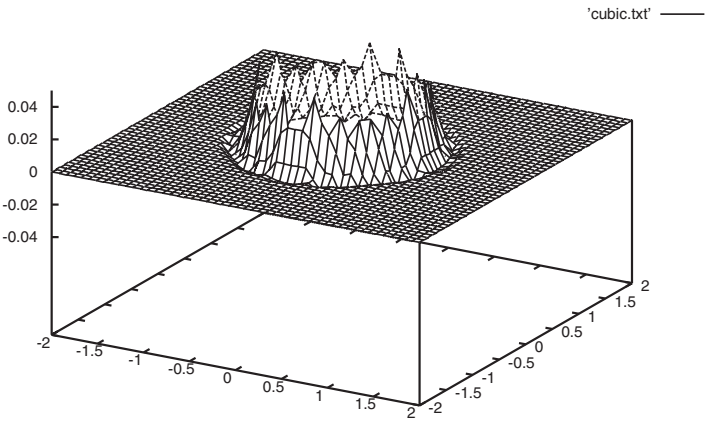


FIG. 2. *Approximate solution for test* 2. (a) *Initial condition.* (b) *Solution at* $t = 0.5$.

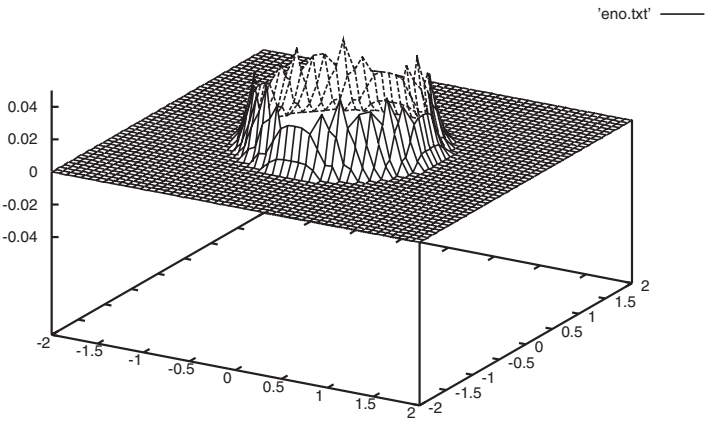
TABLE 4
Comparison of CPU times for different schemes on test 2.

Scheme	CPU
P_1	34s
P_2	41s
cubic	52s
ENO 3	3m36s
WENO 2/3 S	1m43s
WENO 2/3 D2	1m55s
WENO 3/5 S	3m55s
WENO 3/5 D2	3m55s
WENO 3/5 D3	4m02s
WENO 3/5 D2+D3	4m05s

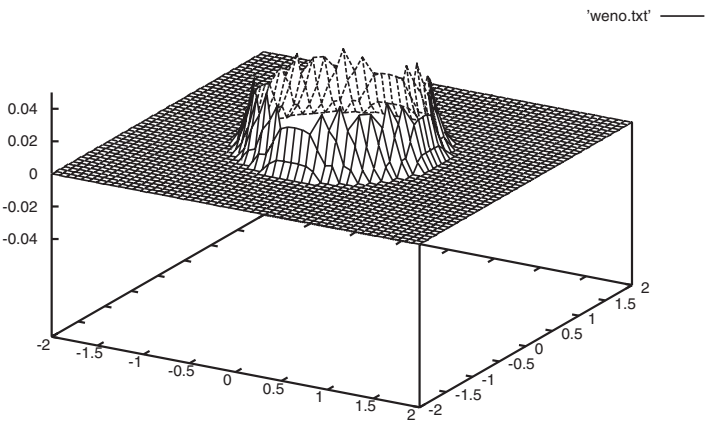
that the plain cubic reconstruction, as expected, introduces a zone of undershoot in the neighborhood of the circle $|x| = 1$, where a discontinuity in the second derivative appears. This artifact is avoided by nonoscillatory schemes, among which, however, the WENO scheme performs apparently better in terms of CPU time. Also, the behavior of the scheme is remarkably isotropic, as shown in the scatter plot of Figure 4,



(a)



(b)



(c)

FIG. 3. *Zoom of the approximate solutions for test* 2. (a) *Linear third order scheme.* (b) *Third order ENO scheme.* (c) *Third order WENO scheme.*

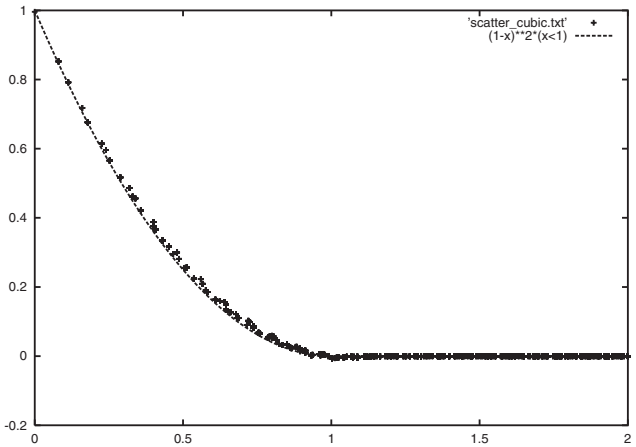


FIG. 4. *Scatter plot of the approximate solution for test* 2.

in which at any node the numerical solution is plotted as a function of $|x|$ for the cubic Lagrange reconstruction (ENO and WENO reconstructions give similar results).

Test 3: Uniformly semiconcave solution. This test refers again to the HJ equation

$$\begin{cases} v_t(x,t) + \frac{1}{2}|\nabla v(x,t)|^2 = 0, \\ v(x,0) = v_0(x) = \min(0, |x|^2 - 1), \end{cases} \tag{6.6}$$

considered in $[-2,2]^2$, in which the initial condition v_0 has been changed to a semiconcave function.

The exact solution of (6.6) is now uniformly semiconcave for any positive time; more precisely, it has the explicit form

$$v(t,x) = \begin{cases} \frac{|x|^2}{2t+1} - 1 & \text{if} |x| \le 2t+1, \\ 0 & \text{if} |x| \ge 2t+1. \end{cases}$$

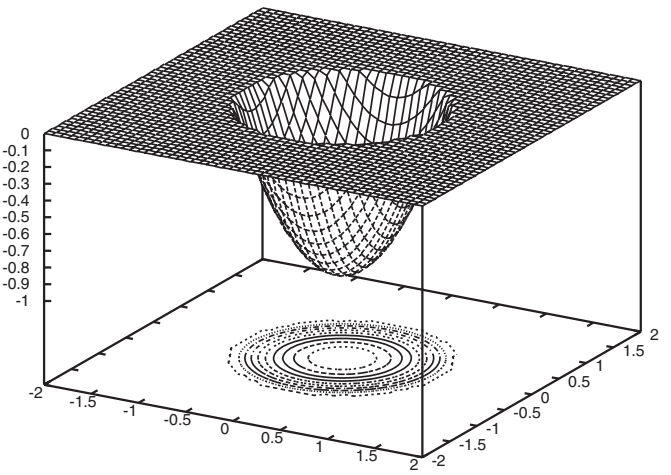
The approximate solution is shown in Figure 5 at $t = 0$ and $t = 0.5$ with a 50×50 nodes computational grid. Table 7 reports a strong reduction of the error with respect to the previous test. This shows that in the case of semiconcave solutions, numerical schemes can take advantage of higher consistency rates, although the piecewise quadratic structure of the solution is too simple to be really discriminating in this respect. Among different smoothness indicators, the best accuracy is achieved by D2+D3 indicator, whereas other choices yield similar results.

Test 4: Two-dimensional periodic solution. This test (taken again from [6], [17], [20]) deals with the HJ equation

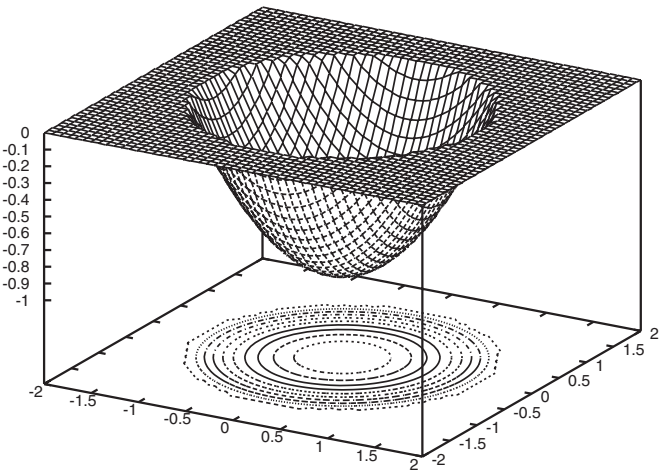
$$\begin{cases} v_t(x,t) + \frac{(v_{x_1}(x,t) + v_{x_2}(x,t)+1)^2}{2} = 0, \\ v(x,0) = v_0(x) = -\cos\frac{\pi(x_1+x_2)}{2}, \end{cases} \tag{6.7}$$

considered in $[-2,2]^2$, with doubly periodic boundary conditions.

In this situation, the structure of the solution is more complex, and we have tested the various schemes in terms of convergence rate using computational meshes



(a)



(b)

FIG. 5. *Approximate solutions for test* 3. (a) *Initial condition.* (b) *Solution at* $t = 0.5$.

of 25×25, 50×50, 100×100, and 200×200 nodes. The approximate solution has been computed before the singularity, at $T = 0.8/\pi^2$, with 4 time steps, and after the singularity, at $T = 1.5/\pi^2$, with 5 time steps. The Euler method has been used to approximate characteristics and reconstruction has been performed by the WENO Shu version. Relative errors are reported in Tables 5–6, whereas Figure 6 shows the approximate solution plotted at $t = 0$ and $t = T = 1.5/\pi^2$.

Error tables confirm the theoretical convergence rate of the schemes in the presence of a smooth solution, whereas, after the singularity, convergence rate of the WENO5 scheme is heavily underestimated by the theory (we suppose this might be due to particular circumstances of the test).

The comparison among the different schemes, in Table 7, shows (at fixed 50×50 grid and after the singularity) similar performances except for a loss in accuracy of the WENO3 D2 scheme. At this level the fifth order scheme has no better performance,

TABLE 5
Errors before and after the singularity for test 4, *WENO*3 *scheme.*

WENO 3rd order				
	Before singularity		After singularity	
N	Relative L^∞-error	L^∞ order	Relative L^∞-error	L^∞ order
25	3.00×10^{-3}		1.00×10^{-2}	
50	4.66×10^{-4}	2.6	8.68×10^{-5}	6.8
100	2.68×10^{-5}	4.1	9.35×10^{-6}	3.2
200	1.48×10^{-6}	4.1	2.97×10^{-7}	4.4

TABLE 6
Errors before and after the singularity for test 4, *WENO*3 *scheme.*

WENO 5th order				
	Before singularity		After singularity	
N	Relative L^∞-error	L^∞ order	Relative L^∞-error	L^∞ order
25	1.28×10^{-3}		8.55×10^{-3}	
50	4.89×10^{-5}	4.3	8.53×10^{-4}	3.6
100	2.07×10^{-6}	4.5	2.08×10^{-6}	8.6
200	2.37×10^{-8}	6.4	5.34×10^{-9}	8.6

TABLE 7
Comparison of relative L^∞-error for different schemes on tests 1–4.

Scheme	Test 1	Test 2	Test 3	Test 4
P_1	5.93×10^{-3}	3.79×10^{-2}	1.20×10^{-2}	1.23×10^{-2}
P_2	2.47×10^{-4}	1.69×10^{-2}	5.82×10^{-3}	7.24×10^{-4}
Cubic	5.12×10^{-5}	2.54×10^{-2}	1.20×10^{-2}	3.18×10^{-4}
ENO 3	3.51×10^{-5}	2.75×10^{-2}	5.80×10^{-4}	1.43×10^{-4}
WENO 2/3 S	5.12×10^{-5}	2.69×10^{-2}	1.20×10^{-2}	8.68×10^{-5}
WENO 2/3 D2	6.81×10^{-4}	3.15×10^{-2}	1.20×10^{-2}	1.29×10^{-3}
WENO 3/5 S	5.83×10^{-6}	2.52×10^{-2}	3.17×10^{-3}	8.53×10^{-4}
WENO 3/5 D2	7.52×10^{-6}	2.58×10^{-2}	1.14×10^{-3}	1.45×10^{-4}
WENO 3/5 D3	3.00×10^{-6}	2.97×10^{-2}	3.34×10^{-3}	4.13×10^{-5}
WENO 3/5 D2+D3	6.17×10^{-5}	2.68×10^{-2}	8.73×10^{-5}	1.78×10^{-4}

although looking at Table 5 we see that it would have a great improvement passing from 50 to 100 nodes.

Conclusions. We have built a class of WENO large time-step schemes based on an SL formulation. Under suitable assumptions, convergence of the schemes can be proved up to the ninth order. Theoretical consistency analysis is generally confirmed by numerical experiments, even when singular solutions are taken into account. Among all the smoothness indicators considered, there has been now and then a considerably better or worse accuracy for some of them. The original Shu indicator, however, seems to be a robust and accurate choice in all the cases tested.

While this work has been focused on the special case of (1.1), this class of schemes can be immediately adapted (see the appendix in [4]) to treat dynamic programming equations of the Bellman type

$$v + \max_{a \in A} \{ -b(x,a) \cdot Dv - f(x,a) \} = 0 \tag{6.8}$$

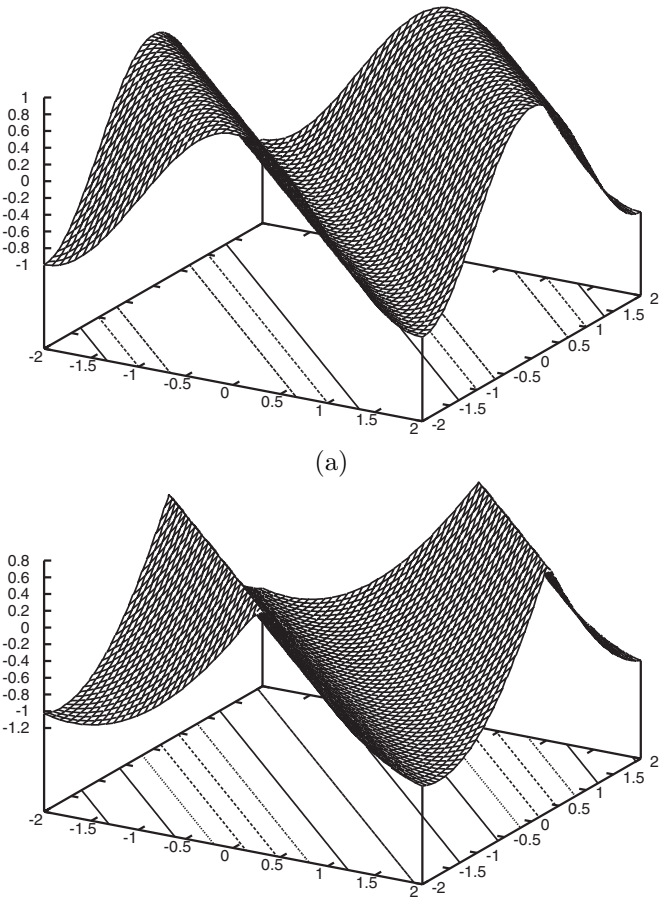


FIG. 6. *Approximate solutions for test* 4. (a) *Initial condition.* (b) *Solution at* $t = 1.5/\pi^2$.

(which leads to a convex Hamiltonian and includes (1.1) as a special case) or, in even greater generality, of the nonconvex Isaacs type

$$v + \min_{a_2 \in A_2} \max_{a_1 \in A_1} \{ -b(x, a_1, a_2) \cdot Dv - f(x, a_1, a_2) \} = 0, \tag{6.9}$$

although in such settings no complete theory exists on the high-order case.

The advantage of WENO over ENO reconstructions is apparent in terms of CPU time, while the accuracy in both cases seems to be comparable with the accuracy of linear Lagrange reconstructions. WENO techniques can be valuable, however, when special information associated to singularities (e.g., switching curves for dynamic programming equations) requires a nonoscillatory behavior of the approximation scheme.

REFERENCES

[1] R. ABGRALL, *On essentially non-oscillatory schemes on unstructured meshes: Analysis and implementation,* J. Comput. Phys., 114 (1994), pp. 45–58.

[2] R. ABGRALL, *Numerical discretization of the first-order Hamilton–Jacobi equation on triangular meshes,* Comm. Pure Appl. Math., 49 (1996), pp. 1339–1373.

[3] M. Abramowitz and I.A. Stegun, eds., *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables,* National Bureau of Standards, Washington, D.C., 1964.
[4] M. Bardi and I. Capuzzo Dolcetta, *Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations,* Birkhäuser Boston, Boston, 1997.
[5] G. Barles, *Solutions de viscositè des equations d'Hamilton–Jacobi,* Springer-Verlag, Berlin, 1998.
[6] S. Bryson and D. Levy, *Central schemes for multidimensional Hamilton–Jacobi equations,* SIAM J. Sci. Comput., 25 (2003), pp. 767–791.
[7] S. Bryson and D. Levy, *High-order central schemes for multidimensional Hamilton–Jacobi equations,* SIAM J. Numer. Anal., 41 (2004), pp. 1339–1369.
[8] R. Courant, E. Isaacson, and M. Rees, *On the solution of nonlinear hyperbolic differential equations by finite differences,* Comm. Pure Appl. Math., 5 (1952), pp. 243–255.
[9] M. Falcone, *Numerical solution of dynamic programming equations,* in Optimal Control and Viscosity Solutions of Hamilton–Jacobi–Bellman Equations, Appendix A, Birkhäuser Boston, Boston, 1997.
[10] M. Falcone and R. Ferretti, *Discrete-time high-order schemes for viscosity solutions of Hamilton–Jacobi equations,* Numer. Math., 67 (1994), pp. 315–344.
[11] M. Falcone and R. Ferretti, *Semi-Lagrangian schemes for Hamilton–Jacobi equations, discrete representation formulae and Godunov methods,* J. Comput. Phys., 175 (2002), pp. 559–575.
[12] M. Falcone and T. Giorgi, *An approximation scheme for evolutive Hamilton–Jacobi equations,* in Stochastic Analysis, Control, Optimization and Applications: A Volume in Honor of W.H. Fleming, W.M. McEneaney, G. Yin, and Q. Zhang, eds., Birkhäuser Boston, Boston, 1999, pp. 289–303.
[13] R. Ferretti, *Convergence of semi-Lagrangian approximations to convex Hamilton–Jacobi equations under (very) large Courant numbers,* SIAM J. Numer. Anal., 40 (2003), pp. 2240–2253.
[14] A. Harten and S. Osher, *Uniformly high-order accurate nonoscillatory schemes.* I, SIAM J. Numer. Anal., 24 (1987), pp. 279–309.
[15] A. Harten, B. Engquist, S. Osher, and S.R. Chakravarthy, *Uniformly high-order accurate essentially nonoscillatory schemes.* III, J. Comput. Phys., 71 (1987), pp. 231–303.
[16] C. Hu and C.-W. Shu, *Weighted essentially non-oscillatory schemes on triangular meshes,* J. Comput. Phys., 150 (1999), pp. 97–127.
[17] G.S. Jiang and D. Peng, *Weighted ENO schemes for Hamilton–Jacobi equations,* SIAM J. Sci. Comput., 21 (2000), pp. 2126–2143.
[18] G.S. Jiang and C.-W. Shu, *Efficient implementation of weighted ENO schemes,* J. Comput. Phys., 126 (1996), pp. 202–228.
[19] R.J. LeVeque, *Numerical Methods for Conservation Laws,* Birkhäuser, Basel, 1992.
[20] C.T. Lin and E. Tadmor, *High-resolution nonoscillatory central schemes for Hamilton–Jacobi equations,* SIAM J. Sci. Comput., 21 (2000), pp. 2163–2186.
[21] P.L. Lions, *Generalized Solutions of Hamilton–Jacobi Equations,* Pitman, London, 1982.
[22] X.-D. Liu, S. Osher, and T. Chan, *Weighted essentially non-oscillatory schemes,* J. Comput. Phys., 115 (1994), pp. 200–212.
[23] S. Osher and J.A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton–Jacobi formulations,* J. Comput. Phys., 79 (1988), pp. 12–49.
[24] S. Osher and C.-W. Shu, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations,* SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[25] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws,* in Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture Notes in Math. 1697, Papers from the C.I.M.E. Summer School held in Cetraro, June 23–28, 1997, A. Quarteroni, ed., Springer-Verlag, Berlin; Centro Internazionale Matematico Estivo (C.I.M.E.), Florence, 1998, pp. 325–432.
[26] J. Shi, C. Hu, and C.-W. Shu, *A technique of treating negative weights in WENO schemes,* J. Comput. Phys., 175 (2002), pp. 108–127.
[27] A.N. Staniforth and J. Côtè, *Semi-Lagrangian integration schemes for atmospheric models: A review,* Mon. Weather Rev., 119 (1991), pp. 2206–2223.
[28] J. Strain, *Semi-Lagrangian methods for level set equations,* J. Comput. Phys., 151 (1999), pp. 498–533.
[29] Y.-T. Zhang and C.-W. Shu, *High-order WENO schemes for Hamilton–Jacobi equations on triangular meshes,* SIAM J. Sci. Comput., 24 (2002), pp. 1005–1030.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1092–1117
© 2005 Society for Industrial and Applied Mathematics

COMPONENT-AVERAGED ROW PROJECTIONS: A ROBUST, BLOCK-PARALLEL SCHEME FOR SPARSE LINEAR SYSTEMS*

DAN GORDON† AND RACHEL GORDON‡

Abstract. A new method for the parallel solution of large sparse linear systems is introduced. It proceeds by dividing the equations into blocks and operating in block-parallel iterative mode; i.e., all the blocks are processed in parallel, and the partial results are "merged" to form the next iterate. The new scheme performs Kaczmarz row projections within the blocks and merges the results by certain *component-averaging* operations—hence it is called component-averaged row projections, or CARP. The system matrix can be general, nonsymmetric, and ill-conditioned, and the division into blocks is unrestricted. For partial differential equations (PDEs), if the blocks are domain-based, then only variables at the boundaries between domains are averaged, thereby minimizing data transfer between processors. CARP is very robust; its application to test cases of linear systems derived from PDEs shows that it converges in difficult cases where state-of-the-art methods fail. It is also very memory efficient and exhibits an almost linear speedup ratio, with efficiency greater than unity in some cases. A formal proof of convergence is presented: It is shown that the component-averaging operations are equivalent to row projections in a certain superspace, so the convergence properties of CARP are identical to those of Kaczmarz's algorithm in the superspace. CARP and its convergence proof also apply to the consistent convex feasibility problem.

Key words. block-parallel, component-averaging, convex feasibility problem, domain decomposition, linear equations, parallel processing, partial differential equations, row projections, sparse linear systems, superspace

AMS subject classifications. 15A06, 65F10, 65F50, 65N12, 65Y05, 90C51

DOI. 10.1137/040609458

1. Introduction.

Iterative methods for solving large sparse linear systems of equations are advantageous over the classical direct solvers, especially for huge systems. Methods of parallelizing iterative algorithms [20, 29] divide the equations into blocks and usually fall into one of two modes of operation. In the block-sequential (also called block-iterative) mode, the blocks are processed sequentially, but the computations on each block's equations are done in parallel. Examples of block-sequential methods are found in [1, 6, 10]. In the second mode of operation, sometimes referred to as block-parallel, the blocks are assigned to different processors to be processed in parallel, and the results from all the blocks are then combined in some manner to produce the next iterate. Typical examples of block-parallel schemes are parallel versions of Bi-CGSTAB [35], RGMRES [30], conjugate gradient (CG), and CG-squared (CGS) [31]; other examples include those in [2, 4, 9, 12, 13].

Kaczmarz's row projection method (KACZ) [23] was one of the first iterative methods used for large nonsymmetric systems. Its main advantages are robustness, guaranteed convergence on consistent systems, and cyclic convergence on inconsistent systems [8, 16, 32]. KACZ was also independently discovered in the context of image reconstruction from projections, where it is called ART (algebraic reconstruction technique) [21]. Kaczmarz's algorithm, by its nature and mathematical definition, is inherently sequential since, at each iterative step, the current iterate is projected

*Received by the editors June 4, 2004; accepted for publication (in revised form) April 21, 2005; published electronically December 30, 2005.
http://www.siam.org/journals/sisc/27-3/60945.html
†Department of Computer Science, University of Haifa, Haifa 31905, Israel (gordon@cs.haifa.ac.il).
‡Department of Aerospace Engineering, The Technion–Israel Institute of Technology, Haifa 32000, Israel (rgordon@tx.technion.ac.il).

onto the hyperplane defined by the next equation. In the context of partial differential equations (PDEs), current approaches to parallelizing KACZ have taken the block-sequential approach by replacing row projections with block projections and processing the blocks sequentially; see [6, 7, 24].

In this paper we present a block-parallel method of parallelizing Kaczmarz's algorithm for PDEs without block projections. Our approach stems from domain-decomposition considerations: After partitioning the domain into subdomains and assigning subdomains to processors, each processor is assigned the block of equations whose central node lies in its subdomain(s). Starting from some initial estimate, each processor performs a sequence of row projections on the hyperplanes defined by its block's equations, as in KACZ, and the results from the various blocks are then "merged" together to form the next iterate. The merge operation takes variables, which belong to two or more blocks, and replaces their different values (in the different domains) by their average. Thus, only variables bordering subdomain boundaries are averaged, so communication between processors is limited. Domain boundaries can even overlap, which means that the blocks of equations need not be disjoint.

We call this algorithm [illegible], or CARP. The principle of component-averaging also appeared, in a different form, in the CAV and BICAV algorithms of Censor, Gordon, and Gordon in [11, 10]. The main idea there was to utilize the sparsity of the system matrix of image-reconstruction problems. Note that the "string-averaging" algorithm [9, 12] also performs block-parallel row projections, but the resulting vectors of the separate blocks are combined by taking their regular (weighted) average.

CARP is completely general and places no restriction on the system matrix or the selection of the blocks. A formal proof of convergence is presented. It relies on a novel result which shows that the averaging operations are equivalent to row projections in a certain superspace. From this, it follows that the convergence properties of CARP are identical to those of Kaczmarz's algorithm in the superspace, and that CARP can also be used for nonsquare systems. CARP and its convergence proof, which uses a result of Aharoni and Censor [1], are also valid for the consistent case of the convex feasibility problem.

CARP provides a very robust parallel iterative method for solving large sparse linear systems derived from PDEs. It converges in difficult cases where state-of-the-art parallel versions of RGMRES, CGS, and Bi-CGSTAB fail. Since CARP is a linear iterative method, its convergence rate is theoretically limited; see [20, sec. 3.2.5]. Hence, it is not as efficient as some of the nonlinear methods to which it is compared in this paper. However, some of the nonlinear methods fail to converge on several of the test cases. In terms of its parallel behavior, CARP exhibits an almost linear speedup ratio, as well as efficiency greater than unity on some of the test cases.

CARP is memory efficient, compared to block-based projection methods such as those in [2, 3, 6, 7] or Krylov-based methods, since each processor operates on its equations one at a time. The memory requirement for each processor is just the submatrix of its local equation coefficients and the local part of the iterated solution vector; there is no need to store any other data related to a domain's submatrices, as in other methods. Furthermore, for many problems, there is no need even to store the equation coefficients since they can be calculated "on the fly" as needed. Another advantage is that there is no need to preprocess the system matrix, so CARP is suitable for quasi-linearization methods for nonlinear problems, in which the system matrix is modified repeatedly.

The rest of the paper is organized as follows. Section 2 places our work in relation

to previous work. Section 3 presents the mathematical background, the formulation of CARP, the convergence proofs, and some extensions. Section 4 presents numerical results comparing the new approach with state-of-the-art parallel techniques, and section 5 concludes with a discussion and future research directions.

2. Relation to previous work. Block projection methods (see Elfving [17]) use block projections instead of the row projections of KACZ [23] or Cimmino's algorithm [14]. Hence, such methods require computation and storage of data related to the inverses of submatrices. This data consists of Cholesky factors of the normal equation matrices or the LU factors of the augmented system matrices. As mentioned, this gives CARP an advantage with regard to memory and applicability to nonlinear systems.

Bramley and Sameh [6], following Kamath and Sameh [24], parallelized block KACZ with CG-acceleration under five partitioning schemes on regular grids. The method was very robust, but there was no clear-cut "best" partitioning scheme: The RP9 scheme produced the fastest runtime results on most of the test problems, but it requires inverse submatrix data, and it is unsuitable for unstructured grids. The RP27 scheme does not need submatrix data, but it performed best on only one problem and it is suitable only for structured grids. The cube-4 scheme is adaptable to unstructured grids, but it also uses data related to submatrix inverses, and its runtime is the worst or second worst of the schemes tested.

In [7], the above approach was extended to include also CG-acceleration of block-Cimmino and a new projection method called V-RP. These methods, using the RP9 scheme from [6], were compared with RGMRES and CGNE, both with and without preconditioning. Arioli et al. [2] studied parallel CG-acceleration of block-Cimmino for different block partitionings. Their study was restricted to block-tridiagonal systems, though their technique may be applicable to other cases. This method also requires data related to submatrix inverses, with the above disadvantages. In [3], this research was extended to include block-Lanczos acceleration, which improved the runtime performance (over the CG-acceleration) for some ill-conditioned systems.

Three of the most successful and commonly used parallel techniques for large sparse nonsymmetric systems are RGMRES [29], CGS [31], and Bi-CGSTAB [35]. These methods, which can be used with or without preconditioners, often converge faster than the (linear) projection methods. However, they are not as robust as row projection methods and fail in some difficult cases.

As mentioned, the convergence proof of CARP relies on extending the Euclidean space into a certain superspace. Pierra [27, 28] introduced the concept of extending the Euclidean space into a product space, and this idea was used by Combettes [15] to prove the convergence of Cimmino's algorithm. In the product space extension, each coordinate is "extended" to a fixed number of coordinates in the superspace. Our technique of a superspace extension is a generalization of the product space concept, allowing extensions of only some of the coordinates, as well as different-sized extensions for different coordinates.

3. Mathematical development.

3.1. Background, motivation, and presentation of CARP. Consider the following system of m linear equations in n variables:

$$\sum_{j=1}^{n} x_j a_j^i \;=\; b_i \;\text{ for }\; 1 \le i \le m, \quad \text{or, in matrix form: } \; Ax \;=\; b. \tag{1}$$

In KACZ, each iterate is projected onto a hyperplane (defined by one of the equations) in a cyclic manner. KACZ has been studied very extensively, both theoretically and experimentally. Its convergence with relaxation parameters, for consistent systems, has been shown by Herman, Lent, and Lutz [22] and by Trummer [33]. Tanabe [32] proved that when the system is inconsistent, it converges [illegible]; i.e., for each hyperplane, the sequence of projections on that hyperplane converges to a limit. See also [8] for further results in that direction.

Throughout the rest of the paper we assume that every column of A is nonzero. Such a column, if it exists, can be removed as a preliminary step since it corresponds in effect to coefficients of a "fictitious" variable, i.e., one whose value can be arbitrary. For $1 \le i \le m$, we denote by a^i the ith row vector of the matrix A in (1) and by $\langle u, v \rangle$ the dot product of two vectors u, v. KACZ can be described as follows: Starting from an arbitrary point x^0 in the n-dimensional Euclidean space, the kth iterate x^k is projected toward the next hyperplane, and the hyperplanes are chosen in cyclic order. To simplify our notation, we denote by $i(k)$ the kth index taken cyclically from 1 to n, i.e., $i(k) = (k \bmod m) + 1$. The regular KACZ algorithm (with relaxation parameters) is the following.

ALGORITHM 1 (KACZ).

Initialization: $x^0 \in \mathbb{R}^n$ [illegible]

Iterative step: [illegible] x^k [illegible]

$$x^{k+1} = x^k + \lambda_k \frac{b_{i(k)} - \langle a^{i(k)}, x^k \rangle}{\|a^{i(k)}\|^2} a^{i(k)} . \tag{2}$$

If we assume that the equations are normalized, i.e., for $1 \le i \le n$, the ith equation is divided by $\|a^i\|$, then (2) can be replaced by

$$x^{k+1} = x^k + \lambda_k \left(b_{i(k)} - \langle a^{i(k)}, x^k \rangle \right) a^{i(k)} . \tag{3}$$

It is easy to see that the sequence of points produced by KACZ is identical to the one obtained by first normalizing the equations and replacing (2) by (3).

The motivation for CARP was to parallelize KACZ for PDEs using domain decomposition, but without using block projections with their inherent limitations. This led to a block-parallel method in which the domain is divided into subdomains (possibly overlapping), which can be assigned to different processors. Each processor is assigned the block of those equations whose central node lies in its subdomain(s). Starting from some initial estimate, the following two main steps are repeated until convergence: In the first step, consecutively executed row projections are performed in every block (as in KACZ) on all the equations of the block; in fact, more than one sweep can be carried out. In the second step, every variable shared by two or more blocks is replaced by the average of its values in the separate blocks. Thus, only variables bordering a neighboring domain are averaged. The above approach is formalized as follows.

The equations of the linear system (1) are divided into blocks, $B_1, \ldots, B_t$, which are not necessarily disjoint. For each $1 \le j \le n$, denote by I_j the index set of the blocks which contain an equation with a nonzero coefficient of x_j; i.e., $I_j = \{1 \le q \le t \mid x_j$ has a nonzero coefficient in some equation of $B_q\}$. Let $s_j = |I_j|$ (the size of I_j). The following definition explains and formalizes the merge operation of CARP.

DEFINITION 1. [illegible] $B = \{B_1, \ldots, B_t\}$ [illegible] B [illegible] $\mathrm{CA}_B : (\mathbb{R}^n)^t \to (\mathbb{R}^n)$ [illegible]

$x^1, \ldots, x^t \in \mathbb{R}^n$ ... $\mathrm{CA}_B(x^1, \ldots, x^t)$ ... $\mathbb{R}^n$ ... j ...

$$\mathrm{CA}_B(x^1, \ldots, x^t)_j \;=\; \frac{1}{s_j} \sum_{q \in I_j} x_j^q \;,$$

... x_j^q ... j ... x^q ... $1 \le q \le t$

We can now present the CARP algorithm. In the following, given a block of equations B_q and a point $x \in \mathbb{R}^n$, we use the term "KACZ sweep" to refer to the operation of successively projecting x onto the hyperplanes defined by the equations of B_q, as in KACZ (Algorithm 1); this is formalized as follows.

DEFINITION 2. ... B_q $1 \le q \le t$ ... (1)
... $\{i_1, \ldots, i_r\}$ ... B_q ... $x \in \mathbb{R}^n$ ...
... $\mathrm{KSWP}(B_q, x) : \mathbb{R}^n \to \mathbb{R}^n$ ... $x^0 = x$ ... $0 \le j < r$ ...

$$x^{j+1} \;=\; x^j \;+\; \lambda_j \frac{b_{i_j} - \langle a^{i_j}, x^j \rangle}{\|a^{i_j}\|^2} a^{i_j} \;,$$

... λ_j ... $\mathrm{KSWP}(B_q, x) = x^r$

ALGORITHM 2 (CARP).
Initialization: $x^0 = (x_1^0, \ldots, x_n^0) \in \mathbb{R}^n$ ...
Iterative step: ... x^k, ...
1. ... $1 \le q \le t$, ... B_q. ... $1 \le q \le t$, ... $\bar{x}^q$. ... $p_k \ge 1$, $\bar{x}^q = \mathrm{KSWP}^{p_k}(B_q, x^k)$.
2. $x^{k+1} = \mathrm{CA}_B(\bar{x}^1, \ldots, \bar{x}^t)$.

Appendix A provides a detailed implementation of CARP, allowing separate processors for the different blocks.

3.2. Convergence proofs. The proof of convergence of CARP proceeds along the following lines. We first transform the system (1) into a system of equations in some superspace $\mathbb{R}^s$of $\mathbb{R}^n$. In $\mathbb{R}^s$, the equations belonging to different blocks do not share any common variables, so the parallel processing of the separate blocks is equivalent to regular sweeps of KACZ in $\mathbb{R}^s$. We then prove a general ..., according to which, operations of component averaging are equivalent to certain row projections. From this it follows that CARP is just KACZ in $\mathbb{R}^s$.

Assume w.l.o.g. that for some $1 \le r \le n$ the variables $x_1, \ldots, x_r$ are exactly the ones shared by two or more blocks, i.e., $s_1, \ldots, s_r \ge 2$ and $s_{r+1} = \cdots = s_n = 1$. Denote $s = \sum_{j=1}^n s_j$, and consider the space $\mathbb{R}^s$. In order to simplify our notation, we will use y instead of x for vectors and components of $\mathbb{R}^s$, and index vectors of $\mathbb{R}^s$ as follows:

$$y \;=\; \Big(\underbrace{y_{1,1}, \ldots, y_{1,s_1}}_{s_1 \text{ elements}}, \;\cdots\;, \underbrace{y_{r,1}, \ldots, y_{r,s_r}}_{s_r \text{ elements}}, \; \underbrace{y_{r+1}, \ldots, y_n}_{n-r \text{ elements}} \Big) \,.$$

We now define the following ... $E : \mathbb{R}^n \to \mathbb{R}^s$:

$$E(x_1, \ldots, x_n) \;=\; (y_{1,1}, \ldots, y_{1,s_1}, \;\cdots\;, y_{r,1}, \ldots, y_{r,s_r}, \; y_{r+1}, \ldots, y_n) \,,$$

where $y_{j,1} = \cdots = y_{j,s_j} = x_j$ for $1 \le j \le r$ and $y_j = x_j$ for $r < j \le n$.

Each I_j (the index set of the blocks which contain a nonzero coefficient of x_j) has s_j elements, so it can be represented as $I_j = \{q_{j,1}, q_{j,2}, \ldots, q_{j,s_j}\}$. For every $1 \le j \le n$ and $1 \le \ell \le s_j$, variables of the form $y_{j,\ell}$ will be used to "represent" the "copy" of x_j in the block $B_{q_{j,\ell}}$, as follows: For $1 \le j \le r$ and $1 \le \ell \le s_j$, replace x_j by $y_{j,\ell}$ in block $B_{q_{j,\ell}}$. For $1 \le j \le r$, replace x_j by y_j in all the equations.

Let $B'_1, \ldots, B'_t$ denote the blocks of equations obtained from $B_1, \ldots, B_t$, respectively, by the above replacements. Clearly, in the new blocks, there are no variables belonging to two or more blocks, even if there were shared equations in the original blocks. Denote $B' = \bigcup_{q=1}^{t} B'_q$; this set of equations is defined in the space $\mathbb{R}^s$. We now present an alternative formulation of CARP in which the basic operations are performed in $\mathbb{R}^s$.

ALGORITHM 3 (alternate CARP).

Initialization:

1. $x^0 = (x_1^0, \ldots, x_n^0) \in \mathbb{R}^n$ [illegible]
2. $y^0 = E(x^0)$ [illegible] $\mathbb{R}^s$
3. [illegible] B'

Iterative step: [illegible] y^k

1. [illegible] $1 \le q \le t$ [illegible] B'_q [illegible] B'_q [illegible] $p_k \ge 1$ [illegible] $\bar{y}^q = \mathrm{KSWP}^{p_k}(B'_q, y^k)$
2. (a) [illegible] $1 \le j \le r$ [illegible]

$$y_{j,1}^{k+1} = \cdots = y_{j,s_j}^{k+1} = \frac{1}{s_j}\left(\sum_{\ell=1}^{s_j} \bar{y}_{j,\ell}^{\,q_{j,\ell}}\right).$$

(b) [illegible] $r < j \le n$ $I_j = \{q_{j,1}\}$ [illegible] $y_j^{k+1} = \bar{y}_j^{q_{j,1}}$

(c) [illegible] $y^{k+1} = \left(y_{1,1}^{k+1}, \ldots, y_{1,s_1}^{k+1}, \ldots, y_{r,1}^{k+1}, \ldots, y_{r,s_r}^{k+1}, y_{r+1}^{k+1}, \ldots, y_n^{k+1}\right)$

Final step [illegible] K [illegible] $x^* = (x_1^*, \ldots, x_n^*)$ [illegible]

$$x_j^* = \begin{cases} y_{j,1}^K & \text{[illegible] } 1 \le j \le r, \\ y_j^K & \text{[illegible] } r < j \le n. \end{cases}$$

We can see that other than the initial and final steps, the entire algorithm can be executed iteratively or in parallel in $\mathbb{R}^s$, since the blocks B'_q do not contain any common variables. It follows that iterative step 1 is one (or more) KACZ sweeps on the equations of B'. Furthermore, it is clear that the result (after some K iterative steps) of this alternative form of CARP is the same as that of the regular CARP, assuming that the same relaxation parameters and repetition factors (λ_k and p_k) are used.

The following lemma will be used to show that the averaging operations in iterative step 2 are row projections in $\mathbb{R}^s$.

LEMMA 1 (the averaging lemma). [illegible] $1 \le m \le n$ $y^0 = (y_1^0, \ldots, y_n^0) \in \mathbb{R}^n$ [illegible] $y^1 = (y_1^1, \ldots, y_n^1) \in \mathbb{R}^n$ [illegible] $y_i^1 = (y_1^0 + \cdots + y_m^0)/m$ [illegible] $1 \le i \le m$ [illegible] $y_i^1 = y_i^0$ [illegible] $m < i \le n$ [illegible] y^1 [illegible] y^0 [illegible] $m - 1$ [illegible] $\mathbb{R}^n$ [illegible] KACZ [illegible]. The proof proceeds by induction on m. For $m = 1$, there is nothing to prove. For $m = 2$, project y^0 onto the plane defined by the equation $-y_1 + y_2 = 0$. The vector of coefficients is $a = (-1, 1, 0, \ldots, 0)$, the square of its norm is $\|a\|^2 = 2$,

and the projection $y' = (y'_1, y'_2, \dots)$ is

$$\begin{aligned} y' &= y^0 \; - \; \frac{1}{2}\langle y^0, a\rangle a \\ &= (y^0_1, y^0_2, \dots) \; - \; \frac{1}{2}(-y^0_1 + y^0_2)(-1, 1, 0, \dots, 0) \\ &= \left(\frac{1}{2}(y^0_1 + y^0_2),\; \frac{1}{2}(y^0_1 + y^0_2),\; y^0_3,\; y^0_4, \dots\right). \end{aligned}$$

In other words, for $m = 2$, we obtained $y'_1 = y'_2 = (y^0_1 + y^0_2)/2$ (and left the other components unchanged) by performing one orthogonal projection on a suitable hyperplane.

We assume that the statement is true for m, and prove it for $m + 1$: Let $y^0 = (y^0_1, \dots, y^0_n)$. Project y^0 onto the hyperplane defined by the equation $-y_1 - y_2 - \cdots - y_m + my_{m+1} = 0$. The vector of coefficients is $a = (-1, \dots, -1, m, 0, \dots, 0)$ and the square of its norm is $\|a\|^2 = m + m^2 = m(m + 1)$. The projection is the point $y' = (y'_1, \dots, y'_n)$ defined by

$$\begin{aligned} y' &= y^0 \; - \; \frac{\langle y^0, a\rangle}{m(m+1)}\, a \\ &= (y^0_1, \dots, y^0_n) \; - \; \frac{-y^0_1 - \cdots - y^0_m + my^0_{m+1}}{m(m+1)}\; (-1, \dots, -1, m, 0, \dots, 0)\,. \end{aligned}$$

For each $1 \le i \le m$, we have

$$y'_i \;=\; y^0_i \;-\; \frac{1}{m(m+1)}\left(\sum_{j=1}^{m} y^0_j \;-\; my^0_{m+1}\right),$$

and the $(m + 1)$st coefficient is

$$\begin{aligned} y'_{m+1} &= y^0_{m+1} \;+\; \frac{y^0_1 + \cdots + y^0_m - my^0_{m+1}}{m(m+1)}\, m \\ &= \frac{1}{m+1}\left(y^0_1 + \cdots + y^0_m\right) \;+\; \left(1 - \frac{m}{m+1}\right) y^0_{m+1} \\ &= \frac{1}{m+1}\left(y^0_1 + \cdots + y^0_m\right) \;+\; \frac{m+1-m}{m+1}\, y^0_{m+1} \\ &= \frac{1}{m+1}\left(y^0_1 + \cdots + y^0_m + y^0_{m+1}\right). \end{aligned}$$

In other words, y'_{m+1} already has the required value. For $m + 1 < i \le n$, y^0_i is unchanged, i.e., $y'_i = y^0_i$. To set the other variables to the same value, we rely on the induction hypothesis, according to which we can set

$$y''_1 \;=\; \cdots \;=\; y''_m \;=\; \frac{1}{m}\left(y'_1 + \cdots + y'_m\right)$$

by using $m - 1$ orthogonal projections on hyperplanes of $\mathbb{R}^n$. Note also that by the induction hypothesis, all the other variables will remain unchanged, i.e., $y''_i = y'_i$ for $m + 1 \le i \le n$.

For each $1 \le k \le m$, we have

$$\begin{aligned}
y_k'' &= \frac{1}{m}\sum_{i=1}^{m} y_i' \\
&= \frac{1}{m}\sum_{i=1}^{m}\left[y_i^0 \;-\; \frac{1}{m(m+1)}\left(\sum_{j=1}^{m} y_j^0 \;-\; m y_{m+1}^0\right)\right] \\
&= \frac{1}{m}\sum_{i=1}^{m} y_i^0 \;-\; \frac{1}{m^2(m+1)}\sum_{i=1}^{m}\left[\sum_{j=1}^{m} y_j^0 \;-\; m y_{m+1}^0\right] \\
&= \frac{1}{m}\sum_{i=1}^{m} y_i^0 \;-\; \frac{m}{m^2(m+1)}\left(\sum_{j=1}^{m} y_j^0 \;-\; m y_{m+1}^0\right) \\
&= \frac{1}{m}\sum_{i=1}^{m} y_i^0 \;-\; \frac{1}{m(m+1)}\sum_{j=1}^{m} y_j^0 \;+\; \frac{1}{m+1}\, y_{m+1}^0 \\
&= \left(\frac{1}{m} - \frac{1}{m(m+1)}\right)\sum_{i=1}^{m} y_i^0 \;+\; \frac{1}{m+1}\, y_{m+1}^0 \\
&= \frac{1}{m+1}\sum_{i=1}^{m} y_i^0 \;+\; \frac{1}{m+1}\, y_{m+1}^0 \\
&= \frac{1}{m+1}\sum_{i=1}^{m+1} y_i^0 \,.
\end{aligned}$$

This proves the induction hypothesis and the lemma. □

We shall refer to the equations required by CARP for part 2 of its iterative step as the "averaging" equations. We also denote by B'' the system of equations consisting of B' together with all the averaging equations. As a consequence of the averaging lemma, we have the following.

THEOREM 1. [illegible] (1) [illegible] B'' [illegible] (1) [illegible] λ_k [illegible] $\epsilon < \lambda_k < 2-\epsilon$ [illegible] ϵ [illegible] p_k [illegible] (1) [illegible] 1. [illegible] 2. [illegible] B'' [illegible]. Assume that the system (1) is consistent. It follows immediately that the set of equations of B' is also consistent. In B', each of the original variables x_j belonging to two or more blocks is replaced by a set of variables of the form $y_{j,1}, \ldots, y_{j,s_j}$, so the equations of B', together with the set of equations $\{y_{j,1} = y_{j,2}, \ldots, y_{j,s_{j-1}} = y_{j,s_j} \mid 1 \le j \le r\}$, form a consistent set of equations. Note now that the averaging equations are mathematically equivalent to the above set of equations. Hence, the system B'' (consisting of B' together with the averaging equations) forms a consistent system of linear equations in $\mathbb{R}^s$.

Furthermore, note that the number of averaging equations is exactly equal to $s-n$, which is the number of additional coordinates we get in going from $\mathbb{R}^n$ to $\mathbb{R}^s$. This means that the number of additional equations (in going from B' to B'') is exactly equal to the number of additional variables. In particular, if the system matrix of (1) is square, then the system matrix of B'' is also square. We refer now to the results of Aharoni and Censor [1], according to which KACZ converges on a consistent system even if the projections are not performed cyclically; all that is required is that each equation should be used infinitely often. This allows us to perform, in iterative step 1 of CARP, any positive number of internal sweeps in each block. See subsection 3.4 for details of the results of [1].

Assume now that the system (1) is inconsistent. Tanabe [32] shows that KACZ converges cyclically when there are no relaxation parameters (i.e., $\lambda = 1$). The extension of this result to cyclic relaxation parameters follows from Eggermont, Herman, and Lent [16, Thm. 3.1], from which our result for inconsistent systems follows. ∎

Note that when the averaging operations are considered as row projections, they require a relaxation parameter $\lambda = 1$. Theorem 1 allows us to use $\lambda \neq 1$ for the other equations, in an inconsistent system, provided the projections follow a cyclic pattern and the relaxation parameter associated with every equation is fixed.

The generality of [1, Thm. 1], in the consistent case, allows us to modify the blocks, the number of inner iterations, and the relaxation parameters during the course of CARP, provided the conditions for [1, Thm. 1] are observed. This could be useful in a setting in which the number of processors can vary during the course of a computation.

3.3. CARP1: Blocks of one equation. A special case of CARP, which we call CARP1, occurs when every equation is taken as a separate block. In this case, s_j is simply the number of equations in which x_j has a nonzero coefficient, and the iterative step of CARP1 is given by the equation

$$x_j^{k+1} \;=\; x_j^k \;+\; \frac{\lambda_k}{s_j} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i \;. \tag{4}$$

CARP1 can be implemented on a multiprocessor system in block-parallel mode, though the implementation blocks are not necessarily single equations—they are simply (disjoint) blocks of equations assigned to different processors for the sake of runtime efficiency. Furthermore, the implementation blocks of CARP1 are not intrinsic to the algorithm as in the general CARP. This implementation is detailed in Appendix B.

Equation (4) is similar to the well-known Cimmino algorithm [14], whose iterative step is

$$x_j^{k+1} \;=\; x_j^k \;+\; \frac{\lambda_k}{m} \sum_{i=1}^{m} \frac{b_i - \langle a^i, x^k \rangle}{\|a^i\|_2^2} a_j^i \;. \tag{5}$$

Cimmino's algorithm was the initial motivational step leading to the CAV algorithm [11]: It was observed that when system (1) is sparse, then in the sum of (5) only a "small" number of elements are nonzero, but the sum is divided by the "large" m. This led to an attempt to replace m by s_j, resulting in the iterative step of [11, eq. 1.9], which is identical to (4). Note that even though CARP1 already appeared in [11] (without any specific name), no proof of convergence was given there; naturally,

the convergence properties of CARP1 follow from Theorem 1.[1]

In [11], CARP1 was examined on test cases of image reconstruction from projections. Its initial rate of convergence was identical to that of CAV, but the images obtained were somewhat inferior to those obtained with CAV. Similar comparative results were obtained in [10] with BICAV (a block-sequential version of CAV) and a block-sequential version of CARP1.

CAV made use of sparsity-oriented weights, but differently from CARP: With s_j defined as for CARP1, we define a [illegible] $w_i = 1/(\sum_{j=1}^{n} s_j (a_j^i)^2)$ for $1 \le i \le m$. The iterative step of CAV is given by

$$x^{k+1} = x^k + \lambda_k \sum_{i=1}^{m} w_i \left(b_i - \langle a^i, x^k \rangle\right) a^i \,. \tag{6}$$

Hence, the projections in CAV are orthogonal to the hyperplanes and weighted by w_i (in addition to λ_k). CAV and CARP1 are identical only when all the s_j's are equal.

Note that in CARP1, as given by (4), x^{k+1} can be viewed as the average of certain projections of x^k, but these projections are [illegible], in general, orthogonal to the hyperplanes: Assume for simplicity that the equations are normalized and that $\lambda_k = 1$; then from (4) we get

$$x_j^{k+1} = \frac{1}{m} \sum_{i=1}^{m} \left[x_j^k + \frac{m}{s_j} \left(b_i - \langle a^i, x^k \rangle\right) a_j^i \right] \,. \tag{7}$$

It follows that x^{k+1} is the average of a set of vectors p^i, $1 \le i \le m$, where the jth component of p^i is the summand of (7). If we consider p^i to be a projection of x^k in some direction "toward" the hyperplane defined by the ith equation, then the [illegible] of this projection is $p^i - x^k$. However, from (7), we see that the jth component of this direction vector is $m(bi - \langle a^i, x^k \rangle) a_j^i / s_j$, so the direction vector is not colinear with a^i (unless all the s_j's are equal). Hence, p^i is not, in general, an orthogonal projection of x^k in the direction of the ith hyperplane.

3.4. Applicability of CARP to the convex feasibility problem. Aharoni and Censor [1] proved a general theorem regarding orthogonal projections onto convex sets. This result, together with the averaging lemma, led to the proof of Theorem 1. We shall show that Theorem 1 also applies to the consistent case of the convex feasibility problem (CFP), from which it follows that CARP is also applicable to the CFP.

Some preliminary definitions are needed: Let $I = \{1, 2, \ldots, m\}$ be a set of integers. A function $w : I \to \mathbb{R}$ is called a [illegible] if $w(i) \ge 0$ for all $i \in I$ and $\sum_{i \in I} w(i) = 1$. An infinite sequence of weight functions $\{w_k\}_{k \ge 0}$ is called [illegible] if for every $i \in I$, $w_k(i) > 0$ infinitely often; i.e., the inequality holds for infinitely many values of k.

Let $\{Q_i \mid i \in I\}$ be a finite family of closed convex sets in $\mathbb{R}^n$ such that $Q = \bigcap_{i \in I} Q_i \ne \emptyset$, and let $P_i : \mathbb{R}^n \to Q_i$ be defined as the [illegible] onto Q_i; i.e., for $x \in \mathbb{R}^n$, $P_i(x) = \arg\min\{\|x - y\| \mid y \in Q_i\}$. Now, given a weight function w, we define $P_w : \mathbb{R}^n \to \mathbb{R}^n$ by $P_w(x) = \sum_{i \in I} w(i) P_i(x)$.

[1] While revising this paper, the authors learned that Censor, Elfving, and Herman independently proved that CARP1, with weights, converges for both consistent and inconsistent systems.

Consider now the [illegible] (BIP) algorithm [1, Alg. 1], which starts from an arbitrary point $x^0 \in \mathbb{R}^n$ and whose iterative step is given by

$$x^{k+1} \;=\; x^k \;+\; \lambda_k \left(P_{w_k}(x^k) - x^k \right), \tag{8}$$

where $\{w_k\}_{k\geq 0}$ is a fair sequence of weight functions and $\{\lambda_k\}_{k\geq 0}$ is a sequence of user-determined relaxation parameters. We can see from (6) and (8) that when the convex sets Q_i are hyperplanes, then CAV and BIP have the same form, though the sparsity weights of CAV do not necessarily satisfy the conditions for the weights of BIP.

In [1, Thm. 1], it is shown that if for some $\epsilon > 0$, $\epsilon < \lambda_k < 2 - \epsilon$ for all k, then the sequence of points $\{x^k\}_{k\geq 0}$ generated by the BIP algorithm converges to a point $x^* \in \widehat{Q}$, where $\widehat{Q} = \bigcap_{i\in \widehat{I}} Q_i$ and $\widehat{I} = \{i \in I \mid \sum_{k=0}^{\ell} w_k(i) \;\underset{\ell\to\infty}{\Longrightarrow}\; \infty\}$. (If $\widehat{I} = \emptyset$, then $\widehat{Q}$ is defined as $\mathbb{R}^n$.) The choices of blocks and weight functions determine the behavior of BIP—from sequential to block-sequential and to simultaneous; see [1] for details.

To apply CARP to the CFP, we divide the set of projections (onto the convex sets) into blocks, which need not be disjoint. Starting from an arbitrary initial point, we perform, for each block of projections, a sequence of orthogonal projections onto the convex sets of the block. This is done in parallel on all the blocks. The results from the separate blocks are then merged by component-averaging, as in the CARP algorithm for linear equations, to form the next iterate. The variables that are averaged are only those which are affected by projections belonging to two or more blocks.

The convergence proof for the consistent case is the same as in Theorem 1. This follows from the simple observation that, in the superspace, the hyperplanes defined by the averaging equations are also convex sets. It is easily seen that if the original convex sets have a nonempty intersection, then in the superspace, the convex sets and the hyperplanes defined by the averaging equations also have a nonempty intersection.

4. Numerical results and discussion. Tests were run on two different types of parallel machines: A shared memory SGI origin 2000, and a distributed machine—a PC cluster running under Linux. The PC cluster consists of 16 single Pentium IV, 2.8GHz processors with 1GB memory each, connected with a dedicated 1Gb/sec. ethernet switch. Results were obtained on both machines on six test problems from the literature. The problems were run on the SGI and the PC cluster with 1, 2, 4, 8, and 16 processors, and also with 32 and 64 processors on the SGI machine. This section describes the test problems, the nonsymmetric solvers to which CARP was compared, numerical considerations and implementation, the stopping tests, and the numerical results and discussion.

4.1. The test problems. We examined six PDE problems proposed by Bramley and Sameh [7]. These problems were collected from a variety of sources [18, 24, 25, 26] and were later used as test problems in several other works [2, 3]. The problems are typical of many fields, such as computational fluid dynamics (CFD), heat transfer, and structural mechanics, and are commonly used in a wide variety of industrial and engineering applications.

Each problem has an assigned analytic solution which is used to determine the boundary conditions. Furthermore, the analytic solution enables the calculation of the norm of the error. In the following, we use the common notation $\Delta u = u_{xx}+u_{yy}+u_{zz}$. Our test problems and their predetermined solutions are the following.

1. $\Delta u + 1000 u_x = F$.
2. $\Delta u + 1000 \exp(xyz)(u_x + u_y - u_z) = F$.

3. $\Delta u + 100xu_x - yu_y + zu_z + 100(x+y+z)u/xyz = F$.
4. $\Delta u - 10^5 x^2(u_x + u_y + u_z) = F$.
5. $\Delta u - 1000(1+x^2)u_x + 100(u_y + u_z) = F$.
6. $\Delta u - 1000\,[(1-2x)u_x + (1-2y)u_y + (1-2z)u_z] = F$.

The analytical (preassigned) solutions for the above problems are the following:
Problem 1: $u(x,y,z) = xyz(1-x)(1-y)(1-z)$.
Problem 2: $u(x,y,z) = x+y+z$.
Problems 3–6: $u(x,y,z) = \exp(xyz)\sin(\pi x)\sin(\pi y)\sin(\pi z)$.
The expression for the right-hand-side function F in all six problems is computed analytically for each case, using the preassigned solution.

The test problems were solved on the unit cube domain ($[0,1]\times[0,1]\times[0,1]$) using a seven-point centered difference scheme. The equations were discretized using the same number of grid points in each direction. Results are shown for problems of size $40\times40\times40 = 64{,}000$ equations and of size $80\times80\times80 = 512{,}000$ equations.

4.2. Parallel methods used for comparison. The common wisdom today is that there is no "best" solver for ,, problems. Each class or type of problem has its own most effective solver. Since we propose CARP as a general-purpose solver, we chose to compare it with other general-purpose, state-of-the-art methods, even though better solvers exist for each particular problem. Our methods of choice were three parallel Krylov-based iterative nonsymmetric solvers: RGMRES [30], CGS [31], and Bi-CGSTAB [35]. Each of these methods was used by itself and with the following preconditioners: ILUT, Jacobi, Neumann, and least-squares. We also tried the symmetric Gauss–Seidel preconditioner, but it failed completely in all cases. The comparisons used the AZTEC software library [34], which is designed for the parallel solution of large sparse linear systems of equations.

RGMRES(k) was run with Krylov subspace size $k = 10$; larger values of k do not improve the robustness of RGMRES(k) until k becomes a significant fraction of the problem size (with a large increase of the required storage). The Jacobi, Neumann, and least-squares polynomial preconditioners were run with AZTEC's default polynomial order parameter of 3.

The ILUT preconditioner [29] depends on two parameters which strongly affect its behavior: the ,, , (the value below which elements are taken as zero), and the ,, , which controls the maximum number of nonzeros allowed in each column/row of the incomplete LU factors. Similarly to the other preconditioners, we used the default AZTEC values for ILUT's parameters: drop tolerance = 0 and fill-in = 1.0 (i.e., no additional elements). These default values produced excellent results, so we found no need to experiment with them. Moreover, a thorough study of ILUT should also consider recent improvements, such as reorderings, scalings, and multilevel extensions, all of which go beyond the scope of the present study. Note that ILUT is essentially sequential, and its parallel implementation in AZTEC is based on using it separately on the submatrices of the subdomains.

4.3. Implementation details. In order to implement CARP efficiently on the given problems, we used the domain-decomposition approach and mapped different domains to different processors. Although other approaches are feasible in theory, this approach limits interprocessor communications because only variables bordering a neighboring domain need to be averaged (and hence exchanged between processors).

Both AZTEC and our implementation of CARP used the message passing interface (MPI) library for exchanging messages between the processors. We used SGI's proprietary implementation of MPI on the SGI machine and the public domain

MPICH routines on the PC cluster. CARP was implemented with the MPI Cartesian topology routines for interprocessor communications; these routines are designed for six-way nearest neighbor communications for domain-based data.

Numerical PDE approximations on three-dimensional grids (structured and unstructured) exhibit spatial locality, since each equation centered about a grid point involves only its neighboring grid points. This feature enables domain-decomposition approaches to parallel solvers, in which the coefficient matrix is distributed among the processors with local indexing of submatrices. Besides the memory savings, this allows efficient basic matrix-vector operations. For structured three-dimensional grids, the resulting coefficient matrix is a seven-point stencil matrix.

Our implementation of the data structures used by CARP is similar to that of AZTEC. It is aimed at facilitating the implementation of the communication tasks and gaining efficiency during the iterative procedure. In a preprocessing stage, the following information is saved in each processor:

1. a list of neighboring processors with which it communicates,
2. a list of local nodes that are coupled with external nodes,
3. local representation of the submatrix used by each processor.

The local submatrix in each processor was stored in the sparse matrix format called the "distributed modified sparse row" (DMSR) format; see [34]. This format, which is a generalization of the MSR format [29], stores only the nonzero elements of the coefficient matrix.

In CARP, one can perform any finite number of inner KACZ iterations inside a block before averaging the boundary values that are common to neighboring blocks. Such inner iterations are fast relative to the outer iterations since they do not use interprocessor communications. This allows us to choose the number of inner iterations so as to minimize the overall computation time. Naturally, this choice depends on the specific problem and the machine architecture. Note that when implementing CARP, the residual can be computed efficiently as part of the iterative procedure; see (3).

4.4. Stopping tests. There are several stopping criteria which one may apply to iterative systems. Our stopping criterion was to use the L_2 norm of the residual: $||b - Ax||_2 < \tau$, where τ is a preassigned tolerance. Since this stopping criterion depends on the scaling of the equations, we first normalized the equations by dividing each equation by the norm of its coefficient vector. For problems 1, 2, 4, 5, and 6, τ was defined by setting $\tau^2 = 10^{-11}$ ($\tau \approx 3.16 \times 10^{-5}$). For problem 3, τ was taken as $\tau = 2.3 \times 10^{-3}$. The reason for this difference is that in problem 3, CARP converged well until this residual value was reached, and then continued to converge too slowly to be practically significant. All the other methods failed to converge on problem 3.

In order to limit the time taken by the methods implemented in AZTEC, the maximum number of iterations was set to 5000. The AZTEC library has several other built-in stopping criteria: numerical breakdown, numerical loss of precision, and numerical ill-conditioning.

4.5. Results and discussion.

4.5.1. Main results. In all our runs the initial estimate x^0 was taken as the zero vector. Table 1 presents a summary of the convergence status of all the methods and preconditioners that were examined, for the six test problems, for 512,000 equations on 16 processors. In the case of convergence failure, it also indicates the reason for the failure. The results shown in the table are also valid for four processors with 64,000 and 512,000 equations, unless noted otherwise.

TABLE 1
Convergence status of the six problems on 16 *processors with* 512,000 *equations.*

Method	Problem number					
	1	2	3	4	5	6
CARP	**conv**	**conv**	**conv**	**conv**	**conv**	**conv**
RGMRES	**conv**	**conv**	stgn	**conv**	**conv**	**conv**
RGMRES + ILUT	**conv**	stgn[4]	stgn	nbdn[5,6]	**conv**	stgn
RGMRES + Jacobi	mxit	mxit	divg[3]	mxit	mxit	mxit
RGMRES + Neum.	**conv**	**conv**	stgn	**conv**	**conv**	**conv**
RGMRES + LSQ	**conv**	**conv**	stgn	**conv**	**conv**	mxit[1]
CGS	divg[1]	divg[1]	divg	divg	divg	divg[1]
CGS + ILUT	**conv**	divg	divg	divg	**conv**	**conv**[2]
CGS + Jacobi	divg	divg	divg	divg	divg	divg
CGS + Neum.	**conv**	divg[1]	divg	nbdn[1]	divg[3]	**conv**[4]
CGS + LSQ	divg[1]	divg[1]	divg	nbdn[2]	divg	divg[1]
Bi-CGSTAB	nbdn	nbdn	nbdn	nbdn	nbdn	nbdn
Bi-CGSTAB + ILUT	**conv**	divg	divg	divg[4]	**conv**	**conv**[3]
Bi-CGSTAB + Jacobi	nbdn	nbdn	divg[3]	nbdn	nbdn	nbdn
Bi-CGSTAB + Neum.	**conv**[3]	nbdn	nbdn	nbdn	nbdn	nbdn
Bi-CGSTAB + LSQ	nbdn	nbdn	nbdn	nbdn	nbdn	nbdn

Notation: **conv** – convergence, divg – divergence, mxit – max. iter. reached, nbdn – num. breakdown, stgn – stagnation.

Notes: Status also valid for 4 proc. with 64,000 and 512,000 eqns., unless noted: [1] **conv** with 64,000 eqns. [2] divg with 64,000 eqns. [3] nbdn with 64,000 eqns. [4] nbdn with 512,000 eqns. [5] stgn with 64,000 eqns. [6] stgn with 512,000 eqns.

Table 1 demonstrates the robustness of CARP—it converged in all cases, including problem 3, in which all the other methods failed. However, it should be noted that in problem 3, the residual decreased very slowly after it was reduced to 3×10^{-3}. Of the other methods tested, RGMRES is generally the most successful; it converged by itself (and with some of the preconditioners) in all the problems except problem 3.

Figures 1–6 present a comparison of the (L_2 norm of the) residual versus the execution time for the six test problems, for 64,000 equations, as obtained on the SGI machine with four processors. For each problem, results are presented for all the methods that converged (except for problems 1 and 2 with CGS and CGS+LSQ, which converged extremely slowly). The CARP results are for the optimal relaxation parameters. These figures indicate that there is no "best" solver for all cases; however, RGMRES, either by itself or with ILUT, is among the fastest methods, while ILUT is the most successful preconditioner.

We also tried out the following parallel methods:

- CARP1 performed poorly in all cases except number 3.
- CAV [11], although successful in problems of image reconstruction from projections, performed poorly in all cases except number 3, where it was identical to CARP1.
- String averaging [9, 12] also performs block-parallel row projections, but, as mentioned in the introduction, the results are merged by taking the weighted average of the separate results. This method performed poorly in all six problems, indicating the effectiveness of componentwise averaging in CARP.

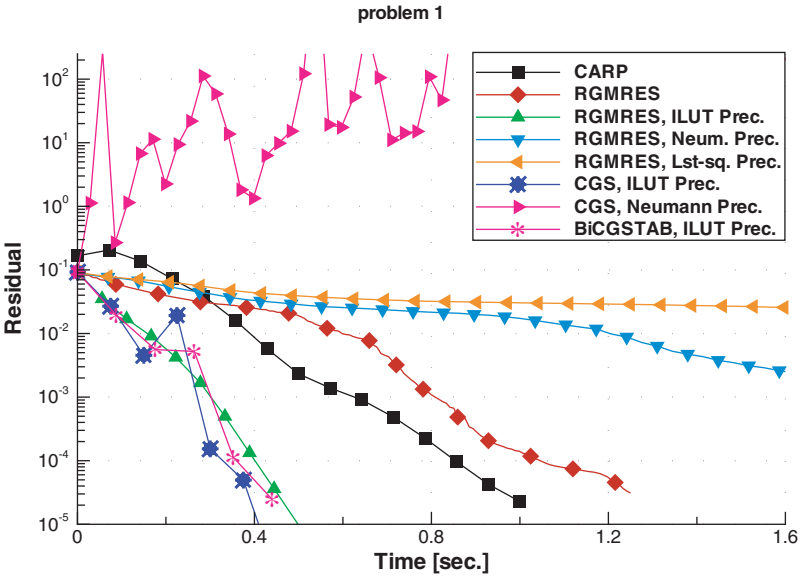


FIG. 1. *Execution times for problem* 1.

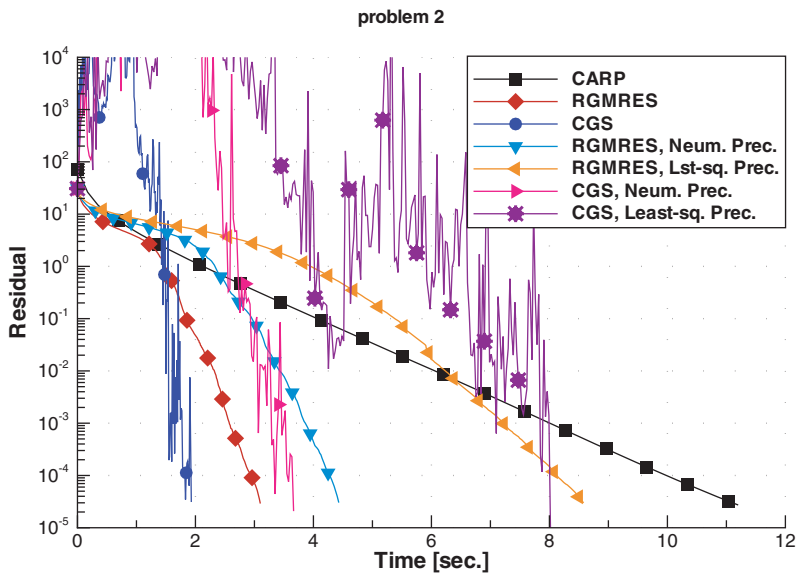


FIG. 2. *Execution times for problem* 2.

The convergence graphs for problem 3 compare CARP, CARP1, KACZ, and CAV. It can be seen that CARP's initial rate of convergence is very fast compared to the others, until a residual of approximately 10^{-2} is reached. From then on, CARP1, KACZ, and CAV exhibit a better rate of convergence. From this we can surmise that a hybrid algorithm, consisting initially of CARP and then CARP1 or KACZ or CAV, will function in this problem better than any single method. Problem 3 is hard

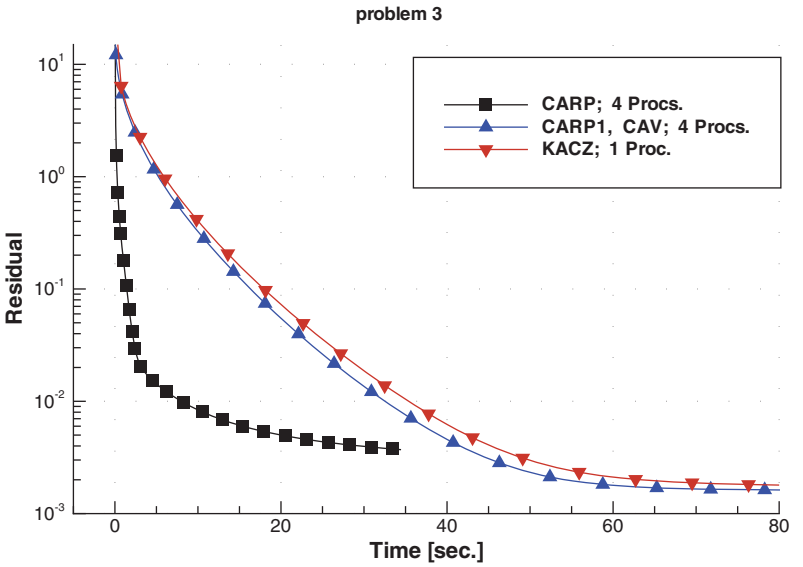


FIG. 3. *Execution times for problem* 3.

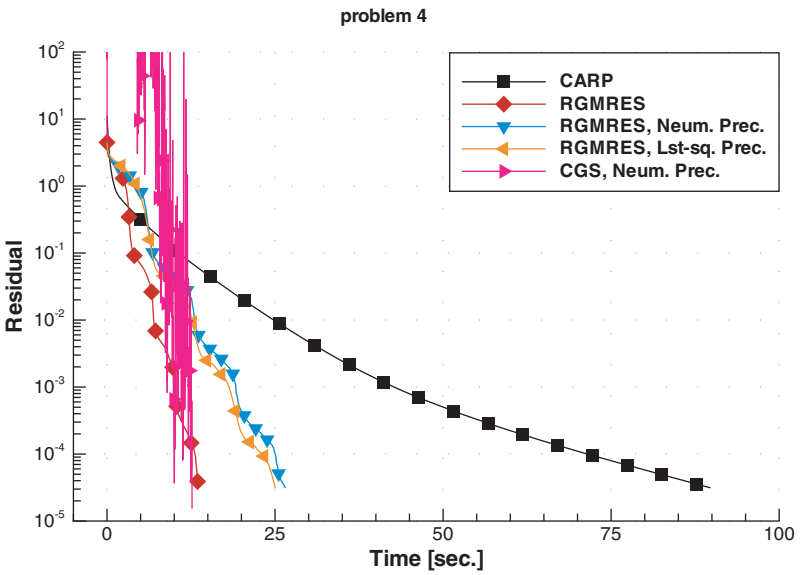


FIG. 4. *Execution times for problem* 4.

for solvers such as GMRES because it is indefinite, with eigenvalues surrounding the origin. It is well known that the eigenvalue distribution and the conditioning of the matrix of eigenvectors are more important for the convergence of GMRES than the conditioning of the system matrix. See also [2, 6, 7].

Tables 2 and 3 present various convergence measures obtained with CARP for 64,000 and 512,000 equations. Denoting by u and x the true solution and current

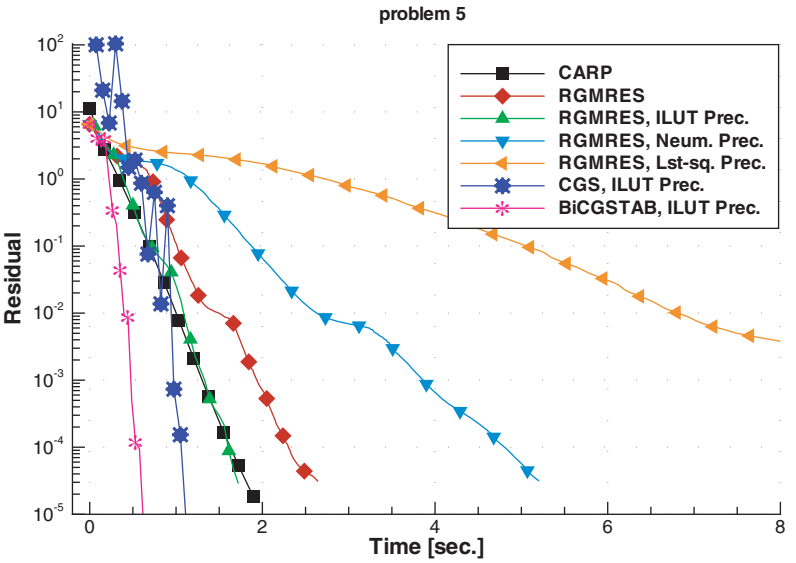


FIG. 5. *Execution times for problem* 5.

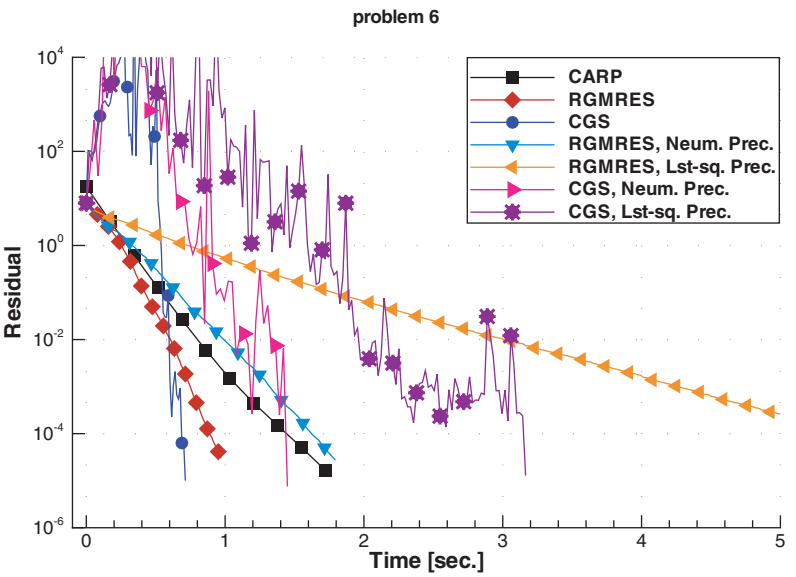


FIG. 6. *Execution times for problem* 6.

estimate, respectively, we computed the following convergence measures:

- The L_2 norm of the residual: $\text{RES}_2 = \|b - Ax\|_2$.
- The relative (or scaled) L_2 norm of the residual: $\text{REL-RES}_2 = \|b - Ax\|_2 / \|b\|_2$ (the initial estimate is taken as zero).
- The L_∞ norm of the residual: $\text{RES}_\infty = \|b - Ax\|_\infty$.
- The relative L_2 norm of the error: $\text{REL-ERR}_2 = \|u - x\|_2 / \|u\|_2$.
- The L_∞ norm of the error: $\text{ERR}_\infty = \|u - x\|_\infty$.

TABLE 2
Various convergence measures for four processors with 64,000 *equations.*

Problem:	1	2	3	4	5	6
RES_2	2.28E-5	2.72E-5	2.70E-3	3.13E-5	1.85E-5	1.66E-5
REL-RES_2	1.37E-4	3.78E-7	8.71E-5	2.87E-6	1.63E-6	9.29E-7
RES_∞	2.28E-6	1.81E-6	1.68E-4	1.21E-6	1.96E-6	1.98E-6
REL-ERR_2	8.19E-5	7.82E-7	8.30E-3	1.72E-3	1.16E-3	9.39E-4
ERR_∞	9.60E-6	2.24E-5	5.60E-2	7.80E-3	1.38E-3	2.19E-3

TABLE 3
Various convergence measures for four processors with 512,000 *equations.*

Problem:	1	2	3	4	5	6
RES_2	2.64E-5	2.93E-5	2.70E-3	3.16E-5	3.08E-5	2.80E-5
REL-RES_2	1.34E-4	2.10E-7	1.47E-4	2.06E-6	2.16E-6	1.37E-6
RES_∞	1.23E-6	1.98E-6	6.51E-5	1.09E-6	1.15E-6	1.36E-6
REL-ERR_2	9.56E-5	6.83E-7	8.93E-3	3.96E-4	2.96E-4	2.40E-4
ERR_∞	9.93E-6	2.74E-5	6.98E-2	2.05E-3	3.33E-4	3.05E-4

As can be seen from these tables, there is a wide variance between the measures achieved for the different problems, even though the convergence criterion for all of them (except problem 3) was taken as a fixed value for RES_2. For example, the achieved values of REL-RES_2 and REL-ERR_2 for problem 2 are much better than for the other problems. Also, REL-RES_2 for problem 3 is similar to that of problem 1, even though they differ greatly in RES_2. The consequence of the above is that there can be no absolute convergence criterion suitable for all cases.

Figures 7 and 8 present the speedup graphs of CARP for problems 1–6 with 512,000 equations, on the PC cluster and the SGI machine, respectively. The figures also include the plot corresponding to an efficiency of 1. It can be seen that all plots exhibit an almost linear behavior, and in some cases, the efficiency is greater than 1. This "super speedup" phenomenon is well known; it is due to the fact that in some cases, the total number of iterations required by CARP decreases as the number of blocks increases. A secondary contributing factor could be due to better utilization of memory cache when a problem of a certain size is divided among more processors.

4.5.2. Parametric studies.

Relaxation parameter $\boldsymbol{\lambda}$. Although the general theory allows us to vary λ during the iterative process, we chose a fixed value in our tests. Varying the value of λ during the iterations is a topic for further study. CARP's rate of convergence depends strongly on the choice of λ, and there is an optimal value for λ for each case, which we determined by numerical experimentation. This optimal value depends mainly on the problem being solved, and to a small extent, on the number of equations and the number of processors. Table 4 presents the optimal (fixed) relaxation parameters used for CARP, for the six test problems, with four processors.

Partitioning scheme. The initial strategy for choosing the partitioning scheme is to minimize the number of shared variables, but depending on the problem at hand, other partitioning schemes may yield better runtime results. The runtime and number of iterations required for convergence of CARP depend, for some problems, on the domain decomposition. Problems 1 and 5 are symmetric with respect to y and z but asymmetric with respect to x. Thus, a decomposition of the computational domain

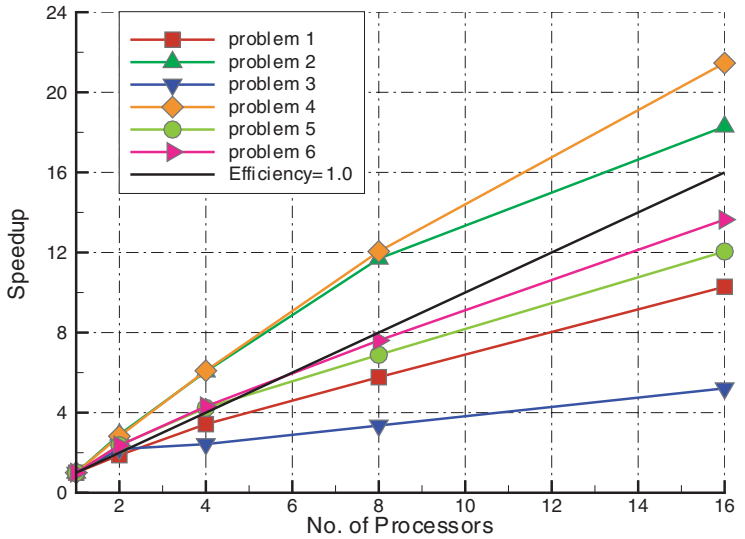


FIG. 7. *Speedup of CARP on the Linux cluster.*

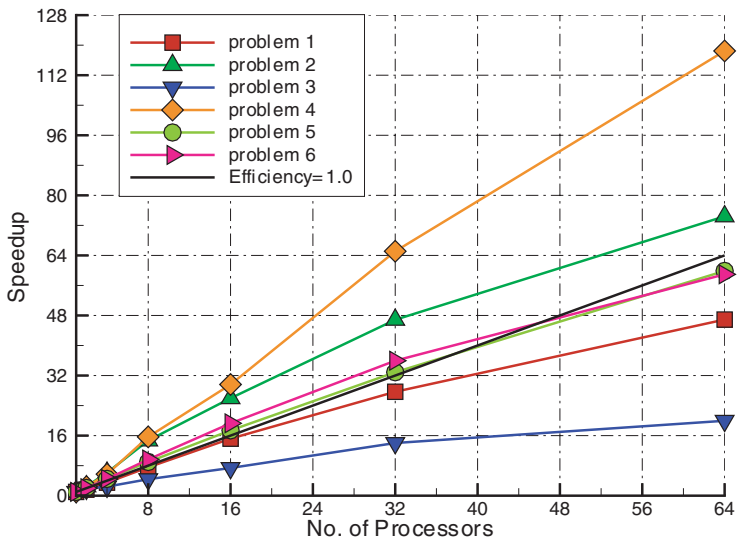


FIG. 8. *Speedup of CARP on the SGI machine.*

TABLE 4
Optimal relaxation parameter λ for CARP, for problems 1–6, with four processors.

Problem:	1	2	3	4	5	6
64,000 eqs.	1.90	1.90	1.60	1.50	1.85	1.35
512,000 eqs.	1.94	1.75	1.60	1.40	1.90	1.50

into four subdomains using a $4\times1\times1$ division in the x, y, z directions, respectively, requires considerably more iterations than a decomposition into $1\times4\times1$ or $1\times1\times4$ subdomains.

Inner iterations. Table 5 shows comparisons of the time and number of iterations required for convergence with one and four inner iterations, with different subdivisions, for problems 1 and 5 with 64,000 equations. In Tables 5 and 6, the number of iterations is the number of outer iterations, i.e., not counting the inner ones. The advantage of the $1\times4\times1$ and $1\times1\times4$ schemes over the $4\times1\times1$ scheme can be attributed to the fact that in the first two schemes, a relatively large number of "almost" parallel row vectors are placed in the same subdomain. Table 5 also shows that in some cases, one inner iteration is better than four, and in other cases, the opposite is true. For symmetric problems, there is no direction preference. The domain should be divided into (approximately) the same number of subdivisions in all directions to minimize the number of boundary nodes and reduce the communication time.

TABLE 5
Time and number of iterations required for different subdivision schemes for problems 1 *and* 5 *with* 64,000 *equations.*

Subdivision scheme	Problem 1				Problem 5			
	1 inner it.		4 inner it.		1 inner it.		4 inner it.	
	time	# it.	time	# it.	time	# it.	time	# it.
$4\times1\times1$	3.26	400	12.35	640	8.95	1050	17.11	880
$1\times4\times1$	**0.99**	**140**	1.28	70	3.42	480	**1.96**	**110**
$1\times1\times4$	**0.99**	**140**	1.28	70	3.56	500	2.07	110

Figures 9 and 10 present a more detailed view of the dependence of the number of iterations (required for convergence) on the relaxation parameter λ and the partitioning scheme for problems 1 and 6, respectively, for 512,000 equations. In both figures, the number of inner iterations was one, unless noted otherwise. Note in particular the shape of the graph for problem 1: It decreases linearly until the minimum and then "shoots" upward superlinearly. Figure 10 is typical for problems 2–6.

Table 6 summarizes the optimal values of the following parameters for the six problems, with 512,000 equations: the number of (outer) iterations, the relaxation parameter λ, the number of inner iterations, and the partitioning scheme.

Communication time. Table 7 provides the communication time required by CARP on problem 1 with 512,000 equations for the SGI machine and Linux cluster. This table emphasizes how the partitioning scheme affects the time required for communications: The $1\times1\times16$ scheme requires some 96,000 transfer edges (between adjacent grid points divided by a boundary), while the $1\times4\times4$ scheme requires only about 38,000 transfer edges. Also, the net computation time (total time minus communication time) for each machine is almost independent of the partitioning scheme. It follows that most of the time difference for this case is due to the communications overhead.

5. Conclusions and further research. CARP is a new block-parallel method for solving large sparse nonsymmetric, general linear systems of equations on parallel machines. It uses single-row projections within each block, as in Kaczmarz's algorithm [23], followed by merging of the block results using certain component-averaging operations. CARP is also applicable in principle to nonsquare systems,

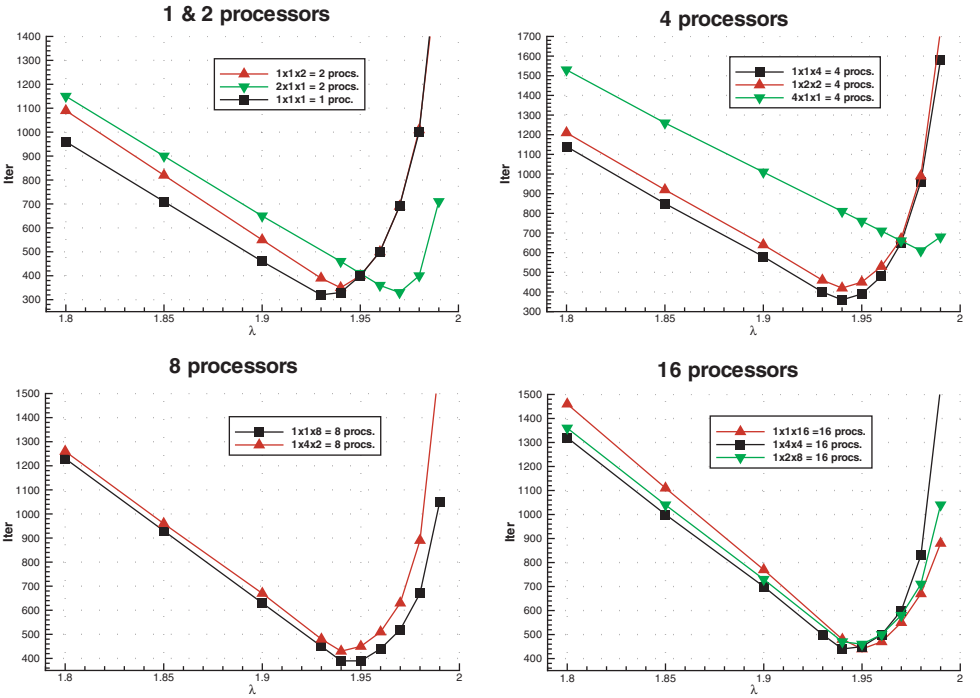


FIG. 9. *Number of iterations as a function of* λ *for problem* 1, 512,000 *equations.*

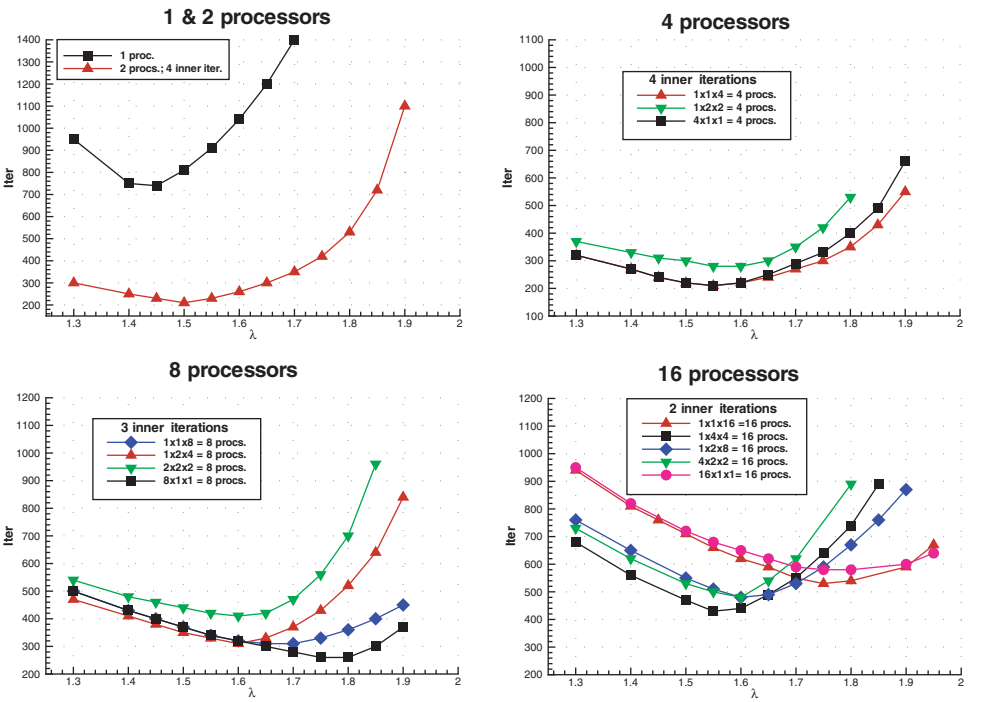


FIG. 10. *Number of iterations as a function of* λ *for problem* 6, 512,000 *equations.*

TABLE 6
Convergence details for CARP on the six problems with 512,000 *equations.*

No. of proc.		Problem no.					
		1	2	3	4	5	6
1	No. of iter.	330	6770	4200	59,600	1000	740
	λ	1.93	1.60	1.60	1.25	1.90	1.45
	Inner iter.	n/a	n/a	n/a	n/a	n/a	n/a
	Partition	1×1×1	1×1×1	1×1×1	1×1×1	1×1×1	1×1×1
2	No. of iter.	350	1,590	980	11,460	360	210
	λ	1.94	1.65	1.60	1.40	1.90	1.50
	Inner iter.	1	4	5	5	3	4
	Partition	1×1×2	1×2×1	2×1×1	2×1×1	1×2×1	1×2×1
4	No. of iter.	360	1,440	2,700	10,230	360	210
	λ	1.94	1.75	1.60	1.40	1.90	1.55
	Inner iter.	1	4	5	5	3	4
	Partition	1×1×4	1×2×2	2×2×1	4×1×1	1×1×4	1×1×4
8	No. of iter.	390	1,840	4,790	10,020	550	260
	λ	1.94	1.85	1.60	1.50	1.90	1.75
	Inner iter.	1	3	5	5	2	3
	Partition	1×1×8	1×4×2	2×2×2	2×4×1	1×1×8	8×1×1
16	No. of iter.	440	3,030	7,530	13,560	630	430
	λ	1.94	1.80	1.60	1.50	1.90	1.55
	Inner iter.	1	2	5	4	2	2
	Partition	1×4×4	2×4×2	2×4×2	1×4×4	1×2×8	1×4×4

TABLE 7
Communication time for problem 1 *with* 512,000 *equations.*

Configuration	SGI			Linux cluster		
	Total time	Comm. time	Comm. perc.	Total time	Comm. time	Comm. perc.
4 proc., 1×1×4	38.67	4.52	11.7%	19.50	3.31	17%
16 proc., 1×1×16	14.35	6.53	45.4%	8.49	3.50	41.2%
16 proc., 1×4×4	9.69	2.17	22.4%	6.24	1.52	24.4%

converges cyclically on inconsistent systems, and is applicable to the consistent case of the convex feasibility problem. A formal proof of convergence shows that CARP is equivalent to Kaczmarz's algorithm in a certain superspace. The new method is robust, memory efficient, and simple to program. It is also suitable for quasi-linearization approaches for nonlinear systems since there is no need to compute and store data related to submatrix inverses after every update of the system matrix.

CARP was compared with other state-of-the-art nonsymmetric solvers, with and without preconditioners, on six well-known test problems and on two different types of parallel machines. Test runs indicate that it is extremely robust compared to these methods, but since CARP is a linear method, its runtime, compared to nonlinear methods, is somewhat limited on some of the problems. On the other hand, CARP exhibits a linear speedup ratio, with efficiency sometimes greater than unity. These results indicate that CARP deserves a place in any robust iterative solution package for parallel machines.

One should note that the optimal performance of CARP depends on the choice of the relaxation parameter and the number of inner iterations in each block, and these are not known a priori. However, the rate of convergence behaves as a simple function of the relaxation parameters (just one local extremum). Hence, an adaptive search scheme can handle this problem: The processors are set to work for a small number of iterations on different values of these parameters, and the initial rates of convergence are compared; the search space is then narrowed by using the most promising parameter values.

Currently, work is in progress to speed up CARP's runtime efficiency by incorporating CG acceleration schemes, with very good results [19]. This is done by applying the techniques of [5] to the Kaczmarz scheme in the superspace. Another potential avenue of research would be to replace Kaczmarz in each block by other solution methods, such as CGNR or CGNE, on the block's submatrix. The solutions from the blocks can then be merged as in CARP. Such a scheme would require a different convergence proof. Another potential for speedup could be the implementation of communication hiding, i.e., information transmission between processors done in parallel with computations. Future research on CARP will also study its applicability to other areas of scientific computing.

Appendix A. Implementation details for CARP.

1. The equations of (1) are divided into blocks $B_1, \ldots, B_t$, which are not necessarily disjoint. A processor P_q is assigned to B_q for $1 \le q \le t$.
2. The (normalized) system (1) and the s_j's are sent to the processors, but each processor retains only the equations belonging to its block. For $1 \le q \le t$, denote $J_q = \{j \mid \text{in some equation of } B_q, \text{ the coefficient of } x_j \neq 0\}$. Processor P_q retains only those s_j's for which $j \in J_q$.
3. Every processor P_q, for every $j \in J_q$, determines if it is the "main owner" of x_j. For every j, there should be exactly one main owner of x_j, even if x_j has nonzero coefficients in several blocks. The main owner of x_j will need access to and from any other processor P_r for which $j \in J_r$.
4. The initial estimate $x^0 = (x_1^0, \ldots, x_n^0)$ is sent to the processors, but each processor P_q holds only the variables x_j^0 for $j \in J_q$.
5. Every processor P_q, for $1 \le q \le t$, sets the iteration index $k = 1$ and repeats the following steps until convergence:
   (a) Perform some finite number of KACZ sweeps on the equations of B_q and denote the result by $\bar{x}^q$; i.e., for some $p_k \ge 1$, $\bar{x}^q = \mathrm{KSWP}^{p_k}(B_q, x^k)$.
   (b) For every $j \in J_q$: If P_q is not the main owner of x_j, send $\bar{x}_j^q$ to the main owner of x_j; if P_q is the main owner of x_j, receive the values $\bar{x}_j^r$ from all other processors P_r for which $j \in J_r$.
   (c) For every $j \in J_q$, if P_q is the main owner of x_j, compute the jth component of the next iterate:

$$x_j^{k+1} = \begin{cases} \bar{x}_j^q & \text{if } s_j = 1, \\ \dfrac{1}{s_j} \displaystyle\sum_{j \in J_r} \bar{x}_j^r & \text{if } s_j > 1. \end{cases}$$

   Note that the sum includes the case $j \in J_q$.
   (d) For every $j \in J_q$, if P_q is the main owner of x_j, send x_j^{k+1} to all other processors P_r for which $j \in J_r$.
   (e) Advance the iteration index k by 1.

6. The last iterate is sent by the processors to the master processor.

Appendix B. Block-parallel implementation of CARP1.

1. The equations of (1) are partitioned into disjoint blocks by partitioning the index set $\{1, 2, \ldots, m\}$ into disjoint sets $I_1, \ldots, I_t$. A processor P_q is assigned to the equations of I_q for $1 \le q \le t$.
2. The (normalized) system (1) and the s_j's are sent to the processors, but each processor retains only the equations belonging to its block. For $1 \le q \le t$, denote $J_q = \{j \mid \text{in some equation of } I_q, \text{ the coefficient of } x_j \neq 0\}$. Processor P_q retains only those s_j's for which $j \in J_q$.
3. Every processor P_q, for every $j \in J_q$, determines if it is the "main owner" of x_j. For every j, there should be exactly one main owner of x_j, even if x_j has nonzero coefficients in several blocks. The main owner of x_j will need access to and from any other processor P_r for which $j \in J_r$.
4. The initial estimate $x^0 = (x_1^0, \ldots, x_n^0)$ is sent to the processors, but each processor P_q holds only the variables x_j^0 for $j \in J_q$.
5. Every processor sets the iteration index $k = 0$.
6. Every processor P_q, for $1 \le q \le t$, repeats the following steps until convergence:
   (a) For every $j \in J_q$, compute the value
   $$d_{q,j} \;=\; \sum_{i \in I_q} \left(b_i - \langle a^i, x^k \rangle\right) a_j^i \;.$$
   (b) For every $j \in J_q$: If P_q is not the main owner of x_j, send $d_{q,j}$ to the main owner of x_j; if P_q is the main owner of x_j, receive the values $d_{r,j}$ from all other processors P_r for which $j \in J_r$.
   (c) For every $j \in J_q$, if P_q is the main owner of x_j, compute the jth component of the next iterate:
   $$x_j^{k+1} \;=\; x_j^k \;+\; \frac{\lambda_k}{s_j} \sum_{j \in J_r} d_{r,j} \;.$$
   Note that the sum includes the case $j \in J_q$.
   (d) For every $j \in J_q$, if P_q is the main owner of x_j, send x_j^{k+1} to all other processors P_r for which $j \in J_r$.
   (e) Advance the iteration index k by 1.
7. The last iterate is sent by the processors to the master processor.

Acknowledgments. The authors are indebted to the anonymous reviewers, whose instructive and detailed comments led to a much improved presentation. Thanks are also due to Y. Censor for his helpful comments.

REFERENCES

[1] R. Aharoni and Y. Censor, *Block-iterative projection methods for parallel computation of solutions to convex feasibility problems*, Linear Algebra Appl., 120 (1989), pp. 165–175.

[2] M. Arioli, I. Duff, J. Noailles, and D. Ruiz, *A block projection method for sparse matrices*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 47–70.

[3] M. Arioli, I. S. Duff, D. Ruiz, and M. Sadkane, *Block Lanczos techniques for accelerating the block Cimmino method*, SIAM J. Sci. Comput., 16 (1995), pp. 1478–1511.

[4] Z.-Z. Bai, V. Migallón, J. Penadés, and D. B. Szyld, *Block and asynchronous two-stage methods for mildly nonlinear systems*, Numer. Math., 82 (1999), pp. 1–20.

[5] Å. Björck and T. Elfving, *Accelerated projection methods for computing pseudoinverse solutions of systems of linear equations*, BIT, 19 (1979), pp. 145–163.
[6] R. Bramley and A. Sameh, *Domain decomposition for parallel row projection algorithms*, Appl. Numer. Math., 8 (1991), pp. 303–315.
[7] R. Bramley and A. Sameh, *Row projection methods for large nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 168–193.
[8] Y. Censor, P. P. B. Eggermont, and D. Gordon, *Strong underrelaxation in Kaczmarz's method for inconsistent systems*, Numer. Math., 41 (1983), pp. 83–92.
[9] Y. Censor, T. Elfving, and G. T. Herman, *Averaging strings of sequential iterations for convex feasibility problems*, in Inherently Parallel Algorithms in Feasibility and Optimization and Their Applications, D. Butnariu, Y. Censor, and S. Reich, eds., Elsevier, Amsterdam, The Netherlands, 2001, pp. 101–114.
[10] Y. Censor, D. Gordon, and R. Gordon, *BICAV: A block-iterative parallel algorithm for sparse systems with pixel-dependent weighting*, IEEE Trans. Medical Imaging, 20 (2001), pp. 1050–1060.
[11] Y. Censor, D. Gordon, and R. Gordon, *Component averaging: An efficient iterative parallel algorithm for large and sparse unstructured problems*, Parallel Comput., 27 (2001), pp. 777–808.
[12] Y. Censor and E. Tom, *Convergence of string-averaging projection schemes for inconsistent convex feasibility problems*, Optim. Methods Software, 18 (2003), pp. 543–554.
[13] Y. Chen, *An accelerated block-parallel Newton method via overlapped partitioning*, in Proceedings of the 15th International Conference on Domain Decomposition Methods, Lecture Notes in Comput. Sci. 40, R. Kornhuber et al., eds., Springer, Berlin, 2003, pp. 547–554.
[14] G. Cimmino, *Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari*, Ricerca Sci. XVI, Ser. II, Anno IX, 1 (1938), pp. 326–333.
[15] P. L. Combettes, *Inconsistent signal feasibility problems: Least-squares solutions in a product space*, IEEE Trans. Signal Process., SP-42 (1994), pp. 2955–2966.
[16] P. P. B. Eggermont, G. T. Herman, and A. Lent, *Iterative algorithms for large partitioned linear systems, with applications to image reconstruction*, Linear Algebra Appl., 40 (1981), pp. 37–67.
[17] T. Elfving, *Block-iterative methods for consistent and inconsistent linear equations*, Numer. Math., 35 (1980), pp. 1–12.
[18] H. Elman and G. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems*, Math. Comput., 54 (1990), pp. 671–700.
[19] D. Gordon and R. Gordon, *CG Acceleration of CARP: Robust and Efficient Solution of Sparse Linear Systems*, in preparation.
[20] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.
[21] G. T. Herman, *Image Reconstruction from Projections: The Fundamentals of Computerized Tomography*, Academic Press, New York, 1980.
[22] G. T. Herman, A. Lent, and P. H. Lutz, *Relaxation methods for image reconstruction*, Comm. ACM, 21 (1978), pp. 152–158.
[23] S. Kaczmarz, *Angenäherte Auflösung von Systemen linearer Gleichungen*, Bull. Acad. Polon. Sci. Lett., A35 (1937), pp. 355–357.
[24] C. Kamath and A. Sameh, *A projection method for solving non-symmetric linear systems on multi processors*, Parallel Comput., 9 (1988), pp. 291–312.
[25] D. Kincaid and D. Young, *Adapting iterative algorithms developed for symmetric systems to non-symmetric systems*, in Elliptic Problem Solvers, M. Schultz, ed., Academic Press, New York, 1981, pp. 353–359.
[26] D. Kuck, E. Davidson, D. Lawrie, and A. Sameh, *Parallel super-computing today and the Cedar approach*, Science, 231 (1986), pp. 967–974.
[27] G. Pierra, *Méthodes de décomposition et croisement d'algorithmes pour des problèmes d'optimisation*, Ph.D. thesis, University of Grenoble, Grenoble, France, 1976.
[28] G. Pierra, *Decomposition through formalization in a product space*, Math. Programming, 28 (1984), pp. 96–115.
[29] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.
[31] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 36–52.
[32] K. Tanabe, *Projection method for solving a singular system of linear equations and its applications*, Numer. Math., 17 (1971), pp. 203–214.

[33] M. R. Trummer, *Reconstructing pictures from projections: On the convergence of the ART algorithm with relaxation*, Computing, 26 (1981), pp. 189–195.
[34] R. S. Tuminaro, M. Herous, S. A. Hutchinson, and J. N. Shadid, *AZTEC User's Guide*, Tech. Report SAND99-8801J, Sandia National Laboratories, Albuquerque, NM, 1999.
[35] H. A. van der Vorst, *BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 631–644.

SIAM J. SCI. COMPUT.
Vol. 27, No. 3, pp. 1118–1139
© 2005 Society for Industrial and Applied Mathematics

HIGH-ORDER COLLOCATION METHODS FOR DIFFERENTIAL EQUATIONS WITH RANDOM INPUTS*

DONGBIN XIU† AND JAN S. HESTHAVEN‡

Abstract. Recently there has been a growing interest in designing efficient methods for the solution of ordinary/partial differential equations with random inputs. To this end, stochastic Galerkin methods appear to be superior to other nonsampling methods and, in many cases, to several sampling methods. However, when the governing equations take complicated forms, numerical implementations of stochastic Galerkin methods can become nontrivial and care is needed to design robust and efficient solvers for the resulting equations. On the other hand, the traditional sampling methods, e.g., Monte Carlo methods, are straightforward to implement, but they do not offer convergence as fast as stochastic Galerkin methods. In this paper, a high-order stochastic collocation approach is proposed. Similar to stochastic Galerkin methods, the collocation methods take advantage of an assumption of smoothness of the solution in random space to achieve fast convergence. However, the numerical implementation of stochastic collocation is trivial, as it requires only repetitive runs of an existing deterministic solver, similar to Monte Carlo methods. The computational cost of the collocation methods depends on the choice of the collocation points, and we present several feasible constructions. One particular choice, based on sparse grids, depends weakly on the dimensionality of the random space and is more suitable for highly accurate computations of practical applications with large dimensional random inputs. Numerical examples are presented to demonstrate the accuracy and efficiency of the stochastic collocation methods.

Key words. collocation methods, stochastic inputs, differential equations, uncertainty quantification

AMS subject classifications. 65C20, 65C30

DOI. 10.1137/040615201

1. Introduction.

This paper is devoted to numerical solutions of differential equations with random inputs. The source of random inputs includes uncertainty in system parameters, boundary/initial conditions, etc. Such kinds of uncertainty are ubiquitous in engineering applications and are often modeled as random processes. The classical approach is to model the random inputs as some idealized processes which typically are white noises, e.g., Wiener processes, Poisson processes, etc., and has led to elegant mathematical analysis and numerical methods by using stochastic calculus, e.g., Ito or Strantonovich calculus. The resulting differential equations are termed "stochastic ordinary/partial differential equations" (SODE/SPDE) (see, for example, [18, 25, 28, 43]). Recently, there has been a growing interest in studying problems with more correlated random inputs ("colored noises") and, in case of random parameters, fully correlated random processes, i.e., random variables. In this context, the classical stochastic calculus does not readily apply and other approaches are required.

*Received by the editors September 16, 2004; accepted for publication (in revised form) August 5, 2005; published electronically December 30, 2005.

http://www.siam.org/journals/sisc/27-3/61520.html

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (dxiu@math.purdue.edu). The work of this author was partly supported by AFOSR contract F49620-01-1-0035.

‡Division of Applied Mathematics, Brown University, Providence, RI 02912 (jansh@dam.brown.edu). The work of this author was partly supported by NSF Career Award DMS-0132967, by AFOSR contract FA9550-04-1-0072, and by the Alfred P. Sloan Foundation through a Sloan Research Fellowship.

The traditional approach for differential equations with random inputs is the Monte Carlo method, which generates ensembles of random realizations for the prescribed random inputs and utilizes repetitive deterministic solvers for each realization. Monte Carlo methods have been applied to many applications and their implementations are straightforward. Although the convergence rate—the ensemble mean of a brute-force Monte Carlo method with K realizations converges asymptotically at a rate $1/\sqrt{K}$—is relatively slow, it is (formally) independent of the dimensionality of the random space, i.e., independent of the number of random variables used to characterize the random inputs (e.g., [13]). To accelerate convergence, several techniques have been developed, for example, Latin hypercube sampling, [34, 48], the quasi-Monte Carlo (QMC) method, [14, 39, 40], and the Markov chain Monte Carlo method (MCMC) [16, 22, 36]. These methods can improve the efficiency of the brute-force Monte Carlo method. However, additional restrictions are imposed based on their specific designs and their applicability is limited.

Another class of methods is the nonsampling methods, where no repetitive deterministic solvers are employed. Methods along this line of approach include perturbation methods [27] and second-moment analysis [31, 32]. These methods do not result in convergent series of expansion and their applicability is restricted to systems with relatively small random inputs and outputs. Such a condition is difficult to satisfy, especially for nonlinear problems where small random inputs may induce large random outputs (e.g., [56]). Stochastic Galerkin methods, whose convergence can be established for problems with relatively large magnitude of random inputs/outputs, have been developed. The stochastic Galerkin methods are generalizations of the Wiener–Hermite polynomial chaos expansion, which was developed in [51] and has been applied to various problems in mechanics [21]. The generalizations, also called generalized polynomial chaos, employ non-Hermite polynomials to improve efficiency for a wider class of problems and include global polynomial expansions based on hypergeometrical polynomials [53, 52, 54], piecewise polynomial expansions [2, 12], and wavelet basis expansions [29, 30]. In general, these methods exhibit fast convergence rates with increasing order of the expansions, provided that solutions are sufficiently smooth in the random space. However, the resulting set of [illegible] equations is often coupled and care is needed to design efficient and robust solver. (See [52, 55] for examples of diffusion equations.) The form of the resulting equations can become very complicated if the underlying differential equations have nontrivial and nonlinear forms. (See [7, 37, 57] for examples of hyperbolic systems.)

In this paper, we propose a class of stochastic collocation methods that combines the strength of Monte Carlo methods and stochastic Galerkin methods. By taking advantage of the existing theory on multivariate polynomial interpolations (e.g., [5, 6]), the stochastic collocation methods achieve fast convergence when the solutions possess sufficient smoothness in random space, similar to stochastic Galerkin methods. On the other hand, the implementations of stochastic collocation methods are straightforward as they only require solutions of the corresponding [illegible] problems at each interpolation point, similar to Monte Carlo methods at each sampling point. Such properties make stochastic collocation methods good alternatives to Monte Carlo methods and stochastic Galerkin methods. The core issue is the construction of the set of interpolation points, and it is nontrivial in multidimensional random spaces. For such a cases, point sets based on tensor products of one-dimensional quadrature points [37, 53] are not suitable, as the number of points grows too fast with increasing dimensions. Therefore, we investigate and propose two other choices. One is based on Stroud's cubature points [49] and the other on sparse grids from the Smolyak

algorithm [46]. The former approach is of relatively low-order accuracy but results in highly efficient algorithms, especially for problems with large dimensional random inputs. The latter offers true high-order accuracy with convergence rate depending weakly on dimensionality. Both methods are more efficient than brute-force Monte Carlo methods in a large range of random dimensionality. Compared to stochastic Galerkin methods, the collocation methods generally result in a larger number of equations than a typical Galerkin method; however, these equations are easier to solve as they are completely decoupled and require only repetitive runs of a deterministic solver. Such a property makes the collocation methods more attractive for problems with complicated governing equations.

This paper is organized as follows. In section 2 we present the mathematical framework of stochastic differential equations—the functional spaces and the strong and weak forms are formulated in section 2.1 and we briefly review the Monte Carlo methods and stochastic Galerkin methods in section 2.2. The stochastic collocation methods are discussed in section 3, where the construction of collocation points is discussed in detail. More technical details and numerical examples are presented in sections 4 and 5, and we conclude the paper in section 6.

2. Random differential equations. Let $(\Omega, \mathcal{A}, \mathcal{P})$ be a complete probability space, where Ω is the event space, $\mathcal{A} \subset 2^\Omega$ the σ-algebra, and $\mathcal{P}$ the probability measure. Consider a d-dimensional bounded domain $D \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ with boundary ∂D, and we study the following problem: find a stochastic function, $u \equiv u(\omega, x) : \Omega \times \bar{D} \to \mathbb{R}$, such that for $\mathcal{P}$-almost everywhere $\omega \in \Omega$, the following equation holds:

$$\mathcal{L}(\omega, x; u) = f(\omega, x), \qquad x \in D, \tag{2.1}$$

subject to the boundary condition

$$\mathcal{B}(\omega, x; u) = g(\omega, x), \qquad x \in \partial D, \tag{2.2}$$

where $x = (x_1, \ldots, x_d)$ are the coordinates in $\mathbb{R}^d$, $\mathcal{L}$ is a (linear/nonlinear) differential operator, and $\mathcal{B}$ is a boundary operator. The operator $\mathcal{B}$ can take various forms on different boundary segments, e.g., $\mathcal{B} \equiv \mathbb{I}$ on Dirichlet segments (where $\mathbb{I}$ is the identity operator) and $\mathcal{B} \equiv n \cdot \nabla$ on Neumann segments (where n is the outward unit normal vector). In the most general settings, the operators $\mathcal{L}$ and $\mathcal{B}$, as well as the driving terms f and g, can all have random components. Finally, we assume that the boundary ∂D is sufficiently regular and the driving terms f and g are properly posed, such that (2.1)–(2.2) is well-posed $\mathcal{P}$-a.e. $\omega \in \Omega$. (With a slight abuse of notation, (2.1)–(2.2) can incorporate time-dependent problems, if we consider D to be in an $(\mathbb{R}^{d+1})$-dimensional space, with the extra dimension designated to time.)

2.1. Strong and weak forms. To solve (2.1)–(2.2) numerically, we need to reduce the infinite-dimensional probability space to a finite-dimensional space. This can be accomplished by characterizing the probability space by a finite number of random variables. Such a procedure, termed the "finite-dimensional noise assumption" in [2], is often achieved via a certain type of decomposition which can approximate the target random process with desired accuracy. One of the choices is the Karhunen–Loève type expansion [33], which is based on the spectral decomposition of the covariance function of the input random process. Following a decomposition and assuming that the random inputs can be characterized by N random variables, we can rewrite the

random inputs in abstract form, e.g.,

$$\mathcal{L}(\omega, x; u) = \mathcal{L}(Y^1(\omega), \dots, Y^N(\omega), x; u), \qquad f(\omega, x) = f(Y^1(\omega), \dots, Y^N(\omega), x), \tag{2.3}$$

where $\{Y^i\}_{i=1}^N$ are real random variables with zero mean value and unit variance. Hence, following the Doob–Dynkin lemma [43], the solution of (2.1)–(2.2) can be described by the same set of random variables $\{Y^i(\omega)\}_{i=1}^N$, i.e.,

$$u(\omega, x) = u(Y^1(\omega), \dots, Y^N(\omega), x). \tag{2.4}$$

By characterizing the probability space with N random variables, we have effectively reduced the infinite-dimensional probability space to a N-dimensional space.

From a mathematical and computational point of view, it is most convenient to further assume that these random variables are mutually independent, and subsequently, we can define the corresponding Hilbert functional spaces via simple tensor product rules. However, such an assumption could potentially introduce more errors in approximating the input random processes (in addition to the errors induced by truncating the infinite series during a decomposition procedure such as the Karhunen–Loève expansion). For example, for non-Gaussian processes, the Karhunen–Loève expansion yields a set of [illegible] random variables that are not necessarily mutually independent. To transform them into independent random variables, a (nonlinear) transformation is needed and its numerical construction can be nontrivial. In fact, a mathematically rigorous transformation exists by using merely the definition of joint cumulative density function (CDF) [44]. However, such a transformation is of little practical use as it requires the knowledge of all the joint CDFs of the underlying random process.

In this paper, we take the customary approach (see, for example, [2, 30, 52]) of assuming that the random inputs are already characterized by a set of mutually independent random variables with satisfactory accuracy. We emphasize that how to (numerically) achieve this, i.e., to represent [illegible] via the Karhunen–Loève decomposition or any other representation and to ensure that the random variables involved are independent, falls into the topic of numerical representation of non-Gaussian processes. It remains an active research area (e.g., [23, 45, 58]) and is beyond the scope of this paper. We also remark that it is possible to construct multidimensional functional spaces based on finite number of [illegible] random variables [47]. However, such a construction does not, in its current form, allow straightforward numerical implementations.

Let us now assume that $\{Y^i\}_{i=1}^N$ are independent random variables with probability density functions $\rho_i : \Gamma^i \to \mathbb{R}^+$ and their images $\Gamma^i \equiv Y^i(\Omega)$ are bounded intervals in $\mathbb{R}$ for $i = 1, \dots, N$. Then

$$\rho(y) = \prod_{i=1}^N \rho_i(Y^i) \qquad \forall y \in \Gamma \tag{2.5}$$

is the joint probability density of $y = (Y^1, \dots, Y^N)$ with the support

$$\Gamma \equiv \prod_{i=1}^N \Gamma^i \subset \mathbb{R}^N. \tag{2.6}$$

This allows us to rewrite (2.1)–(2.2) as an $(N+d)$-dimensional differential equation in the strong form

$$\mathcal{L}(y,x;u) = f(y,x), \qquad (y,x) \in \Gamma \times D, \tag{2.7}$$

subject to the boundary condition

$$\mathcal{B}(y,x;u) = g(y,x), \qquad (y,x) \in \Gamma \times \partial D, \tag{2.8}$$

where N is the dimensionality of the random space Γ and d is the dimensionality of the physical space D.

Similar to the weak formulations in deterministic problems, we often seek to solve (2.7)–(2.8) in a weak form. For example, we can define a (finite-dimensional) subspace $V_\Gamma \subset L^2_\rho(\Gamma)$, the space of all square integrable function in Γ with respect to the measure $\rho(y)dy$, and seek $u_V(y,x) \in V_\Gamma(y)$, such that

$$\int_\Gamma \rho(y)\mathcal{L}(y,x;u_V)\, v(y)dy = \int_\Gamma \rho(y) f(y,x) v(y)dy \qquad \forall v(y) \in V_\Gamma, x \in D, \tag{2.9}$$

$$\int_\Gamma \rho(y)\mathcal{B}(y,x,u_V) v(y)dy = \int_\Gamma \rho(y) g(y,x) v(y)dy \qquad \forall v(y) \in V_\Gamma, x \in \partial D. \tag{2.10}$$

The resulting problem (2.9)–(2.10) thus becomes a deterministic problem in the physical domain D and can be solved by a standard discretization technique, e.g., finite elements, finite volume.

2.2. Existing numerical methods. Several numerical methods can be applied to problem (2.7)–(2.8), or (2.9)–(2.10). Here we briefly review two of the more popular methods—Monte Carlo methods and stochastic Galerkin methods—whose strengths complement each other.

2.2.1. Monte Carlo methods. Monte Carlo simulation is one of the most developed methods for solving stochastic differential equations. Although the convergence rate is relatively slow—a typical Monte Carlo simulation consisting of K realizations converges asymptotically as $1/\sqrt{K}$ [13]—it is independent of the number of random variables $\{Y^i(\omega)\}_{i=1}^N$. The procedure of applying a Monte Carlo method to problem (2.7)–(2.8) takes the following steps:

1. For a prescribed number of realizations K, generate independent and identically distributed (i.i.d.) random variables $\{Y_j^i\}_{i=1}^N \equiv \{Y^i(\omega_j)\}_{i=1}^N$, for $j = 1, \ldots, K$;
2. For each $j = 1, \ldots, K$, solve a deterministic problem (2.7)–(2.8) with $y_j = (Y_j^1, \ldots, Y_j^N)$ and obtain solution $u_j \equiv u(y_j, x)$, i.e., solve $\mathcal{L}(y_j, x; u_j) = f(y_j, x)$ in D, subject to $\mathcal{B}(y_j, x; u_j) = g(y_j, x)$ on ∂D;
3. Postprocess the results to evaluate the solution statistics, e.g., $\langle u \rangle \equiv \mathbb{E}[u] = \frac{1}{M}\sum_{j=1}^M u_j$.

Here $\mathbb{E}[u] = \int_\Gamma \rho(y)u(y)dy$ is the expectation. Note that for each realization $j = 1, \ldots, K$, step 2 is a deterministic problem and can be solved by any suitable scheme.

2.2.2. Stochastic Galerkin methods. Stochastic Galerkin methods deal with the weak form (2.9)–(2.10), and they offer fast convergence when the solution is sufficiently smooth in the random space. Forms of stochastic Galerkin methods differ in the construction of the subspace V_Γ in (2.9). Here we briefly review the construction

based on global orthogonal polynomial expansions. Let us define one-dimensional subspaces for $L^2_{\rho_i}(\Gamma^i)$,

$$W^{i,p_i} \equiv \left\{ v : \Gamma^i \to \mathbb{R} : v \in \text{span}\left\{\phi_m(Y^i)\right\}_{m=0}^{p_i} \right\}, \qquad i = 1, \dots, N, \tag{2.11}$$

where $\{\phi_m(Y^i)\}$ are a set of orthogonal polynomials satisfying the orthogonality conditions

$$\int_{\Gamma_i} \rho_i(Y^i)\phi_m(Y^i)\phi_n(Y^i)dY^i = h_m^2 \delta_{mn}, \tag{2.12}$$

with normalization factors $h_m^2 = \int_{\Gamma_i} \rho_i \phi_m^2 dY^i$. The type of the orthogonal polynomials $\{\phi_m(Y^i)\}$ in (2.12) is determined by $\rho_i(Y^i)$, the probability density function of $Y^i(\omega)$, for $i = 1, \dots, N$. For example, uniform distributions are better approximated by Legendre polynomials, and Hermite polynomials are more appropriate for Gaussian distributions. (See [53, 52] for a detailed list of correspondences.) We remark that such a correspondence is not mandatory, so long as $\{\phi_m(Y^i)\}$ forms a complete basis.

The corresponding N-dimensional subspace V_Γ for Γ can be defined as

$$W_N^p \equiv \bigotimes_{|\mathbf{p}| \le p} W^{i,p_i}, \tag{2.13}$$

where the tensor product is over all possible combinations of the multi-index $\mathbf{p} = (p_1, \dots, p_N) \in \mathbb{N}_0^N$ satisfying $|\mathbf{p}| = \sum_{i=1}^N p_i \le p$. Thus, W_N^p is the space of N-variate orthogonal polynomials of total degree at most p, and the number of basis functions in (2.13) is $\dim(W_N^p) = \binom{N+p}{N}$. Hereafter, we will refer to W_N^p as a complete polynomial space. Such spaces are often employed in the (generalized) polynomial chaos expansions [21, 52, 54].

Another construction of V_Γ is based on the tensor products of the one-dimensional polynomial spaces with fixed highest polynomial orders in N dimensions, i.e.,

$$Z_N^p = \bigotimes_{i=1}^N W^{i,p_i}, \quad \max_i p_i = p. \tag{2.14}$$

Such a space will be denoted as tensor polynomial space hereafter. They are employed in [2, 15], and the number of basis functions in Z_N^p is $\dim(Z_N^p) = (p+1)^N$.

If the same one-dimensional subspaces (2.11) are employed, then $W_N^p \subset Z_N^p$. The numbers of basis functions in both spaces depend on the number of dimensions N and the polynomial orders p, and they grow fast with increasing N and p. In particular, the total number of terms can grow exponentially fast with increasing order p when $N \gg 1$ (curse-of-dimensionality). We also remark that when $N \gg 1$ and p is moderate (e.g. $p \le 10$), $\dim(W_N^p) \ll \dim(Z_N^p)$.

For low to moderate value of N, stochastic Galerkin methods are preferred because their high accuracy and fast convergence; for large value of $N \gg 1$, the rapidly growing number of basis functions effectively reduces its efficiency. In this case, the Monte Carlo method is preferred. However, if a highly accurate stochastic solution is desired, then stochastic Galerkin methods can be more efficient even for relatively large value of N (e.g., [2]).

3. Stochastic collocation methods. In this section, we present high-order collocation methods for (2.1)–(2.2). The objective is to combine the strength of the two existing approaches: the high resolution of stochastic Galerkin methods resulting from polynomial approximations in random spaces, and the ease of implementation of Monte Carlo methods by sampling at discrete points in random spaces.

3.1. Formulation. The construction of high-order stochastic collocation methods is based on polynomial interpolations in the multidimensional random space. Let us denote by $y = (Y^1, \ldots, Y^N)$ any point in the random space $\Gamma \subset \mathbb{R}^N$, by Π_N the space of all N-variate polynomials with real coefficients, and by Π_N^p the subspace of polynomials of total degree at most p. (Note that the space W_N^p defined in (2.13) is a subset of Π_N^p.) When derivatives are not interpolated, the [illegible] can be stated in the following form.

DEFINITION 3.1 (Lagrange interpolation). [illegible] $y_1, \ldots, y_M$ [illegible] $b_1, \ldots, b_M$ [illegible] $V_I \in \Pi_N$ [illegible] $l \in V_I$ [illegible]

$$l(y_j) = b_j, \qquad j = 1, \ldots, M. \tag{3.1}$$

The points $y_1, \ldots, y_M$ are called interpolation [illegible], and V_I is the [illegible].

DEFINITION 3.2 (poisedness). [illegible] 3.1, [illegible] $y_1, \ldots, y_M \in \mathbb{R}^N$ [illegible] V_I [illegible] $b_1, \ldots, b_M \in \mathbb{R}$ [illegible] $f \in V_I$ [illegible] $f(y_j) = b_j$ $j = 1, \ldots, M$ [illegible] M [illegible] $\mathbb{R}^N$ [illegible] V_I [illegible] V_I [illegible] M

There is a well-developed and extensive classical theory of univariate Lagrange polynomial interpolation. However, the multivariate case is more difficult. In particular, the Haar spaces exist in abundance for $N = 1$ but not for $N > 1$. In fact, in this case there are no Haar spaces of dimension greater than one—the so-called loss-of-Haar. (For more details and the refinement of this statement, see [11, 35].) Thus, research has focused on the [illegible] of points $\{y_i\}_{i=1}^M$ such that one achieves a good approximation by a Lagrange interpolation.

Let $\Theta_N = \{y_i\}_{i=1}^M \in \Gamma$ be a set of (prescribed) nodes such that the Lagrange interpolation (3.1) in the N-dimensional random space Γ is poised in a corresponding interpolation space V_I. (For sufficient conditions on the distribution of the nodes to ensure a unique interpolation, see [8].) Subsequently, Lagrange interpolation of a smooth function, $f : \mathbb{R}^N \to \mathbb{R}$, can now be viewed as follows: find a polynomial $\mathcal{I}(f) \in V_I$ such that $\mathcal{I}(f)(y_i) = f(y_i)$, $\forall i = 1, \ldots, M$.

The polynomial approximation $\mathcal{I}(f)$ can be expressed by using the Lagrange interpolation polynomials, i.e.,

$$\mathcal{I}(f)(y) = \sum_{k=1}^{M} f(y_k) L_k(y), \tag{3.2}$$

where

$$L_i(y) \in V_I, \quad L_i(y_j) = \delta_{ij}, \quad 1 \le i, j \le M, \tag{3.3}$$

are the Lagrange polynomials.

We equip the space V_I with the supremum-norm $\|\cdot\|$ and introduce

$$\|\mathcal{I}\| = \sup_{f \neq 0} \frac{\|\mathcal{I}f\|_\infty}{\|f\|_\infty}. \tag{3.4}$$

The Lebesgue theorem states that the interpolation error is uniformly bounded as

$$\|f(y) - f^*(y)\|_\infty \le \|f(y) - \mathcal{I}f(y)\|_\infty \le (1+\Lambda)\,\|f(y) - f^*(y)\|_\infty, \tag{3.5}$$

where $f^*(y)$ is the best approximating polynomial, and $\Lambda = \|\mathcal{I}\| = \max_{y\in\Gamma} \sum_{k=1}^{K} |L_i(y)|$ is the Lebesgue constant. We remark that the determination of $f^*(y)$ is an unsolved problem for arbitrary $(N > 1)$-dimensional spaces, and the estimation of the Lebesgue constant, which depends on the particular choice of nodal sets Θ_N, is a nontrivial task even for $N = 1$.

By denoting

$$\hat{u}(y) \equiv \mathcal{I}u(y) = \sum_{k=1}^{M} u(y_k) L_k(y), \tag{3.6}$$

the collocation procedure to solve the stochastic elliptic equation (2.7) is

$$R\left(\hat{u}(y)\right)\big|_{y_k} = 0 \qquad \forall k = 1, \dots, M, \tag{3.7}$$

where $R(u) = \mathcal{L}(u) - f$ is the residual of (2.7). By using the property of Lagrange interpolation (3.3), we immediately obtain: for $k = 1, \dots, M$,

$$\mathcal{L}(y_k, x; u) = f(y_k, x), \qquad x \in D, \tag{3.8}$$

with boundary condition

$$\mathcal{B}(y_k, x; u) = g(y_k, x), \qquad x \in \partial D. \tag{3.9}$$

Thus, the stochastic collocation method is equivalent to solving M deterministic problems (3.8)–(3.9), the deterministic counterpart of problem (2.7)–(2.8), at each nodal point $y_k, k = 1, \dots, M$ in a given nodal set Θ_N. Note that the problem (3.8)–(3.9) for each k is naturally , and existing deterministic solvers can be readily applied. This is in direct contrast to the stochastic Galerkin approaches, where the resulting expanded equations are, in general, coupled.

Once the numerical solutions of (3.8)–(3.9) are obtained at all collocation points, the statistics of the random solution can be evaluated, e.g.,

$$\mathbb{E}(\hat{u})(x) = \sum_{k=1}^{M} u(y_k, x) \int_\Gamma L_k(y)\rho(y)dy. \tag{3.10}$$

The evaluations of such expectations require explicit knowledge of the Lagrange interpolation polynomials $\{L_k(y)\}$. For a given nodal set Θ_N, the polynomials can be determined numerically by inverting a Vandermonde-type matrix. (The matrix is invertible under the assumption that the Lagrange interpolation is poised on Θ_N.) In multivariate cases, such a procedure can be cumbersome, but can be accomplished once and for all at the preprocessing stage.

An alternative is to choose the set Θ_N to be a cubature point set. Recall that a cubature rule approximates an integral by

$$\int_\Gamma f(y)dy \simeq \sum_{i=1}^{M} f(y_i) w_i, \tag{3.11}$$

where $\{w_i\}$ are the weights. It is called "of order s" if the approximation is exact for all $f \in \Pi_N^r, r \leq s$, and not exact for at least one $f \in \Pi_N^{s+1}$. When we choose the interpolation point set Θ_N to be the same as a cubature point set, the evaluation of (3.10) is reduced to

$$\mathbb{E}(\hat{u})(x) = \sum_{k=1}^{M} u(y_k, x) w_k. \tag{3.12}$$

It should be noted that the use of cubature formula to evaluate the solution statistics will introduce extra errors whenever the integrals defining these statistics, e.g., (3.10) for mean solution, cannot be evaluated exactly by an s-order cubature.

Construction of cubature formulas has long been an active and challenging research area, and we refer interested readers to [10, 9] for extensive reviews.

3.2. Choices of collocation points. The computational complexity of the stochastic collocation methods (3.8)–(3.9) is M times that of a deterministic problem, where M is the total number of collocation points. Thus, we need to choose a nodal set Θ_N with fewest possible number of points under a prescribed accuracy requirement. In this section, we present several choices of such collocation points. The main focus here is on multidimensional random spaces, i.e., $\Gamma \subset \mathbb{R}^N$ with $N > 1$. Without loss of generality, we assume that the bounded support of the random variables Y^i is $\Gamma^i = [-1, 1]$ for $i = 1, \ldots, N$ and subsequently the bounded random space Γ is a N-hypercube, i.e., $\Gamma = [-1, 1]^N$. (Note that random variables with bounded support in $[a, b]$ can always be mapped to $[-1, 1]$, and any domain $\Gamma^* = \prod_{i=1}^N [a^i, b^i]$ to $[-1, 1]^N$.)

3.2.1. Tensor products of one-dimensional nodal sets. A natural choice of the nodal set is the tensor product of one-dimensional sets. In interpolating smooth functions $f : [-1, 1]^N \to \mathbb{R}$, much is known about good interpolation formulas for $N = 1$, i.e., for every direction $i = 1, \ldots, N$, we can construct a good one-dimensional interpolation

$$\mathcal{U}^i(f) = \sum_{k=1}^{m_i} f(Y_k^i) \cdot a_k^i \tag{3.13}$$

based on nodal sets

$$\Theta_1^i = (Y_1^i, \ldots, Y_{m_i}^i) \subset [-1, 1], \tag{3.14}$$

where $a_k^i \equiv a_k(Y^i) \in C([-1, 1])$. We assume that a sequence of formulas (3.13) is given. In the multivariate case $N > 1$, the tensor product formulas are

$$\mathcal{I}(f) \equiv \left(\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_N}\right)(f) = \sum_{k_1=1}^{m_{i_1}} \cdots \sum_{k_N=1}^{m_{i_N}} f(Y_{k_1}^{i_1}, \ldots, Y_{k_N}^{i_N}) \cdot (a_{k_1}^{i_1} \otimes \cdots \otimes a_{k_N}^{i_N}). \tag{3.15}$$

Clearly, the above product formula needs $M = m_{i_1} \cdots m_{i_N}$ nodal points. If we choose to use the same interpolating function (3.13) in each dimension with the same number of points, i.e., $m_{i_1} = \cdots = m_{i_N} \equiv m$, the total number of points is $M = m^N$. This number grows quickly in high dimensions $N \gg 1$—even for a poor approximation with two points ($m = 2$) in each dimension, $M = 2^N \gg 1$ for $N \gg 1$.

The tensor product interpolation is easy to construct—the interpolation points and space are obtained by tensor products of the univariate ones. The Lagrange formula are easily extended to this case, as can be found in [4, 24]. However, because of the rapidly growing number of interpolation nodes in high dimensions, we will not consider this approach extensively in this paper.

3.2.2. Sparse grids. In this section we propose to solve the stochastic collocation problem (3.8)–(3.9) on a [illegible] constructed by the Smolyak algorithm. The Smolyak algorithm is a linear combination of product formulas, and the linear combination is chosen in such a way that an interpolation property for $N = 1$ is preserved for $N > 1$. Only products with a relatively small number of points are used and the resulting nodal set has significantly less number of nodes compared to the tensor product rule (3.15). Much research has been devoted to the Smolyak algorithm since its introduction in [46]; see, e.g., [3, 41, 42].

Starting with the one-dimensional interpolation formula (3.13), the Smolyak algorithm is given by (see [50])

$$\mathcal{I}(f) \equiv A(q,N) = \sum_{q-N+1 \le |\mathbf{i}| \le q} (-1)^{q-|\mathbf{i}|_1} \cdot \binom{N-1}{q-|\mathbf{i}|} \cdot \left(\mathcal{U}^{i_1} \otimes \cdots \otimes \mathcal{U}^{i_N}\right), \tag{3.16}$$

where $\mathbf{i} = (i_1, \ldots, i_N) \in \mathbb{N}^N$. To compute $A(q,N)$, we only need to evaluate function on the "sparse grid"

$$\Theta_N \equiv \mathcal{H}(q,N) = \bigcup_{q-N+1 \le |\mathbf{i}| \le q} (\Theta_1^{i_1} \times \cdots \times \Theta_1^{i_N}). \tag{3.17}$$

In this paper, we choose to use Smolyak formulas that are based on one-dimensional polynomial interpolation at the extrema of the Chebyshev polynomials. (Other choices, such as the Gauss quadrature points, can be considered as well.) For any choice of $m_i > 1$, these nodes are given by

$$Y_j^i = -\cos\frac{\pi \cdot (j-1)}{m_i - 1}, \qquad j = 1 \ldots, m_i. \tag{3.18}$$

In addition, we define $Y_1^i = 0$ if $m_i = 1$. The functions a_j^i in (3.13) are characterized by the requirement that $\mathcal{U}^i$ reproduce all polynomials of degree less than m_i. We also choose $m_1 = 1$ and $m_i = 2^{i-1}$ for $i > 1$. By doing so, the one-dimensional nodal sets Θ_1^i are nested, i.e., $\Theta_1^i \subset \Theta_1^{i+1}$, and subsequently $\mathcal{H}(q,N) \subset \mathcal{H}(q+1,N)$.

It has been shown that if we set $q = N + k$, then $A(N+k,N)$ is exact for all polynomials in Π_N^k [42], and the total number of nodes for $N \gg 1$ is [41]

$$M \equiv \dim(A(N+k,N)) \sim \frac{2^k}{k!} N^k, \qquad k \text{ fixed}, N \gg 1. \tag{3.19}$$

On the other hand, $\dim(\Pi_N^k) = \binom{N+k}{N} \sim N^k/k!$ for $N \gg 1$. Thus, $A(N+k,N)$ uses about 2^k times as many points as the degree-of-freedom of Π_N^k. Since this factor is [illegible] of N, the algorithm may be regarded as optimal. Formula (3.19) also indicates that the number of nodes from the Smolyak sparse grid is larger than the number of expansion terms of a stochastic Galerkin method in the complete polynomial space W_N^k. Hereafter, we will refer k in $A(N+k,N)$ as the "level" of the Smolyak construction.

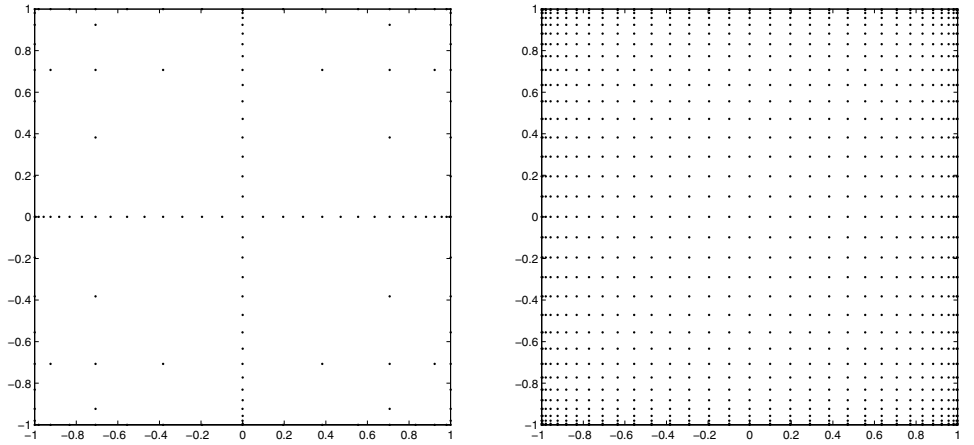


FIG. 3.1. *Two-dimensional ($N = 2$) interpolation nodes based on the extrema of Chebyshev polynomials* (3.18). *Left: sparse grid $\mathcal{H}(N+k, N)$ from Smolyak algorithm, $k = 5$. Total number of points is* 145. *Right: tensor product algorithm* (3.15) *from the same one-dimensional nodes. Total number of nodes is* $1,089$.

TABLE 3.1

The degrees-of-freedom of the Smolyak sparse grid ($\mathcal{H}(N + k, N) = \dim(A(N + k, N))$), the complete polynomial space W_N^k, and the tensor polynomial space Z_N^k, for various dimensions N and order k.

N	k	$\mathcal{H}(N + k, N)$	$\dim(W_N^k)$	$\dim(Z_N^k)$
2	1	5	3	4
	2	13	6	9
	3	29	10	16
	4	65	15	25
10	1	21	11	$1,024$
	2	221	66	$59,049$
	3	$1,581$	286	$1,048,576$
20	1	41	21	$1,048,576$
	2	841	231	$\approx 3.5 \times 10^9$
50	1	101	51	$\approx 1.1 \times 10^{16}$
	2	$5,101$	$1,326$	$\approx 7.2 \times 10^{23}$

In Figure 3.1, we show the two-dimensional ($N = 2$) interpolation nodes by the sparse grid $\mathcal{H}(N + k, N)$ described above, with $k = 5$. For comparison, the two-dimensional tensor product grid based on the same one-dimensional nodes are shown on the right of Figure 3.1, and we observe that the sparse grid has significantly fewer nodes.

In Table 3.1 we list the number of nodes of the Smolyak sparse grid $\mathcal{H}(N+k, N) = \dim(A(N + k, N))$, the degrees-of-freedom of the complete polynomial space W_N^k (2.13), and the tensor polynomial space Z_N^k (2.14). (As shown in [42], the interpolation by $A(N + k, N)$ is exact in a polynomial space richer than W_N^k but not as rich as Z_N^k.) It can be seen that as N grows, the number of nodes from the Smolyak sparse grid is significantly less than $\dim(Z_N^k)$. Also, the relation (3.19) can be observed. This indicates that only at low dimensions, e.g., $N < 5$, should the tensor product construction of interpolation nodes be considered.

The interpolation error in a space $F_N^l = \{f : [-1, 1]^N \to \mathbb{R} \mid \partial^{|\mathbf{m}|} f \text{ continues}, m_i \leq l, \forall i\}$, where $\mathbf{m} \in \mathbb{N}_0^N$ and $\partial^{|\mathbf{m}|}$ is the usual N-variate partial derivative of order $|\mathbf{m}|$,

is (cf. [3])

$$\|\mathbb{I}_N - A(q,N)\|_\infty \le C_{N,l}\cdot M^{-l}\cdot(\log M)^{(l+2)(N-1)+1}, \tag{3.20}$$

where $\mathbb{I}_N$ is the identity operator in a N-dimensional space and $M = \dim(\mathcal{H}(q,N)$ is the number of interpolation points.

Another intriguing property of the sparse grid constructed above is that it is also the nodes of a good cubature formula (3.11). Thus, we can efficiently evaluate the solution statistics in a similar way of (3.12). For analysis on integration error estimates of Smolyak sparse grid for multivariate integrals, see [41].

3.2.3. Stroud's cubature of degrees 2 and 3. For the N-hypercube spaces $[-1,1]^N$ considered in this paper, Stroud constructed a set of cubature points with $(N+1)$-point that is accurate for multiple integrals of polynomials of degree 2, and a set of degree 3 with $2N$-point [49]. The degree 2 formula, termed the Stroud-2 method hereafter, consists of points $\{y_k\}_{k=0}^N$ such that

$$Y_k^{2r-1} = \sqrt{\frac{2}{3}}\cos\frac{2rk\pi}{N+1}, \qquad Y_k^{2r} = \sqrt{\frac{2}{3}}\sin\frac{2rk\pi}{N+1}, \quad r = 1,2,\ldots,[N/2], \tag{3.21}$$

where $[N/2]$ is the greatest integer not exceeding $N/2$, and if N is odd $Y_k^N = (-1)^k/\sqrt{3}$. The degree 3 formula, which will be called as Stroud-3 method, has the points $\{y_k\}_{k=1}^{2N}$ such that

$$Y_k^{2r-1} = \sqrt{\frac{2}{3}}\cos\frac{(2r-1)k\pi}{N}, \qquad Y_k^{2r} = \sqrt{\frac{2}{3}}\sin\frac{(2r-1)k\pi}{N}, \quad r = 1,2,\ldots,[N/2], \tag{3.22}$$

and if N is odd $Y_k^N = (-1)^k/\sqrt{3}$.

It was proved in [38] that the Stroud-2 and -3 methods employ the minimal number of points for their corresponding [illegible] accuracy. However, to the authors' knowledge, no theory exists on the approximation properties of the Lagrange interpolations constructed on these nodes. It is expected that the interpolation errors will be lower than degree 2 and 3 for the Stroud-2 and -3 method, respectively.

In this paper, we will employ the Stroud-2 and -3 nodes for stochastic collocation methods. For relatively low-order accuracy requirement in the solution statistics (expectations), the Stroud methods are highly efficient especially in high-dimensional spaces ($N \gg 1$) because the numbers of nodal points are minimal. However, the accuracy cannot be improved any further and no error estimates can be obtained.

4. Applications to stochastic elliptic equations. In this section we use a stochastic elliptic problem to illustrate the applications of stochastic collocation methods. In particular, we consider for $\mathcal{P}$-a.e. $\omega \in \Omega$,

$$\begin{aligned} -\nabla\cdot(a(\omega,x)\nabla u(\omega,x)) &= f(\omega,x), && \text{in } D,\\ u(\omega,x) &= 0, && \text{on } \partial D, \end{aligned} \tag{4.1}$$

where $a, f : \Omega\times D \to \mathbb{R}$ are known stochastic functions. Standard assumptions are made on the regularity of the input functions so that (4.1) is well-posed, e.g., $a \in L^\infty(\Omega, D)$ is strictly positive.

Again let us assume that the random input functions can be characterized by N independent random variables $\{Y^i(\omega)\}_{i=1}^N$, whose images are bounded intervals

and define an N-dimensional random space $\Gamma \subset \mathbb{R}^N$. Without loss of generality, let $\Gamma = [-1,1]^N$. Problem (4.1) can be then expressed in the strong form,

$$\begin{aligned} -\nabla \cdot (a(y,x)\nabla u(y,x)) &= f(y,x) \quad \forall (y,x) \in \Gamma \times D, \\ u(y,x) &= 0 \qquad \forall (y,x) \in \Gamma \times \partial D, \end{aligned} \tag{4.2}$$

and weak variational form: find $u \in L^2_\rho(\Gamma) \otimes H^1_0(D)$ such that

$$\mathcal{K}_\rho(u,v;\kappa) = \mathcal{I}_\rho(f,v) \quad \forall v \in L^2_\rho(\Gamma) \otimes H^1_0(D), \tag{4.3}$$

where

$$\mathcal{I}_\rho(v,w) = \int_\Gamma \rho(y) \int_D v(y,x) w(y,x) dx dy$$

and

$$\mathcal{K}_\rho(v,w;\kappa) = \int_\Gamma \rho(y) \int_D a(y,x) \nabla v(y,x) \cdot \nabla w(y,x) dx dy.$$

4.1. Formulations and error estimates. A stochastic collocation method, combined with a finite element method (FEM) in the physical space $D \subset \mathbb{R}^d$, takes the following form: for a given set of collocation nodes $\Theta_N = \{y_k\}_{k=1}^M$ on which the Lagrange interpolation is poised in an interpolation space $V_I \subset L^2_\rho(\Gamma)$, and a piecewise (continuous) FEM mesh $X^h_d \in H^1_0(D)$ (where $h > 0$ is its mesh spacing parameter) that satisfies all the standard requirements on a FEM mesh, find as follows: for $k = 1, \ldots, M$, $u^h_k(x) \equiv u^h(y_k, x) \in X^h_d$, such that

$$\int_D a(y_k,x) \nabla u^h_k(x) \cdot \nabla \chi(x) dx = \int_D f(y_k,x) \chi(x) dx \qquad \forall \chi \in X^h_d. \tag{4.4}$$

The final numerical solution takes the form

$$\hat{u}_h(y,x) = \sum_{k=1}^M u^h_k(x) L_k(y). \tag{4.5}$$

To study error estimates, let us define an error in term of the deterministic energy error, i.e.,

$$\mathcal{E}_G \equiv \left(\mathbb{E} \int_D a(\nabla(u - \hat{u}_h))^2 dx \right)^{1/2}. \tag{4.6}$$

We can show that

$$\begin{aligned} \mathcal{E}_G &= \left(\int_\Gamma \rho(y) \int_D a(y,x) (\nabla(u - \hat{u}_h))^2 dx dy \right)^{1/2} \\ &\le \left(\|\rho a\|_{L^\infty(\Gamma \times D)} \right)^{1/2} \min_{v \in V_I \otimes X^d_h} \|u - v\|_{L^2_\rho(\Gamma) \otimes H^1_0(D)} \\ &\le \left(\|\rho a\|_{L^\infty(\Gamma \times D)} \right)^{1/2} (\mathcal{E}_D + \mathcal{E}_\Gamma), \end{aligned} \tag{4.7}$$

where $\mathcal{E}_D = \min_{v \in L^2_\rho(\Gamma) \otimes X^d_h} \|u - v\|_{L^2_\rho(\Gamma) \otimes H^1_0(D)}$ is the standard $H^1_0(D)$ FEM approximation error, and $\mathcal{E}_\Gamma$ is the $L^2_\rho(\Gamma)$ approximation error in the random space satisfying

$$\mathcal{E}_\Gamma = \min_{v \in V_I \otimes H^1_0(D)} \|u - v\|_{L^2_\rho(\Gamma) \otimes H^1_0(D)} \le \|u - \mathcal{I}(u)\|_{L^2_\rho(\Gamma) \otimes H^1_0(D)} \le \|u - \mathcal{I}(u)\|_\infty = \mathcal{E}_I. \tag{4.8}$$

The last term is the interpolation error, which can be bounded by the Lebesgue theorem (3.5). For the interpolation by Smolyak sparse grid, this error is (3.20). We hence have the following proposition.

PROPOSITION 4.1 (separation of error estimates). (4.6)

$$\mathcal{E}_G \le C(\mathcal{E}_D + \mathcal{E}_I), \tag{4.9}$$

$\mathcal{E}_D$ $H_0^1(D)$ D $\mathcal{E}_I$ Γ C D Γ

4.2. Numerical results. Numerical implementations of stochastic Galerkin methods are not considered here as they have been considered in many earlier publications. For Galerkin methods in tensor polynomial spaces, see [12, 1, 2, 15]; in complete polynomial spaces, see [19, 20, 21, 52, 54, 55]. Here we focus on numerical examples of stochastic collocation methods and consider the following problem in one spatial dimension ($d = 1$) and $N > 1$ random dimensions:

$$\frac{d}{dx}\left[a(y,x)\frac{du}{dx}(y,x)\right] = 0, \qquad (y,x) \in \Gamma \times (0,1), \tag{4.10}$$

with boundary conditions

$$u(y,0) = 0, \qquad u(y,1) = 1, \qquad y \in \Gamma.$$

The use of one-dimensional physical space D allows us to focus on the properties of the stochastic collocation methods associated with the random space Γ. Extensions to two- and three-dimensional physical spaces are straightforward. We assume that the random diffusivity has the form

$$a(y,x) = 1 + \sigma \sum_{k=1}^{N} \frac{1}{k^2\pi^2} \cos(2\pi k x) Y^k(\omega), \tag{4.11}$$

where $Y^k(\omega) \in [-1,1], k = 1, \ldots, N$, are independent uniformly distributed random variables. The form of (4.11) is similar to those obtained from a Karhunen–Loève expansion with eigenvalues delaying as $1/k^4$. Here we employ (4.11) to eliminate the errors associated with a numerical Karhunen–Loève solver and to keep the random diffusivity strictly positive as $N \to \infty$. The series in (4.11) converge as $N \to \infty$ and we have

$$\mathbb{E}(a(y,x)) = 1, \qquad 1 - \frac{\sigma}{6} < a(y,x) < 1 + \frac{\sigma}{6}.$$

The spatial discretization is done on a p-type spectral element with high-order modal expansions. The basis functions $\psi(\xi)$ in a standard interval $\xi \in (-1,1)$ are defined as

$$\psi_p(\xi) = \begin{cases} \frac{1-\xi}{2}, & p = 0, \\ \frac{1-\xi}{2}\frac{1+\xi}{2} P_{p-1}^{1,1}(\xi), & 0 < p < P, \\ \frac{1+\xi}{2}, & p = P, \end{cases} \tag{4.12}$$

where $P_p^{\alpha,\beta}(\xi)$ is the Jacobi polynomial of order p with parameters $\alpha, \beta > -1$. For more details of such a basis, see [26]. In all the following computations, the spatial

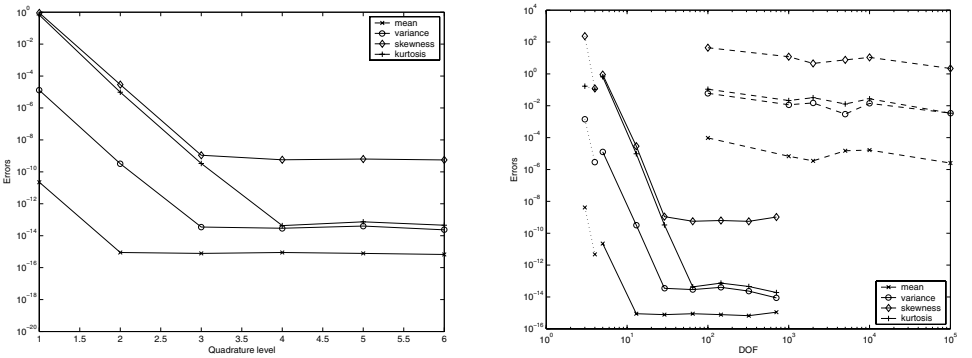


FIG. 4.1. *Convergence of errors in moments with two-dimensional random input ($N = 2$). Left: convergence with respect to level of approximation k of Smolyak formula $A(N + k, N)$; Right: convergence with respect to number of quadratures of different methods, Smolyak sparse grid (solid lines), Stroud's cubature (dotted lines), and Monte Carlo simulation (dashed lines).*

discretizations are conducted with sufficiently high-order basis such that the spatial errors are subdominant.

The exact solution to this problem is

$$u_e(y, x) = c_1 \int \frac{1}{a(y, x)} dx + c_2,$$

where the two constants c_1, c_2 are determined by the boundary conditions. However, the closed form expression for the exact solution is not readily available.

4.2.1. Low-dimensional random inputs: Accuracy. In this section, a low-dimensional random input is assumed. In particular, we choose $N = 2$. The "exact" solution is obtained by using the Smolyak sparse grid with level 8, i.e., $A(N+8, N) = A(10, 2)$.

In Figure 4.1, the errors in the first four moments of the solution are shown. On the left, the error of the Smolyak collocation method is shown. We observe that as the approximation level k of $A(N + k, N)$ increases, the errors converge exponentially fast before reaching saturate levels.

On the right of Figure 4.1, the error convergence of a few different methods is shown with respect to the degree-of-freedom, i.e., the total number of interpolation points. It can be seen that the Smolyak method has the best accuracy; the Stroud-2 and -3 methods approximate the lower moments (mean and variance) well, with considerable less degrees-of-freedom (only three and four points for the Stroud-2 and -3 method, respectively). On the other hand, the standard Monte Carlo method has the expected $O(\mathrm{DOF}^{-1/2})$ convergence.

4.2.2. Moderate-dimensional random inputs. Here we choose a moderately high-dimensional random space with $N = 8$. Figure 4.2 shows the error convergence with respect to degrees-of-freedom for different methods. Again, the Stroud-3 method offers solutions with accuracy comparable to the Smolyak method at level one ($k = 1$ for $A(N + k, N)$), and with fewer nodes. In this case, both the Stroud method and Smolyak method are more efficient than Monte Carlo methods. We also observe that there is no clear sign of accuracy improvement in kurtosis, when comparing the Stroud-3 and Stroud-2 methods. (See also the results in Figure 4.1.) The reason is

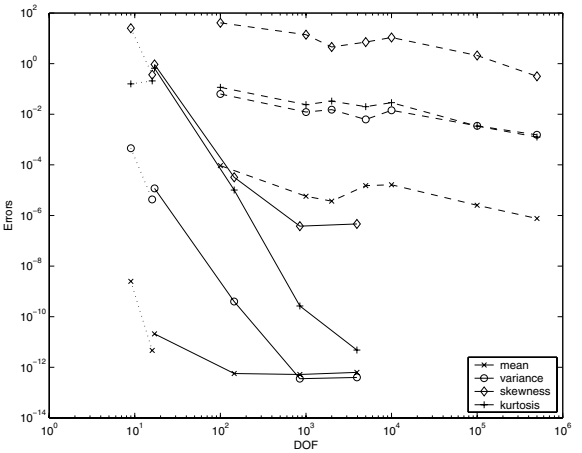


FIG. 4.2. *Convergence of errors in moments with eight-dimensional random input ($N = 8$). Smolyak sparse grid $A(N + k, N), k \leq 4$ (solid lines), Stroud's cubature (dotted lines), and Monte Carlo simulation (dashed lines).*

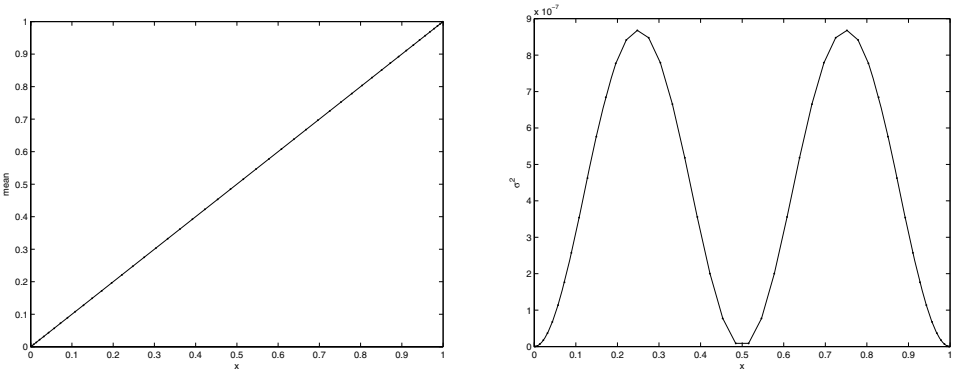


FIG. 4.3. *Stochastic solution of* (4.10) *with* 50*-dimensional ($N = 50$) random input. Left: mean solution. Right: variance.*

TABLE 4.1

Solutions at high-dimensional random input ($N = 50$), with Stroud-2 and -3 method, Smolyak method $A(N+k, N)$, and Monte Carlo method. Degrees-of-freedom for polynomial spaces: for $k = 1$, $\dim(W_N^k) = 51$, $\dim(Z_N^k) \approx 1.1 \times 10^{16}$; for $k = 2$, $\dim(W_N^k) = 1,326$, $\dim(Z_N^k) \approx 7.2 \times 10^{23}$.

Methods	DOF	$\max \sigma_u^2(x)$
Stroud 2	51	8.67914(−7)
Stroud 3	100	8.67916(−7)
Smolyak ($k = 1$)	101	8.67925(−7)
Smolyak ($k = 2$)	5101	8.67919(−7)
MCS	100	8.16677(−7)
MCS	5000	8.73434(−7)

that kurtosis is the fourth moment of the solution and neither of the Stroud's methods is precise beyond third-order polynomials.

4.2.3. High-dimensional random inputs: Efficiency. In this section, we focus on problem (4.10) with high-dimensional random input, $N = 50$. In Figure 4.3, the solution profiles of mean and variance are shown. The maximums of variance are tabulated in Table 4.1. It can be seen that the Stroud-2, -3 method and Smolyak method with $k = 2$ agree with each other well. With similar numbers of degrees-of-freedom, Monte Carlo methods show noticeable differences. In this particular case, both the Stroud methods (Stroud-2 and Stroud-3) and the Smolyak method ($k = 1$ and $k = 2$) are superior to the Monte Carlo method, and the Stroud-2 method is to be preferred, as it gives sufficiently accurate results (in variance) with only 51 nodes. The collocation points based on tensor products and Galerkin methods in tensor polynomial spaces Z_N^k are clearly not practical, as $\dim(Z_N^k) \approx 7.2 \times 10^{23}$. The Galerkin methods in the complete polynomial spaces have number of expansion terms of 51 and 1,326 for $k = 1, 2$, respectively. The choice between a Galerkin method and a Smolyak collocation method in this case will depend on users, i.e., whether to use a deterministic solver more times or to design a solver for a coupled system of equations with less terms.

5. Time dependent problem: An illustrative example. Although the stochastic collocation algorithm is discussed in the context of boundary value problems, it is straightforward to apply it to time-dependent problems. Let us consider a damped motion of a particle in a bistable potential in the presence of a noisy periodic forcing. In particular, we consider for $\omega \in \Omega$

$$\frac{dx}{dt}(t,\omega) = x - x^3 + f(t,\omega), \qquad x(0,\omega) = x_0, \tag{5.1}$$

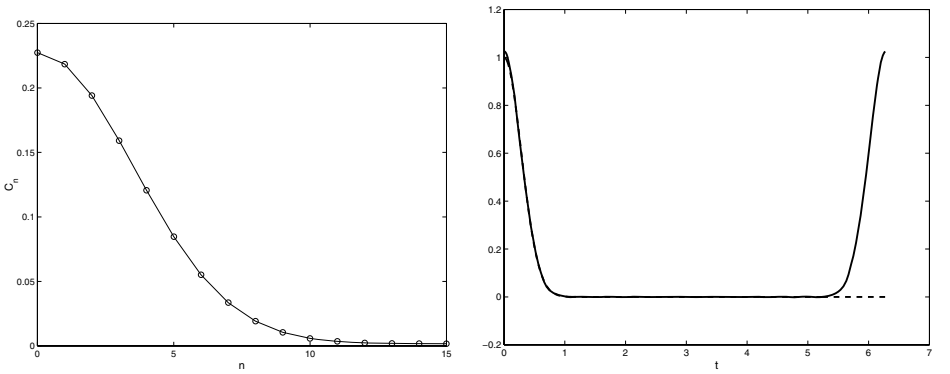
where $f(t,\omega)$ is a 2π-periodic random process with zero mean value and a 2π-periodic covariance function, i.e., $\mathbb{E}[f(t,\omega)] = 0$, and $\mathbb{E}[f(t_1,\omega)f(t_2,\omega)] = \sigma_f^2 C_f(|t_1 - t_2|)$, where σ_f^2 is the variance and $C_f(t) = C_f(t + 2\pi)$. We remark that with proper noisy input on top of a periodic driving force, the system (5.1) can exhibit complicated behavior such as stochastic resonance. (See, for example, [17].) However, the purpose of this example is on computational aspects of the algorithm, and we will restrict ourselves to cases with relatively simple dynamical behavior.

To model the periodic random process $f(t,\omega)$, we use a Fourier-type expansion

$$\hat{f}(t,\omega) = \sum_{n=-K}^{K} f_n(\omega) e^{-int} \tag{5.2}$$

for its characterization by a finite number of random variables. Here, the coefficients $f_n(\omega) = f_n^r(\omega) + i f_n^i(\omega)$ are complex random variables. It is straightforward to show that if f_n^r and f_n^i are statistically independent for all n and have zero mean and variance of $C_n/4$, where $C_n = \frac{1}{\pi}\int_0^{2\pi} C_f(t)\cos(nt)dt$ are the coefficients of the Fourier cosine series of the 2π-periodic covariance function C_f, then the random field $\hat{f}(t,\omega)$ given by (5.2) approximates the prescribed 2π-periodic covariance function $C_{\hat{f}} \to C_f$ as $K \to \infty$. Let $y(\omega) = (f_0^r, f_1^r, f_1^i, \ldots, f_K^r, f_K^i)$, then $\hat{f}(t,\omega) \equiv \hat{f}(t, y(\omega))$ has $(2K+1)$ random dimensions. (Note f_0^i does not take effect in (5.2).) Obviously, $\hat{f}(t, y(\omega))$ is an approximation of $f(t,\omega)$, as (5.2) is a finite-term summation. In practice, we choose the number of expansion terms so that the contributions from the truncated terms are sufficiently small.

In this numerical example, we choose a 2π-periodic covariance function C_f by extending the standard Gaussian covariance function $C_G(t) = \exp(-t^2/b^2)$ to the



(a) (b)

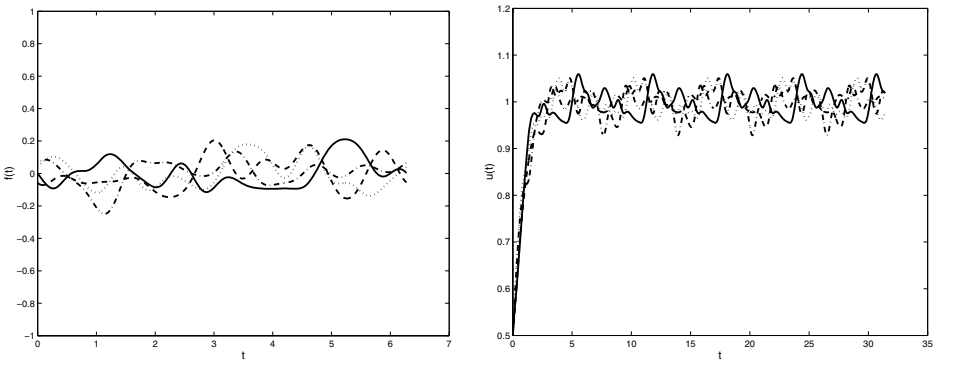
FIG. 5.1. *Construction of a 2π-periodic covariance function C_f.* (a) *Decay of Fourier cosine coefficients $C_n, n = 0, 1, \ldots$;* (b) *reconstructed periodic covariance function (solid line) by extending the regular Gaussian covariance function (dashed line) $C_G(t) = \exp(t^2/b^2)$ with correlation length $b = 0.4$.*

periodic domain $(0, 2\pi)$, where b is the correlation length. Hereafter we choose a moderately short correlation length $b = 0.4$. In Figure 5.1(a), the decay of the Fourier cosine coefficient of C_f is shown. Due to the relatively fast decay of these coefficients, we choose to use $K = 10$ in the truncated series (5.2). Thus, $\hat{f}(t, y)$ has $N = 2K+1 = 21$ random dimensions. The constructed covariance function $C_{\hat{f}}(t)$ is shown in Figure 5.1(b), along with the standard nonperiodic Gaussian covariance function C_G, which we extended to the periodic domain.

Figure 5.2(a) shows four (arbitrarily chosen) realizations of $\hat{f}(t, y)$, constructed by $N = 21(K = 10)$ dimensions with correlation length $b = 0.4$ and $\sigma = 0.1$. Corresponding to these four realizations of forcing, the four realizations of solution to (5.1), with initial condition $x_0 = 0.5$, are shown in Figure 5.2(b). In this case, all realizations are attracted to one of the stable wells ($x = 1$) and oscillate nonlinearly around it. Subsequently, $\mathbb{E}[u(t)] \to 1$ and $\mathbb{E}[u^2(t)] \to \sigma_u^2 = \text{const.}$ as $t \to \infty$. The computational results of the evolution of the standard deviation $\sigma_u(t)$ is shown in Figure 5.3. Stochastic collocation based on the Smolyak grid $A(N + k, N)$ is conducted for $k = 1$ and $k = 2$. With $N = 21$, $A(N+1, N)$ has 43 points and $A(N+2, N)$ has 925 points. The results of standard deviation with $k = 2$ shows little visual difference compared to $k = 1$, and is regarded as the well-resolved solution. Results by Monte Carlo simulation are also shown in Figure 5.3. We observe the numerical oscillations resulted from the statistical errors, and the reduction of such errors from 100 realizations to 1,000 realizations is rather modest.

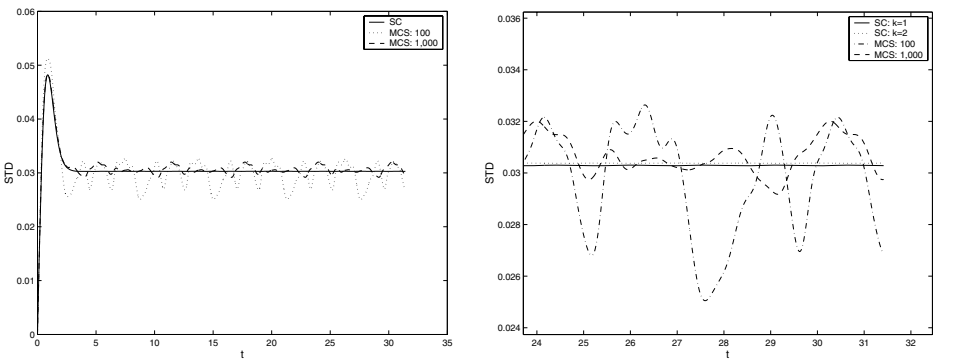
Problem (5.1) can also be solved via a stochastic Galerkin method. In the stochastic Galerkin method, we expand the solution as $u(t, y) = \sum_{i=1}^{M} u_i(t)\Phi_i(y)$, where $\{\Phi_i(y)\}$ is the multidimensional orthogonal basis defined in (2.13). The resulting equations for the expansion coefficients take the form

(5.3)
$$\frac{du_m}{dt}(t) = x_m - \frac{1}{\langle \Phi_m^2 \rangle} \sum_{i=1}^{M} \sum_{j=1}^{M} \sum_{k=1}^{M} x_i x_j x_k \langle \Phi_i \Phi_j \Phi_k \Phi_m \rangle + f_m(t), \qquad m = 1, \ldots, M,$$



(a) (b)

FIG. 5.2. *Realizations of problem* (5.1). (a) *Four realizations of the periodic random forcing* $\hat{f}(t, y(\omega))$*;* (b) *four realizations of the solution* $u(t, y(\omega))$ *corresponding to the four realizations of forcing on the left.*



(a) (b)

FIG. 5.3. *Numerical results of the evolution of standard deviation (STD) of* (5.1)*, computed by stochastic collocation (SC) and Monte Carlo simulations (MCS).* (a) *Solutions to* $0 \le t \le 10\pi$*;* (b) *close-up view near* $t \approx 10\pi$.

where $\{f_i(t)\}$ are the expansion coefficients for the random forcing term. Solution of each of the M number of equations requires a M^3 triple summations and storage (or on-the-fly evaluations) of the constants $\langle \Phi_i \Phi_j \Phi_k \Phi_m \rangle$ of the size M^4. In the case here, when $N = 21$, $M = 22$ for first-order expansion, and $M = 252$ for second-order expansion. Neither the storage nor computing of these is negligible.

6. Summary. A class of high-order collocation methods is proposed in this paper. The method is a forge of several known mathematical and numerical components, e.g., sampling method, cubature formula/sparse grid, and collocational method. Its formulation is similar to that of a Monte Carlo method, where a deterministic equation is solved repetitively at each discrete point in the random space. However, in collocation methods the distribution of collocation points is fixed deterministically a priori and is determined through the aid of existing theory of multivariate polynomial inter-

polation (or integration). Furthermore, by constructing the appropriate polynomial interpolations, stochastic collocation methods can achieve fast convergence (exponential convergence under sufficient smoothness conditions), similar to stochastic Galerkin methods. Thus, the stochastic collocation methods can be regarded as high-order "deterministic sampling methods." Also, when the interpolation error of certain collocation nodes is known, as in the case of the Smolyak sparse grid, we can obtain error estimates in strong form (e.g., the error estimate in (4.7)), whereas in cubature sampling or statistical sampling methods the interpolation errors are usually not known.

Several choices of the collocation nodal sets were investigated. It was found that the sparse grid based on Smolyak algorithm offers high accuracy and fast convergence, with a weak dependence on the number of random dimensions. An alternative, based on the Stroud-2 and -3 cubature, is optimal at lower orders. However, their accuracy is fixed and cannot be improved. Numerical experiments were conducted on an elliptic equation with random inputs, and we demonstrate that the error estimate can be effectively separated into two parts: one for the standard spatial discretization error and one for the interpolation error in random space. It was found that for random dimensions as high as 50, the stochastic collocation methods (including both Stroud's methods and Smolyak methods) are more efficient than brute-force Monte Carlo methods. With low accuracy, the Stroud-3 method is preferred to the Smolyak method with degree one ($k = 1$). On the other hand, the implementation of a collocation method is "embarrassingly" trivial, when compared to a stochastic Galerkin method which requires a specialized solver for the resulting coupled set of equations. However, the high-order collocation method based on Smolyak sparse grid has more degrees-of-freedom than Galerkin methods in complete polynomial spaces. An example of a time dependent nonlinear ODE is also presented. The collocation method exhibits good convergence and accuracy for a 21-dimensional random input. This problem has a cubic nonlinearity, and a stochastic Galerkin approach, albeit with less expansion terms compared to the sparse grid collocation method, would incur nonnegligible extra computation cost due to the nonlinearity. The choice between a collocation method and a Galerkin method is problem dependent, and the advantage of collocation methods will be more noticeable for problems with more complicated forms of governing equations. A complete comparison between collocation and Galerkin methods depends on many factors, including detailed error estimates (particularly, in this paper we did not study the aliasing error by collocation methods) and assessment of the algorithm efficiency for a Galerkin method (for solving the coupled set of equations). This is our ongoing research.

REFERENCES

[1] I. Babuška and P. Chatzipantelidis, *On solving elliptic stochastic partial differential equations*, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 4093–4122.

[2] I. Babuška, R. Tempone, and G. Zouraris, *Galerkin finite element approximations of stochastic elliptic differential equations*, SIAM J. Numer. Anal., 42 (2004), pp. 800–825.

[3] V. Barthelmann, E. Novak, and K. Ritter, *High dimensional polynomial interpolation on sparse grid*, Adv. Comput. Math., 12 (1999), pp. 273–288.

[4] I. Berezin and N. Zhidkov, *Computing Methods*, Addison-Wesley, Reading, MA, 1965.

[5] C. Bernardi and Y. Maday, *Properties of some weighted Sobolev spaces and application to spectral approximations*, SIAM J. Numer. Anal., 26 (1989), pp. 769–829.

[6] C. Canuto and A. Quarteroni, *Approximation results for orthogonal polynomials in Sobolev spaces*, Math. Comput., 38 (1982), pp. 67–86.

[7] Q.-Y. Chen, D. Gottlieb, and J. Hesthaven, *Uncertainty analysis for the steady-state flows in a dual throat nozzle*, J. Comput. Phys., 204 (2005), pp. 387–398.

[8] K. Chung and T. Yao, *On lattices admitting unique Lagrange interpolations*, SIAM J. Numer. Anal., 14 (1977), pp. 735–743.
[9] R. Cools, *Monomial cubature rules since 'Stroud': A compilation, part* 2, J. Comput. Appl. Math., 112 (1999), pp. 21–27.
[10] R. Cools and P. Rabinowitz, *Monomial cubature rules since 'Stroud': A compilation*, J. Comput. Appl. Math., 48 (1993), pp. 309–326.
[11] P. Davis, *Interpolation and Approximation*, Dover, New York, 1975.
[12] M. Deb, I. Babuška, and J. Oden, *Solution of stochastic partial differential equations using Galerkin finite element techniques*, Comput. Methods Appl. Mech. Engrg., 190 (2001), pp. 6359–6372.
[13] G. Fishman, *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, New York, 1996.
[14] B. Fox, *Strategies for Quasi-Monte Carlo*, Kluwer, Dordrect, The Netherlands, 1999.
[15] P. Frauenfelder, C. Schwab, and R. Todor, *Finite elements for elliptic problems with stochastic coefficients*, Comput. Methods Appl. Mech. Engrg., 194 (2005), pp. 205–228.
[16] D. Gamerman, *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*, Chapman & Hall, London, 1997.
[17] L. Gammaitoni, P. Hanggi, P. Jung, and F. Marchesoni, *Stochastic resonance*, Rev. Modern Phys., 70 (1998), pp. 223–287.
[18] C. Gardiner, *Handbook of Stochastic Methods: For Physics, Chemistry and the Natural Sciences*, 2nd ed., Springer-Verlag, New York, 1985.
[19] R. Ghanem, *Ingredients for a general purpose stochastic finite element formulation*, Comput. Methods Appl. Mech. Engrg., 168 (1999), pp. 19–34.
[20] R. Ghanem, *Stochastic finite elements for heterogeneous media with multiple random non-Gaussian properties*, ASCE J. Engrg. Mech., 125 (1999), pp. 26–40.
[21] R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, Springer-Verlag, New York, 1991.
[22] W. Gilks, S. Richardson, and D. Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall, London, 1995.
[23] M. Grigoriu, *Simulation of stationary non-Gaussian translation processes*, J. Engrg. Mech., 124 (1998), pp. 121–126.
[24] E. Isaacson and H. Keller, *Analysis of Numerical Methods*, John Wiley, New York, 1966.
[25] I. Karatzas and S. Shreve, *Brownian Motion and Stochastic Calculus*, Springer-Verlag, New York, 1988.
[26] G. Karniadakis and S. Sherwin, *Spectral/hp Element Methods for CFD*, Oxford University Press, London, 1999.
[27] M. Kleiber and T. Hien, *The stochastic finite element method*, John Wiley, New York, 1992.
[28] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*, Springer-Verlag, New York, 1999.
[29] O. Le Maitre, O. Knio, H. Najm, and R. Ghanem, *Uncertainty propagation using Wiener-Haar expansions*, J. Comput. Phys., 197 (2004), pp. 28–57.
[30] O. Le Maitre, H. Najm, R. Ghanem, and O. Knio, *Multi-resolution analysis of Wiener-type uncertainty propagation schemes*, J. Comput. Phys., 197 (2004), pp. 502–531.
[31] W. Liu, T. Belytschko, and A. Mani, *Probabilistic finite elements for nonlinear structural dynamics*, Comput. Methods Appl. Mech. Engrg., 56 (1986), pp. 61–81.
[32] W. Liu, T. Belytschko, and A. Mani, *Random field finite elements*, Internat. J. Numer. Methods Engrg., 23 (1986), pp. 1831–1845.
[33] M. Loève, *Probability Theory*, 4th ed., Springer-Verlag, New York, 1977.
[34] W. Loh, *On Latin hypercube sampling*, Ann. Statist., 24 (1996), pp. 2058–2080.
[35] G. Lorentz, *Approximation of Functions*, Holt, Rinehart and Winston, New York, 1966.
[36] N. Madras, *Lectures on Monte Carlo Methods*, American Mathematical Society, Providence, RI, 2002.
[37] L. Mathelin and M. Hussaini, *A stochastic collocation algorithm for uncertainty analysis*, Tech. Report NASA/CR-2003-212153, NASA Langley Research Center, 2003.
[38] I. Mysovskih, *Proof of the minimality of the number of nodes in the cubature formula for a hypersphere*, USSR Comput. Math. Math. Phys., 6 (1966), pp. 15–27.
[39] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992.
[40] H. Niederreiter, P. Hellekalek, G. Larcher, and P. Zinterhof, *Monte Carlo and Quasi-Monte Carlo Methods* 1996, Springer-Verlag, New York, 1998.
[41] E. Novak and K. Ritter, *High dimensional integration of smooth functions over cubes*, Numer. Math., 75 (1996), pp. 79–97.
[42] E. Novak and K. Ritter, *Simple cubature formulas with high polynomial exactness*, Con-

structive Approx., 15 (1999), pp. 499–522.
[43] B. Oksendal, *Stochastic Differential Equations. An Introduction with Applications*, 5th ed., Springer-Verlag, New York, 1998.
[44] M. Rosenblatt, *Remark on a multivariate transformation*, Ann. Math. Statist., 23 (1953), pp. 470–472.
[45] S. Sakamoto and R. Ghanem, *Simulation of multi-dimensional non-Gaussian non-stationary random fields*, Probab. Engrg. Mech., 17 (2002), pp. 167–176.
[46] S. Smolyak, *Quadrature and interpolation formulas for tensor products of certain classes of functions*, Soviet Math. Dokl., 4 (1963), pp. 240–243.
[47] C. Soize and R. Ghanem, *Physical systems with random uncertainties: Chaos representations with arbitrary probability measure*, SIAM. J. Sci. Comput., 26 (2004), pp. 395–410.
[48] M. Stein, *Large sample properties of simulations using Latin hypercube sampling*, Technometrics, 29 (1987), pp. 143–151.
[49] A. Stroud, *Remarks on the disposition of points in numerical integration formulas*, Math. Comput., 11 (1957), pp. 257–261.
[50] G. Wasilkowski and H. Wozniakowski, *Explicit cost bounds of algorithms for multivariate tensor product problems*, J. Complexity, 11 (1995), pp. 1–56.
[51] N. Wiener, *The homogeneous chaos*, Amer. J. Math., 60 (1938), pp. 897–936.
[52] D. Xiu and G. Karniadakis, *Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos*, Comput. Methods Appl. Math. Engrg., 191 (2002), pp. 4927–4948.
[53] D. Xiu and G. Karniadakis, *The Wiener-Askey polynomial chaos for stochastic differential equations*, SIAM J. Sci. Comput., 24 (2002), pp. 619–644.
[54] D. Xiu and G. Karniadakis, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, J. Comput. Phys., 187 (2003), pp. 137–167.
[55] D. Xiu and G. Karniadakis, *A new stochastic approach to transient heat conduction modeling with uncertainty*, Internat. J. Heat Mass Trans., 46 (2003), pp. 4681–4693.
[56] D. Xiu and G. Karniadakis, *Supersensitivity due to uncertain boundary conditions*, Internat. J. Numer. Meth. Engrg., 61 (2004), pp. 2114–2138.
[57] D. Xiu and S. Sherwin, *Uncertainty modelling in hyperbolic systems and its applications in reduced modelling of a human arterial network*, in Proceedings of International Conference of High-Order and Spectral Methods (ICOSAHOM), Providence, RI, 2004.
[58] F. Yamazaki and M. Shinozuka, *Digital generation of non-Gaussian stochastic fields*, J. Engrg. Mech., 114 (1988), pp. 1183–1197.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1141–1167
© 2006 Society for Industrial and Applied Mathematics

A SECOND ORDER ACCURATE EMBEDDED BOUNDARY METHOD FOR THE WAVE EQUATION WITH DIRICHLET DATA*

HEINZ-OTTO KREISS† AND N. ANDERS PETERSSON‡

Abstract. The accuracy of Cartesian embedded boundary methods for the second order wave equation in general two-dimensional domains subject to Dirichlet boundary conditions is analyzed. Based on the analysis, we develop a numerical method where both the solution and its gradient are second order accurate. We avoid the small-cell stiffness problem without sacrificing the second order accuracy by adding a small artificial term to the Dirichlet boundary condition. Long-time stability of the method is obtained by adding a small fourth order dissipative term. Several numerical examples are provided to demonstrate the accuracy and stability of the method. The method is also used to solve the two-dimensional TM_z problem for Maxwell's equations posed as a second order wave equation for the electric field coupled to ordinary differential equations for the magnetic field.

Key words. wave equation, stability, accuracy, embedded boundary

AMS subject classifications. 65M06, 65M12

DOI. 10.1137/040604728

1. Introduction.

Consider the Dirichlet problem for the second order wave equation in a two-dimensional domain Ω with boundary Γ:

$$u_{tt} = \Delta u + F(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, \quad t \geq 0, \tag{1}$$

$$u(\boldsymbol{x}, t) = f(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Gamma, \quad t \geq 0, \tag{2}$$

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega, \tag{3}$$

$$u_t(\boldsymbol{x}, 0) = u_1(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{4}$$

In this paper we continue the approach started in [1] and [2] and develop a numerical method that directly discretizes the second order wave equation. This approach is motivated by wave propagation problems from applications such as seismology, acoustics, and general relativity, where the underlying differential equations are systems of second order hyperbolic partial differential equations. For every second order hyperbolic system, there is an equivalent but larger first order hyperbolic system. For example, two-dimensional acoustic wave propagation is governed by a scalar second order wave equation for the pressure or by a system of three first order hyperbolic equations for the two velocity components and the pressure. Hence, the obvious advantage of the second order formulation is that the size of the system is reduced. Both continuous formulations need one boundary condition to form a well-posed initial-boundary value problem, which means that two extrapolation boundary conditions must be supplied to close the discretized first order system. This can be done, but one has to be careful not to introduce instabilities [3]. Furthermore, if the solution is not properly resolved

*Received by the editors March 3, 2004; accepted for publication (in revised form) December 8, 2004; published electronically January 6, 2006. This work was performed under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory, under contract W-7405-Eng-48.

http://www.siam.org/journals/sisc/27-4/60472.html

†Department of Mathematics, University of California, Los Angeles, Los Angeles, CA 90024 (kreiss@math.ucla.edu).

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 (andersp@llnl.gov).

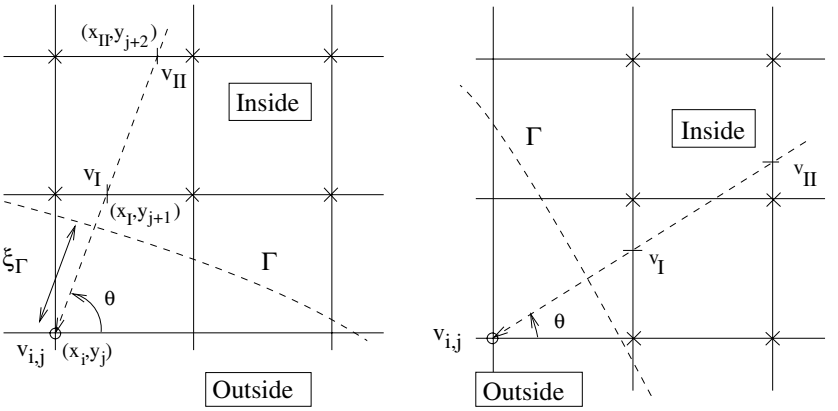


FIG. 1. *The points used for discretizing the Dirichlet boundary condition for the cases of horizontal interpolation (left) and vertical interpolation (right).*

on the grid, the discretized first order system allows for spurious waves that travel in the wrong direction [4]. The latter two problems can be circumvented by using a staggered grid (which amounts to nothing else but solving the original second order problem in disguise). However, when the boundary is not aligned with the grid, it is hard to find an accurate and stable staggered grid discretization of the boundary conditions. In contrast, the second order formulation does not need extrapolation boundary conditions and, as we have shown in [1], a node based discretization of the second order formulation avoids the difficulties with spurious waves traveling in the wrong direction. The subject of the present paper is to derive a stable second order accurate discretization of the wave equation subject to Dirichlet boundary conditions on boundaries that are not aligned with the grid.

A Cartesian embedded boundary approach will be used to solve the above problem numerically. We cover the domain Ω by a regular Cartesian grid with points $\boldsymbol{x}_{i,j} = (ih, jh)^T$, where i, j are integers and h is the grid size. The boundary is allowed to intersect the grid in an arbitrary manner; see Figure 1. Centered finite differences are used to discretize the wave equation in both space and time and the boundary condition is approximated using straightforward interpolation. However, the details of how the boundary condition is discretized are important since they determine the accuracy and stability of the numerical method. In the current paper, we discretize the Dirichlet boundary condition such that both the solution and its gradient become second order accurate. Furthermore, we add an artificial term to the discrete boundary condition that removes the stiffness due to small cells cut by the boundary without sacrificing the second order accuracy of the method. As a result, we can use an explicit time-integration method where the time step essentially equals that of a periodic domain. Stability of the method is obtained by using a small fourth order stabilizing term of the form $h^3 A^T A u_t$, where A is the matrix representation of the discrete Laplace operator satisfying the discrete boundary condition [2].

The embedded boundary technique for discretizing partial differential equations dates back to the first order technique by Weller and Shortley [5] and the higher order generalizations of Collatz [6]. More recently, several embedded boundary methods have been presented for various types of partial differential equations. For example, Pember et al. [7] used a Cartesian grid method for solving the time-dependent equa-

tions of gas dynamics. For the one- and two-dimensional Euler equations, Berger, Helzel, and LeVeque [8] developed a Cartesian "h-box" method which aims at avoiding the small-cell time step restriction without sacrificing accuracy. Zhang and LeVeque [9] solved the acoustic wave equation with discontinuous coefficients written as a first order system. A staggered grid method was used by Ditkowski, Dridi, and Hesthaven [10] for solving Maxwell's equations on a Cartesian grid. The methods described in these papers all solve first order systems (in time). For Poisson's equation with Dirichlet boundary conditions, Johansen and Colella [11] derived an embedded boundary technique based on the finite volume method combined with multigrid. More recently, Li and Greengard [12] developed a method for the scalar second order wave equation in two space dimensions, which combines a free-space Green's function evolution formula with an integral equation technique to satisfy Dirichlet or Neumann boundary conditions.

We have previously developed a Cartesian embedded boundary method for solving the wave equation subject to Dirichlet conditions [1]. That numerical method produces a solution that is second order accurate in max-norm, but the gradient of the solution is only first order accurate. Having accurate gradients is important in applications such as the elastic wave equation, where the stresses depend on the gradient of the displacement and the boundary condition on a free surface is expressed in terms of the stresses. As a step toward solving the elastic wave equation, we will here consider Maxwell's equations written as a second order wave equation for the electric field. In this application it is important to compute the gradient of the electric field to second order accuracy, since the time derivative of the magnetic field depends on the curl of the electric field.

There are two reasons why difference methods that use the embedded boundary discretization technique have problems calculating gradients with full accuracy:

1. Even if the initial data are fully compatible with the boundary condition for the analytic problem, the spatial discretization can destroy the compatibility on the truncation error level, which can degrade the rate of convergence for the gradient. This problem occurs already in one space dimension, and in section 2 we will discuss how it can be avoided by using a "smooth start" procedure, where we change variables

$$u(\boldsymbol{x},t) = \tilde{u}(\boldsymbol{x},t) + (u_0(\boldsymbol{x}) + t u_1(\boldsymbol{x}))e^{-t^2} \tag{5}$$

   and numerically solve the problem for $\tilde{u}$. Since the new variable $\tilde{u}$ satisfies a modified wave equation subject to homogeneous initial conditions, there will be no incompatibilities on the truncation error level for the solution or its first time derivative.
2. Because the boundary is embedded in a regular Cartesian grid, the truncation error in the boundary condition can oscillate wildly between consecutive grid points along the boundary. This problem occurs in two or more space dimensions and can again degrade the rate of convergence for the gradient. In section 3, we first analyze this problem for Poisson's equation and then show that the wave equation behaves in the same way as long as the truncation error is smooth in time and compatible with the initial conditions. The analysis shows that due to the highly oscillatory truncation error, the Dirichlet boundary condition must be discretized using a third order accurate formula to obtain a second order accurate gradient of the solution. The analysis also shows that a second order truncation error that varies smoothly along the

boundary also results in a second order accurate gradient.

Guided by these results, we develop a new Cartesian embedded boundary method for the Dirichlet problem which we outline now. Let $v_{i,j}(t)$ denote the semidiscrete approximation of $u(x_i, y_j, t)$. A second order approximation of the Laplacian of u is given by

$$\Delta_h v_{i,j} =: \frac{1}{h^2}(v_{i+1,j} + v_{i-1,j} + v_{i,j+1} + v_{i,j-1} - 4v_{i,j}). \tag{6}$$

To be able to evaluate $\Delta_h v_{i,j}$ at all grid points inside Ω, we use ghost points just outside the domain; see Figure 1. Let x_I, x_{II} be the intersections between the normal going through $\boldsymbol{x}_{i,j}$ and the grid lines $y = y_{j+1}$ and $y = y_{j+2}$, respectively. Denote by ξ_I, $\xi_{II} = 2\xi_I$ the distances between $\boldsymbol{x}_{i,j}$ and (x_I, y_{j+1}), (x_{II}, y_{j+2}), respectively. To aid in the approximation of the Dirichlet condition, we construct a Lagrange interpolant between three points along the normal: $(0, v_{i,j})$, (ξ_I, v_I), (ξ_{II}, v_{II}),

$$Lv =: g_0(\xi_\Gamma)v_{i,j} + g_I(\xi_\Gamma)v_I + g_{II}(\xi_\Gamma)v_{II}, \tag{7}$$

where the coefficients are given by

$$g_0(\xi) = \frac{(\xi_I - \xi)(2\xi_I - \xi)}{2\xi_I^2}, \quad g_I(\xi) = \frac{\xi(2\xi_I - \xi)}{\xi_I^2}, \quad g_{II}(\xi) = \frac{\xi(\xi - \xi_I)}{2\xi_I^2}. \tag{8}$$

To approximate v_I and v_{II}, we interpolate along the horizontal grid lines y_{j+1} and y_{j+2}:

$$v_I = d_1 v_{i,j+1} + d_2 v_{i+1,j+1} + d_3 v_{i+2,j+1}, \tag{9}$$

$$v_{II} = d_4 v_{i,j+2} + d_5 v_{i+1,j+2} + d_6 v_{i+2,j+2}, \tag{10}$$

where $d_k = \mathcal{O}(1)$ are the quadratic Lagrange interpolation coefficients. The interpolant $Lu(t)$ is a third order approximation of the boundary value, i.e., $Lu(t) = u(\boldsymbol{x}_{i,j}^\Gamma, t) + \mathcal{O}(h^3)$, where $\boldsymbol{x}_{i,j}^\Gamma$ is the intersection point between the boundary and the normal going through the ghost point $\boldsymbol{x}_{i,j}$.

The interpolation formulas for v_I and v_{II}, (9)–(10), hold when the angle θ between the normal and the x-axis satisfies $\pi/4 \le \theta \le \pi/2$. When $0 \le \theta \le \pi/4$, the horizontal interpolations are replaced by corresponding interpolations in the vertical direction; see Figure 1. Despite the sudden switching between horizontal and vertical interpolation at $\theta = \pi/4$, the interpolation formulas for v_I and v_{II} are Lipschitz continuous in θ. Approaching the limit from above, the coefficients in the horizontal interpolation formulas (9)–(10) tend to $d_2 = d_6 = 1$, $d_1 = d_3 = d_4 = d_5 = 0$. Geometrically, this corresponds to the normal intersecting the mesh along the diagonal, so $v_I = v_{i+1,j+1}$ and $v_{II} = v_{i+2,j+2}$. The same limit is found with the corresponding vertical interpolation formulas as $\theta \to \pi/4$ from below. The Lipschitz continuity also holds at $\theta = \pi/2$, where the horizontal interpolation switches from using points to the right of the ghost point to using points to the left of it. At $\theta = \pi/2$, both the left and the right interpolation formulas use $v_I = v_{i,j+1}$ and $v_{II} = v_{i,j+2}$. The same argument shows Lipschitz continuity at $\theta = 0$, where the vertical interpolation switches from using points above the ghost point to using points below it. The formulas in the other three quadrants are simply obtained by reflections in index space, leading to a total of 8 different cases to treat all possible directions of the boundary.

When the normal is close to vertical (or horizontal), the boundary can intersect the grid such that ξ_Γ is arbitrarily close to ξ_I; i.e., $g_0(\xi_\Gamma)$ can be arbitrarily close to

zero. The limit $\xi_\Gamma = \xi_I$ corresponds to v_I being on the boundary and coinciding with a grid point, in which case one could eliminate the ghost point from the computation and simply use $v_I = f$ as a discrete boundary condition. This could be achieved for one ghost point by shifting the mesh appropriately. But for a general two-dimensional domain discretized on a fine grid, there can be thousands of ghost points and it is not possible to move the mesh to ensure that ξ_Γ stays bounded away from ξ_I for all ghost points simultaneously. Hence, it is important that the discretization of the boundary condition works for arbitrarily small $g_0(\xi_\Gamma)$.

If (7) were used to approximate the boundary condition, i.e., $Lv(t) = f(\boldsymbol{x}_{i,j}^\Gamma, t)$, we would need to divide by $g_0(\xi_\Gamma)$ to solve for the ghost point value using

$$v_{i,j} = \frac{1}{g_0(\xi_\Gamma)} \left(f(\boldsymbol{x}_{i,j}^\Gamma, t) - g_I(\xi_\Gamma) v_I - g_{II}(\xi_\Gamma) v_{II} \right). \tag{11}$$

The ghost point value is used to evaluate the discrete Laplacian (6), and as we demonstrate for the one-dimensional case in Appendix A, (11) would make the time-stepping of (1) very stiff as $g_0 \to 0$. This phenomenon is known as the small-cell stiffness problem. To mitigate the stiffness, we add an artificial term to the interpolant (7) and instead consider

$$B_h v(t) =: g_0(\xi_\Gamma) v_{i,j} + g_I(\xi_\Gamma) v_I + g_{II}(\xi_\Gamma) v_{II} + \gamma(v_{i,j} - 2v_I + v_{II}) = f(\boldsymbol{x}_{i,j}^\Gamma, t), \tag{12}$$

where the constant $\gamma > 0$. The artificial term is an undivided second difference in the normal direction, and thus it inflicts an $\mathcal{O}(h^2)$ error in the boundary condition approximation. When the direction of the normal changes smoothly along the boundary, the error due to the artificial term also varies smoothly along the boundary. This is precisely the situation analyzed in section 3, since the truncation error in the boundary condition is composed of a highly oscillatory $\mathcal{O}(h^3)$ part and a smooth $\mathcal{O}(h^2)$ component. The accuracy of both the solution and its discrete gradient should therefore be $\mathcal{O}(h^2)$.

Because of the artificial term in the boundary condition, the coefficient in front of the ghost point $v_{i,j}$ is bounded uniformly away from zero, since

$$\gamma \le g_0(\xi_\Gamma) + \gamma < 1 + \gamma.$$

As a result, the small-cell stiffness problem is removed. By estimating the spectrum for the one-dimensional wave equation, we find that the eigenvalue with the largest magnitude is independent of small cells near the boundary when $\gamma \ge 0.25$; see Appendix A. Numerical computations indicate that $0.2 \le \gamma \le 0.5$ works well in practice; i.e., it is big enough to allow the time step to be chosen independently of the small cells near the boundary but small enough to prevent the artificial term from dominating the error in the numerical solution.

We can use the boundary condition (12) to eliminate all ghost points in the discrete Laplacian (6). For example, at the point $(i, j+1)$ in the left part of Figure 1 we get

$$\begin{aligned} \Delta_h v_{i,j+1} = {} & \frac{1}{h^2}(v_{i+1,j+1} + v_{i-1,j+1} + v_{i,j+2} - 4v_{i,j+1}) \\ & - \frac{(g_I - 2\gamma)}{h^2(g_0 + \gamma)}(d_1 v_{i,j+1} + d_2 v_{i+1,j+1} + d_3 v_{i+2,j+1}) \\ & - \frac{(g_{II} + \gamma)}{h^2(g_0 + \gamma)}(d_4 v_{i,j+2} + d_5 v_{i+1,j+2} + d_6 v_{i+2,j+2}) + \frac{f(\boldsymbol{x}_{i,j}^\Gamma, t)}{h^2(g_0 + \gamma)}, \end{aligned} \tag{13}$$

assuming that (i,j) is the only nearest neighbor of $(i,j+1)$ that is outside of Ω. (If additional points are outside, other formulas of the type (12) would be used to eliminate those points as well.) The expression (13) defines the discrete boundary forcing function $\boldsymbol{b}$ at point $(i,j+1)$:

$$\boldsymbol{b}_{i,j+1}(t) = \frac{f(\boldsymbol{x}^{\Gamma}_{i,j}, t)}{h^2(g_0+\gamma)}. \tag{14}$$

In the interior of Ω, i.e., for points $\boldsymbol{x}_{i,j}$ where $\boldsymbol{x}_{i\pm1,j}$ and $\boldsymbol{x}_{i,j\pm1}$ are all inside Ω, $\boldsymbol{b}^n_{i,j} = 0$.

After all ghost points have been eliminated, the discrete approximation of the Laplacian of u (for functions satisfying the boundary condition $u = f(\boldsymbol{x}^{\Gamma}, t)$) can be written in matrix form

$$\Delta u(\boldsymbol{x}, t) \approx A\boldsymbol{v}(t) + \boldsymbol{b}(t). \tag{15}$$

Here the array $\boldsymbol{v}$ contains the solution at all grid points inside Ω. Using this notation, the semidiscrete approximation of (1)–(4) becomes

$$\boldsymbol{v}_{tt} = A\boldsymbol{v} + \boldsymbol{b}(t) + \boldsymbol{F}(t), \quad t \geq 0, \tag{16}$$

$$\boldsymbol{v}(0) = u_0(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{17}$$

$$\boldsymbol{v}_t(0) = u_1(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{18}$$

where $\boldsymbol{F}_{i,j}(t) = F(\boldsymbol{x}_{i,j}, t)$ for all $\boldsymbol{x}_{i,j} \in \Omega$.

We discretize the second order time derivative in (16) using a centered finite difference formula on the regular grid $t_n = nk$, $n = 0, 1, 2, \ldots$, where $k > 0$ is the time step. Let $\boldsymbol{v}^n = \boldsymbol{v}(t_n)$ and $\boldsymbol{b}^n = \boldsymbol{b}(t_n)$. Because of the discretized form of the Dirichlet boundary condition, the matrix A in (16) will not be symmetric. To avoid the resulting mild instability which was analyzed for the Neumann problem in [2], we use the discrete stabilization operator $h^3 A^T[A(\boldsymbol{v}^n - \boldsymbol{v}^{n-1})/k + d\boldsymbol{b}/dt(t_n)]$, which was proposed in the same paper. Note the time derivative of the boundary forcing is included within the brackets to make that expression a consistent approximation of Δu_t, for functions satisfying the boundary condition $u_t = f_t(\boldsymbol{x}^{\Gamma}, t)$. Also note that the discrete stabilization operator can be applied all the way up to the boundary. Away from the boundary, it is equivalent to $h^3\Delta_h^2(\boldsymbol{v}^n - \boldsymbol{v}^{n-1})/k$, which is a very efficient damping term for highly oscillatory instabilities. For the case with inhomogeneous boundary conditions and internal forcing, the proposed scheme becomes

$$\frac{\boldsymbol{v}^{n+1} - 2\boldsymbol{v}^n + \boldsymbol{v}^{n-1}}{k^2} = A\boldsymbol{v}^n + \boldsymbol{b}^n + \boldsymbol{F}(t_n) - \alpha h^3 A^T \left((A\boldsymbol{v}^n - A\boldsymbol{v}^{n-1})/k + \frac{d\boldsymbol{b}}{dt}(t_n) \right). \tag{19}$$

The numerical experiments in section 4 indicate that $\alpha = \mathcal{O}(10^{-3})$ suffices to stabilize the scheme for long time computations. We note that the sparse structure of A can be used to efficiently evaluate both $A\boldsymbol{v}$ and $A^T\boldsymbol{v}$ without the need to store the matrix explicitly; see [2]. Also note that this scheme bears many similarities with our method for the Neumann problem [2]. In particular, both methods use the same set of grid points for discretizing the boundary condition and the same type of stabilization term. Hence, the two methods can be combined in a straightforward manner to solve problems with mixed Neumann–Dirichlet conditions.

The remainder of the paper is organized as follows. The influence of incompatibilities between initial and boundary data on the truncation error level is analyzed in section 2 and the effect of highly oscillatory truncation errors in the boundary condition is studied in section 3. Two-dimensional numerical experiments are performed in section 4. We first construct a smooth solution against which the error in the numerical solution can be evaluated. The accuracy of the scheme and effects of the smooth start technique are evaluated in section 4.1. In section 4.2 we test the method by solving the TM_z problem for Maxwell's equations, i.e., the case when the electric field only has a component in the z-direction and the magnetic field has no component in the z-direction. The TM_z problem can be formulated as a scalar wave equation for the electric field in the (x, y) plane, subject to a Dirichlet boundary condition. The wave equation for the electric field is coupled to ODEs for the magnetic field that are integrated in time as the electric field is evolved. This is a good test of the scheme, since the accuracy of the magnetic field is determined by the accuracy of the curl of the electric field. Numerical experiments are performed for a circular domain where the numerical solution is compared to an analytic solution, as well as for more complex domains. Concluding remarks are presented in section 5.

2. Smooth start. We start by considering the one-dimensional half-plane problem

$$u_{tt} = u_{xx} + F(x,t), \quad a \le x < \infty, \quad t \ge 0, \tag{20}$$

with initial data

$$u(x,0) = u_0(x), \quad u_t(x,0) = u_1(x) \tag{21}$$

and the boundary condition

$$u(a,t) = f(t). \tag{22}$$

We henceforth assume that all data belong to C^∞. Therefore also the solution is smooth, provided compatibility conditions are satisfied. In the one-dimensional setting, these are

$$\begin{aligned}
u(a,0) &= u_0(a) = f(0),\\
u_t(a,0) &= u_1(a) = f_t(0),\\
u_{tt}(a,0) &= u_{0xx}(a) + F(a,0) = f_{tt}(0),\\
u_{ttt}(a,0) &= u_{1xx}(a) + F_t(a,0) = f_{ttt}(0),\\
u_{tttt}(a,0) &= u_{0xxxx}(a) + F_{xx}(a,0) + F_{tt}(a,0) = f_{tttt}(0),\\
u_{ttttt}(a,0) &= u_{1xxxx}(a) + F_{xxt}(a,0) + F_{ttt}(a,0) = f_{ttttt}(0),\\
&\vdots
\end{aligned} \tag{23}$$

We introduce a grid

$$x_\nu = \nu h, \quad \nu = 0, 1, 2, \ldots, \tag{24}$$

where $h > 0$ is the grid size. When the boundary is aligned with the grid, e.g., $a = 0$, a second order centered finite difference approximation of (20)–(22) yields the expected second order convergence of the numerical solution and its gradient. However, we are

interested in the case when the boundary is not aligned with the grid. We therefore take $a = (1-\beta)h$, $0 < \beta < 1$, so that the boundary falls between x_0 and x_1. Hence, the grid point x_0 is outside the domain but will be used as a ghost point to aid in the difference approximation. Let $v(x_j, t)$ be a grid function approximating $u(x_j, t)$. The one-dimensional counterpart of the boundary condition (12) is

$$B_h^{(1)} v(t) =: c_0(\beta)v(x_0,t) + c_1(\beta)v(x_1,t) + c_2(\beta)v(x_2,t) + \gamma(v(x_0,t) - 2v(x_1,t) + v(x_2,t)) = f(t), \quad \gamma > 0, \tag{25}$$

where the coefficients satisfy

$$c_0(\beta) = \frac{1}{2}\beta(1+\beta), \quad c_1(\beta) = (1-\beta)(1+\beta), \quad c_2(\beta) = -\frac{1}{2}\beta(1-\beta). \tag{26}$$

Similar to the two-dimensional case, we remove the stiffness in the time integration by taking $\gamma > 0$ such that the coefficient in front of $v(x_0, t)$ is bounded away from zero for $0 \le \beta \le 1$.

The semidiscrete difference approximation of (20)–(22) when the boundary is not aligned with the grid becomes

$$v_{tt}(x_\nu, t) = D_+^x D_-^x v(x_\nu, t) + F(x_\nu, t), \quad \nu = 1, 2, \ldots, \tag{27}$$

$$v(x_\nu, 0) = u_0(x_\nu), \quad v_t(x_\nu, 0) = u_1(x_\nu), \quad \nu = 1, 2, \ldots, \tag{28}$$

$$B_h^{(1)} v(t) = f(t), \tag{29}$$

where $D_+^x v(x_\nu, t) = (v(x_{\nu+1}, t) - v(x_\nu, t))/h$ is the usual forward divided difference operator in the x-direction and $D_-^x v(x_\nu, t) = D_+^x v(x_{\nu-1}, t)$. For later use, we also define the centered operator $D_0^x v(x_\nu, t) = (v(x_{\nu+1}, t) - v(x_{\nu-1}, t))/(2h)$. Using Taylor series expansions, we see that the error $e(x_\nu, t) = u(x_\nu, t) - v(x_\nu, t)$ satisfies

$$e_{tt}(x_\nu, t) = D_+^x D_-^x e(x_\nu, t) - \frac{h^2}{12} u_{xxxx}(x_\nu, t) + \mathcal{O}(h^4), \quad \nu = 1, 2, \ldots, \tag{30}$$

$$e(x_\nu, 0) = e_t(x_\nu, 0) = 0, \quad \nu = 1, 2, \ldots, \tag{31}$$

$$B_h^{(1)} e(t) = \gamma h^2 u_{xx}(a, t) + h^3 (C + \gamma\beta) u_{xxx}(a, t) + \mathcal{O}(h^4). \tag{32}$$

Since the data are $\mathcal{O}(h^2)$, $|e(\cdot, t)|_\infty = \mathcal{O}(h^2)$. To derive an estimate for the gradient of e, we first derive an equation for $w = e_{tt}$. By taking two time derivatives of the differential equation (30) and the boundary condition (32), we get

$$w_{tt}(x_\nu, t) = D_+^x D_-^x w(x_\nu, t) - \frac{h^2}{12} u_{xxxxtt}(x_\nu, t) + \mathcal{O}(h^4), \quad \nu = 1, 2, \ldots,$$

$$B_h^{(1)} w(t) = \gamma h^2 u_{xxtt}(a, t) + h^3 (C + \gamma\beta) u_{xxxtt}(a, t) + \mathcal{O}(h^4).$$

To derive initial data for w, we can apply the differential equation (30) to the initial data (31). However, this only applies away from the boundary. For $\nu \ge 2$ we have

$$w(x_\nu, 0) = -\frac{h^2}{12} u_{xxxx}(x_\nu, 0) + \mathcal{O}(h^4), \quad \nu = 2, 3, \ldots,$$

$$w_t(x_\nu, 0) = -\frac{h^2}{12} u_{xxxxt}(x_\nu, 0) + \mathcal{O}(h^4), \quad \nu = 2, 3, \ldots .$$

To evaluate the initial data at the first interior point x_1, we must first apply the boundary condition (32) to define $e(x_0, 0)$:

$$B_h^{(1)} e(0) =: (c_0 + \gamma)e(x_0, 0) + (c_1 - 2\gamma)e(x_1, 0) + (c_2 + \gamma)e(x_2, 0) \\ = \gamma h^2 u_{xx}(a, 0) + h^3(C + \gamma\beta)u_{xxx}(a, 0) + \mathcal{O}(h^4).$$

Since $e(x_1, 0) = e(x_2, 0) = 0$, $D_+^x D_-^x e(x_1, 0) = e(x_0, 0)/h^2$. Therefore, (30) gives us

$$w(x_1, 0) = e_{tt}(x_1, 0) = \frac{\gamma}{c_0 + \gamma} u_{xx}(a, 0) + \frac{h(C + \gamma\beta)}{c_0 + \gamma} u_{xxx}(a, 0) + \mathcal{O}(h^2).$$

By taking the time derivative of the boundary condition, the same procedure can be used to obtain

$$w_t(x_1, 0) = e_{ttt}(x_1, 0) = \frac{\gamma}{c_0 + \gamma} u_{xxt}(a, 0) + \frac{h(C + \gamma\beta)}{c_0 + \gamma} u_{xxxt}(a, 0) + \mathcal{O}(h^2).$$

Hence, in general the forcing in the initial data is $\mathcal{O}(1)$ at the first interior point, which makes $|w(\cdot, t)|_\infty = \mathcal{O}(1)$. Consequently, $|D_+^x D_-^x e(\cdot, t)|_\infty = \mathcal{O}(1)$ and we can not obtain the desired estimate for the gradient of e.

The situation is much better in the case with homogeneous initial data, $u_0(x) = u_1(x) = 0$. Now all spatial derivatives of u and u_t are initially zero, so the initial data for w are

$$w(x_\nu, 0) = w_t(x_\nu, 0) = 0, \quad \nu = 1, 2, \ldots .$$

Hence the forcing for w is $\mathcal{O}(h^2)$. Therefore, we can derive an estimate for $|e_{tt}(\cdot, t)|_\infty = |w(\cdot, t)|_\infty = \mathcal{O}(h^2)$ and

$$|D_+^x D_-^x e(\cdot, t)|_\infty \le |w(\cdot, t)|_\infty + \frac{h^2}{12} |u_{xxxx}(\cdot, t)|_\infty = \mathcal{O}(h^2).$$

Since $|D_+^x e(\cdot, t)|_\infty$ can be estimated in terms of e and $D_+^x D_-^x e$ using a discrete Sobolev inequality (see, e.g., [13]), we have

$$|D_+^x e|_\infty = \mathcal{O}(h^2).$$

A problem with general, but compatible, initial data can be reformulated into a problem with homogeneous initial data by changing variables according to (5). If the original variable u satisfies (1)–(4), the variable $\tilde{u}$ satisfies the modified problem

$$\tilde{u}_{tt} = \Delta\tilde{u} + \tilde{F}(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, \quad t \ge 0, \tag{33}$$

$$\tilde{u}(\boldsymbol{x}, t) = \tilde{f}(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Gamma, \quad t \ge 0, \tag{34}$$

$$\tilde{u}(\boldsymbol{x}, 0) = 0, \quad \boldsymbol{x} \in \Omega, \tag{35}$$

$$\tilde{u}_t(\boldsymbol{x}, 0) = 0, \quad \boldsymbol{x} \in \Omega, \tag{36}$$

where the modified internal forcing function is

$$\tilde{F}(\boldsymbol{x}, t) = F(\boldsymbol{x}, t) + \left[\Delta u_0(\boldsymbol{x}) + t\Delta u_1(\boldsymbol{x}) + (2 - 4t^2)u_0(\boldsymbol{x}) + (6t - 4t^3)u_1(\boldsymbol{x})\right] e^{-t^2}$$

and the boundary forcing becomes

$$\tilde{f}(\boldsymbol{x}, t) = f(\boldsymbol{x}, t) - [u_0(\boldsymbol{x}) + tu_1(\boldsymbol{x})]e^{-t^2}, \quad \boldsymbol{x} \in \Gamma.$$

To solve the problem for $u(\boldsymbol{x},t)$, we first compute $\tilde{u}(\boldsymbol{x},t)$ numerically and then add in $(u_0(\boldsymbol{x})+tu_1(\boldsymbol{x}))e^{-t^2}$ to obtain the solution of the original problem. By doing so, we obtain a second order accurate solution where the gradient and second derivative of the numerical solution are also second order accurate.

To verify the above theory, we solve the embedded boundary problem (27)–(29) numerically. Now we consider the bounded domain

$$a \le x \le b$$

and add a boundary condition at $x=b$,

$$u(b,t) = f_2(t).$$

The interior grid points are $q_0 \le \nu \le q_1$ such that $a+\beta h = x_{q_0}$ and $x_{q_1}+\beta_2 h = b$, $0<\beta_2<1$. We take initial data, interior forcing function, and boundary forcing functions such that the analytic problem is solved by $u(x,t)=\sin(2x-t+\pi/4)$, i.e.,

$$\begin{aligned} F(x_\nu,t) &= 3\sin(2x_\nu - t + \pi/4),\\ f(t) &= \sin(2a-t+\pi/4),\\ f_2(t) &= \sin(2b-t+\pi/4),\\ u_0(x_\nu) &= \sin(2x_\nu+\pi/4), \quad u_1(x_\nu) = -\cos(2x_\nu+\pi/4). \end{aligned}$$

We introduce a grid in time $t_n = nk$, $k>0$ and discretize time derivatives by centered differences. We set $h=2\pi/N$, where N is a positive integer, so that $x_N = 2\pi$. The boundary condition at $x=a$ is discretized by (25), shifted to involve the points (q_0-1, q_0, q_0+1). The boundary condition at $x=b$ is discretized using a corresponding formula involving the points (q_1-1, q_1, q_1+1) with β_2 replacing β. We arrive at the following explicit time integration scheme:

$$\begin{aligned} D_+^t D_-^t v(x_\nu,t_n) &= D_+^x D_-^x v(x_\nu,t_n) + F(x_\nu,t_n), \quad \nu = q_0, q_0+1,\dots,q_1,\\ v(x_\nu,0) &= u_0(x_\nu), \quad \nu = q_0, q_0+1,\dots,q_1,\\ v(x_\nu,-k) &= u_0(x_\nu) - ku_1(x_\nu) + \frac{k^2}{2}\left(u_{0xx}(x_\nu,0)+F(x_\nu,0)\right)\\ &\qquad -\frac{k^3}{6}\left(u_{1xx}(x_\nu,0)+F_t(x_\nu,0)\right), \quad \nu = q_0, q_0+1,\dots,q_1,\\ B_h^{(a)} v(t_n) &= f(t_n), \quad B_h^{(b)} v(t_n) = f_2(t_n). \end{aligned} \tag{37}$$

One can show that the above scheme is stable for $k/h<1$ when $\gamma \ge 0.25$; see [1] and Appendix A. Numerical computations (see Table 1) indicate that the solution is second order accurate, but the gradient is only a little better than first order accurate and the second derivative has an $\mathcal{O}(1)$ error. However, by using the smooth start procedure, i.e., changing variables according to (5) and numerically solving (33)–(36), second order accuracy is also obtained for the gradient and the second derivative; see Table 2.

We have also observed that the convergence of the gradient improves when the initial data is inhomogeneous, but the interior and boundary forcing functions are homogeneous, $F(x,t)=0$, $f(t)=0$. From the compatibility relations (23), we see that all even spatial derivatives of u and u_t are zero at $x=a$, $t=0$. In particular, $u_{xx}(a,0)=0$ and $u_{xxt}(a,0)=0$, so the leading order terms in the initial data for w become one order smaller in h. Hence, $w(x_1,0)=\mathcal{O}(h)$ and $w_t(x_1,0)=\mathcal{O}(h)$,

TABLE 1

Direct start: Error in the computed solution when the boundary is embedded between grid points. The errors are reported at time $t = 6.333$, the domain is $0.37 \le x \le 5.1$, $\gamma = 0.2$, and $k/h = 0.9$.

N	$\|u - v\|_\infty$	$\|\frac{\partial u}{\partial x} - D_0^x v\|_\infty$	$\|\frac{\partial^2 u}{\partial x^2} - D_+^x D_-^x v\|_\infty$	β	β_2
200	1.22×10^{-3}	9.36×10^{-3}	1.88×10^{0}	0.222	0.338
400	2.53×10^{-4}	3.13×10^{-3}	3.59×10^{-1}	0.445	0.676
800	6.29×10^{-5}	1.26×10^{-3}	2.98×10^{-1}	0.890	0.352

TABLE 2

Smooth start: Error in the computed solution when the boundary is embedded between grid points. The errors are reported at time $t = 6.333$, the domain is $0.37 \le x \le 5.1$, $\gamma = 0.2$, and $k/h = 0.9$.

N	$\|u - v\|_\infty$	$\|\frac{\partial u}{\partial x} - D_0^x v\|_\infty$	$\|\frac{\partial^2 u}{\partial x^2} - D_+^x D_-^x v\|_\infty$	β	β_2
200	8.90×10^{-4}	1.14×10^{-3}	1.02×10^{-3}	0.222	0.338
400	2.22×10^{-4}	2.86×10^{-4}	2.56×10^{-4}	0.445	0.676
800	5.55×10^{-5}	7.09×10^{-5}	6.38×10^{-5}	0.890	0.352

and we get $|w(\cdot,t)|_\infty = \mathcal{O}(h)$. Hence, $|e(\cdot,t)|_\infty = \mathcal{O}(h^2)$ but $|D_+^x D_-^x e(\cdot,t)|_\infty = \mathcal{O}(h)$. Numerical experiments suggest that the solution and its gradient are second order accurate but the second derivative is only first order accurate. Naturally, the error in the second derivative can be made second order accurate also in this case by using smooth start.

There are also cases when the smooth start procedure is not necessary. This occurs, for example, when the boundary data are homogeneous, $f(t) = 0$, and the internal forcing and initial data have compact support in the interior of Ω.

3. An analytic model of the Dirichlet problem. In this section we first consider Poisson's equation

$$u_{xx} + u_{yy} = F(x,y) \tag{38}$$

on the half-plane $y - 2x \le 0$ with Dirichlet boundary conditions

$$u(x,y) = f(x,y) \quad \text{for} \quad y = 2x. \tag{39}$$

We want to solve the problem numerically, use the same Cartesian grid as before, and approximate (38) by the second order approximation

$$\Delta_h v(x_m, y_l) = F_h(x_m, y_l)$$

in all interior points. Here F_h is the restriction of F to the grid.

The grid is not aligned with the boundary. Along the boundary only every second point is on the grid. A second order accurate approximation of (39) is, for example, given by

$$v(lh, 2lh) = f(lh, 2lh), \tag{40}$$

$$\frac{1}{2}\left\{v\big((l-1)h, (2l-1)h\big) + v\big(lh, (2l-1)h\big)\right\} = f\left(\left(l - \frac{1}{2}\right)h, (2l-1)h\right) \tag{41}$$

for $l = 0, \pm 1, \pm 2, \ldots$. We write (40)–(41) formally as

$$B_h v = f_h.$$

To obtain an error estimate we assume that the solution of (38)–(39) is smooth. (This is, for example, the case if F, f are smooth and decay rapidly as $|x| + |y| \to \infty$; cf. [14].) The usual truncation error analysis gives us, for the error $e = v - u$ on the grid,

$$\Delta_h e = h^2 G \quad \text{in the interior,} \tag{42}$$

$$B_h e = h^2 g \quad \text{on the boundary.} \tag{43}$$

Here G is the restriction of a smooth function to the grid but g is, in general, not smooth. The boundary condition (41) implies that g oscillates between 0 and $u_{xx} + \mathcal{O}(h^2)$ from grid point to grid point. This rapid change is typical for embedded boundary difference approximations in general domains.

If we want only to estimate e, this is tolerable because one can prove, using the discrete maximum principle, that the error is $\mathcal{O}(h^2)$. However, if we also want to calculate the gradient, then the error degrades, in general, to $\mathcal{O}(h)$. This degradation does not depend on G. If $G \neq 0$, we solve an auxiliary problem

$$\Delta_h e_1 = h^2 G \quad \text{in the interior,} \tag{44}$$

$$B_h e_1 = 0 \quad \text{on the boundary.} \tag{45}$$

The problem (44)–(45) is a second order accurate approximation of

$$\Delta u_1 = h^2 G \quad \text{in the interior,}$$
$$u_1 = 0 \quad \text{on the boundary.}$$

The function u_1 and its gradient are of order $\mathcal{O}(h^2)$. Since $e_1 - u_1 = \mathcal{O}(h^4)$ on the grid, the discrete gradient of e_1 is $\mathcal{O}(h^2)$. Subtracting the solution of (44)–(45) from the solution of (42)–(43) shows that we can assume that $G \equiv 0$; i.e., we discuss

$$\Delta_h e = 0 \quad \text{in the interior,} \tag{46}$$

$$B_h e = h^2 g \quad \text{on the boundary.} \tag{47}$$

To gain insight into the effects of highly oscillatory boundary data, we analyze a modified equation model of (46)–(47). A more sophisticated analysis, where the discrete nature of the problem is taken into account, will be presented elsewhere. The problem (46)–(47) is a second order accurate approximation of

$$\Delta e = 0, \quad y - 2x \leq 0, \tag{48}$$

$$e = h^2 g, \quad y = 2x, \tag{49}$$

and we proceed by studying the properties of its solution, in particular when g is highly oscillatory. To simplify the notation, we rotate the coordinate system and introduce the new independent variables

$$\xi = (2x - y)/\sqrt{5}, \tag{50}$$

$$\eta = (x + 2y)/\sqrt{5}. \tag{51}$$

The boundary is aligned with $\xi = 0$ and we consider the half-plane problem for $\xi \geq 0$. We assume that all functions are 2π-periodic in η. Fourier transforming (48)–(49) in η gives us

$$\hat{e}_{\xi\xi} - \omega^2 \hat{e} = 0, \quad \omega \text{ integer},$$
$$\hat{e}(0,\omega) = h^2 \hat{g}(\omega).$$

Since we are mainly interested in highly oscillatory g, we always assume that $\hat{g}(0) = 0$. Then the unique solution in L_2 is given by

$$e = h^2 \sum_{\omega \neq 0} e^{-|\omega|\xi + i\omega\eta} \hat{g}(\omega), \tag{52}$$

and we have

$$e_\xi = -h^2 \sum_{\omega \neq 0} |\omega| e^{-|\omega|\xi + i\omega\eta} \hat{g}(\omega).$$

We make the following observations:

(1) If the boundary data are smooth, i.e., $\hat{g}(\omega)$ decays rapidly, then the solution is also smooth up to the boundary.

(2) If we only know that $g \in L_2$, then we obtain, for every fixed ξ,

$$\begin{aligned}\|e_\xi(\xi,\cdot)\|^2 &= \int_{-\pi}^{\pi} |e_\xi(\xi,\eta)|^2 d\eta = \sum_{\omega \neq 0} |\omega|^2 e^{-2|\omega|\xi} |h^2 \hat{g}(\omega)|^2 \\ &= \frac{1}{\xi^2} \sum_{\omega \neq 0} |\xi\omega|^2 e^{-2|\omega|\xi} |h^2 \hat{g}(\omega)|^2 \leq \frac{\text{const.}}{\xi^2} \sum_{\omega \neq 0} e^{-|\omega|\xi} |h^2 \hat{g}(\omega)|^2.\end{aligned}$$

Thus the solution becomes smooth away from the boundary layer at $\xi = 0$.

(3) If the boundary data are highly oscillatory, i.e.,

$$g(\eta) = \sum_{|\omega| \geq 1/h} e^{i\omega\eta} \hat{g}(\omega), \quad 0 < h \ll 1,$$

we obtain

$$\begin{aligned}\|e_\xi(\xi,\cdot)\|^2 &\leq \frac{\text{const.}}{\xi^2} \sum_{|\omega| \geq 1/h} e^{-|\omega|\xi} |h^2 \hat{g}(\omega)|^2 \\ &\leq \frac{\text{const.}}{\xi^2} e^{-\xi/h} \sum_{|\omega| \geq 1/h} e^{-(|\omega| - 1/h)\xi} |h^2 \hat{g}(\omega)|^2;\end{aligned}$$

i.e., the thickness of the boundary layer is $\mathcal{O}(h)$.

(4) The highest frequency on a grid is $|\omega h| = \pi$. Hence, if $\hat{g}(\omega) = 0$ for $|\omega| > \pi/h$, then

$$\|e_\xi(\xi,\cdot)\|^2 = \frac{1}{h^2} \sum_{|\omega h| \leq \pi} |\omega h|^2 e^{-2|\omega|\xi} |h^2 \hat{g}(\omega)|^2 \leq \pi^2 \sum_{|\omega h| \leq \pi} e^{-2|\omega|\xi} |h\hat{g}(\omega)|^2.$$

Thus the gradient is of the order $\mathcal{O}(h)$.

We would like to point out that these estimates correspond to Schauder-type interior regularity estimates; see, e.g., Chapters 4 and 6 of [14].

To summarize our results for the Dirichlet problem for Poisson's equation, the gradient is only of the order $\mathcal{O}(h^2)$ if the boundary data $h^2 g(\eta)$ are the restriction of a smooth function. For highly oscillatory $g(\eta)$, the gradient will be $\mathcal{O}(h)$. We can also use a third order accurate approximation of the boundary condition. Then the solution is of the order $\mathcal{O}(h^3)$ and the discrete gradient is of the order $\mathcal{O}(h^2)$, even if the boundary data are highly oscillatory. Since the problem is linear, we also get a discrete gradient of the order $\mathcal{O}(h^2)$ when the boundary data consist of a smooth $\mathcal{O}(h^2)$ term together with a highly oscillatory $\mathcal{O}(h^3)$ component.

3.1. The wave equation. We next consider the wave equation (1)–(4) on the half-plane $y - 2x \le 0$. We use the same grid and the same spatial discretization as for Poisson's equation, and study the following semidiscrete approximation:

$$\begin{aligned} v_{tt}(x_m, y_l, t) &= \Delta_h v(x_m, y_l, t) + F_h(x_m, y_l, t), \qquad y_l - 2x_m < 0,\ t \ge 0, \\ B_h v(t) &= f_h(t) \qquad\qquad \text{on the boundary},\ t \ge 0, \\ v(x_m, y_l, 0) &= u_0(x_m, y_l), \qquad\qquad y_l - 2x_m < 0, \\ v_t(x_m, y_l, 0) &= u_1(x_m, y_l), \qquad\qquad y_l - 2x_m < 0. \end{aligned}$$

Again we are interested in the accuracy of the solution and its gradient. We assume that F, u_0, and u_1 are smooth functions that can be expanded smoothly to the entire plane such that they decay rapidly as $|x| + |y| \to \infty$. Therefore, we can assume that

$$F \equiv u_0 \equiv u_1 \equiv 0. \tag{53}$$

Otherwise, we would first solve the continuous and semidiscrete Cauchy problems which we would then subtract from u and v, respectively. The gradient of the solution of the semidiscrete Cauchy problem has full accuracy since it satisfies the same equation as the solution itself (with different data). Thus we consider

$$\begin{aligned} u_{tt} &= \Delta u, \qquad\qquad y - 2x \le 0, \quad t \ge 0, \\ u(x, y, t) &= f(x, y, t), \qquad y - 2x = 0, \quad t \ge 0, \\ u(x, y, 0) &= u_t(x, y, 0) = 0, \qquad y - 2x \le 0. \end{aligned}$$

If $f(x, 2x, 0) \ne 0$, then the boundary data are not compatible with the initial data. If we want the solution to have p bounded time derivatives, we need

$$\partial^j f(x, y, t)/\partial t^j \big|_{t=0} = 0, \quad j = 0, 1, 2, \ldots, p-1, \quad y = 2x. \tag{54}$$

The corresponding difference approximation is

$$\begin{aligned} v_{tt}(x_m, y_l, t) &= \Delta_h v(x_m, y_l, t), \qquad y_l - 2x_m < 0,\ t \ge 0, \\ B_h v(t) &= f_h(t) \qquad\qquad \text{on the boundary},\ t \ge 0, \\ v(x_m, y_l, 0) &= v_t(x_m, y_l, 0) = 0, \qquad y_l - 2x_m < 0. \end{aligned}$$

We want to obtain an estimate for the error $e = v - u$ on the grid, and by using the same truncation error technique as for Poisson's equation, we get the following discrete error equation:

$$\begin{aligned} e_{tt}(x_m, y_l, t) &= \Delta_h e(x_m, y_l, t), \qquad y_l - 2x_m < 0,\ t \ge 0, \\ B_h e(t) &= h^2 g_h(t) \qquad\qquad \text{on the boundary},\ t \ge 0, \\ e(x_m, y_l, 0) &= e_t(x_m, y_l, 0) = 0, \qquad y_l - 2x_m < 0, \end{aligned}$$

where g_h is highly oscillatory in space but smooth in time (because the interpolation formulas do not depend on time). We proceed as for Poisson's equation and study the continuous modified equation model of the discrete error equation:

$$\begin{aligned} e_{tt} &= \Delta e, && y - 2x < 0,\ t \geq 0, \\ e &= h^2 g, && y - 2x = 0,\ t \geq 0, \\ e(x, y, 0) &= e_t(x, y, 0) = 0, && y - 2x < 0. \end{aligned}$$

By rotating the coordinate system according to (50)–(51), we align the boundary with $\xi = 0$ and arrive at a half-plane problem for $\xi \geq 0$, $-\infty < \eta < \infty$. Up to now, we have assumed a second order accurate discretization of the boundary condition, but the same arguments also apply to a higher order boundary discretization. We therefore take $\phi = h^2 g$ and study the half-plane problem

$$\begin{aligned} e_{tt} &= \Delta e, \\ e(0, \eta, t) &= \phi(\eta, t), \\ e(\xi, \eta, 0) &= e_t(\xi, \eta, 0) = 0. \end{aligned} \tag{55}$$

We assume that all data are 2π-periodic functions in η, Fourier transform the problem in the η-direction, and obtain

$$\begin{aligned} \hat{e}_{tt} &= \hat{e}_{\xi\xi} - \omega^2 \hat{e}, \\ \hat{e}(0, \omega, t) &= \hat{\phi}(\omega, t), \\ \hat{e}(\xi, \omega, 0) &= \hat{e}_t(\xi, \omega, 0) = 0, \end{aligned} \tag{56}$$

with

$$\partial^j \hat{\phi}(\omega, t)/\partial t^j \big|_{t=0} = 0, \quad j = 0, 1, 2, \ldots, p-1.$$

Because of the compatibility assumption (54), the solution will be smooth in time. If the boundary data are smooth in η, i.e., the Fourier coefficients $\hat{\phi}(\omega, t)$ decay rapidly for large ω, the solution will also be smooth in space. However, as we have seen before, the truncation error in the boundary condition is highly oscillatory in space, and we proceed by studying large $|\omega|$.

When $|\omega| \gg 1$, we solve (56) by iteration. For $j = 1, 2, \ldots$, let

$$\begin{aligned} \hat{e}^{(j)}_{\xi\xi} - \omega^2 \hat{e}^{(j)} &= \hat{e}^{(j-1)}_{tt}, \\ \hat{e}^{(j)}(0, \omega, t) &= \hat{\phi}(\omega, t). \end{aligned}$$

We take $\hat{e}^{(0)} = 0$, so $\hat{e}^{(1)}$ satisfies

$$\begin{aligned} \hat{e}^{(1)}_{\xi\xi} - \omega^2 \hat{e}^{(1)} &= 0, \\ \hat{e}^{(1)}(0, \omega, t) &= \hat{\phi}(\omega, t). \end{aligned}$$

The only bounded solution is $\hat{e}^{(1)}(\xi, \omega, t) = \hat{\phi}(\omega, t) e^{-|\omega|\xi}$. The problem for $\hat{e}^{(2)}$ becomes

$$\begin{aligned} \hat{e}^{(2)}_{\xi\xi} - \omega^2 \hat{e}^{(2)} &= \hat{\phi}_{tt}(\omega, t) e^{-|\omega|\xi}, \\ \hat{e}^{(2)}(0, \omega, t) &= \hat{\phi}(\omega, t), \end{aligned}$$

which is solved by

$$\hat{e}^{(2)} = \hat{e}^{(1)} - \frac{1}{2\omega^2}\hat{\phi}_{tt}|\omega|\xi e^{-|\omega|\xi}.$$

Note that $\alpha e^{-\alpha} \leq$ const. for $\alpha \geq 0$. Hence, for $|\omega| \gg 1$ the solution of (56) is

$$\hat{e}(\xi,\omega,t) = \hat{\phi}e^{-|\omega|\xi} - \frac{1}{2\omega^2}\hat{\phi}_{tt}|\omega|\xi e^{-|\omega|\xi} + \mathcal{O}\left(\frac{1}{\omega^4}\right).$$

As long as ϕ_{tt} is of the same order as ϕ, the gradient satisfies

$$\hat{e}_\xi(\xi,\omega,t) = \hat{e}^{(1)}_\xi(\xi,\omega,t)\left(1 + \mathcal{O}\left(\frac{1}{\omega^2}\right)\right).$$

Therefore, the properties of the gradient of $\hat{e}$ can be reduced to the properties of $\hat{e}^{(1)}$, which was studied in the previous section. Hence, for boundary data $\phi(\xi,t)$ which are highly oscillatory in space but smooth in time, the gradient of e only becomes large in a boundary layer near $\xi = 0$, while the solution remains smooth in the interior of the domain. This means that if the boundary approximation is third order accurate with a highly oscillatory truncation error, i.e., $\phi = h^3 g$, the gradient of the solution will be third order accurate except near the boundary $\xi = 0$, where it will be second order accurate.

Note that the time-smoothness assumption is essential because ϕ_{tt} appears as forcing for $\hat{e}^{(2)}$ and the number of time derivatives in the forcing will increase by two for each iteration. However, this is not only a technicality due to our way of constructing the solution. The error in the solution is studied numerically in the next section, and it can clearly be seen that the error in the gradient becomes large throughout the domain when the truncation error is not smooth in time; cf. Figure 2. This illustrates a major difference in the behavior of hyperbolic and elliptic problems. For elliptic problems, a high frequency boundary error will lead to a solution error that is unsmooth only in a boundary layer. In contrast, for hyperbolic problems, a boundary error which is highly oscillatory in both space and time will make the solution error highly oscillatory throughout the domain. The unsmooth solution error will be confined to a boundary layer only if the boundary error is smooth in time.

4. Numerical experiments. In this section we numerically solve (1)–(4) with the scheme described above. We denote the CFL-number by CFL $\equiv k/h$. Note that for a two-dimensional periodic domain our time-integration scheme (19) is stable for CFL $\leq 1/\sqrt{2} \approx 0.71$. Also note that all errors reported below are measured in max-norm over all grid points inside Ω.

We start the time integration at $n = 0$ and take $v^0_{i,j} = u_0(x_i, y_j)$. We need to use a fourth order accurate approximation of $u(x_i, y_j, -k)$ for $v^{-1}_{i,j}$. This is achieved by using the differential equation to approximate the second and third time derivatives,

(57)
$$v^{-1}_{i,j} = u_0(x_i,y_j) - ku_1(x_i,y_j) + \frac{k^2}{2}\left(D^x_+D^x_- + D^y_+D^y_-\right)u_0(x_i,y_j) + \frac{k^2}{2}F(\boldsymbol{x}_{i,j},0) - \frac{k^3}{6}\left(D^x_+D^x_- + D^y_+D^y_-\right)u_1(x_i,y_j) - \frac{k^3}{6}F_t(\boldsymbol{x}_{i,j},0).$$

Note that when the smooth start procedure is employed, the initial data are homogeneous, and thus the above formula simplifies to

$$\tilde{v}_{i,j}^{-1} = \frac{k^2}{2}\tilde{F}(\boldsymbol{x}_{i,j}, 0) - \frac{k^3}{6}\tilde{F}_t(\boldsymbol{x}_{i,j}, 0),$$

where the modified forcing initially satisfies

$$\tilde{F}(\boldsymbol{x}, 0) = F(\boldsymbol{x}, 0) + 2u_0(\boldsymbol{x}) + \Delta u_0(\boldsymbol{x}), \quad \tilde{F}_t(\boldsymbol{x}, 0) = F_t(\boldsymbol{x}, 0) + 6u_1(\boldsymbol{x}) + \Delta u_1(\boldsymbol{x}).$$

To use the smooth start procedure, we need to evaluate Δu_k, $k = 0, 1$, at all interior grid points. When $u_0(x, y)$ and $u_1(x, y)$ are complicated analytic expressions of x and y, an appealing alternative is to approximate Δu_k by finite differences. Numerical experiments indicate that a centered second or fourth order accurate difference approximation of Δu_k, $k = 0, 1$, can replace analytic expressions without any noticeable degradation of the computed solutions, as long as u_0 and u_1 are well resolved on the grid. However, to define all values in the five- or seven-point formula for the centered difference approximation of the Laplacian, this approach works only if u_0 and u_1 can be evaluated at all interior grid points as well as one or two grid points outside the boundary, respectively.

4.1. Trigonometric exact solution. To evaluate the accuracy of the method and the properties of the smooth start technique, we begin by considering the case when the exact solution is known. Given a smooth function $U(\boldsymbol{x}, t)$, this is accomplished by constructing balancing interior and boundary forcing functions $F(\boldsymbol{x}, t) = U_{tt}(\boldsymbol{x}, t) - \Delta U(\boldsymbol{x}, t)$ and $f(\boldsymbol{x}_\Gamma, t) = U(\boldsymbol{x}_\Gamma, t)$, respectively. The initial data are obtained by setting $u_0(\boldsymbol{x}) = U(\boldsymbol{x}, 0)$, $u_1(\boldsymbol{x}) = U_t(\boldsymbol{x}, 0)$. The continuous problem is then discretized and the error in the discrete solution can be obtained by taking the difference between the numerical and analytic solutions at each grid point.

We choose the exact solution to be the trigonometric traveling wave

$$U(x, y, t) = \sin(\omega(x - t)) \sin(\omega y), \quad \omega = 2\pi. \tag{58}$$

The domain Ω is taken to be an ellipse centered at the origin with semiaxes $x_s = 1$ and $y_s = 0.75$. The Cartesian grid covers the rectangle $-1.04 \le x \le 1.04$, $-0.78 \le y \le 0.78$, the grid size is $h = 2.08/(N - 1)$, and CFL $= 0.5$. To assess the accuracy of the scheme we run the computations until time $t = 2.0$. Note that no damping is necessary for these short runs ($\alpha = 0$). In Table 3, we present a grid refinement study for the scheme (19) started directly (from inhomogeneous initial data) and smoothly, respectively. In both cases, the solution itself converges as $\mathcal{O}(h^2)$, but the gradient and Laplacian are second order accurate only when smooth start is being used. (The second order formulas $D_0^x v_{i,j}$ and $D_0^y v_{i,j}$ are used to approximate the discrete gradient, and the component of the gradient with the largest error is reported. The Laplacian is approximated by $\Delta_h v_{i,j}$.) Note that when the computation is started directly, the error in the gradient is only a little better than first order accurate and the error in the Laplacian is $\mathcal{O}(1)$. This agrees with our observations for the one-dimensional case; cf. Table 1.

To more clearly see the impact of the smooth start procedure on the solution, in Figure 2 we plot the error in the solution at time $t = 2.0$, with direct and smooth start. Clearly, the smooth start procedure efficiently removes the high frequency errors present in the computation using direct start.

TABLE 3

Grid refinement study showing the errors in the computed solutions at $t = 2.0$ when the exact solution is the trigonometric function (58)*, with direct and smooth start.*

	Direct start			Smooth start		
N	$\|u_{\text{err}}\|_\infty$	$\|\nabla u_{\text{err}}\|_\infty$	$\|\Delta u_{\text{err}}\|_\infty$	$\|u_{\text{err}}\|_\infty$	$\|\nabla u_{\text{err}}\|_\infty$	$\|\Delta u_{\text{err}}\|_\infty$
201	$2.23 \cdot 10^{-3}$	$3.48 \cdot 10^{-2}$	14.3	$1.39 \cdot 10^{-3}$	$8.60 \cdot 10^{-3}$	$7.25 \cdot 10^{-2}$
401	$5.38 \cdot 10^{-4}$	$1.27 \cdot 10^{-2}$	9.86	$3.43 \cdot 10^{-4}$	$2.15 \cdot 10^{-3}$	$1.83 \cdot 10^{-2}$

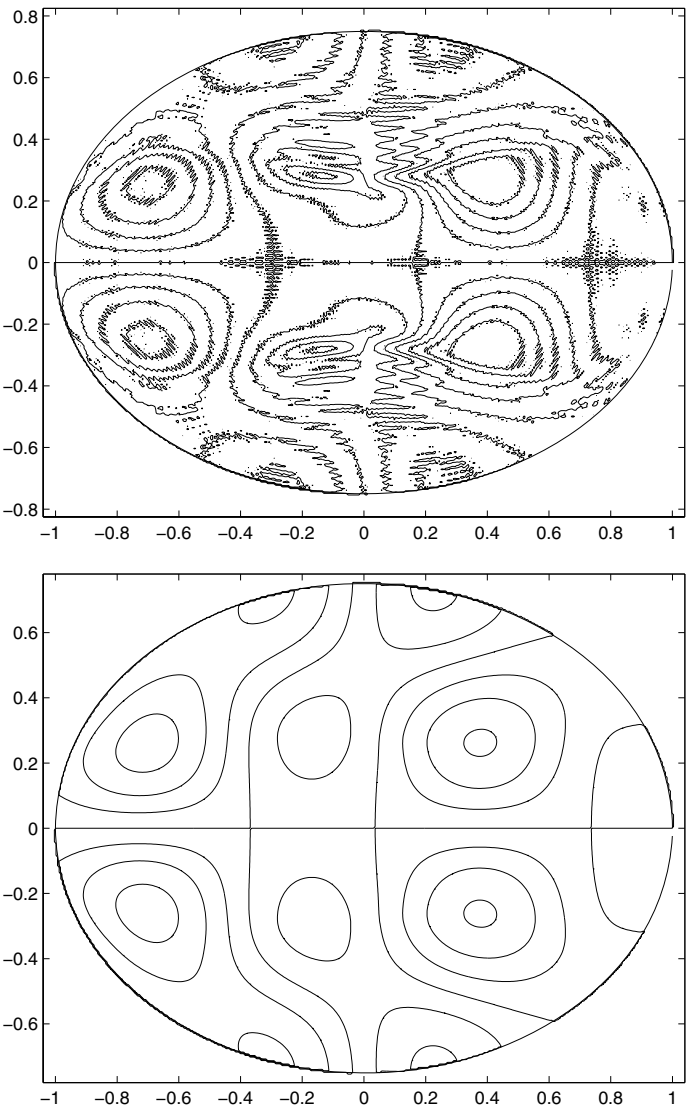


FIG. 2. *Error in the solution at time $t = 2.0$, with direct start (top) and smooth start (bottom). In both cases, the grid size was $h = 2.08/400$ ($N = 401$) and CFL $= 0.5$. Contours are equally spaced in $[-4 \cdot 10^{-4}, 4 \cdot 10^{-4}]$ with step size 10^{-4}.*

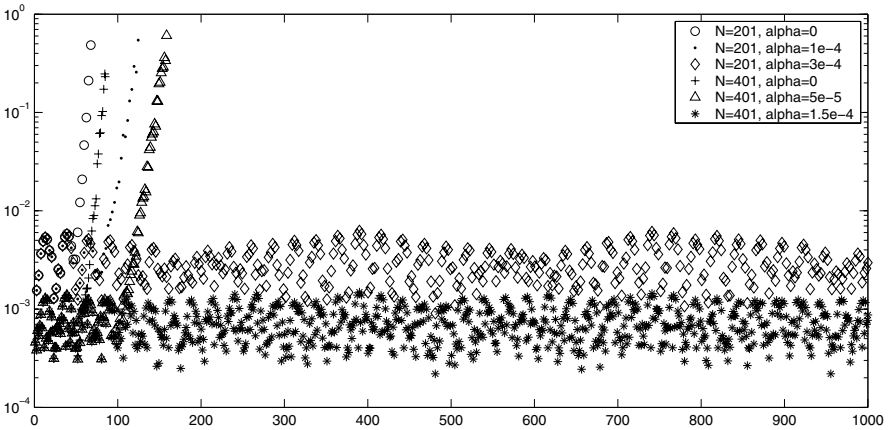


FIG. 3. *Max error in the solution as a function of time for different damping coefficients α and grid sizes N. For $N = 401$, time 1000 was reached after taking 384,615 time steps.*

Next we investigate the long-time properties of the time-integration scheme. In Figure 3 we report the error in the solution for different values of the damping coefficient α and for grid sizes $N = 201$ and $N = 401$. The domain is the same ellipse as before, CFL $= 0.5$, and smooth start is enabled. Damping is clearly needed to reach time 1000, corresponding to $384,615$ time steps on the finer grid, and we need to take $\alpha = 3 \cdot 10^{-4}$ to stabilize the scheme for the coarse grid. However, for the finer grid it suffices with $\alpha = 1.5 \cdot 10^{-4}$. This suggests that $\alpha = h\tilde{\alpha}$; i.e., the damping term can be of the order $\mathcal{O}(h^4)$. Note that there is no apparent increase in the error after long times, which indicates that the damping is very mild and that the scheme is appropriate for long time computations.

4.2. The TM$_z$ problem. We proceed by testing our method on the TM$_z$ problem for Maxwell's equations, i.e., the two-dimensional case where

$$\boldsymbol{H} = H^{(x)}(x, y, t)\boldsymbol{e}_x + H^{(y)}(x, y, t)\boldsymbol{e}_y$$

and $\boldsymbol{E} = E^{(z)}(x, y, t)\boldsymbol{e}_z$. By scaling the dependent variables and time, Maxwell's equations describing a homogeneous, lossless material without charges simplify to (cf. [15])

$$\frac{\partial H^{(x)}}{\partial t} = -\frac{\partial E^{(z)}}{\partial y}, \tag{59}$$

$$\frac{\partial H^{(y)}}{\partial t} = \frac{\partial E^{(z)}}{\partial x} \qquad \text{in } \Omega, \quad t \geq 0, \tag{60}$$

$$\frac{\partial E^{(z)}}{\partial t} = \frac{\partial H^{(y)}}{\partial x} - \frac{\partial H^{(x)}}{\partial y}, \tag{61}$$

subject to the constraint

$$\frac{\partial H^{(x)}}{\partial x} + \frac{\partial H^{(y)}}{\partial y} = 0 \quad \text{in } \Omega, \quad t \geq 0. \tag{62}$$

By cross-differentiation,

$$\frac{\partial^2 E^{(z)}}{\partial t^2} = \frac{\partial^2 E^{(z)}}{\partial x^2} + \frac{\partial^2 E^{(z)}}{\partial y^2} \quad \text{in } \Omega, \quad t \geq 0. \tag{63}$$

The perfectly electric conducting (PEC) boundary condition $\boldsymbol{n} \times \boldsymbol{E} = 0$, where $\boldsymbol{n} = n^{(x)}\boldsymbol{e}_x + n^{(y)}\boldsymbol{e}_y$ is the outward normal of the boundary, becomes

$$(\boldsymbol{e}_x n^{(y)} - \boldsymbol{e}_y n^{(x)}) E^{(z)} = 0 \quad \text{on } \Gamma,$$

i.e.,

$$E^{(z)} = 0 \quad \text{on } \Gamma. \tag{64}$$

Instead of directly solving the first order system (59)–(61), we solve the second order equation (63) subject to the Dirichlet boundary condition (64). Once $E^{(z)}$ is computed, the equations for $H^{(x)}$ and $H^{(y)}$, (59) and (60), reduce to ordinary differential equations at each point in Ω.

On the discrete side, (59) and (60) are integrated in time using the second order Adams–Bashforth method:

$$H_{i,j}^{(x)}(t_{n+1}) = H_{i,j}^{(x)}(t_n) - \frac{k}{2}\left(3D_0^y E_{i,j}^{(z)}(t_n) - D_0^y E_{i,j}^{(z)}(t_{n-1})\right), \tag{65}$$

$$H_{i,j}^{(y)}(t_{n+1}) = H_{i,j}^{(y)}(t_n) + \frac{k}{2}\left(3D_0^x E_{i,j}^{(z)}(t_n) - D_0^x E_{i,j}^{(z)}(t_{n-1})\right). \tag{66}$$

As a result, the following second order accurate centered approximation of the divergence constraint (62) satisfies

$$D_0^x H_{i,j}^{(x)}(t_n) + D_0^y H_{i,j}^{(y)}(t_n) = D_0^x H_{i,j}^{(x)}(0) + D_0^y H_{i,j}^{(y)}(0), \quad t_n > 0.$$

Hence, if the discrete divergence is initially zero, it will remain so for all subsequent times.

We use the scheme (19) together with the smooth start procedure to evolve (63). This is a good test of the numerical accuracy of $E^{(z)}$, since $H^{(x)}$ and $H^{(y)}$ depend on the gradient of $E^{(z)}$. Initial conditions for (59)–(61) are $H^{(x)}(\boldsymbol{x}, 0)$, $H^{(y)}(\boldsymbol{x}, 0)$, and $E^{(z)}(\boldsymbol{x}, 0)$. The second order formulation (63) also needs the time derivative of $E^{(z)}$ at $t = 0$. This follows from (61):

$$E_t^{(z)}(\boldsymbol{x}, 0) = H_x^{(y)}(\boldsymbol{x}, 0) - H_y^{(x)}(\boldsymbol{x}, 0).$$

Furthermore, to start the Adams–Bashforth scheme (65)–(66), we need an accurate approximation of $E^{(z)}(\boldsymbol{x}_{i,j}, -k)$:

$$E^{(z)}(\boldsymbol{x}_{i,j}, -k) = E^{(z)}(\boldsymbol{x}_{i,j}, 0) - kE_t^{(z)}(\boldsymbol{x}_{i,j}, 0) + \frac{k^2}{2}\Delta_h E^{(z)}(\boldsymbol{x}_{i,j}, 0).$$

When Ω is a unit circular disc, Maxwell's equations can be solved analytically using separation of variables using polar coordinates (ρ, θ):

$$x(\rho, \theta) = \rho \cos\theta,$$
$$y(\rho, \theta) = \rho \sin\theta.$$

TABLE 4
Error in the computed solution of the TM_z problem for the mode $m = 3$, $n = 1$ at time $t = 2.0$.

N	$\|E^{(z)}_{\text{error}}\|_\infty$	$\|H^{(x)}_{\text{error}}\|_\infty$	$\|H^{(y)}_{\text{error}}\|_\infty$	$\|E^{(z)}_{x\,\text{error}}\|_\infty$	$\|E^{(z)}_{y\,\text{error}}\|_\infty$	$\|\Delta E^{(z)}_{\text{error}}\|_\infty$
Direct start						
201	$1.03 \cdot 10^{-3}$	$1.47 \cdot 10^{-3}$	$1.98 \cdot 10^{-3}$	$1.32 \cdot 10^{-2}$	$1.63 \cdot 10^{-2}$	$8.52 \cdot 10^{0}$
401	$2.53 \cdot 10^{-4}$	$3.54 \cdot 10^{-4}$	$4.90 \cdot 10^{-4}$	$3.99 \cdot 10^{-3}$	$4.77 \cdot 10^{-3}$	$4.02 \cdot 10^{0}$
Smooth start						
201	$1.59 \cdot 10^{-3}$	$3.93 \cdot 10^{-3}$	$3.28 \cdot 10^{-3}$	$2.61 \cdot 10^{-3}$	$5.53 \cdot 10^{-3}$	$4.22 \cdot 10^{-2}$
401	$3.95 \cdot 10^{-4}$	$1.08 \cdot 10^{-3}$	$8.41 \cdot 10^{-4}$	$6.58 \cdot 10^{-4}$	$1.54 \cdot 10^{-3}$	$1.06 \cdot 10^{-2}$

We can alternatively express $\boldsymbol{H}$ in terms of its Cartesian or polar components, related by

$$H^{(x)} = H^{(\rho)} \cos\theta - H^{(\theta)} \sin\theta,$$
$$H^{(y)} = H^{(\rho)} \sin\theta + H^{(\theta)} \cos\theta.$$

Let $J_m(\xi)$ denote the Bessel function of the first kind of order m ($m = 0, 1, 2, \dots$) and let $\mathcal{X}_{mn}$ be the nth zero of J_m ($n = 1, 2, 3, \dots$), i.e., $J_m(\mathcal{X}_{mn}) = 0$, $0 < \mathcal{X}_{m1} < \mathcal{X}_{m2} < \cdots$. In terms of polar coordinates and components, the TM_z problem is solved by (cf. [15])

$$E^{(z)}_{mn}(\rho, \theta) = \mathcal{X}_{mn} J_m(\mathcal{X}_{mn}\rho) \cos(m\theta) \cos(\mathcal{X}_{mn} t), \tag{67}$$

$$H^{(\rho)}_{mn}(\rho, \theta) = \frac{m}{\rho} J_m(\mathcal{X}_{mn}\rho) \sin(m\theta) \sin(\mathcal{X}_{mn} t), \tag{68}$$

$$H^{(\theta)}_{mn}(\rho, \theta) = \mathcal{X}_{mn} J'_m(\mathcal{X}_{mn}\rho) \cos(m\theta) \sin(\mathcal{X}_{mn} t). \tag{69}$$

Since the equations are linear, any linear combination of the above solution is also a solution.

We evaluate the numerical scheme by solving the TM_z problem on a unit circular disc and compare the numerical solution to the exact analytic solution (67)–(69). We consider the case $m = 3$, $n = 1$, where $\mathcal{X}_{31} \approx 6.3801618959$. The problem is discretized on a computational grid with grid size $h = 2.08/(N - 1)$ and CFL $= 0.5$. In this computation, the coefficient of the artificial term in the Dirichlet boundary condition is $\gamma = 0.2$. No stabilization is necessary for these short time computations, and we set $\alpha = 0$. In Table 4 we compare the errors when the computation is started directly (from inhomogeneous initial data) and smoothly. In both cases, the errors in $E^{(z)}$, $H^{(x)}$, $H^{(y)}$, $E^{(z)}_x$, and $E^{(z)}_y$ are all $\mathcal{O}(h^2)$. However, the errors in $\Delta E^{(z)}$ reveal a highly oscillatory component in $E^{(z)}$ when the computation is started directly. Since $\partial \boldsymbol{H}/\partial t$ depends on the gradient of $E^{(z)}$, it is surprising that the errors in $\boldsymbol{H}$ are smaller when the computation is started directly, since the errors in the gradient of $E^{(z)}$ are smaller when the smooth start procedure is used.

In terms of our scaled variables, the total field energy within Ω is

$$E_{\text{field}}(t) = \int_\Omega \boldsymbol{E} \cdot \boldsymbol{E} + \boldsymbol{H} \cdot \boldsymbol{H} \, d\Omega. \tag{70}$$

Maxwell's equations (59)–(61) subject to a homogeneous PEC boundary condition (64) imply $E_{\text{field}}(t) = E_{\text{field}}(0)$, $t > 0$. Since our scheme uses damping to ensure

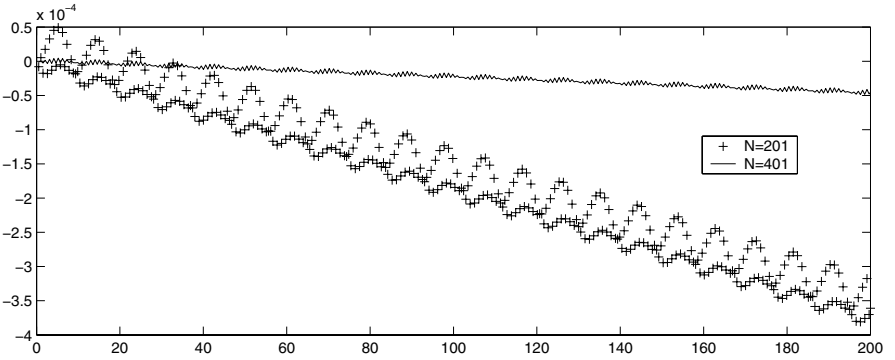


FIG. 4. *The error in the relative energy* $E_{\text{field}}(t)/E_{\text{field}}(0)$ *as a function of time for a circular domain for the mode* $m = 3$, $n = 1$ *of* (67)–(69).

stability, it is of interest to evaluate how well the energy (70) is conserved by the numerical approximation.

To evaluate (70), we need to integrate a grid function defined on the embedded boundary grid. Let $g(\boldsymbol{x})$ be a smooth function defined in Ω and take

$$g_{i,j} = \begin{cases} g(\boldsymbol{x}_{i,j}), & \boldsymbol{x}_{i,j} \in \Omega, \\ \text{extrap}(g)_{i,j}, & \boldsymbol{x}_{i,j} \notin \Omega,\ \boldsymbol{x}_{i\pm1,j} \in \Omega \text{ or } \boldsymbol{x}_{i,j\pm1} \in \Omega, \\ 0, & \text{otherwise}, \end{cases}$$

where $\text{extrap}(g)_{i,j}$ is defined by linear extrapolation from the interior points. We approximate the integral by a sum of grid cell averages of g:

$$\int_\Omega g(\boldsymbol{x})\, d\Omega \approx \sum_{i,j} C_{i+1/2,j+1/2} \frac{g_{i,j} + g_{i+1,j} + g_{i,j+1} + g_{i+1,j+1}}{4}. \tag{71}$$

For interior cells where all four grid vertices are inside Ω, the cell area is $C_{i+1/2,j+1/2} = h^2$. When all four grid vertices are outside Ω, the cell area is $C_{i+1/2,j+1/2} = 0$. For cells cut by the boundary, we first compute the intersections between the boundary and the four grid lines x_i, x_{i+1}, y_j, y_{j+1}. To estimate the cell area, the cell boundary is approximated by a linear segment between each intersection point. This procedure together with the linear extrapolation to define $\text{extrap}(g)$ in points just outside the boundary has been verified to give a second order accurate quadrature formula.

In Figure 4 we plot the error in the relative energy $E_{\text{field}}(t)/E_{\text{field}}(0)$ as a function of time for a circular domain. As before, we study the mode $m = 3$, $n = 1$ and use the same parameters as before. For these long time computations, the stabilization term is necessary and we set $\alpha = 10^{-3}$. As can be expected, the conservation error is $\mathcal{O}(h^2)$ and grows approximately linearly in time. For the fine mesh, the relative energy is conserved to within $5 \cdot 10^{-5}$ at time $t = 200$, and we conclude that the effect of the stabilization term is very small.

Next we study the evolution of a pulse centered at $(x, y) = (x_F, y_F)$ in a more complicated region where the boundary is represented by a cubic spline; see Figure 5. We start the computation from a localized perturbation in $E^{(z)}$ near (x_F, y_F):

$$E^{(z)}(x, y, 0) = \phi(\sqrt{(x - x_F)^2 + (y - y_F)^2}).$$

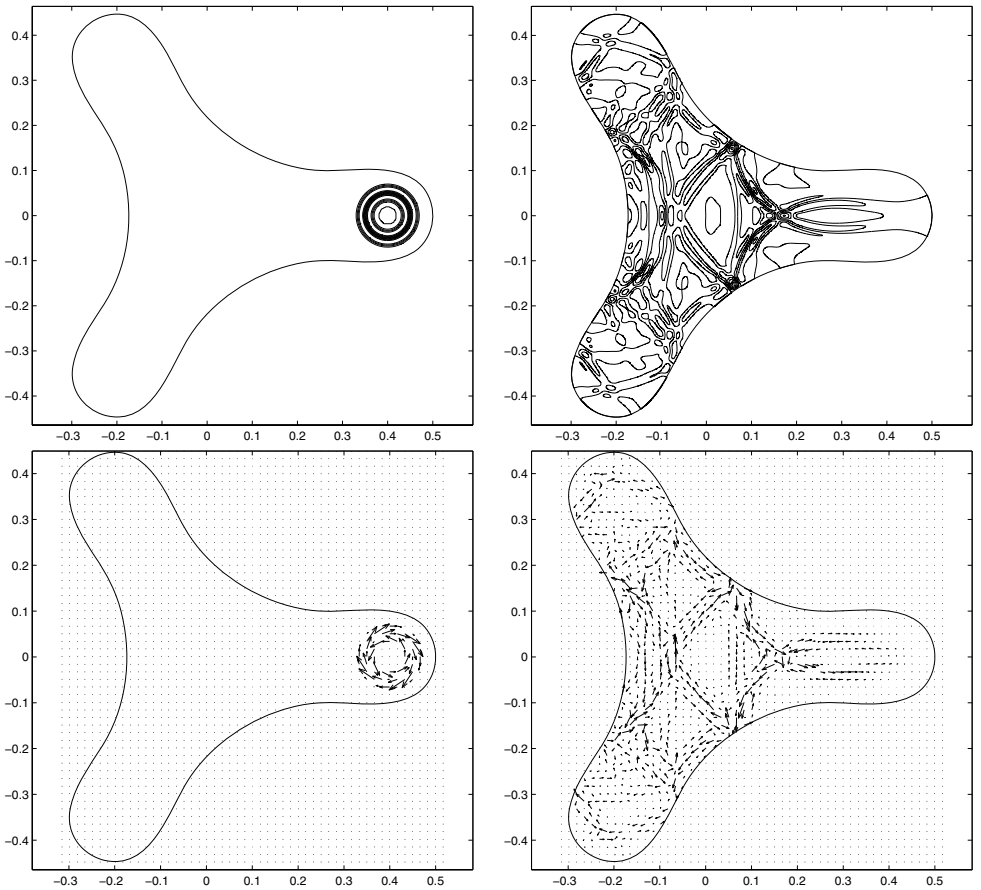


FIG. 5. *The evolution of* $E^{(z)}$ *(top) and* $\boldsymbol{H}$ *(bottom) in the problem for the grid refinement test. The left-hand figures show initial data and the right-hand figures show the solution at time* $t = 0.8$*. The solutions are shown on the finest grid with* 1601×1790 *grid points. Contours are spaced equally in* $[-1.0, 1.0]$ *with step size* 0.2*. Every* 32*nd grid point is used in the vector plots in the bottom row.*

To make the perturbation propagate (essentially) radially outward from (x_F, y_F), we take

$$\frac{\partial E^{(z)}}{\partial t}(x, y, 0) = -\phi'(\sqrt{(x - x_F)^2 + (y - y_F)^2}).$$

However, to start the computation we must specify compatible initial data for $\boldsymbol{H}$. Let $\psi(x, y)$ be a scalar field. The initial $\boldsymbol{H}$ field will be divergence free if

$$H^{(x)}(x, y, 0) = -\frac{\partial \psi}{\partial y}, \tag{72}$$

$$H^{(y)}(x, y, 0) = \frac{\partial \psi}{\partial x}. \tag{73}$$

Note that (61) yields

$$\frac{\partial E^{(z)}}{\partial t}(x, y, 0) = \Delta \psi.$$

Hence, we can find the initial data for $(E^{(x)}, E^{(y)})$ by solving the Poisson problem

$$\Delta\psi = -\phi' \tag{74}$$

and inserting the gradient of ψ in (72)–(73). In special cases, this can be done analytically. Introduce polar coordinates (ρ, θ) centered around the point (x_F, y_F),

$$\begin{aligned} x &= x_F + \rho\cos\theta, \\ y &= y_F + \rho\sin\theta, \end{aligned}$$

and let ϕ be independent of θ, i.e., $\phi = \phi(\rho)$. The solution of (74) is then

$$\psi_\rho(\rho) = -\phi(\rho) + \frac{1}{\rho}\int_0^\rho \phi(\rho')\,d\rho', \tag{75}$$

and since ψ is independent of θ, $\psi_x = (x - x_F)\psi_\rho/\rho$ and $\psi_y = (y - y_F)\psi_\rho/\rho$. If we take

$$\phi(\rho) = \begin{cases} 0, & \rho < r_0, \\ P\left(\frac{\rho - r_0}{r_1 - r_0}\right), & r_0 \le \rho \le r_1, \\ 0, & r > r_1, \end{cases} \qquad P(\xi) = a\xi^5(1-\xi)^5(\xi - 1/2),$$

we have $\int_0^1 P(\xi)\,d\xi = 0$, and thus $\psi_\rho(\rho)$ has compact support in $r_0 \le \rho \le r_1$.

In the computation shown in Figure 5, we used $x_F = 0.4$, $y_F = 0$, $r_0 = 0.02$, and $r_1 = 0.08$ to give the initial data compact support inside Ω, and set $a = 10938.8$, which normalizes the initial data to $-1 \le E^{(z)}(x, y, 0) \le 1$. Since the initial data have compact support, the smooth start procedure was not necessary in this computation. No exact solution is known for this problem and we resort to a grid refinement study to access the accuracy of the computed solution. Three calculations were performed where the grid size was $h = 5.193 \cdot 10^{-4}$, $2h$, and $4h$, respectively. The corresponding grids had 1601×1790, 801×896, and 401×448 grid points, respectively. In this computation, we used $\alpha = 10^{-3}$, $\gamma = 0.25$, and CFL $= 0.5$. Let $(E_h^{(z)}, H_h^{(x)}, H_h^{(y)})$ be the discrete solutions corresponding to grid size h. If the discrete solution is resolved on the grid, we expect it to satisfy

$$E_h^{(z)}(x_i, y_j, t_n) = E^{(z)}(x_i, y_j, t_n) + h^2 R(x_i, y_j, t_n) + \mathcal{O}(h^3)$$

and similar expressions for $H_h^{(x)}$ and $H_h^{(y)}$. Hence,

$$\frac{\|E_h^{(z)} - E_{4h}^{(z)}\|_\infty}{\|E_h^{(z)} - E_{2h}^{(z)}\|_\infty} \to 5, \quad h \to 0,$$

and corresponding expressions for $H_h^{(x)}$ and $H_h^{(y)}$. In Table 5, we report the calculated convergence rates at time $t = 0.8$. The calculated rates are near the asymptotic value of 5, which indicates that the solution is second order accurate. The relatively large difference between the solution on the fine and coarse grids indicates that the solution on that grid is not very accurate, despite being in the asymptotic range.

For completeness, we give the (x, y)-coordinates of the 21 node points in the periodic interpolating cubic spline: $(-0.0384, -0.2665)$; $(0.25, -0.1)$; $(0.4, -0.1)$; $(0.4707, -0.0707)$; $(0.5, 0.0)$; $(0.4707, 0.0707)$; $(0.4, 0.1)$; $(0.25, 0.1)$; $(-0.0384, 0.2665)$; $(-0.1134, 0.3964)$; $(-0.1741, 0.443)$; $(-0.25, 0.433)$; $(-0.2966, 0.3722)$; $(-0.2866, 0.2964)$; $(-0.2116, 0.1665)$; $(-0.2116, -0.1665)$; $(-0.2866, -0.2964)$; $(-0.2966, -0.3722)$; $(-0.25, -0.433)$; $(-0.1741, -0.443)$; and $(-0.1134, -0.3964)$.

TABLE 5
Convergence of the discrete solution.

	$X = E^{(z)}$	$X = H^{(x)}$	$X = H^{(y)}$
$\|X_h - X_{4h}\|_\infty$	0.3636	0.3552	0.2491
$\|X_h - X_{2h}\|_\infty$	0.0744	0.0621	0.0622
Ratio	4.89	5.24	4.00

5. Conclusions. We have developed a Cartesian embedded boundary method for the second order wave equation in general two-dimensional domains subject to Dirichlet boundary conditions, where both the solution and its gradient are second order accurate. By adding a small artificial term to the discrete boundary condition, we avoid the small-cell stiffness problem and can use an explicit time-integration method where the time step essentially equals that of a periodic domain.

Work is under way to solve Maxwell's equations written as a system of second order wave equations by combining the current method with our previous method for the Neumann problem [2]. Future developments include treatment of discontinuous wave propagation speeds, unbounded domains (far field boundaries), and generalizations to three space dimensions.

Appendix A. Estimating γ in the Dirichlet boundary condition.

Consider the semidiscrete one-dimensional wave equation (27)–(29). After eliminating the ghost point v_0, we can write the system in matrix form as follows:

$$\boldsymbol{v}_{tt} = A\boldsymbol{v} + \boldsymbol{F} + \boldsymbol{b}, \quad \boldsymbol{v} = (v_1, v_2, v_3, \dots)^T.$$

As before, $\boldsymbol{F}$ and $\boldsymbol{b}$ are the interior and boundary forcing terms, respectively. The tridiagonal matrix A satisfies

$$A = \frac{1}{h^2}\begin{pmatrix} -2 - \frac{c_1 - 2\gamma}{c_0 + \gamma} & 1 - \frac{c_2 + \gamma}{c_0 + \gamma} & 0 & 0 & \cdots \\ 1 & -2 & 1 & 0 & \cdots \\ 0 & 1 & -2 & 1 & \\ \vdots & & \ddots & \ddots & \ddots \end{pmatrix}.$$

Let the eigenvalues of A be λ_j. Since A can be symmetrized by a diagonal matrix, all eigenvalues of A are real valued. For all rows except the first, the Gershgorin circles (see, e.g., [16]) for the matrix $h^2 A$ are

$$\Lambda_j = \{z : |z + 2| \le 2\}, \quad j \ge 2,$$

that is, a circle in the complex z-plane with radius 2 centered at $z = -2$. The circle corresponding to the first row is

$$\Lambda_1 = \left\{ z : \left| z + 2 + \frac{c_1 - 2\gamma}{c_0 + \gamma} \right| \le 1 - \frac{c_2 + \gamma}{c_0 + \gamma} \right\}.$$

Since all eigenvalues of A are real, the circles correspond to intervals on the real axis, and the Gershgorin circle theorem says that all eigenvalues of A are located in the union of these intervals.

We begin by demonstrating the small-cell stiffness when $\gamma = 0$ and $\beta \to 0$. In this case, the Gershgorin circle from the first row in A is disjoint from the remaining circles when $0 \le \beta < (\sqrt{3}-1)/2$. For such β, there must be one eigenvalue of A that lies within the interval

$$-3 - \frac{c_1 - c_2}{c_0} \le h^2\lambda_1 \le -1 - \frac{c_1 + c_2}{c_0},$$

and we conclude that $h^2\lambda_1 \to -\infty$ as $\beta \to 0$. As a result, the CFL condition for the explicit time integration scheme (37) becomes $k/h \to 0$ as $\beta \to 0$; see [1].

Next, we want to estimate the eigenvalue with the largest magnitude for $\gamma > 0$. The interval corresponding to all circles except the first is $-4 \le \Re(z) \le 0$ and the interval from Λ_1 is

$$-3 - \frac{c_1 - c_2 - 3\gamma}{c_0 + \gamma} \le \Re(z) \le -1 + \frac{\gamma - c_1 - c_2}{c_0 + \gamma}.$$

Hence, we study how large

$$q(\beta, \gamma) = \frac{c_1(\beta) - c_2(\beta) - 3\gamma}{c_0(\beta) + \gamma}$$

can be for $0 \le \beta \le 1$, $\gamma > 0$. The coefficients in q are given by (26). When $\gamma \le 0.25$, the maximum of q is attained at $\beta = 0$ and satisfies

$$\max_{0 \le \beta \le 1} q(\beta, \gamma) = q(0, \gamma) = \frac{1 - 3\gamma}{\gamma}, \quad \gamma \le 0.25. \tag{76}$$

For $\gamma > 0.25$, $\max q(\beta, \gamma) < q(0, 0.25)$. Hence, to obtain the same lower bound from the first Gershgorin circle as from the rest, we require $\max q \le 1$ and (76) gives $\gamma \ge 0.25$. The upper bound from Λ_1 is always less than zero for $0 \le \beta \le 1$. Hence, all eigenvalues of A satisfy the estimate

$$-4 \le h^2\lambda_j \le 0 \quad \text{for } \gamma \ge 0.25,$$

and the CFL condition for the explicit time integration scheme (37) becomes $k/h < 1$, see [1].

REFERENCES

[1] H.-O. Kreiss, N. A. Petersson, and J. Ystrom, *Difference approximations for the second order wave equation*, SIAM J. Numer. Anal., 40 (2002), pp. 1940–1967.

[2] H.-O. Kreiss, N. A. Petersson, and J. Ystrom, *Difference approximations of the Neumann problem for the second order wave equation*, SIAM J. Numer. Anal., 42 (2004), pp. 1292–1323.

[3] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time Dependent Problems and Difference Methods*, Wiley–Interscience, New York, 1995.

[4] G. Browning, H-.O. Kreiss, and J. Oliger, *Mesh refinement*, Math. Comp., 27 (1973), pp. 29–39.

[5] R. Weller and G. H. Shortley, *Calculation of stresses within the boundary of photoelastic models*, J. Appl. Mech., 6 (1939), pp. A-71–A-78.

[6] L. Collatz, *The Numerical Treatment of Differential Equations*, 3rd ed., Springer-Verlag, Berlin, 1960.

[7] R. B. Pember, J. B. Bell, P. Colella, W. Y. Crutchfield, and M. L. Welcome, *An adaptive Cartesian grid method for unsteady compressible flow in irregular regions*, J. Comput. Phys., 120 (1995), pp. 278–304.

[8] M. J. Berger, C. Helzel, and R. J. LeVeque, *h-Box methods for the approximation of hyperbolic conservation laws on irregular grids*, SIAM J. Numer. Anal., 41 (2003), pp. 893–918.

[9] C. Zhang and R. LeVeque, *The immersed interface method for acoustic wave equations with discontinuous coefficients*, Wave Motion, 25 (1997), pp. 237–263.

[10] A. Ditkowski, K. Dridi, and J. S. Hesthaven, *Convergent Cartesian grid methods for Maxwell's equations in complex geometries*, J. Comput. Phys., 170 (2001), pp. 39–80.

[11] H. Johansen and P. Colella, *A Cartesian grid embedded boundary method for Poisson's equation on irregular domains*, J. Comput. Phys., 147 (1998), pp. 60–85.

[12] J.-R. Li and L. Greengard, *High order marching schemes for the wave equation in complex geometry*, J. Comput. Phys., 198 (2004), pp. 295–309.

[13] H.-O. Kreiss and J. Lorenz, *Initial-Boundary Value Problems and the Navier-Stokes Equations*, Academic Press, San Diego, 1989.

[14] D. Gilbarg and N. S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, Springer-Verlag, Berlin, 1983.

[15] C. A. Balanis, *Advanced Engineering Electromagnetics*, Wiley, New York, 1989.

[16] G. H. Golub and J. M. Ortega, *Scientific Computing and Differential Equations*, Academic Press, San Diego, 1992.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1168–1180
© 2006 Society for Industrial and Applied Mathematics

A COMPARATIVE STUDY OF LAGRANGIAN METHODS USING AXISYMMETRIC AND DEFORMING BLOBS*

LOUIS F. ROSSI†

Abstract. This paper presents results from a head-to-head comparison of two Lagrangian methods for solutions to the two-dimensional, incompressible convection-diffusion equations. The first Lagrangian method is an axisymmetric core spreading method using Gaussian basis functions. The second method uses deforming elliptical Gaussian basis functions. Previous results show that the first method has second-order spatial accuracy and the second method has fourth-order spatial accuracy. However, the deforming basis functions require more computational effort per element, so this paper examines computational performance as well as overall accuracy. The test problem is the deformation and diffusion of ellipsoidal distribution of scalar with an underlying flow field that has closed circular streamlines. The test suite includes moderate, high, and infinite Péclet number problems. The results indicate that the performance tradeoff for the sample flow calculation occur at modest problem sizes and that the fourth-order method offers distinct advantages as a general approach for challenging problems.

Key words. convection-diffusion, particle methods, computational fluid dynamics, deforming blobs

AMS subject classifications. 35Q30, 41A25, 65M12, 65M60, 76D05

DOI. 10.1137/030600679

1. Introduction.

Lagrangian schemes for solving convection-diffusion equations are naturally adaptive in the sense that information about the computed field closely follows physical path lines. These schemes are a natural reduction from vortex methods for Navier–Stokes and other nonlinear schemes and differ only in the sense that the velocity is specified rather than determined from the computed field. Since these methods express the computed field as a linear combination of basis functions, sometimes called blobs, that move with the local flow velocity, one can argue that computational resources are dedicated only where the computed field is substantial and nowhere else. To boost the efficiency of Lagrangian methods, many investigators have explored anisotropic and dynamically deforming computational elements as a way to use fewer elements in complex flow geometries. For instance, Marshall and Grant [12], Meiburg [14], and Teng [22] have specified anisotropic vortex elements to develop more efficient means of capturing boundary layers and mixing layers. Furthermore, it makes sense that improvements to the accuracy of moving, localized basis functions should come through local corrections in the shape of the basis function. Leonard [11], Moeleker and Leonard [15], Rossi [19], and Teng [23, 24] have developed methods that permit the computational elements to deform dynamically based on linearized deviations in the flow velocity. All these papers report improved accuracy, though not necessarily an order of spatial convergence beyond the anticipated second order and not necessarily improved computational performance. Since anisotropic and deforming basis functions require more computational resources than simple anisotropic elements, one would hope that the more sophisticated scheme will have substantial improvements

*Received by the editors August 12, 2003; accepted for publication (in revised form) December 29, 2004; published electronically January 6, 2006. This work was supported by National Science Foundation grant DMS-9971800.

http://www.siam.org/journals/sisc/27-4/60067.html

†Department of Mathematical Sciences, University of Delaware, Newark, DE 19716 (rossi@math.udel.edu).

in accuracy to justify the additional cost. To the contrary, Moeleker and Leonard [15] report that the increased computational effort over calculations with axisymmetric elements for the convection-diffusion equations does not justify the cost of using dynamically deforming elements. Recently, Rossi determined that this general category of method, when used with velocity and velocity derivative data evaluated at the basis function centroid, has second-order spatial correctness so that the deforming method and axisymmetric methods have the same order of convergence. However, this same analysis indicates that a curvature correction to the velocity field coupled with the adaptive deformations achieves fourth-order spatial accuracy [20]. Tests using deforming methods with and without the correction conclusively show that the theoretical convergence rates are indicative of the method's actual convergence rate and that the method using curvature corrections outperforms the same method without the corrections. What remains to be studied is whether it is better to use the more expedient axisymmetric elements or the more accurate but computationally intensive deforming basis functions for reasonably challenging problems. This paper makes a head-to-head comparison between Lagrangian methods with axisymmetric blobs and high-order deforming blobs to understand flow-dependent performance tradeoffs between the efficiency and accuracy of the two techniques when solving problems that are challenging on current desktop machines.

This paper focuses on solutions to the two-dimensional, incompressible, convection-diffusion equations,

$$\partial_t \rho + (\vec{u} \cdot \nabla)\rho = \frac{1}{\mathrm{Pe}} \nabla^2 \rho, \tag{1.1a}$$

$$\nabla \cdot \vec{u} = 0, \tag{1.1b}$$

on the unbounded plane where ρ is a passive scalar quantity that moves and diffuses with the flow. The $\vec{u}$ is a known fluid velocity field and Pe is the Péclet number, a dimensionless quantity expressing the ratio of convective to diffusive effects in the flow. Both the method using axisymmetric blobs and the method using deforming blobs are known as [illegible] because the width of each computational element grows with time to capture the diffusive term. Core-spreading methods are known to be inconsistent except when $\mathrm{Pe} = \infty$ because the growth of the core is governed by the Péclet number (or Reynolds number for vortex methods) and cannot be reduced below a nontrivial value. C. Greengard discusses the matter in some detail in [4]. Regarding practical issues when computing flows, Kida [6], Kida and Nakajima [7], and Kida, Nakajima, and Suemitsu [8] have since shown that there are important applications where these spatial errors are considerably smaller than other sources of numerical errors, so meaningful calculations can be performed with these inconsistent techniques. Essentially, the inconsistent formulation is useful for applications where the timescale of interest is much smaller than the growth in core size. If one strives for consistency, there are a number of ways to check the growth of core widths, including remeshing onto a regular array elements (see [2] for a survey) as well as adaptive refinement and merging [17, 18], but this is secondary to considerations of the accuracy and performance of the underlying method.

This comparative study has two goals. First, we will determine roughly where the high spatial order method outperforms the low-order method. It worth knowing whether or not the method has concrete advantages over lower-order but more efficient methods for reasonable applications on current computing platforms. Second, we use this study to highlight quantitative and qualitative differences in the convergent behavior of the two methods. While both methods are inconsistent methods,

using deforming elements extends the range of applicability before one must apply corrections to the method. In other words, both algorithms are naturally adaptive in the sense that computational effort is expended only where the field is nontrivial, but the scheme using deforming elements is more adaptive than the method using rigid elements in a concrete sense that will be explored in section 4.

These goals are similar but not identical to Legras and Dritschel's comparison of contour surgery and pseudospectral methods [10]. Both Legras and Dritschel's study and this work represent an effort to understand the strengths and weaknesses of a new method by comparing it to an established method. This study has a more intense focus on the computational performance and efficiency of the methods in addition to their convergence toward an exact solution. In Legras and Dritschel's study, they examine a complex flow with the two numerical schemes and compare the results to one another. In this paper, we test two similar Lagrangian methods on a flow problem using three different Péclet numbers by comparing approximate solutions to a highly resolved reference solution. Where the Lagrangian methods are general algorithms, the reference solution is calculated using a spectral scheme that is customized for this particular problem geometry. Legras and Dritschel's study compares quantities such as contour moments and area increments to measure the relative proximity of solutions. This paper will present the root-mean-squared distances (L_2) between the two methods that are being tested and the reference solution. Legras and Dritschel use two levels of refinement, coarse and fine. This paper examines solutions over different flow parameters at six distinct resolutions to establish convergence properties and performance trends.

This paper is organized as follows. This section (section 1) describes the background of the problem and provides a review of relevant works by other investigators. In section 2, we present the two methods under examination and discuss the time integration scheme used by both. In section 3, we define the fluid flow problems in the test suite and the practicalities of the error measurements used in this paper. The results of these tests are in section 4. Finally, section 5 contains summary remarks and future directions in this work.

2. The semidiscrete systems and temporal integration. This study compares two methods, one using spreading Gaussian blobs and the other using deforming elliptical Gaussian blobs. As noted before, both methods are inconsistent when $\mathrm{Pe} < \infty$ but can be corrected with adaptive refinement or remeshing. The spreading term arises when we capture the diffusive term in the same discrete dynamical system used to capture the convective terms in (1.1). Some popular alternatives for capturing the diffusive term include field redistribution and random walks, but these suffer from a lack of adaptivity in the former and low efficiency and accuracy in the latter. Both require operator splitting with the associated splitting errors. The advantages of core-spreading methods are that they are fully deterministic and naturally adaptive and the particle dynamics capture the underlying partial differential equation without splitting the operator.

In this paper, we define $\widehat{\rho}$ to be the approximation using a linear combination of moving basis functions:

$$\widehat{\rho} = \sum_{i=0}^{N} \gamma_i \phi(\vec{x} - \vec{x}_i, \dots). \tag{2.1}$$

The shape of the basis function is determined by ϕ and most often is chosen to be axisymmetric. For instance, if one is trying to solve the Navier–Stokes equations

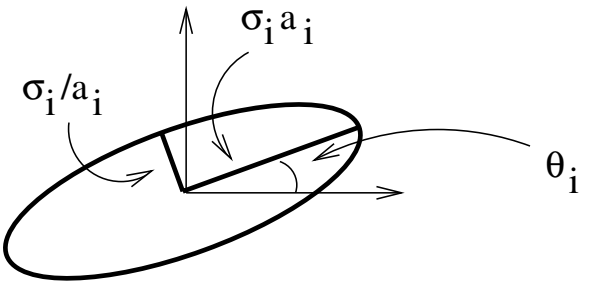


FIG. 2.1. *A schematic diagram of an elliptical Gaussian basis function as defined in* (2.3a). *The ellipse is representative of a single level set.*

where $\vec{u}$ and ρ are related via a Biot–Savart integral, then radial symmetry makes the velocity integration much simpler. The first of the two methods, which we shall refer to as the axisymmetric blob method, is the simplest available core-spreading method:

$$\phi(\vec{x},t;\vec{x}_i,\sigma_i) = \frac{1}{4\pi\sigma_i^2}\exp\left[-\frac{|\vec{x}-\vec{x}_i|^2}{4\sigma_i^2}\right], \tag{2.2a}$$

$$\frac{d\vec{x}_i}{dt} = \widehat{u}_i, \tag{2.2b}$$

$$\frac{d\sigma_i^2}{dt} = \frac{1}{\mathrm{Pe}}, \tag{2.2c}$$

where $\vec{x} = \left[\begin{smallmatrix} x \\ y \end{smallmatrix}\right]$. This method can use centroid velocity data ($\widehat{u}_i = \vec{u}(\vec{x}_i)$) with no penalty to its spatial accuracy. We shall refer to the second method in this study as the deforming blob method:

$$\phi(\vec{x};\vec{x}_i,\sigma_i,a_i,\theta_i) = \frac{1}{4\pi\sigma_i^2}\exp\left\{-\frac{|A_{\theta_i,a_i}(\vec{x}-\vec{x}_i)|^2}{4\sigma_i^2}\right\}, \tag{2.3a}$$

$$\frac{d}{dt}\vec{x}_i = \widehat{u}_i, \tag{2.3b}$$

$$\frac{d}{dt}(\sigma_i^2) = \frac{1}{2\mathrm{Pe}}(a_i^2 + a_i^{-2}), \tag{2.3c}$$

$$\frac{d}{dt}(a_i^2) = 2[d_{11}(c_i^2 - s_i^2) + (d_{12}+d_{21})s_ic_i]a_i^2 + \frac{1}{2\sigma^2\mathrm{Pe}}(1-a_i^4), \tag{2.3d}$$

$$\frac{d}{dt}\theta_i = \frac{d_{21}-d_{12}}{2} + \left[\frac{d_{21}+d_{12}}{2}(s_i^2-c_i^2) + 2d_{11}s_ic_i\right]\frac{(a_i^{-2}+a_i^2)}{(a_i^{-2}-a_i^2)}, \tag{2.3e}$$

where d_{ij} are the constituent elements of $D\vec{u}$, the matrix of partial derivatives of the velocity field $\vec{u}$, $c_i = \cos(\theta_i)$, $s_i = \sin(\theta_i)$, and

$$A_{\theta,a} = \begin{bmatrix} \cos\theta/a & \sin\theta/a \\ -a\sin\theta & a\cos\theta \end{bmatrix}. \tag{2.4}$$

So, in addition to the usual parameters, these basis functions have an orientation (θ_i) and an aspect ratio (a_i^2) as shown in Figure 2.1. The system (2.3) is larger and more complex than (2.2), and so there is a computational penalty for using the deforming blob method. Also, to have fourth-order spatial accuracy [20], the velocity field is

adjusted away from centroid velocity data as

$$\widehat{u}_i = \vec{u}(\vec{x}_i) + \sigma_i^2 \left(\vec{u}_{xx}(\vec{x}_i) M_{xx} + 2\vec{u}_{xy}(\vec{x}_i) M_{xy} + \vec{u}_{yy}(\vec{x}_i) M_{yy} \right), \tag{2.5}$$

where

$$M_{xx} = c_i^2 a_i^2 + s_i^2 / a_i^2, \tag{2.6a}$$

$$M_{xy} = c_i s_i (a_i^2 - a_i^{-2}), \tag{2.6b}$$

$$M_{yy} = c_i^2 / a_i^2 + s_i^2 a_i^2. \tag{2.6c}$$

There are no known advantages to curvature corrections in $D\vec{u}$ when using this method, though this was explored somewhat in [20].

When we approximate solutions to (1.1) using one of these schemes, we must choose a temporal integrator. For a problem where the velocity field is known and easy to compute, a Runge–Kutta method might be appropriate. However, there are many nonlinear problems where Lagrangian methods would be desirable, and in these cases, the velocity field might be expensive to compute. For instance, in vortex methods, the velocity field depends upon $\widehat{\rho}$ though a Biot–Savart integral. This is the most expensive part of the computation, even if one is using fast summation. To be general and open to the reality of extended applications, we perform the study with an explicit multistep integrator.

We use a third-order Adams–Bashforth multistep integrator for the trajectories in this comparative study. Also, we use a composite scheme because the deformation of the elements can occur on a timescale that is much faster than changes in the flow field. The deformation of each element occurs on half-steps because we would like to remain flexible and achieve second-order accuracy if the velocity were to depend upon ρ (even though it does not in this paper). Thus, the time-stepping algorithm is as follows:

1. Integrate either (2.2b) for the axisymmetric blob method or (2.3b) for the deforming blob method to move particles forward one-half a time step using an explicit stepper. Evaluate the velocity field.
2. Integrate (2.2c) for the axisymmetric blob method or (2.3c)–(2.3e) for the deforming blob method to deform basis functions for a full timestep. (2.2c) [illegible] Evaluate the velocity field.[1]
3. Integrate (2.2b) or (2.3b) to move particles forward one more half time step. Evaluate the velocity field.
4. To continue, return to step 1.

It can be shown that this simple splitting technique is second-order correct in time though it would be worthwhile to investigate whether or not there is a high-order multistep method for the entire system. The algorithm is implemented in a modular program, and the codes are identical except for the evolution equations governing the computational elements. That is, input/output, the looping structures, and all other routine tasks are absolutely identical in both codes.

[1] In the case of (2.3c)–(2.3e), an adaptive fourth-order Runge–Kutta algorithm is used with half-stepping for error control. We experimented with an adaptive Cash–Karp error control algorithm as outlined in [16] but found this to be less efficient than a simple half-stepping for this particular application.

3. The test problem. To examine the performance of the two methods, each will approximate the solution of (1.1) with an underlying differentially rotating flow given by the stream function

$$\psi = \frac{\pi}{4} r^2 + \delta(r^2 + c_1 r^4 + c_2 r^6), \tag{3.1a}$$

$$\vec{u} = \begin{bmatrix} \frac{\partial \psi}{\partial y} \\ -\frac{\partial \psi}{\partial x} \end{bmatrix}, \tag{3.1b}$$

where $\delta = 3/10$, $c_1 = -7/6$, and $c_2 = 1/10$ on an unbounded domain. This is the same flow field used in [20], but these tests will use different initial data. Where we applied axisymmetric initial data in previous work so that an exact solution could be used for convergence studies, in this study we will use a normalized, anisotropic distribution of scalar,

$$\rho(\vec{x}, 0) = \frac{32}{\pi} e^{-32\left(x^2/8 + 8y^2\right)}, \tag{3.2}$$

so that, in the absence of dissipation, fine filaments will be generated after a small number of rotations as seen in Figure 3.1. We select this sample problem because one uses Lagrangian methods when the quantity of interest, ρ, occupies a fraction of the computational domain at any given time during the simulation. While axisymmetric structures are useful for a convergence study, it is unlikely anyone would choose a Lagrangian method if the computed field was expected to fill the domain of interest. Instead, a performance measurement should involve a distribution that is more typical for the method being tested.

With this flow structure, the turnover time near the origin is about 3 but the turnover time near the edge of the structure is roughly $\frac{9}{2}$. The computation is performed until $T = 9$ so that the inside of the initial distribution is twisted a full revolution relative to the edges as seen in Figure 3.1. To test the two Lagrangian methods, the calculation is performed with $\mathrm{Pe} = 10^4$, $\mathrm{Pe} = 10^5$, and $\mathrm{Pe} = \infty$. The initial data for both methods is identical. Since the initial data has infinite support, we initially arrange computational elements over the domain

$$8(x^2/8 + 8y^2) \leq 1, \tag{3.3}$$

beyond which ρ has decayed to less than 2% of its initial maximum value. The strength of each element is determined using an exact deregularization procedure, as discussed in [21], so that the growth of spatial errors in solving (1.1) are much greater than the growth of initial errors. Isotropic particles are arranged on the domain (3.3) in a regular square lattice of width $\frac{\sigma_0}{2}$, where σ_0 is the initial core width for the trial simulation. Both methods were tested in trials with initial core widths of 8×10^{-4}, 4×10^{-4}, 2×10^{-4}, 10^{-4}, 5×10^{-5}, and 2.5×10^{-5}.

A high resolution spectral calculation will serve as a reference ("exact") solution for the convergence study. The spectral code is implemented in Matlab using scripts from Trefethen's monograph [25]. The spectral code takes advantage of the fact that we know the domain of interest a priori, so we can compute spatial derivatives spectrally on the unit disk (see Chapter 11 of the monograph) using Chebyshev interpolation in the radial direction and trigonometric interpolation in the azimuthal direction. The spectral code used in this paper modifies Trefethen's MATLAB program to take advantage of sparsity in the two-dimensional differentiation matrix.

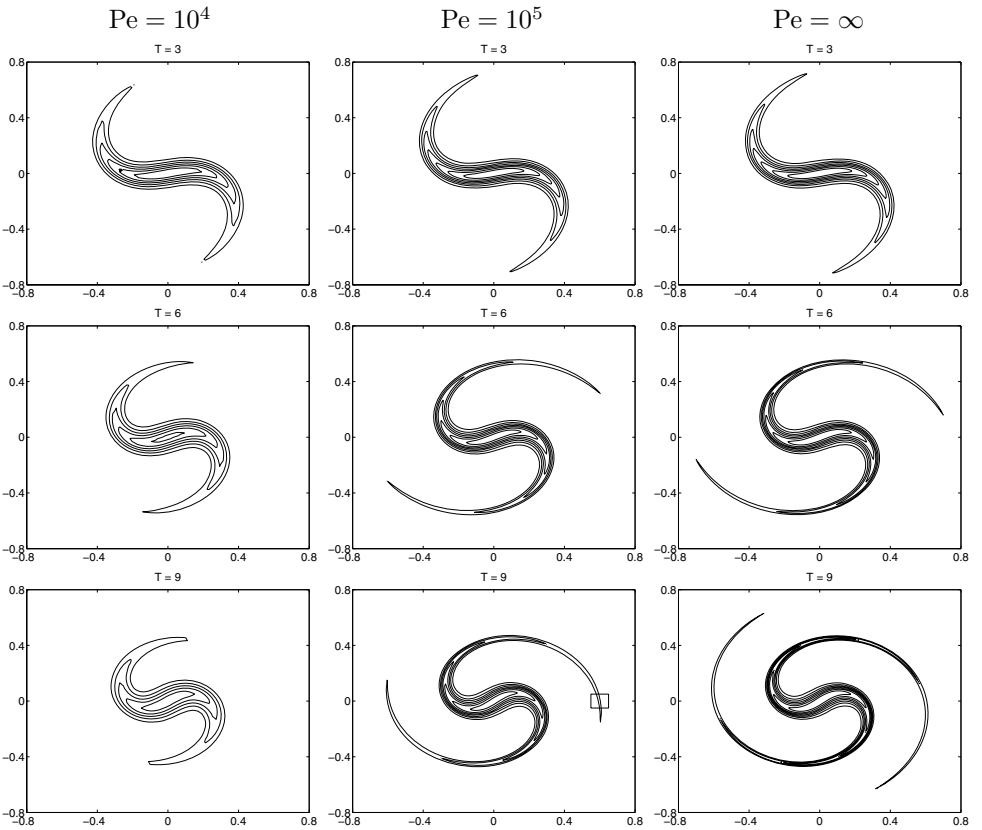


FIG. 3.1. *Solutions for* $\mathrm{Pe} = 10^4$ *(left),* $\mathrm{Pe} = 10^5$ *(center), and* $\mathrm{Pe} = \infty$ *(right). The contour increment is* 4 *units of* ρ *per division. The differential rotation of the underlying flow intensifies gradients while diffusion suppresses gradients. The long, slender filaments at high Péclet number pose a challenge to any numerical method. The area of detail indicated at the bottom center is expanded in Figure* 4.2.

Time-stepping is performed using third-order Adams–Bashforth with timesteps of either 10^{-3} or 6×10^{-5} for $\mathrm{Pe} = 10^4$ and 10^5, respectively, depending upon the stability requirements of the spectral differentiation matrix. Refinement studies indicate that all computations are temporally resolved. While the purpose of this paper is to examine the trade-offs between the axisymmetric blob method and the deforming blob method, it is worthwhile to discuss three notable distinctions between the spectral reference solution and the Lagrangian solutions.

- The most important difference is that we should achieve spectral accuracy in space from the spectral code since the exact solution is smooth for all finite time. That is, we expect the spatial accuracy to scale like C^{-N}, where N is the number of modes used in the computation. The Lagrangian methods are second order $O(\sigma^2)$ and fourth order $O(\sigma^4)$ in space, respectively. However, for very high Péclet numbers, the scalar filaments can become quite sharp, and the resulting large gradients make the spectral convergence rate very slow.
- The spectral code is designed to take advantage of the simple geometry of the test problem, whereas the Lagrangian code is general and is applicable

to any flow geometry. For instance, if we were to superpose a straining flow such as $\psi_2 = Cxy$ into the flow field, we could directly apply the same Lagrangian code, whereas the spectral code would need to be modified for a new computational domain.

- The spectral method operates on a finite domain where $\widehat{\rho} = 0$ at $r = 1$ in polar coordinates, whereas the Lagrangian code operates on an unbounded domain and the choice of basis function automatically satisfies $\widehat{\rho} \to 0$ as $r \to \infty$. Since (3.2) decays rapidly and the Péclet number is high, solutions from either method can be compared to one another.

Solutions to all calculations are compared on a uniform grid that is sufficiently fine to resolve differences down to four decimal places. For $\mathrm{Pe} = 10^4$, a 100×100 mesh is sufficient, but for $\mathrm{Pe} = 10^5$ and $\mathrm{Pe} = \infty$, a 400×400 mesh is used. For the Lagrangian computations, $\widehat{\rho}$ is computed directly at each mesh point. For the spectral reference solution, we acquire regularly spaced data using the corresponding spectral projection. To measure the difference between two solutions, u and v, we use an l_2 approximation to the L_2 norm:

$$\|u - v\|_2 = \sqrt{\sum_{i,j} (u_{i,j} - v_{i,j})^2 \Delta x \Delta y}, \tag{3.4}$$

where the sum is over all mesh points and Δx and Δy are, respectively, the width and height of the regular adjoint grid cells. Thus, the error in a test calculation is the l_2 difference between the test solution and the reference solution.

4. Results. It is no surprise that both Lagrangian methods achieve the predicted spatial convergence rate as seen in Figure 4.1 and reported in [20] because only the flow parameters and initial data have been changed. Examining (2.2) and (2.3), we see that the core sizes for the deforming elements will always be at least as large as those of the axisymmetric system. In Figure 4.1, we see that for the same initial widths, the deforming elements grow substantially larger. Yet, despite having wider elements, the deforming blob scheme realizes better results because elements conform to the local flow geometry as seen in Figure 4.2. If one extrapolates to a hypothetical situation where the initial widths are zero, one can see that the final state will correspond to the deforming blob method having wider elements but a more accurate solution. In other words, while both techniques are inconsistent if left uncorrected by remeshing or adaptive refinement techniques, the deforming blob method suffers less for it.

To measure the performance of both methods, the overall precision of the computation can be seen as a function of computational effort expended to achieve the resulting accuracy. The problem size, computational time, and overall accuracy of each computation are shown in Table 4.1. Typically, the deforming elements require between five and six times the computational effort of an axisymmetric blob because there are more degrees of freedom and the evolution of σ^2 is coupled to a^2 so neither can be solved analytically as is the case for the axisymmetric elements. In Figure 4.1, we see that the tradeoff between complexity and accuracy occurs even at small problem sizes of about 10,000 (axisymmetric) elements at $\mathrm{Pe} = 10^4$, and the improvements are realized at increasingly large problem sizes as we increase the Péclet number. This does not correspond to a loss of performance at large Pe because accurate solutions to larger Pe flow will require a larger number of computational elements to resolve it. Since it is not unusual for high-performance Lagrangian computations to use millions of computational elements, these results indicate that the use of deforming elements

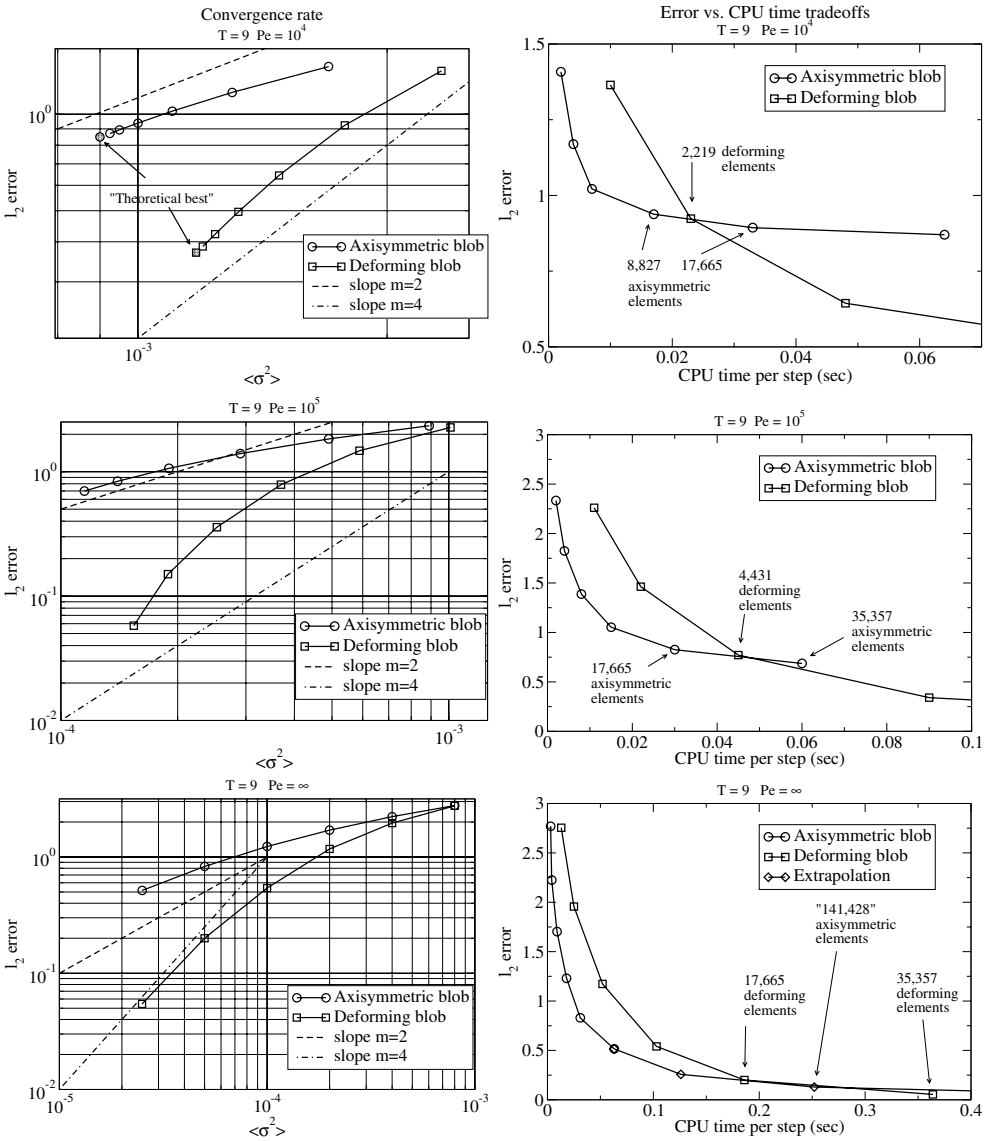


FIG. 4.1. *Convergence rates using logarithmic scales (left) and performance data (right) for test problems using the axisymmetric blob and deforming blob methods. Theoretical convergence rates are achieved for both Lagrangian methods, and the deforming blob method works better. In the absence of remeshing or adaptive refinement, both methods are inconsistent, meaning that* $\langle \sigma^2 \rangle$ *has a lower bound. The "theoretical best" value is extrapolated back to a position where initial core widths would be zero for the* $\mathrm{Pe} = 10^4$ *calculation. By examining the accuracy gained as a function of ticks of the CPU clock, we can measure the tradeoff between the simpler low-order method and the more complex high-order method. One can see that the tradeoff depends upon the Péclet number.*

would benefit many investigations, should lead to significant improvements in accuracy, and will make aggressive calculations more accessible with modest computational resources.

As a final note, it is also meaningful to report upon the spectral method used to generate reference solutions. Though the references themselves are not the focal point

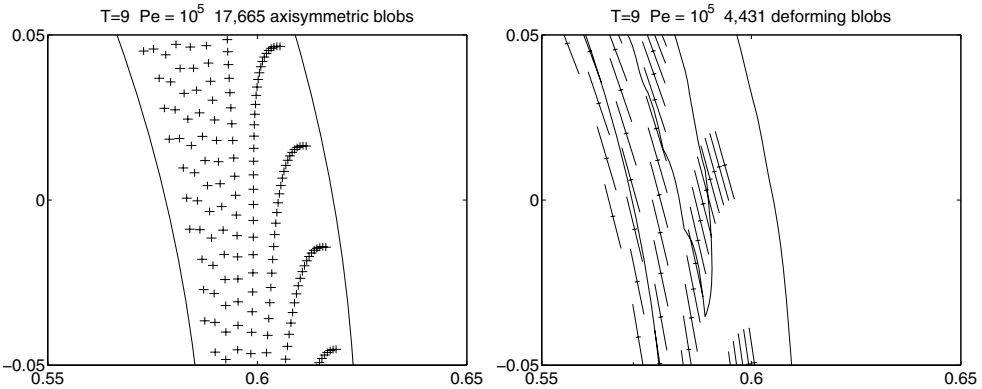


FIG. 4.2. *Sample particle positions using the axisymmetric (left) and deforming blob methods (right). Particle positions are indicated by oriented X's whose arms correspond to* $\sigma a/10$ *and* $\sigma/a/10$. *Scalar concentrations are indicated by the solid contour line. The area of detail is indicated in Figure* 3.1 *in the bottom center figure. The axisymmetric calculation uses more blobs to achieve slightly less accurate results than the deforming computation on the right. The deforming elements are more than twice as wide, but they naturally conform to the geometry of the computed field.*

TABLE 4.1

Table of computation times and accuracy of the methods used in this paper. N is the total number of computational elements. CPU is the total computational time in minutes required to run the full simulation from start to finish, including overhead costs associated with initializations, startup procedures for multistep methods, etc. Accuracy is the l_2 *error associated with the computation at various Péclet numbers.*

	Axisymmetric blob method				Deforming blob method			
N	CPU	l_2 error			CPU	l_2 error		
	(min)	$\mathrm{Pe} = 10^{-4}$	10^{-5}	∞	(min)	$\mathrm{Pe} = 10^{-4}$	10^{-5}	∞
1,107	0.30	1.41	2.34	2.77	1.92	1.36	2.270	2.754
2,219	0.71	1.17	1.83	2.22	3.88	0.92	1.475	1.956
4,431	1.38	1.02	1.40	1.70	7.75	0.64	0.787	1.173
8,827	2.66	0.94	1.06	1.23	15.15	0.50	0.358	0.541
17,665	5.14	0.89	0.84	0.83	30.03	0.42	0.150	0.200
35,357	9.85	0.87	0.70	0.52	57.12	0.39	0.058	0.054

of the paper, the efforts required to generate them merit some attention. At $\mathrm{Pe} = 10^4$, the spectral computation ran with about 100 modes azimuthally and radially in less than two hours to produce a reference solution that was well below the theoretical best solution that could be obtained with either Lagrangian method. Like the Lagrangian methods that were being tested, the spectral algorithm struggled with computations at $\mathrm{Pe} = 10^5$ and $\mathrm{Pe} = \infty$. The spectral reference solution at $\mathrm{Pe} = 10^5$ required more than 250 modes azimuthally and radially and needed roughly 28 days of CPU time. Part of the problem is that the spatial resolution required a small time step of 6×10^{-5} for the time-stepping to be stable. Before being critical about this performance, one should consider that the Lagrangian method could never reach this accuracy at all without remeshing or refinement. Also, a crafty spectral guru would know of some tricks to save memory and clock cycles or do a better job of avoiding stability boundaries. For instance, one might remap the domain to allocate more resolution away from the outer boundary and possibly dedicate more nodes closer to the filaments (see [9] for an example). When $\mathrm{Pe} = \infty$, the Lagrangian methods are consistent, and the standard spectral scheme is prohibitively expensive in both memory and time. The deforming

Lagrangian method was used to generate its own reference solution. Even with close to 150,000 particles, it ran in three and a half hours and so was the natural choice for generating a reference solution.

5. Conclusions. A number of common themes emerge from this quantitative comparison of low- and high-order Lagrangian methods. The general conclusion is that there are significant performance advantages in using the deforming blob method over the axisymmetric blob method for problems of general interest that are accessible with currently available computing devices. The more specific findings are as follows.

- The performance tradeoff depends upon the Péclet number of the problem, so the tradeoffs occur at larger problem sizes when diffusion is less dominant. The other side of this result is that one might be better off using methods with axisymmetric blobs if the resolution requirements are low.
- While both the axisymmetric and deforming blob methods are inconsistent unless one applies a correction method, the deforming blob method has a better limiting accuracy. Therefore, one can argue that the range of applicability of the deforming blob method is greater than the axisymmetric blob method, and the deforming blob method requires fewer corrections if one desires consistency.
- Both Lagrangian methods compare very favorably to the reference spectral scheme at high Péclet numbers, and this really highlights the advantage of natural adaptivity in Lagrangian schemes. As a general tool for fluid flows, the Lagrangian algorithm dedicates resources more effectively where the field is substantial. This is not intended to criticize the use of spectral methods for high Péclet number flows but rather to point out that it might require some a priori knowledge of the solution to tune the spectral scheme. The results in this paper suggest that these Lagrangian methods are high quality general tools for flow problems where the field of interest occupies a fraction of the domain of interest.

This study was designed around a problem where the exact solution can be approximated to a high degree of precision by a standard spectral method, but the problem itself is only one of many possibilities one might try. This particular problem focuses on a flow with a large sustained deformation rate. In the final analysis, we see a significant performance tradeoff where fourth-order errors are important. Since the velocity curvature governs the fourth-order corrections (2.5), (2.6), one infers that the performance advantages will be most pronounced in flows where velocity curvatures are large.

These results point toward closely related projects. To go beyond the convection-diffusion equations, one might solve the nonlinear vorticity equations where ρ would represent the vorticity of a fluid flow, but one must extract the velocity field from $\widehat{\rho}$. Some progress in this area has already been made by determining a fourth-order asymptotic expression for the Biot–Savart integral of an elliptical Gaussian [19]. Naturally, approximating the velocity field of an elliptical Gaussian is more computationally expensive than the exact evaluation for an axisymmetric Gaussian. However, in practical calculations, one uses fast summation algorithms which reduce the computational complexity by separating near and far field velocity contributions. For examples of different adaptive summation methods, see [1, 3, 5, 13]. Anisotropic elements in the far field introduce no added complexity or computational effort, so contributions from the far field would not differ significantly for axisymmetric and nonaxisymmetric blobs. However, direct interactions in the near field would be more

costly for anisotropic elements than for isotropic elements.

Along different lines, one might examine the three-dimensional convection-diffusion equations using deforming ellipsoidal Gaussian blobs. Many of the ideas and techniques in this paper and in [20] would apply. Finally, most of the work on remeshing and adaptive refinement has focused on axisymmetric elements. Generalizations to these techniques for highly anisotropic blobs should be explored to make methods like these fully consistent in flow calculations with finite diffusivity.

Acknowledgments. The author would like to thank Prof. Toby Driscoll at the University of Delaware for his helpful advice on the practicalities of implementing the spectral code used in this paper, and the referees for their thoughtful comments and suggestions.

REFERENCES

[1] C. Anderson, *A method of local corrections for computing the velocity field due to a distribution of vortex blobs*, J. Comput. Phys., 62 (1986), pp. 111–123.

[2] G.-H. Cottet and P. D. Koumoutsakos, *Vortex Methods: Theory and Practice*, Cambridge University Press, Cambridge, New York, 2000.

[3] C. I. Draghicescu, *A fast algorithm for vortex blob interactions*, J. Comput. Phys., 116 (1995), pp. 69–78.

[4] C. Greengard, *The core spreading vortex method approximates the wrong equation*, J. Comput. Phys., 61 (1985), pp. 345–348.

[5] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, J. Comput. Phys., 73 (1987), pp. 325–348.

[6] T. Kida, *Theoretical and numerical results of a deterministic two-dimensional vortex method*, Sādhanā, 23 (1998), pp. 419–441.

[7] T. Kida and T. Nakajima, *Core spreading vortex methods in two-dimensional viscous flows*, Comput. Methods Appl. Mech. Engrg., 160 (1998), pp. 273–298.

[8] T. Kida, T. Nakajima, and H. Suemitsu, *Second order core spreading vortex method in two-dimensional viscous flows*, JSME Int. J. Series B, 41 (1998), pp. 441–446.

[9] D. Kosloff and H. Tal-Ezer, *A modified Chebyshev pseudospectral method with an $O(N^{-1})$ time step restriction*, J. Comput. Phys., 104 (1993), pp. 457–469.

[10] B. Legras and D. G. Dritschel, *A comparison of the contour surgery and pseudo-spectral methods*, J. Comput. Phys., 104 (1993), pp. 287–302.

[11] A. Leonard, *Large-eddy simulation of chaotic convection and beyond*, in 35th Aerospace Sciences Meeting and Exhibit, American Institute of Aeronautics and Astronautics, 1997.

[12] J. S. Marshall and J. R. Grant, *A method for determining the velocity induced by highly anisotropic vorticity blobs*, J. Comput. Phys., 126 (1996), pp. 286–298.

[13] J. S. Marshall, J. R. Grant, A. A. Gossler, and S. A. Huyer, *Vorticity transport on a Lagrangian tetrahedral mesh*, J. Comput. Phys., 161 (2000), pp. 85–113.

[14] E. Meiburg, *Incorporation and test of diffusion and strain effects in the two-dimensional vortex blob technique*, J. Comput. Phys., 82 (1989), pp. 85–93.

[15] P. Moeleker and A. Leonard, *Lagrangian methods for the tensor-diffusivity subgrid model*, J. Comput. Phys., 167 (2001), pp. 1–21.

[16] W. H. Press, S. A. Teukolsky, W. T. Vettering, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, Cambridge, UK, 1992.

[17] L. F. Rossi, *Resurrecting core spreading vortex methods: A new scheme that is both deterministic and convergent*, SIAM J. Sci. Comput., 17 (1996), pp. 370–397.

[18] L. F. Rossi, *Merging computational elements in vortex simulations*, SIAM J. Sci. Comput., 18 (1997), pp. 1014–1027.

[19] L. F. Rossi, *High Order Vortex Methods with Deforming Elliptical Gaussian Blobs* 1: *Derivation and Validation*, Technical Report 2001-11, University of Delaware, Newark, DE, 2002.

[20] L. F. Rossi, *Achieving high-order convergence rates with deforming basis functions*, SIAM J. Sci. Comput., 26 (2005), pp. 885–906.

[21] L. F. Rossi, J. F. Lingevitch, and A. J. Bernoff, *Quasi-steady monopole and tripole attractors for relaxing vortices*, Phys. Fluids, 9 (1997), pp. 2329–2339.

[22] Z.-H. Teng, *Elliptic-vortex method for incompressible flow at high Reynolds number*, J. Comput. Phys., 46 (1982), pp. 54–68.

[23] Z.-H. Teng, *Variable-elliptical-vortex method for incompressible flow simulation*, J. Comput. Math., 4 (1986), pp. 255–262.
[24] Z.-H. Teng, L.-A. Ying, and P. Zhang, *Convergence of the variable-elliptic-vortex method for Euler equations*, SIAM J. Numer. Anal., 32 (1995), pp. 754–774.
[25] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia, 2000.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1181–1201
© 2006 Society for Industrial and Applied Mathematics

AUTHORITY RANKINGS FROM HITS, PAGERANK, AND SALSA: EXISTENCE, UNIQUENESS, AND EFFECT OF INITIALIZATION*

AYMAN FARAHAT†, THOMAS LOFARO‡, JOEL C. MILLER§, GREGORY RAE¶, AND LESLEY A. WARD¶

Abstract. Algorithms such as Kleinberg's HITS algorithm, the PageRank algorithm of Brin and Page, and the SALSA algorithm of Lempel and Moran use the link structure of a network of web pages to assign weights to each page in the network. The weights can then be used to rank the pages as authoritative sources. These algorithms share a common underpinning; they find a dominant eigenvector of a nonnegative matrix that describes the link structure of the given network and use the entries of this eigenvector as the page weights. We use this commonality to give a unified treatment, proving the existence of the required eigenvector for the PageRank, HITS, and SALSA algorithms, the uniqueness of the PageRank eigenvector, and the convergence of the algorithms to these eigenvectors. However, we show that the HITS and SALSA eigenvectors need not be unique. We examine how the initialization of the algorithms affects the final weightings produced. We give examples of networks that lead the HITS and SALSA algorithms to return nonunique or nonintuitive rankings. We characterize all such networks in terms of the connectivity of the related HITS authority graph. We propose a modification, Exponentiated Input to HITS, to the adjacency matrix input to the HITS algorithm. We prove that Exponentiated Input to HITS returns a unique ranking, provided that the network is weakly connected. Our examples also show that SALSA can give inconsistent hub and authority weights, due to nonuniqueness. We also mention a small modification to the SALSA initialization which makes the hub and authority weights consistent.

Key words. link analysis, web search, HITS algorithm, Kleinberg's algorithm, PageRank algorithm, SALSA algorithm, hubs, authorities, networks

AMS subject classifications. 68W40, 15A18, 68R10

DOI. 10.1137/S1064827502412875

1. Introduction.

The rapid growth of the World Wide Web has created a need for search tools. In the past, search engines ranked pages by using word frequency or similar measures. Recently, new algorithms have been created that greatly improve rankings, using the network structure of the web. Two prominent ranking algorithms are PageRank [3] (the underlying method of the search engine) and HITS [9, 15] (Hypertext Induced Topic Search). A third algorithm, SALSA [17] (Stochastic Approach for Link Structure Analysis), combines ideas from PageRank and HITS. Some search tools, such as HITS, first use conventional information-retrieval techniques to identify a subnetwork of pages relevant to a particular query, and then order the results so that the most relevant web pages are presented near the top of the list. Others, such as PageRank, rank the whole web on an absolute scale and then select

*Received by the editors August 8, 2002; accepted for publication (in revised form) January 7, 2005; published electronically January 6, 2006. A brief preliminary version of this work appeared in the Proceedings of the ACM SIGIR 2001 Conference. This work was supported in part by the Harvey Mudd College Mathematics Clinic and by HNC Software, Inc.

http://www.siam.org/journals/sisc/27-4/41287.html

†PARC, 3333 Coyote Hills Rd., Palo Alto, CA 94304 (Ayman.Farahat@parc.com).

‡Department of Mathematics and Computer Science, Gustavus Adolphus College, St. Peter, MN 56082 (tlofaro@gustavus.edu). The work of this author was supported in part by a Research, Scholarship and Creativity Grant from Gustavus Adolphus College.

§Department of Applied Mathematics and Theoretical Physics, Cambridge University, Cambridge, UK (jcm52@cam.ac.uk).

¶Department of Mathematics, Harvey Mudd College, Claremont, CA 91711 (grae@hmc.edu, ward@math.hmc.edu).

those pages relevant to the query, preserving the order.

The common thread running through these methods is linear algebra. For each method, the ranking vector is the dominant eigenvector of some matrix describing the network. How this matrix is defined differs in each method. However, this commonality allows us to establish theorems (see section 3) providing conditions for each method which ensure the convergence of the algorithm, existence of solutions, and uniqueness of solutions. These results also apply to any other methods that reduce to the computation of a dominant eigenvector.

Introducing the HITS algorithm in [15, p. 613], Kleinberg writes:

> For the sake of simplicity, we will make the following technical assumption ... $|\lambda_1(M)| > |\lambda_2(M)|$ [that is, the dominant eigenvalue is simple]. ... When the assumption does not hold, the analysis becomes less clean, but it is not affected in any substantial way.

This idea has been repeated in (among others) [2, 5, 6, 11, 18, 25]. See, for example, this passage from [6, p. 5]:

> iteration will converge to the principal eigenvector for any "nondegenerate" choice of initial vector — ... for example, for any vector all of whose entries are positive. This says that the hub and authority weights we compute are truly an [illegible] feature of the collection of linked pages, not an artifact of our choice of initial weights or the tuning of arbitrary parameters. [Italics in the original.]

In this paper we take up the question of the choice of initial weights. We examine the dependence of the results of the algorithms on changes in initialization. The key point is that the dominant eigenvalue of the appropriate matrix need not be simple. When it is simple, the convergence asserted above for HITS is valid. However, when the dominant eigenvalue is repeated, the ranking vector could be any nonnegative vector in the multidimensional dominant eigenspace. (This occurs for HITS in our example in Figure 4.1, section 4.) In particular, the ranking vector may not be unique, but may depend on the initial seed vector.

Even when the dominant eigenvalue is simple, HITS can return inappropriate zero weights (Figure 4.3, section 4).

We show that the HITS and SALSA algorithms are "badly behaved" on certain networks, meaning that (i) they can return ranking vectors that are not unique but depend on the initial seed vector or (ii) HITS can return ranking vectors that inappropriately assign zero weights to parts of the network. In some cases, this behavior can cause SALSA to return hub and authority weights that are inconsistent.

We characterize all networks where HITS and SALSA are badly behaved in terms of the connectivity of a certain related graph (our [illegible], Definition 4.3). We propose a modification, [illegible], of the adjacency matrix input to the HITS algorithm; our new input matrix adds information about longer paths in the network. We prove that with Exponentiated Input, the HITS algorithm returns a unique eigenvector, without inappropriate zero weights, as long as the network is weakly connected.

Here is an outline of the paper. In section 2, we review four eigenvector ranking algorithms, namely, the PageRank, HITS, and SALSA algorithms, along with our Exponentiated Input modification to the adjacency matrix input to the HITS algorithm. For each algorithm, we identify the nonnegative matrix that describes the structure of the network being considered.

In section 3, we review the Perron–Frobenius theorems which deal with the properties of dominant eigenvalues and eigenvectors of nonnegative matrices. We present existence and (where applicable) uniqueness results for the dominant eigenvectors the ranking algorithms find. In particular, we show that Exponentiated Input to HITS does return a unique ranking (independent of the initialization), without inappropriate zero weights, when the network graph is weakly connected.

In section 4, we give examples of simple networks for which the HITS and SALSA algorithms return nonunique dominant eigenvectors and for which HITS assigns inappropriate zero weights to significant parts of the network. We say that an algorithm is [illegible] on a network if it has either of these two features. We give a characterization of all networks for which HITS and SALSA are badly behaved in terms of the connectivity of our HITS authority graph G' (Definition 4.3), which is an undirected graph related to the directed graph representing the network.

For our example in section 4, Figure 4.1, SALSA returns authority and hub vectors which are inconsistent. In section 4.3 we suggest a modification to the SALSA initialization which makes the vectors consistent.

In section 5 we discuss related issues, including stability of the algorithms, sample networks for which the dominant and second eigenvalues are arbitrarily close for Exponentiated Input to HITS, and other possible modifications of the adjacency matrix input to HITS. Section 6 considers related work, and section 7 gives our conclusions.

2. Eigenvector-based ranking algorithms. To prepare our review of the eigenvector-based ranking algorithms (or [illegible]) PageRank, HITS, Exponentiated Input to HITS, and SALSA, we give some definitions.

The network of web pages and links defines a directed graph G, with the web pages defining the nodes $1, \ldots, n$ and the links defining the edges. This graph is described by an $n \times n$ [illegible] A, where $a_{ij} = 1$ if there is a link from page i to page j and $a_{ij} = 0$ otherwise. We use the following standard definitions from graph theory.

DEFINITION 2.1. [illegible] indegree (outdegree) [illegible] i [illegible] G [illegible] ([illegible]) i

DEFINITION 2.2. [illegible] G weakly connected [illegible]

The notion of dominant eigenvectors and eigenvalues is central to the descriptions of the algorithms in terms of linear algebra.

DEFINITION 2.3. [illegible] dominant eigenvector [illegible] ([illegible]) dominant [illegible] largest [illegible]

We note that some authors use the term [illegible] in a slightly different sense. Also, in [15] the term [illegible] indicates an eigenvector whose eigenvalue is simple and of [illegible] largest modulus.

With our definitions, we can describe the four linear link analysis algorithms.

2.1. PageRank. The PageRank method, developed by Brin and Page [3], is a random walk with random resets on G. This process describes a Markov chain. Let W be the transition probability matrix describing the Markov chain without reset, so that the (i, j)-entry in W gives the probability of following a link on page i to page j. Assuming all links are equally likely, W is derived from A by

$$w_{ij} = \frac{a_{ij}}{\sum_{k=1}^{n} a_{ik}}. \tag{2.1}$$

If at least one node has zero outdegree, then the Markov chain is absorbing, and a modification to W is needed. For instance, PageRank temporarily removes the links to all such nodes, then replaces them after the ranking is calculated [23].

The random reset part of PageRank is described by a probability $\varepsilon < 1$, which determines whether we restart, and a (positive) uniform transition probability matrix U having $u_{ij} = 1/n$ for all i and j. (Positive, nonuniform choices for U are also possible.) Combining these two elements we conclude that the PageRank algorithm is described by the stochastic matrix

$$P = \varepsilon U + (1 - \varepsilon)W. \tag{2.2}$$

With probability ε we randomly choose a new page. Otherwise we follow one of the directed edges from our present node.

The relationship between Markov chains such as this and linear algebra is well understood (see Gantmacher [8], for example). The stationary distribution of this Markov chain is simply the dominant eigenvector of the matrix P^T (the transpose of P). We return to this point in section 3.

2.2. HITS. The HITS algorithm was developed by Kleinberg [15] and is now a part of the CLEVER Searching project of the IBM Almaden Research Center [13]. Similar ideas are used in the Teoma search engine, subsequently acquired by Ask Jeeves [24]. The premise of the algorithm is that a web page serves two purposes: to provide information on a topic and to provide links to other pages giving information on a topic. This gives rise to two ways of categorizing a web page. First, we consider a web page to be an [illegible] on a subject if it provides good information about the subject. Second, we consider the web page to be a [illegible] if it provides links to good authorities on the subject. The HITS algorithm is an iterative algorithm developed to quantify each page's value as an authority and as a hub.

The HITS algorithm is not applied to the graph representing the whole web, but rather to a subgraph, typically of 1000–5000 nodes, derived from traditional text matching of the query terms in the search topic.

We again consider the directed graph G. Each node i is given an initial authority weight $a_0(i)$ and hub weight $h_0(i)$, although the initial authority weights are not used by the algorithm as presented in [15, p. 612]. To begin the algorithm, these nodes are uniformly weighted. The original paper [15, p. 614] states "one can compute weights ... starting from any initial [weights]." We will later discuss complications that can arise if the initial weighting is changed or if the algorithm starts from the initial authority weights instead of the initial hub weights.

The algorithm proceeds iteratively, until values from successive iterations are within a specified error, as follows.

1. In the kth iteration, node i is assigned a new authority weight $a_k(i)$ equal to the sum of $h_{k-1}(j)$, where the sum runs over each node j which points to node i. This is repeated for all nodes in G.

2. The new hub weight $h_k(i)$ is the sum of $a_k(j)$, where the sum runs over the nodes j to which node i points. This is repeated for all nodes in G. Note that the hub weights are computed from the [illegible] authority weights, which in turn were computed from the [illegible] hub weights.

3. Still in the kth iteration, after new weights are computed for all nodes, the weights are normalized so that $\sum_i a_k(i)^2 = \sum_i h_k(i)^2 = 1$.

To construct a linear algebra formulation of this method, we create a vector $\vec{a}_k$ of

the authority weights at iteration k and another vector $\vec{h}_k$ of the hub weights, where

$$\vec{a}_k = [a_k(1), a_k(2), \ldots, a_k(n)]^T, \tag{2.3}$$

$$\vec{h}_k = [h_k(1), h_k(2), \ldots, h_k(n)]^T. \tag{2.4}$$

A uniform initialization gives

$$\vec{a}_0 = [1/\sqrt{n},\ 1/\sqrt{n},\ \ldots,\ 1/\sqrt{n}]^T, \tag{2.5}$$

$$\vec{h}_0 = [1/\sqrt{n},\ 1/\sqrt{n},\ \ldots,\ 1/\sqrt{n}]^T \tag{2.6}$$

up to normalization. When A is the adjacency matrix of the directed graph G, the algorithm becomes

$$\vec{a}_k = \phi_k A^T \vec{h}_{k-1}, \qquad \vec{h}_k = \psi_k A \vec{a}_k, \tag{2.7}$$

where ϕ_k and ψ_k are the normalization constants chosen such that the sum of the squares of the authority weights, as well as that of the hub weights, is 1 in the kth iteration.

Combining the formulas in (2.7), we reach

$$\vec{a}_k = \phi_k \psi_{k-1} A^T A \vec{a}_{k-1} \qquad \text{for } k > 1; \tag{2.8}$$

$$\vec{h}_k = \psi_k \phi_k A A^T \vec{h}_{k-1} \qquad \text{for } k > 0. \tag{2.9}$$

When this method converges, the resulting authority and hub vectors satisfy

$$\vec{a}^* = \frac{1}{\lambda^*} A^T A \vec{a}^*, \qquad \vec{h}^* = \frac{1}{\lambda^*} A A^T \vec{h}^*. \tag{2.10}$$

In other words, the authority vector $\vec{a}^*$ is an eigenvector of $A^T A$, and the hub vector $\vec{h}^*$ is an eigenvector of AA^T. It is simple to show (see section 3.2) that these eigenvectors are, as in the case of PageRank, dominant eigenvectors corresponding to the dominant eigenvalue λ^*. To avoid confusion, it should be noted that throughout this paper, we do not concern ourselves with the eigenvalues of A, but just those of $A^T A$ and AA^T.

2.3. Exponentiated Input to HITS. In [20], we showed that certain network topologies caused the standard HITS algorithm to return either nonunique or non-intuitive results (see section 4). We developed the Exponentiated Input method to address these limitations. HITS's direct dependence on the adjacency matrix A restricts it, in a given iteration step, to considering only paths of length 1. That is, to determine the hub score of a page, at each iteration HITS looks only at the authority scores of adjacent pages. We replace A by a new matrix with information about longer paths.

To consider paths of length greater than 1, we note that given an adjacency matrix A, the number of paths of length m from node i to node j is given by the (i, j)-entry of A^m. We want to consider paths of length 1 to be more important than paths of length 2, 3, and so on. To accomplish this we use the new matrix

$$A + A^2/2! + A^3/3! + \cdots + A^m/m! + \cdots = e^A - I. \tag{2.11}$$

This series converges, since each entry in the mth term of the series can be bounded by $n^m/m!$. While the original adjacency matrix A is binary, $e^A - I$ need not be.

Using this new "exponentiated" matrix $e^A - I$, the HITS algorithm with exponentiated input can be expressed in linear algebra terms as follows. We initialize the authority and hub vectors so that $a_0(i) > 0$ and $h_0(i) > 0$ for all i, and $\sum_i a_0(i)^2 = \sum_i h_0(i)^2 = 1$. (Again, the uniform initialization $a_0(i) = h_0(i) = 1/\sqrt{n}$ is one possibility, and again, $\vec{a}_0$ is never used.) For $k \geq 1$, we [illegible] A [illegible] $e^A - I$, so that we define

$$\vec{a}_k = \phi_k (e^A - I)^T \vec{h}_{k-1}, \qquad \vec{h}_k = \psi_k (e^A - I)\vec{a}_k \tag{2.12}$$

for $k > 0$, with normalization constants ϕ_k, ψ_k. We will show that the sequence $\{\vec{a}_k\}$ approaches the eigenspace of the largest eigenvalue of $(e^A - I)^T(e^A - I)$, and that the sequence $\{\vec{h}_k\}$ approaches the eigenspace of the largest eigenvalue of $(e^A - I)(e^A - I)^T$.

We prove in sections 3.2 and 3.3 that given a weakly connected input graph, the Exponentiated Input modification of HITS prevents the possibilities of (i) returning nonunique rankings that depend on the initialization and (ii) returning inappropriate zero weights. Other matrices such as $A + A^2/2$ or $I + A$ rather than $e^A - I$ would also exclude these possibilities; again, so long as G is weakly connected.

2.4. SALSA. The SALSA (stochastic approach for link structure analysis) algorithm, developed by Lempel and Moran [17], combines the random walk idea of PageRank with the hub/authority idea of HITS. Given a graph G, a bipartite undirected graph H is constructed by building the subset V_a of all the nodes with positive indegree (the potential authorities), and the subset V_h of all the nodes with positive outdegree (the potential hubs). After renumbering, the elements of V_a and V_h become the nodes of H. Since some nodes may have both positive indegree and outdegree, the number m of nodes of H satisfies $m \leq 2n$, where n is the number of nodes of G. The (undirected) edges of H are defined from G as follows: if G has a link from i to j, then we put an edge between the nodes corresponding to i in V_h and j in V_a. The algorithm corresponds to a two-step random walk on the graph H.

If we pick a node in V_a and randomly follow an edge, we traverse to a node in V_h. A second step returns us to V_a. Thus a pair of two-step random walks, one starting in V_a and the other in V_h, can be used to determine the hub and authority vectors. As in the PageRank scheme, each random walk is a Markov chain with a corresponding transition probability matrix. As before, let A be the adjacency matrix of G. Let W_r be the matrix generated from A by dividing each entry of A by its row sum. Similarly define W_c to be generated by dividing each entry of A by its column sum. We arrive at $\vec{a}_k = W_c^T W_r \vec{a}_{k-1}$ and $\vec{h}_k = W_r W_c^T \vec{h}_{k-1}$. Note that the sequence $\vec{a}_k$ depends on the initialization $\vec{a}_0$, and $\vec{h}_k$ depends on the initialization $\vec{h}_0$. This contrasts with HITS, where both sequences depend on $\vec{h}_0$, and leads to subtle differences in the behavior of the two algorithms. In particular the SALSA limits $\vec{a}$ and $\vec{h}$ may not satisfy $\vec{a} = W_c^T \vec{h}$.

A significant advantage of SALSA is that the weightings can be computed explicitly without the iterative process described here [17]. Although the initial paper provides an explicit calculation only in the case of uniform initial weights, this quick method of calculating the weightings can be easily generalized to accommodate nonuniform initial weights.

3. Existence and convergence. Each of the algorithms presented in section 2 reduces to computing the dominant eigenvector of some matrix that describes the connectivity of the given graph. In this section, we discuss the graph properties that ensure the existence and uniqueness of these eigenvectors. If the dominant eigenvalue is repeated, it will have a full eigenspace, and a unique ranking does not exist.

Therefore, perhaps unexpectedly, the ranking returned by an algorithm depends on the initialization used to start the algorithm. We discuss this in greater detail in section 4.

3.1. The Perron–Frobenius theorems. We begin by presenting the necessary theorems concerning dominant eigenvalues and eigenvectors of nonnegative matrices. These theorems describe why the ranking vectors of the above methods exist and when the rankings are unique. In addition, the different hypotheses allow us to describe shortcomings of HITS and SALSA (see section 4). The following definitions and theorems (including proofs) can be found in Gantmacher [8].

DEFINITION 3.1. M positive M nonnegative M

DEFINITION 3.2. M matrix permutation σ_{ij} (i,j) i j i j

A matrix permutation applied to the adjacency matrix of a graph corresponds to a relabeling of the nodes. We use the notation $\circ$ for composition of matrix permutations.

DEFINITION 3.3. M reducible $\sigma = \sigma_{i_1,j_1} \circ \cdots \circ \sigma_{i_m,j_m}$

$$\sigma(M) = \begin{pmatrix} M_1 & 0 \\ M_2 & M_3 \end{pmatrix}, \tag{3.1}$$

M_1 M_3 0 M irreducible

A graph is said to be strongly connected if every node can be reached from every other node following the direction of the edges. It can be shown that a graph whose (weighted) adjacency matrix is M is strongly connected if and only if the matrix is irreducible. This is commonly used as an equivalent definition of irreducibility.

The following three theorems concern the existence and multiplicity of a dominant real eigenvalue and the properties of its corresponding eigenvector. For our results, it is particularly important whether or not the dominant eigenvalue is simple. The first theorem, due to Perron, has the most restrictive hypothesis.

THEOREM 3.4. M r r

Frobenius generalized Perron's result to irreducible nonnegative matrices. The conclusions are naturally weaker in the sense that there may be other eigenvalues having the same modulus as r. However, the dominant real eigenvalue has multiplicity 1.

THEOREM 3.5. M r r r

Finally, if we remove the irreducibility hypothesis, even less can be said. (Consider, for instance, the identity matrix, whose dominant eigenvalue is of course not simple.) However, the dominant real eigenvalue r does exist, is nonnegative, and has a nonnegative eigenvector.

THEOREM 3.6. M r r r

We give a lemma that will help to show that for HITS, the nonnegative eigenvalue found using the above theorems is in fact larger in modulus than all other eigenvalues.

LEMMA 3.7. B B^TB .

. The eigenvalues are real and have full eigenspaces because $M = B^TB$ is symmetric. To show that they are nonnegative, we must show that $x^TMx \geq 0$ for any vector x [16]. This is straightforward: $x^TMx = x^TB^TBx = |Bx|^2 \geq 0$. ∎

3.2. Existence and uniqueness of ranking vectors. In this section, we present existence and uniqueness results for positive or nonnegative dominant eigenvectors of the associated matrices for the four algorithms. In section 3.3, we address convergence to these eigenvectors. Theorems similar to those below can be found in the original papers, but the results on uniqueness (or lack thereof) in Theorems 3.9 and 3.10 seem to have been largely overlooked.

THEOREM 3.8 (PageRank). G A ε (2.2) U .

Here, by the phrase , we mean that the PageRank matrix P for G has an eigenvector (which we call the PageRank vector) corresponding to a positive eigenvalue that is at least as large as the modulus of every other eigenvalue of P, and similarly for the HITS, SALSA, and Exponentiated Input to HITS hub and authority vectors in Theorems 3.9, 3.10, and 3.11 below.

3.8. The result follows immediately from Theorem 3.4. If $\varepsilon > 0$ and U is a positive matrix, then the matrix P of (2.2) is positive, and hence P^T has a simple dominant eigenvalue, which is positive, with a unique eigenvector having positive entries. Convergence of the PageRank algorithm to this eigenvector follows from the theory of Markov chains. See, for instance, [19, p. 490]. ∎

THEOREM 3.9 (HITS). G A

. AA^T and A^TA are nonnegative. Theorem 3.6 gives the result. ∎

THEOREM 3.10 (SALSA). G A

. The matrices $W_c^TW_r$ and $W_rW_c^T$ defined in section 2.4 are nonnegative. Theorem 3.6 gives the result. ∎

We have proved that the eigenvector exists for HITS and SALSA. As we show in section 4, it may not be unique, since the dominant eigenvalue may be repeated (although all other eigenvalues are smaller in modulus, by Lemma 3.7 for HITS and by the theory of Markov chains for SALSA as shown in [17]). However, for Exponentiated Input to HITS, the eigenvector is unique, as we now show.

THEOREM 3.11 (Exponentiated Input to HITS). G A G G

. We prove this result by showing that the authority matrix $(e^A - I)^T(e^A - I)$ has a simple dominant eigenvalue whose eigenvector is positive exactly on those nodes whose indegree is positive. The proof for the hub matrix is analogous.

Without loss of generality, we assume that G has n nodes, t of which have indegree equal to zero, and that those nodes with indegree zero are labeled $n-t+1,\dots,n$. Then the adjacency matrix is given by

$$A = \begin{pmatrix} \widetilde{A} & 0 \\ \widetilde{B} & 0 \end{pmatrix}, \tag{3.2}$$

where $\widetilde{A}$ is $(n-t)\times(n-t)$ and the matrix $\widetilde{B}$ is $t\times(n-t)$. The last t columns of

$$e^A - I = A + A^2/2! + \cdots + A^m/m! + \cdots \tag{3.3}$$

have only zero entries. Therefore

$$(e^A - I)^T(e^A - I) = \begin{pmatrix} C & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.4}$$

where $C = (e^{\widetilde{A}} - I)^T(e^{\widetilde{A}} - I)$. Moreover, the characteristic polynomial of $(e^A - I)^T(e^A - I)$ is $\lambda^t p_C(\lambda)$ where $p_C(\lambda)$ is the characteristic polynomial of C. Hence if C has a simple, dominant, positive eigenvalue, then the same is true for $(e^A - I)^T(e^A - I)$. Further, if the corresponding eigenvector of C is positive, then the corresponding eigenvector of $(e^A - I)^T(e^A - I)$ is the same vector, padded with 0 entries for all the zero indegree nodes. Lemma 3.7 shows that the eigenvalues of C are real and nonnegative. We need to show that C is irreducible so that Theorem 3.5 applies.

Let G' be the graph whose (weighted) adjacency matrix is given by C. This graph G' is the analogue, for Exponentiated Input to HITS, of our HITS authority graph; see section 4.2 and especially Definition 4.3. Note that G' can be considered an undirected graph because C is symmetric. C is irreducible if and only if G' is connected. We seek to show that G' is connected.

The (i,j)-entry of C is nonzero if and only if there is a node k such that the original graph G has a path from k to i and a path from k to j. From this we conclude that all vertices that can be reached from a node k following the links in G must be in the same component of G'. Further, if k has nonzero indegree, it follows that k is in that component of G' as well.

Let us assume that G' is disconnected. Then it can be separated into two components H_1 and H_2 (which may themselves be disconnected). It must be true that there is no link between H_1 and H_2 in G. Since G is weakly connected, there must be a node (with zero indegree) that links to some node l in H_1 and to another node m in H_2. However, that would lead to a connection between these nodes (l and m) in G', contradicting the assumption that G' is disconnected.

Thus G' is connected and C is irreducible. This completes the proof. □

A close inspection of the proof shows that if the more easily computable matrix $A + A^2/2$ were used instead of $e^A - I$, then the result would still hold. The main observation needed is that all nodes that can be reached from a given node i following edges in G are still in the same component of G'. This is because a link from i to j and a link from j to k in G will imply a link between j and k in G'. Consequently the proof above still applies.

Alternatively we could use $I + A$, as the graph G' contains the graph G (ignoring direction) as a subgraph. Consequently, if G is weakly connected, then G' is connected. Since G' is connected, the same arguments then show that the algorithm converges uniquely. The graph G' will be discussed in more detail later.

3.3. Convergence of the ranking algorithms. All of the methods described in section 2 reduce to the computation of a dominant eigenvector. However, PageRank and SALSA compute this eigenvector using a Markov chain method, while HITS and Exponentiated Input to HITS are formulated in terms of summing the weights of linked nodes at each step.

A probability transition matrix is [illegible] if some power of the matrix is positive. Moreover, a regular transition matrix has a unique dominant eigenvector that (after scaling) gives the stationary distribution of the associated Markov chain.

The transition matrix P of the PageRank method is positive, and hence regular, if the random reset parameter $\varepsilon > 0$. The ergodic theorem guarantees that the random walk process that PageRank uses converges to the stationary vector of the transition matrix. Therefore PageRank converges to a unique ranking vector.

SALSA also uses this approach. However, because SALSA does not incorporate random reset, the graph structure affects whether the ranking vector is unique. In particular, if the bipartite graph H is not connected, then the authority and hub vectors are not unique. The original SALSA paper showed, using the ergodic theorem, that each component of the graph has a simple dominant eigenvalue which is equal to one. Each component will converge on its own, and the final result will depend on how much weight each component is given in the initialization.

The HITS and Exponentiated Input to HITS methods differ only in their input matrices, and so the proof of convergence is identical for both methods. We offer a heuristic proof, followed by a short rigorous proof.

It is sufficient to consider the convergence of the hub vector $\vec{h}_k$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$ be the eigenvalues of AA^T. Because AA^T is symmetric, each eigenvalue has a full eigenspace and the eigenvectors can be chosen to form an orthogonal basis for $\mathbf{R}^n$.

We can express $\vec{h}$ as a linear combination of the eigenvectors $\vec{v}_1, \ldots, \vec{v}_n$. So $\vec{h}_1 = \alpha_1\vec{v}_1 + \cdots + \alpha_n\vec{v}_n$, where $\alpha_i = \vec{h}_1 \cdot \vec{v}_i / \|\vec{v}_i\|^2$. Then $\vec{h}_2 = \lambda_1\alpha_1\vec{v}_1 + \cdots + \lambda_n\alpha_n\vec{v}_n$. Suppose $\lambda_1 = \lambda_2 = \cdots = \lambda_r \neq \lambda_{r+1}$. We can rewrite $\vec{h}_2$ as $\lambda_1(\alpha_1\vec{v}_1 + \cdots + \alpha_r\vec{v}_r) + \lambda_{r+1}\alpha_{r+1}\vec{v}_{r+1} + \cdots + \lambda_n\alpha_n\vec{v}_n$. Iterating, we obtain

$$\vec{h}_{k+1} = \xi_k \left(\lambda_1^k [\alpha_1\vec{v}_1 + \cdots + \alpha_r\vec{v}_r] + \sum_{i=r+1}^{n} \lambda_i^k \alpha_i \vec{v}_i \right), \tag{3.5}$$

where $\xi_k = \psi_k\phi_k\xi_{k-1}$ is the normalization constant. As k increases, the λ_1^k term dominates. Thus, $\vec{h}_k \to c\,(\alpha_1\vec{v}_1 + \cdots + \alpha_r\vec{v}_r)$. A closer analysis shows that $\psi_k\phi_k \to \lambda_1$, which gives a numerical test for convergence.

We now offer a rigorous proof of convergence.

THEOREM 3.12. [illegible] $\vec{a}_k$ [illegible]
[illegible] A^TA [illegible] $\vec{h}_k$ [illegible]
[illegible] AA^T [illegible]
$(e^A - I)^T(e^A - I)$ [illegible] $(e^A - I)(e^A - I)^T$

The HITS algorithm computes all future hub and authority vectors, $\vec{h}_k$ and $\vec{a}_k$, $k \geq 1$, from the initial [illegible] vector $\vec{h}_0$; it does not use the initial authority vector $\vec{a}_0$ (section 2.2). A fine point is that if instead the HITS algorithm were implemented so that it made use of the initial [illegible] vector and ignored the initial hub vector, then the resulting authority vectors $\vec{a}_k$ could converge to a [illegible] vector. Figure 4.1 gives

an example where this would happen. Theorem 4.4 provides a sufficient condition for HITS to converge to the same authority vector, regardless of whether HITS makes use of $\vec{h}_0$ or $\vec{a}_0$. If G' is weakly connected, Theorem 3.11 proves that Exponentiated Input to HITS also enjoys that uniqueness property.

3.12. The eigenvalues of AA^T are real and nonnegative. It follows that while the eigenvalue of largest modulus may be repeated, all other eigenvalues have strictly smaller modulus. Since AA^T is symmetric, the eigenspaces are orthogonal. In the dominant eigenspace, we can use Gram–Schmidt orthonormalization to choose orthogonal vectors such that one of them is nonnegative. Because $\vec{h}_0$ is positive, its dot product with this nonnegative vector is positive, and thus $\vec{h}_0$ has a nontrivial component in the eigenspace of the dominant eigenvalue. This ensures that the algorithm converges to an eigenvector of the largest eigenvalue [10, pp. 351–353]. By the construction of the HITS algorithm, the limit cannot have negative entries. Note that although the algorithm converges, it could converge to any nonnegative, normalized vector in the eigenspace of the dominant eigenvalue λ_1, depending on the initial choice $\vec{h}_0$ of the hub weights.

The same argument applies to the HITS authority vector and to the authority and hub vectors for the Exponentiated Input to HITS algorithm. □

We note that one could apply the PageRank idea of adding a matrix εU, where $\varepsilon > 0$ and U is a positive matrix, to the matrices AA^T and A^TA for HITS, $W_rW_c^T$ and $W_c^TW_r$ for SALSA, and $(e^A - I)(e^A - I)^T$ and $(e^A - I)^T(e^A - I)$ for Exponentiated Input to HITS. (For the last-mentioned algorithm, this would be needed only when G is not weakly connected and so the dominant eigenvalue is not already known to be simple.) Since the resulting perturbed matrices are positive, Theorem 3.4 then guarantees the existence and uniqueness of a dominant eigenvector for each perturbed matrix. One could then use perturbation analysis to study the stability of the resulting eigenvector and how close this eigenvector is to the dominant eigenspace of the original unperturbed matrix—in particular, how close the resulting authority vector is to the possible authority vectors given by the original matrix.

4. Limitations of the HITS and SALSA algorithms. In the previous section, we presented conditions for the existence and uniqueness of ranking vectors. The weakest conclusions were for the HITS and SALSA algorithms. In this section we explore in more detail the limitations of these two methods. The key questions here are whether the largest eigenvalue of A^TA for HITS, or of $W_c^TW_r$ for SALSA, is simple or repeated and whether its eigenvector has inappropriate zero entries.

4.1. Examples. We begin by considering a few simple examples that illustrate some limitations of the HITS and SALSA algorithms.

We first demonstrate the problem of repeated largest eigenvalue as the initialization changes. Examples of networks with repeated largest eigenvalue for HITS have appeared in [20] and independently in [2, section 5] (where they were considered in the context of finding communities). The initial SALSA paper [17] mentioned the possibility of a repeated dominant eigenvalue but did not investigate changes in the initialization.

With the uniform initialization $\vec{h}_0 = [1/\sqrt{n}, 1/\sqrt{n}, \ldots, 1/\sqrt{n}]^T$ for the hub vector, the HITS algorithm applied to the network in Figure 4.1 yields $\vec{a} = [2/\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 0]^T$ and $\vec{h} = [0, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}]^T$. This means that vertex 6 is no better a hub than vertices 2, 3, 4, and 5. In contrast, if the chain of $\vec{h}_i$ and $\vec{a}_i$ were to start from a uniformly initialized $\vec{a}_0$ rather than $\vec{h}_0$, we would find

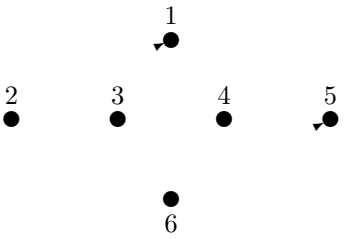


FIG. 4.1. *A network with repeated largest eigenvalue for A^TA. The HITS algorithm with standard input yields different authority vectors for different initial seed vectors, and the SALSA algorithm with standard input gives inconsistent authority and hub vectors.*

that $\vec{a} = [1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 0]^T$ and $\vec{h} = [0, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 2/\sqrt{5}]^T$ so that vertex 1 is now no better an authority than vertices 2, 3, 4, and 5.

More generally, choosing different (positive) initial seed vectors for the HITS algorithm, the final authority vector for the network in Figure 4.1 can be

$$\vec{a} = [\alpha, \beta, \beta, \beta, \beta, 0]^T \tag{4.1}$$

for any positive numbers α and β such that $\alpha^2 + 4\beta^2 = 1$. In other words, the possible outcomes for $\vec{a}$ can be , normalized, nonnegative vector in the two-dimensional eigenspace of the dominant eigenvalue for A^TA except $[1, 0, 0, 0, 0, 0]^T$ and $[0, 1/2, 1/2, 1/2, 1/2, 0]^T$.

Similar behavior occurs when the SALSA algorithm is applied to the same network (in Figure 4.1). The rankings from SALSA result from the limits of two different chains which do not interact. Each of these chains will converge, but their limits will be unrelated.

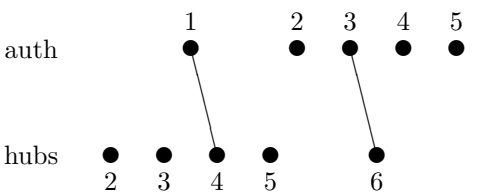


FIG. 4.2. *The SALSA graph H associated with Figure* 4.1.

The random walk performed by the SALSA algorithm is confined to whichever component of Figure 4.2 it begins in. Using the standard uniform initialization, we find that $\vec{a} = [1/5, 1/5, 1/5, 1/5, 1/5, 0]^T$ and $\vec{h} = [0, 1/5, 1/5, 1/5, 1/5, 1/5]^T$, so that all nodes with nonzero outdegree are equally good hubs and all nodes with nonzero indegree are equally good authorities.

These two weightings are inconsistent. Because the premise of HITS and SALSA is that good hubs point to good authorities, if four out of five equally good hubs all point to one node and no others, then we expect that node to be a better authority than any of the four nodes pointed to by the fifth hub. In other words, we expect that $\vec{a} = W_c^T \vec{h}$ and $\vec{h} = W_r \vec{a}$. The fact that this is not the case results from the repeated eigenvalue of $W_c^T W_r$. For arbitrary positive initial vectors, the final authority vector can be any vector of the form shown in (4.1) such that $\alpha, \beta > 0$ and $\alpha + 4\beta = 1$.

The reason for the inconsistency is that the initial hub vector assigns the first

component four times as much weight as the second component. In contrast, the authority vector assigns the second component four times as much weight as the first.

For both HITS and SALSA, there are some graphs that give rise to repeated eigenvalues. The output of all these graphs is sensitive to the initial vector chosen. A second problem, inappropriate zero weights, can arise in HITS regardless of the output's dependence on or independence of the initial vector.

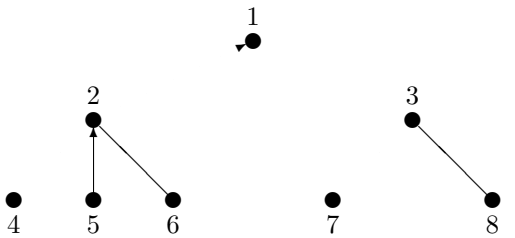


FIG. 4.3. *A tree with simple largest eigenvalue for A^TA.*

Consider the graph in Figure 4.3. If we remove vertex 5 from the graph, obtaining the two-level binary tree, then the characteristic polynomial of A^TA is $(\lambda-2)^3\lambda^4$ and so A^TA has a repeated dominant eigenvalue leading to nonunique behavior. However, the inclusion of vertex 5 creates a significantly different situation. Now A^TA has a simple dominant eigenvalue 3 and repeated eigenvalues of 2 and 0.

In this case, as long as the initial seed vector is positive, the output of HITS is

$$\vec{a} = [0,\ 1,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]^T, \tag{4.2}$$

$$\vec{h} = [0,\ 0,\ 0,\ 1/\sqrt{3},\ 1/\sqrt{3},\ 1/\sqrt{3},\ 0,\ 0]^T. \tag{4.3}$$

This means that node 2 gets authority weight 1, while all other nodes get authority weight 0. The majority of the graph is deemed to be completely unimportant. However, we would expect that node 1 should be a better authority than any other node, because it is accessible from every node. Meanwhile, node 3 should be only slightly less important than node 2, since there is only one more node pointing to node 2.

For the same graph (Figure 4.3), SALSA once more has repeated eigenvalues and nonunique rankings. In this case, if we use the uniform initialization, we again have inconsistent results: $\vec{a} \neq W_c^T\vec{h}$. In most cases where the SALSA algorithm has repeated eigenvalues, the uniform initialization will result in this inconsistency between the authority and hub vectors. The following definition introduces terminology to describe these behaviors.

DEFINITION 4.1. badly behaved

4.2. Characterization of badly behaved graphs. Theorem 4.2 in this section gives a linear algebra interpretation of what it means for the HITS and SALSA algorithms to be badly behaved on a graph. We also characterize (Theorems 4.4 and 4.5) the graphs for which the HITS and SALSA algorithms are badly behaved.

THEOREM 4.2.

1. AA^T
2. AA^T (A^TA)

. Assume that condition 1 of Theorem 4.2 holds; that is, AA^T has a repeated dominant eigenvalue.

Let the seed vector given to the HITS algorithm be $\vec{h}_0$. Each iteration preserves the direction while changing the magnitude of the component in each eigenspace. Once the nondominant eigenvectors are effectively scaled to zero, the final result is the (normalized) component of $\vec{h}_0$ in the eigenspace of the dominant eigenvalue. Since the eigenvalue is repeated, there are many possibilities for this component. Conversely, if there are multiple possible outcomes depending on the seed vector, then the component in the eigenspace of the dominant eigenvalue is not unique, and so the eigenspace is multidimensional.

Now assume that condition 2 of Theorem 4.2 holds; that is, the dominant eigenvector has a 0 in entry i, corresponding to a node i with positive outdegree. As the iterates converge to the dominant eigenvector, the hub weight of node i converges to 0.

Conversely, if the hub weight of node i converges to 0, then the corresponding entry from the dominant eigenvector must be 0. □

In fact, if the dominant eigenvalue is repeated, the eigenspace is spanned by several nonnegative eigenvectors that satisfy condition 2.

Next we characterize the set of graphs on which the HITS algorithm is badly behaved, and describe some results about its behavior on these graphs.

DEFINITION 4.3. G $1, \ldots, n$ HITS authority graph G' G' G ( ) G' $\{i,j\} \in E(G')$ $k \in G$ (k,i) (k,j) G

This graph G' was first described in [20]. Shortly thereafter, it was independently described in [2] in the context of stability (see section 5.1).

We give some intuition for why the HITS authority graph G' is important for understanding the behavior of the HITS algorithm on G. Under the HITS algorithm, the authority weight of i has an impact on the hub weight of k because (k,i) is an edge of G. Similarly, the hub weight of k affects the authority weight of j. Thus, the edge $\{i,j\}$ represents a relation in two iterations of the algorithm between the authority weights of nodes i and j.

If we start the HITS algorithm with node i having authority weight 1 and all other nodes having authority weight 0, then after two iterations, all nodes joined to i in G' will have nonzero weight. After four iterations, all nodes that can be reached in at most two steps in G' will have nonzero weight. Continuing, all nodes that are in the same component as i will get nonzero weight, but the weight will not leak into any other components.

If we allow repeated edges $\{i,j\}$ in G' whenever there are multiple nodes pointing to i and j in G, and if we allow a loop at i in G' for each k pointing to i in G, then the adjacency matrix of G' is A^TA (up to deletion of zero rows and columns in A^TA). Using appropriate matrix permutations, this matrix can be made block diagonal with each block corresponding to a different component of G'. Therefore, the eigenvectors (and eigenvalues) of A^TA can be partitioned into disjoint subsets corresponding to the connected components of the HITS authority graph G'. (Note that allowing repeated edges and loops does not change the connectivity of G'.) This leads to our theorem characterizing the cases where HITS is badly behaved.

THEOREM 4.4. G G'

. The "only if" direction of the proof follows from Theorem 3.5 along with Lemma 3.7. If HITS is badly behaved, then the matrix $\widetilde{C}$ defined below must be reducible, and therefore G' is disconnected.

Assume that there are t nodes with positive indegree. Defining $C = A^T A$ and labeling the nodes so that those with 0 indegree have the highest index, we can rewrite C as

$$C = \begin{pmatrix} \widetilde{C} & 0 \\ 0 & 0 \end{pmatrix}, \tag{4.4}$$

where $\widetilde{C}$ is $t \times t$. We are interested in the dominant eigenvalue and eigenvector of $\widetilde{C}$.

If the dominant eigenvalue is not simple, or the eigenvector has zero entries for nodes with nonzero indegree, then $\widetilde{C}$ must be reducible. However, as noted in the proof of Theorem 3.11 above, $\widetilde{C}$ is reducible if and only if G' is disconnected.

To prove the "if" direction, it suffices to show that for each component there is a nonnegative eigenvector that is positive exactly on that component. Since the matrix is symmetric, all eigenvectors of different eigenvalues are orthogonal. If there exists a unique dominant eigenvector, it is nonnegative, and thus it cannot be orthogonal to all of these. In fact, it must be one of these. Using this and Theorem 4.2, it follows that the HITS algorithm is badly behaved.

(Another way to see this is, for G' with more than one connected component, to relabel the vertices of G' with the vertices of the first component first, then those of the second component, and so on. The matrix $\widetilde{C}$ becomes block diagonal. Each eigenvector of $\widetilde{C}$ arises as an eigenvector of one of the blocks, padded with zeroes "outside that block." Thus any unique dominant eigenvector $\vec{v}$ of $\widetilde{C}$ is of this special form. But then $\vec{v}$ has zero entries everywhere "outside that block," and in particular in those positions corresponding to vertices in other components of G'. But these vertices have positive indegree in G. Thus the HITS algorithm on G is badly behaved.) □

There is a strong connection between the HITS authority graph G' and the SALSA bipartite graph H. In particular, each edge of G' corresponds to a "vee" in H. More specifically, if there is an edge $\{i, j\}$ in G', then there exists a node k that links to both i and j. This means that on the graph H there is a hub node corresponding to k that has an edge to both i and j. It follows immediately that G' is disconnected if and only if H is disconnected. This proves the following.

THEOREM 4.5. . G G' .

This theorem can also be proven easily from the results of section 6.1 of [17]. It is not mentioned in [17] that multiple components can lead to inconsistent hub and authority rankings.

We do not know how likely it is in practice for the HITS authority graph G' corresponding to a particular query to be disconnected.

4.3. Modification to the SALSA initialization. The results of this section show that the inconsistency encountered with SALSA is due to the SALSA graph being disconnected. In particular, when we uniformly initialize the authority and hub vectors, the total authority weight given to a component may differ from the total hub weight given to that component. In all such cases, the final authority vector will be inconsistent with the hub vector.

This suggests the following modified initialization. To each component C_i we assign a component weight equal to $|C_i|/|H|$, the proportion of the nodes in the SALSA graph H which are in C_i. Note that $|C_i|$ counts nodes in both the authority and the

hub subsets, V_a and V_h, of H. When we initialize the authority weights in component C_i, we ensure that their sum equals the component weight, and similarly we initialize the hub weights. Then the final hub and authority weights will be consistent.

Although this correction will ensure that the weightings given are consistent, it does not guarantee that the weightings are meaningful in terms of the application. If some components are a priori more important than others because of extra information, then that should be considered when assigning the component weights.

When applied to Figure 4.1, this initialization results in $\vec{a} = [1/2, 1/8, 1/8, 1/8, 1/8, 0]^T$ and $\vec{h} = [0, 1/8, 1/8, 1/8, 1/8, 1/2]^T$. This contrasts with the results of the standard initialization, which were that all of the nonzero weights were 1/5.

5. Related issues. In this section, we discuss the stability of HITS to changes in the network structure; two scenarios involving closely clustered top eigenvalues; and a different modification, [illegible], of the adjacency matrix.

5.1. Stability. Recent work in [21, 22] studied the stability of HITS and other algorithms under small changes in the network structure. We can view the network perturbation as a continuous change in the adjacency matrix. An added link can be given weight t and a deleted link can be given weight $1-t$ as t goes from 0 to 1. Thus we move from an original network to the modified network by a continuous process.

If the first two eigenvectors switch, this corresponds to their associated eigenvalues switching order. In order for this to happen, the eigenvalues must be equal at some value t_0. Consider the network structure at that value. The only restriction on the value of the link weights in our proof of Theorem 4.4 was that they be positive. This theorem gives a necessary condition for the dominant eigenvalue to be repeated.

The authors of [21, 22] found examples of real networks on which small changes in the link structure appeared to cause the sort of switch in eigenvalues described above. This suggests that the HITS authority graph G' may have been disconnected to have caused that effect, or nearly so to have caused a similar effect.

More generally, small changes to a matrix tend to result in large changes to eigenvectors only when eigenvalues are close. We would expect the leading eigenvalues to be close together when G' is less tightly connected. This implies a close relationship between the connectivity of G' and the stability of HITS.

Similar results are obtained in [2]. The authors of [2] also describe the HITS authority graph, and focus their attention on cases where it is connected (their "authority-connected graphs").

5.2. Arbitrarily close eigenvalues. Although we have shown that for Exponentiated Input to HITS the largest eigenvalue of $(e^A - I)^T(e^A - I)$ is not repeated, a potential problem arises when the second largest eigenvalue is very close to the largest. The number of iterations needed for convergence increases without bound as the eigenvalues get close to one another. The following examples illustrate two scenarios where this problem appears to occur.

[illegible] 5.1. We begin by describing a family of graphs where numerical experiments suggest that the dominant and second eigenvalues can be made arbitrarily close. These graphs are of the [illegible]" form shown in Figure 5.1. The graph has a root vertex from which two identical branches begin. Each branch consists of m consecutive nodes of outdegree 1; we call $l = m - 1$ the [illegible] of the graph. The last node in this sequence has outdegree b, producing a [illegible]" at the end of each branch. We call b the [illegible] of the graph.

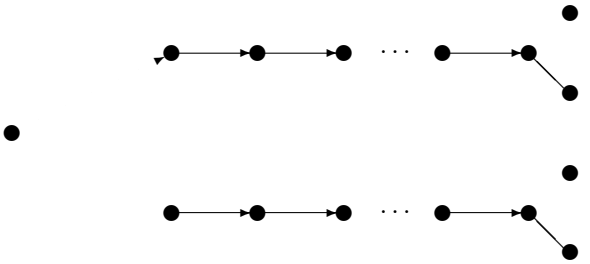


FIG. 5.1. *Broom tree with brush width* $b = 2$ *and large handle length* l.

The ratio between the second largest eigenvalue and the dominant eigenvalue of $(e^A - I)^T(e^A - I)$ is called the [illegible] of the graph. Using a program written in MATLAB [14] we fixed the brush width and let the handle length increase from $l = 5$ to $l = 50$. For each value of l we computed the corresponding eigenvalue ratio. When the brush width is held at $b = 1$ we obtain a family of brooms for which the eigenvalue ratio increases monotonically from 0.7796 at $l = 5$ to 0.9277 at $l = 50$. Fixing the brush width at $b = 2$ and letting l increase produces a dramatically faster convergence of the gap ratio from 0.9524 at $l = 5$ to 1.000 at $l = 50$ (all calculations rounded to 4 significant digits). Figure 5.2 shows the graph of handle length versus eigenvalue ratio for $b = 1, 2, 3$.

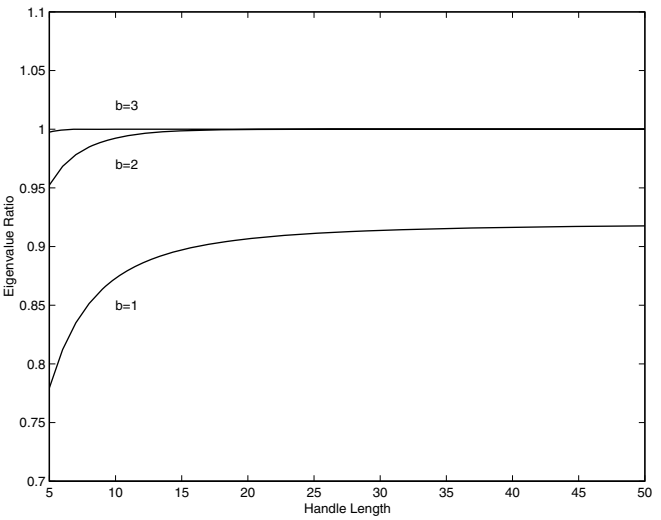


FIG. 5.2. *Ratio between second and dominant eigenvalues appears to approach one as* $l \to \infty$.

One can consider many different modifications to the adjacency matrix A input to HITS, beyond our Exponentiated Input $e^A - I$. In the example above we fixed the modification of A, and varied the graph, in order to get an eigenvalue ratio close to one. We now give an example showing that we can vary the modification of A for a fixed graph and thus obtain an eigenvalue ratio close to one.

[illegible] 5.2. Consider the two-level binary tree (delete vertex 5 from the graph shown in section 4.1, Figure 4.3). Here the standard HITS algorithm is badly behaved because the graph has a repeated dominant eigenvalue. Consider a modification of A where paths of length 2 are counted with value ε. Let $\widetilde{A} = A + \varepsilon A^2$. With this $\widetilde{A}$,

we can prove that the dominant eigenvalue will be simple for any weakly connected graph. Then the input to the HITS algorithm corresponds to finding eigenvectors of

$$\widetilde{A}^T\widetilde{A} = \begin{bmatrix} 2+4\varepsilon^2 & 2\varepsilon & 2\varepsilon & \cdots & 0 \\ 2\varepsilon & 2 & 0 & & \\ 2\varepsilon & 0 & 2 & & \\ \vdots & & & \ddots & \\ 0 & & & & 0 \end{bmatrix}, \tag{5.1}$$

where all omitted entries are 0. As $\varepsilon \to 0$, Gershgorin's disk theorem [12, p. 344] shows that the three largest eigenvalues of $\widetilde{A}^T\widetilde{A}$ become arbitrarily close.

To sum up, the examples in this subsection indicate that the top few eigenvalues, either of $(e^A - I)^T(e^A - I)$ or of $\widetilde{A}^T\widetilde{A}$, may be very close to each other, even if they are not equal in modulus.

5.3. Usage Weighted Input to HITS. A different type of modification to the adjacency matrix is possible when web server logs are available for a given network. In [20], we proposed [illegible], in which we initialize the matrix to zero, and then increment the (i, j)-entry every time a user travels from node i to node j. (The resulting input matrix need not mirror the network, since users may ignore some hyperlinks and may navigate directly between pages that are not hyperlinked.) In each iteration the most frequently followed links play the largest role in determining new authority weights. The effect is similar to that of lifting by gradient ascent [7] but does not require direct querying of users. Similar approaches have recently been advocated by [26], [27].

6. Related work. Kleinberg's HITS algorithm is presented in [15] and discussed in more detail in [9]. Several authors have extended the HITS algorithm using textual analysis. In their paper [1] on topic distillation in hyperlinked environments, Bharat and Henzinger combine the connectivity analysis of the HITS algorithm with content analysis. They identify three limitations of the HITS algorithm: (i) mutually reinforcing relationships between hosts; (ii) automatically generated links that do not reflect a human assessment of the value of pages; and (iii) nonrelevant documents leading to topic drift. They propose modifications to the HITS algorithm, using content analysis, that address these limitations. For instance, they address the common problem of topic drift by introducing the [illegible] of a node, which measures the similarity of the node to a query document. Relevance weights are then used to prune nonrelevant nodes and to regulate the influence of a node according to its relevance, specifically by multiplying the vectors of hub and authority weights componentwise by the vector of relevance weights. Both the limitations and the solutions are different from those considered in the present paper; in particular we do not use content analysis.

The ARC or [illegible] system [5], for compiling a list of authoritative web documents on a given topic, also augments the HITS algorithm with content analysis. Specifically, the ARC system multiplies the weight assigned to an [illegible] link to a given node by a positive factor that increases with the amount of topic-related text occurring around the [illegible] link. These ideas are developed further in the paper [4] on spectral filtering. Here nodes may be web pages or even smaller units, allowing a finer analysis of the link topology and in particular inhibiting topic drift. In [4], the HITS algorithm is also modified to address several other idiosyncratic features of the web. The adjacency matrix is replaced by an [illegible] in which

the entry a_{ij} represents the strength of the affinity between nodes i and j. The value of a_{ij} is a sum of three components, one of which quantifies the occurrence of query terms in text near the hyperlink to node j.

The paper [7] on customized authority lists modifies the HITS algorithm by [illegible] of authority weights. The idea is to incorporate user feedback to adjust the authority weights to reflect an individual user's evaluation of authoritativeness.

Borodin et al. [2] discuss other related theoretical aspects of authority and hub algorithms including both HITS and SALSA. They provide two mechanisms for determining when different algorithms provide similar rankings and then use this framework to compare several ranking algorithms. The stability of ranking algorithms to graph modifications is also discussed in detail. These ideas are similar in spirit to those in [21, 22].

Our work differs from [4, 5] in that we do not use textual analysis and from [7] in that we do not require direct querying of users. Our Exponentiated Input work differs from the other work discussed here in that the adjacency matrix is modified to take into account paths of length greater than one. Our work differs from [2, 21, 22] in that we are not concerned with the stability of the result to changes in the network structure. We are interested in networks for which the results of the ranking algorithm are sensitive to changes in the initialization.

The use of links in ranking documents and finding authoritative sources is similar to the techniques of citation analysis in bibliometrics. See also [4, 7] for fuller surveys of the literature.

7. Conclusions and further work. We have given a unified treatment of the existence and uniqueness of ranking vectors generated by four link analysis algorithms, and of convergence of the algorithms. We have shown that the HITS and SALSA algorithms behave unexpectedly on some graphs. In particular, their output can depend on the initial seed vector; the output of HITS can depend on whether the implementation uses the initial authority vector or the initial hub vector; HITS can inappropriately assign zero weights to parts of a graph; and the authority and hub vectors produced by SALSA can be inconsistent. The PageRank algorithm does not display this behavior; nor does Exponentiated Input to HITS if the graph is weakly connected.

It is important to understand when this unexpected behavior occurs. We have characterized the graphs for which HITS and SALSA are badly behaved; for this we presented a general method that extends ideas in [2, 17]. This method can be easily applied to other algorithms and can help motivate modifications that eliminate the bad behavior. In this light, we have introduced a new modification of the HITS algorithm, namely Exponentiated Input to HITS, which eliminates bad behavior as long as the network is weakly connected. We have also proposed a change to the initialization of SALSA which eliminates inconsistency in hub and authority weights.

The idea of Exponentiated Input can be extended to SALSA by changing the input matrix A with similar results. The following natural extension leads to a similar algorithm. Consider a web surfer surfing for n time steps. At each step, she either remains where she is, with probability $(n-1)/n$, or she follows a link at random, with probability $1/n$. After n time steps the expected number of links followed is 1. The surfer then follows links backward in a similar manner for n time steps. Taking the limit as $n \to \infty$, the algorithm reduces to finding the dominant eigenvector of $e^{W_r} e^{W_c^T}$.

The choice of scaling factors (for paths of each length) used in constructing the Exponentiated Input matrix would be interesting to study. This is especially true because calculating powers of a large matrix is computationally expensive. At present

we have no principled way to decide which scaling factors are most appropriate for a given application.

While the results of this paper are concerned with existence and uniqueness and with the dependence of rankings on initialization, we have discussed how the methods applied here may relate to other important ideas of link analysis. In particular, the HITS authority graph G' can be used to explore both stability and community issues addressed in other works.

Acknowledgment. We are grateful for the detailed comments provided by two anonymous reviewers.

REFERENCES

[1] K. Bharat and M. R. Henzinger, *Improved algorithms for topic distillation in a hyperlinked environment*, in Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1998, pp. 104–111.

[2] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas, *Finding authorities and hubs from link structures on the World Wide Web*, in Proceedings of the Tenth International Conference on the World Wide Web, 2001, pp. 415–429.

[3] S. Brin and L. Page, *The anatomy of a large-scale hypertextual (web) search engine*, in Proceedings of the Seventh International Conference on the World Wide Web, 1998, pp. 107–117.

[4] S. Chakrabarti, B. Dom, D. Gibson, R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, *Spectral filtering for resource discovery*, in ACM SIGIR Workshop on Hypertext Information Retrieval on the Web, 1998.

[5] S. Chakrabarti, B. Dom, S. Rajagopalan, D. Gibson, and J. Kleinberg, *Automatic resource compilation by analyzing hyperlink structure and associated text*, in Proceedings of the Seventh International Conference on the World Wide Web, 1998, pp. 65–74.

[6] S. Chakrabarti, B. E. Dom, S. Ravi Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg, *Mining the link structure of the World Wide Web*, IEEE Computer Magazine, 32 (1999), pp. 60–67.

[7] H. Chang, D. Cohn, and A. McCallum, *Creating customized authority lists*, in Proceedings of the Seventeenth International Conference on Machine Learning, 2000, pp. 49–54.

[8] F. R. Gantmacher, *Matrix Theory*, Vol. 2, Chelsea, New York, 1974.

[9] D. Gibson, J. Kleinberg, and P. Raghavan, *Inferring web communities from link topology*, in Proceedings of the Ninth ACM Conference on Hypertext and Hypermedia, 1998, pp. 225–234.

[10] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[11] M. Henzinger, *Link analysis in web information retrieval*, IEEE Data Engrg. Bull., 23 (2000), pp. 3–8.

[12] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[13] IBM, *The CLEVER Project*, http://www.almaden.ibm.com/cs/k53/clever.html.

[14] The Mathworks Inc., *MATLAB version* 5.3, 1999.

[15] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, J. ACM, 46 (1999), pp. 604–632.

[16] P. D. Lax, *Linear Algebra*, John Wiley and Sons, New York, 1997.

[17] R. Lempel and S. Moran, *The stochastic approach for link-structure analysis (SALSA) and the TKC effect*, in Proceedings of the Ninth International Conference on the World Wide Web, 2000, pp. 387–401.

[18] L. Li, Y. Shang, and W. Zhang, *Improvement of HITS-based algorithms on web documents*, in Proceedings of the Eleventh International Conference on the World Wide Web, 2002, pp. 527–535.

[19] C. R. MacCluer, *The many proofs and applications of Perron's theorem*, SIAM Rev., 42 (2000), pp. 487–498.

[20] J. C. Miller, G. Rae, F. Schaefer, L. A. Ward, A. Farahat, and T. LoFaro, *Modifications of Kleinberg's HITS algorithm using matrix exponentiation and web log records*, in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001, pp. 444–445.

[21] A. Y. Ng, A. X. Zheng, and M. I. Jordan, *Link analysis, eigenvectors, and stability*, in Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

[22] A. Y. Ng, A. X. Zheng, and M. I. Jordan, *Stable algorithms for link analysis*, in Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001, pp. 258–266.

[23] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation ranking: Bringing order to the Web*, http://newdbpubs.stanford.edu:8090/pub/1999-66 (1998).

[24] S. Robinson, *The ongoing search for efficient Web search algorithms*, SIAM News, 37 (November 2004), pp. 4, 11.

[25] N. Sundaresan, J. Yi, and A. Huang, *Using metadata to enhance a web information gathering system*, in WebDB (Informal Proceedings), 2000, pp. 11–16.

[26] M. Wang, *A significant improvement to Clever algorithm in hyperlinked environment*, in Proceedings of the Eleventh International Conference on the World Wide Web, 2002.

[27] G.-R. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, H.-J. Zhang, and C.-J. Lu, *User access pattern enhanced small web search*, in Proceedings of the Twelfth International Conference on the World Wide Web, 2003.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1202–1223
© 2006 Society for Industrial and Applied Mathematics

ANALYSIS OF BOX SCHEMES FOR REACTIVE FLOW PROBLEMS*

S. L. MITCHELL†, K. W. MORTON‡, AND A. SPENCE‡

Abstract. Key properties of the box scheme are shown to be advantageous for reactive flow problems. Unconditional stability and compact conservation are shown by a detailed modified equation analysis to enable the scheme to reflect exactly the "reduced speed," enhanced diffusion, and dispersion which are typical of such "hyperbolic conservation laws with relaxation." A novel modified equation analysis is also used to show how the spurious checkerboard mode behaves and can be controlled. Numerical experiments for some nonlinear one-dimensional problems and a two-dimensional problem demonstrate that the behavior of the scheme deduced from a simple model problem has general validity.

Key words. box scheme, reactive flow problems, modified equation analysis

AMS subject classifications. 65M06, 65M12, 35L65

DOI. 10.1137/030601910

1. Introduction.

Our objectives in this paper are twofold. First, we wish to develop effective numerical methods for the differential equations which arise in the transport of reacting substances by groundwater flow of the general form

$$\mathbf{u}_t + \operatorname{div} \mathcal{F}(\mathbf{u}) = \mathbf{R}(\mathbf{u}), \quad \mathbf{u}(\mathbf{x}, t) : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^p, \tag{1.1}$$

where the fluxes $\mathcal{F}$ represent the groundwater transport and the terms $\mathbf{R}$ on the right represent the chemical reactions. In typical problems the reaction rates vary widely, and some may be on a time scale much shorter than the advection velocities occurring in the transport terms. The result is the phenomenon of "reduced speed" whereby the transport occurs at a speed very much lower than the advection velocity; this precludes the use of simple explicit numerical schemes for modeling the transport.

Such situations are common in other applications, such as river-flow modeling. And there the standard numerical remedy is to use the box scheme, often called the Preissmann box scheme; see [6] and [15]. The name of the scheme derives from its construction by integrating the differential equation over a box in $(\mathbf{x}, t)$-space, applying the divergence theorem to the differential terms, and approximating all the integrals by quadrature formulae based on values at the corners of the box. This gives a very compact difference scheme on a uniform mesh (or a finite volume scheme on a general mesh) which has natural conservation properties. It is also implicit and unconditionally stable. For it to be fully effective, two steps are necessary. First, an efficient marching scheme is needed to solve the implicit system of equations by sweeping out from the given initial and boundary conditions. Second, the time step needs to be chosen so that the dominant flow of "information" corresponds roughly to a main diagonal of the discretization box. Then the remaining disadvantage of the scheme is the presence of spurious oscillations in the numerical solution, which have to be controlled in some way.

*Received by the editors November 19, 2003; accepted for publication (in revised form) January 10, 2005; published electronically January 6, 2006.
http://www.siam.org/journals/sisc/27-4/60191.html

†Institute of Applied Mathematics, University of British Columbia, Vancouver, BC, V6T 1Z2, Canada (sarah@iam.ubc.ca).

‡Department of Mathematical Sciences, University of Bath, Claverton Down, BA2 7AY, UK (bill.morton@comlab.ox.ac.uk, as@maths.bath.ac.uk).

Thus our second objective is to explore the properties of the box scheme and show how it may best be used for the problems under consideration. In the next section we begin with the study of a linear model problem in one space dimension in order to introduce most of the key ideas. We show by an asymptotic analysis how the reaction terms lead to the reduced speed, and we also introduce diffusion and dispersion into the simple model. In section 3 we consider the box scheme for this problem—its stability by an energy method and its modified equation analysis. For smooth approximations it reproduces the reduced speed, diffusion, and dispersion of the differential equation, with the last achieved by choosing the time step so that the CFL number based on the reduced speed is set to unity; a novel modified equation analysis of the spurious oscillatory modes shows how they move at a completely different speed and how they can be damped by applying a general θ-weighting to the advective terms with $\theta = \frac{1}{2} + \mathcal{O}(\Delta t)$. To demonstrate that these key properties extend to more general problems, in section 4 the method is applied to one-dimensional problems with nonlinear reaction terms and in section 5 to a two-dimensional problem with an incompressible flow field and a nonuniform mesh.

To conclude this introduction, we should point out that the standard schemes in widespread use for modeling reactive transport in groundwater flow make use of operator-splitting techniques—see, for example, [8], [19], [7], [24], [2], [3], and [1]. That is, one solves alternately the equilibrium equation system $\mathbf{R}(\mathbf{u}) = 0$ and the transport equation. There are several disadvantages of such a procedure, as has been pointed out in [3] and [7]. These include the fact that the transport can occur at the incorrect speed unless the reaction rates are sufficiently fast, an $\mathcal{O}(\Delta t)$ error arises due to the splitting, and stability constraints determined by the velocity of the transport equation are possible. Nonetheless, splitting schemes are computationally attractive and form the basis for several software packages, notably PARSIM [1] and MINTRAN [19].

2. A linear model in one space dimension. The simplest model that displays the key features we wish to study is for a single chemical component being transported through rock by steady groundwater flow and reacting with the rock. Let c be the concentration of the chemical in the water, s the concentration in the rock, λ the rate of adsorption into the rock, μ the rate of desorption from the rock into the water, and V the groundwater flow velocity. Then we have the following pair of equations, with λ, μ, and V all positive,

$$\frac{\partial c}{\partial t} + V\frac{\partial c}{\partial x} = \mu s - \lambda c, \tag{2.1}$$

$$\frac{\partial s}{\partial t} = \lambda c - \mu s, \tag{2.2}$$

to hold on $(x, t) \in [0, \infty) \times [0, \infty)$.

In a typical problem the concentrations are zero initially, but then there is a release of the chemical at one point over a short period of time, followed by its dispersal through the rock. Thus we shall take as initial and boundary conditions

$$c(x, 0) = s(x, 0) = 0 \text{ for } x \geq 0, \quad c(0, t) = h(t), \text{ given, for } t > 0; \tag{2.3}$$

then $s(0, t)$ is found from integrating (2.2) with respect to t.

The general behavior of this system is easy to deduce. At each point in the rock the concentration s is obtained by integrating (2.2), while the concentration c

is advected to the right with velocity V but decays through the term $-\lambda c$ and is augmented by μs. For this linear model the result can be expressed in the form of two integral equations which can be solved by the use of Laplace transforms—see [16] and [14] for details. Examples of the resulting (exact) solution are shown in Figure 1, which can be used to judge the accuracy of our numerical methods; note that, in common with the usual practice in this field, we plot the solution against time at a given point in space. Key features of the solutions obtained for different values of λ and μ are the speed of propagation of the initial pulse and its change of form. Some of these features can be deduced by simple, generally applicable, asymptotic analysis, the validity of which can again be checked for this model problem against the Laplace transform solution.

2.1. Asymptotic analysis. For large values of λ and μ, one may apply the analysis of [13] (see also [4]) for [illegible]. These are defined as systems of the form (1.1) in which there exists a constant matrix Q with rank $r < p$ such that $Q\mathbf{R}(\mathbf{u}) = 0$ and $\mathbf{R} = \epsilon^{-1}\tilde{\mathbf{R}}$, where ϵ is a small parameter. Then this defines an [illegible] given by the mapping $\mathcal{M} : \mathbf{v} \in \mathbb{R}^r \to \mathbf{u}^e \in \mathbb{R}^p$, on which $\tilde{\mathbf{R}}(\mathbf{u}^e) = 0$. In one space dimension, where we write (1.1) as $\mathbf{u}_t + \mathbf{f}_x = \mathbf{R}$ with $\mathbf{f} : \mathbb{R}^p \to \mathbb{R}^p$, we can define a new flux $\mathbf{g} : \mathbb{R}^r \to \mathbb{R}^r$ by $\mathbf{g}(\mathbf{v}) = Q\mathbf{f}(\mathcal{M}\mathbf{v})$; thence, by operating on the equation with the constant matrix Q, we obtain an [illegible]

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{\partial \mathbf{g}(\mathbf{v})}{\partial x} = 0, \tag{2.4}$$

whose solution provides a lowest order approximation to $\mathbf{u}$ as $\epsilon \to 0$.

For the linear model (2.1), (2.2), the equilibrium manifold is given by $\lambda c = \mu s$ with Q the row vector $(1, 1)$. Hence we add the two equations to give

$$s_t + c_t + V c_x = 0, \tag{2.5}$$

which is an important equation in its own right. Then, substituting for s in this equation gives the equilibrium model

$$c_t + \frac{\mu V}{\lambda + \mu} c_x = 0, \tag{2.6}$$

which shows how the reduced speed $V' = \mu V/(\lambda + \mu)$ comes into play in the limit of large λ and μ. Strictly speaking, in the sense of [13], the equilibrium model should be written in terms of $c + s$, but for this linear problem c, s, and $c + s$ all satisfy (2.6) so we use the simpler form.

Further levels of approximation are obtained by expanding the mapping $\mathcal{M}$ in powers of the small parameter, the reciprocals of λ and μ in the present case. From differentiating (2.2) we obtain

$$s_t = (\lambda/\mu)c_t - (1/\mu)s_{tt}, \tag{2.7}$$

and then to the lowest order we can derive the approximation

$$s_{tt} = (\lambda/\mu)c_{tt} - (1/\mu)s_{ttt} \sim (\lambda/\mu)V'^2 c_{xx}. \tag{2.8}$$

Substitution back into the conservation equation (2.5) then gives

$$c_t + \frac{\mu V}{\lambda + \mu} c_x = \frac{\lambda \mu V^2}{(\lambda + \mu)^3} c_{xx}, \tag{2.9}$$

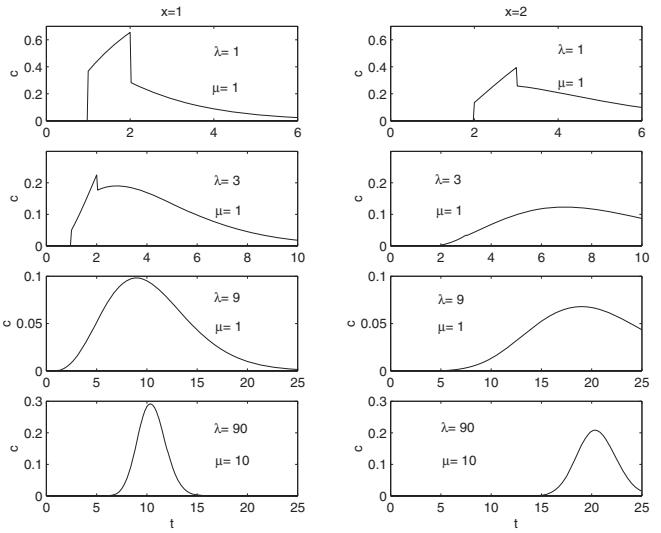


FIG. 1. *The Laplace transform solution* c *at* $x = 1$ *and* $x = 2$ *for* $V = 1$ *for various values of* (λ, μ). *The boundary condition is a square pulse of height and width* 1.

commonly called the [illegible]. Continuing to one order higher introduces a dispersion term to give

$$c_t + \frac{\mu V}{\lambda + \mu} c_x = \frac{\lambda \mu V^2}{(\lambda + \mu)^3} c_{xx} - \frac{\lambda \mu (\lambda - \mu) V^3}{(\lambda + \mu)^5} c_{xxx}. \tag{2.10}$$

From the plots in Figure 1 we can clearly see the effect not only of the reduced speed but also of the diffusion term in the improved model (2.9) and the dispersion term in (2.10). In the four cases shown we have $V = 1$ and the values of λ and μ are $(1, 1)$, $(3, 1)$, $(9, 1)$, and $(90, 10)$. These give values of the reduced velocity $V' = \mu V/(\lambda + \mu)$ equal to $1/2$, $1/4$, $1/10$, and $1/10$, respectively. In Table 2.1 these are compared with estimates of the actual speed of the initial pulse deduced from the solutions in the figures. Note too how the solutions are smoothed and damped, particularly in the last two cases, and in the table the values of the diffusion and dispersion coefficients in the improved equilibrium models are also given.

TABLE 2.1

Table showing observed and reduced speed, and comparing the estimate for $c_{\max}$ *from* (2.12) *with the observed value (at* $x = 2$ *with the speed* $V = 1$*). Also shown are the diffusion and dispersion coefficients in* (2.10).

	$(\lambda, \mu) = (1, 1)$	$(\lambda, \mu) = (3, 1)$	$(\lambda, \mu) = (9, 1)$	$(\lambda, \mu) = (90, 10)$
Observed speed	1.00	0.31	0.11	0.10
Reduced speed	0.50	0.25	0.10	0.10
Max observed c	0.394	0.123	0.068	0.208
$c_{\max}$ (from (2.12))	0.200	0.115	0.067	0.210
Diffusion coefficient	0.125	0.047	0.009	9×10^{-4}
Dispersion coefficient	0	0.006	7.2×10^{-4}	7.2×10^{-6}

We have already referred to the similarity of the present problems to those encountered in river modeling. In a classic paper [11], [12], Lighthill and Whitham carried out an asymptotic analysis of flood waves in which they derived an expression

for the maximum flood height as well as the speed of the wave. If we differentiate (2.1) with respect to t and add to it μ times the conservation equation (2.5), we can eliminate s and obtain the following second order equation for c:

$$c_{tt} + Vc_{tx} + (\lambda + \mu)c_t + \mu V c_x = 0. \tag{2.11}$$

Following the procedure used in [11], [12], and [22] we can apply a Laplace transform to this equation, with the initial and boundary conditions given in (2.3), and obtain an expression for the solution in terms of a contour integral. Then the solution near the dynamic wave-front can be deduced from the behavior of the integrand for large values of the transform variable. In particular, we find that the maximum occurs where $x = V't$ and for large values of $\lambda + \mu$ is given by

$$c_{\max} \sim \frac{1}{2}\left(\frac{(\lambda+\mu)^3}{\pi\lambda\mu t}\right)^{\frac{1}{2}}. \tag{2.12}$$

Values of this expression are also compared with the observed maxima for the four exact solutions in Table 2.1.

3. The box scheme for the linear model. The box scheme became a popular method for approximating the St. Venant equations of river flow through the work of Preissmann and his colleagues [6]. The scheme is also associated with the name of Keller, who applied it to parabolic problems [10] and to two-point boundary value problems [9]. It was also analyzed in the early paper [18] and very recently in [15]. It can be identified as the valuable cell-vertex finite volume method for approximating the Euler equations (and also the Navier–Stokes equations) of steady transonic aerodynamics (see, for example, [5]), and its multisymplectic structure has been recognized and has led to its application to the KdV equation [23]. When applied to these cases, a valuable feature of the scheme, which follows from the derivation outlined in the introduction, is the accurate and compact discretization that it yields on a nonuniform quadrilateral mesh. However, we shall concentrate here on its finite difference formulation on a uniform rectangular mesh, because we can demonstrate that its asymptotic behavior closely mirrors that of the differential system as outlined in the previous section. We shall also begin by using the simple trapezoidal rule weighting in time, although we shall show later, as is well known, that the more general θ-weighting is important in controlling the spurious checkerboard mode.

Suppose we have a uniform mesh over the rectangle $0 \le t \le T$, $0 \le x \le L$, with time step $\Delta t = T/N$ and spatial step $\Delta x = L/J$ for a given N and J. Let U_j^n denote the numerical approximation of u at $(x_j, t^n) := (j\Delta x, n\Delta t)$ for $j = 0, 1, \ldots, J$ and $n = 0, 1, \ldots, N$. On the mesh we define the finite difference operators

$$\delta_x U_{j+\frac{1}{2}}^{n+\frac{1}{2}} = U_{j+1}^{n+\frac{1}{2}} - U_j^{n+\frac{1}{2}}, \quad \mu_x U_{j+\frac{1}{2}}^{n+\frac{1}{2}} = \tfrac{1}{2}\left(U_{j+1}^{n+\frac{1}{2}} + U_j^{n+\frac{1}{2}}\right) \tag{3.1}$$

and, similarly, δ_t and μ_t. Then the box scheme consistently uses the operator $\mu_x\delta_t$ to approximate ∂_t and, similarly, $\mu_t\delta_x$ to approximate ∂_x in a partial differential equation (PDE). If we apply this scheme to the conservation law (2.6), we obtain

$$\mu_x\delta_t C_{j+\frac{1}{2}}^{n+\frac{1}{2}} + \mu_x\delta_t S_{j+\frac{1}{2}}^{n+\frac{1}{2}} + p\mu_t\delta_x C_{j+\frac{1}{2}}^{n+\frac{1}{2}} = 0, \text{ where } p = V\Delta t/\Delta x \tag{3.2}$$

and p denotes the CFL number. This is a very compact scheme which uses four neighboring values of the unknowns, and it is actually , , , as it involves two points

at the new time level. However, for appropriate initial and boundary conditions such as those given in (2.3), the discrete equations can be solved by marching away from a boundary. To complete the approximation we have to discretize the reaction equation (2.2), which is an ordinary differential equation in time, to which we apply the one-dimensional box scheme (i.e., the trapezoidal rule) to obtain

$$\delta_t S_j^{n+\frac{1}{2}} = \mu_t\big(\lambda' C_j^{n+\frac{1}{2}} - \mu' S_j^{n+\frac{1}{2}}\big), \tag{3.3}$$

where $\lambda' := \lambda\Delta t$ and $\mu' := \mu\Delta t$. Note that this equation needs no boundary condition as its application for $j = 0$ integrates the boundary data for c up the t-axis.

Equations (3.2) and (3.3) can be written out as

$$\begin{aligned}&(1+p)C_{j+1}^{n+1} + (1-p)C_j^{n+1} - (1-p)C_{j+1}^n - (1+p)C_j^n\\ &\quad + \big(S_{j+1}^{n+1} - S_{j+1}^n\big) + \big(S_j^{n+1} - S_j^n\big) = 0,\end{aligned} \tag{3.4}$$

$$S_{j+1}^{n+1} = \frac{\frac{1}{2}\lambda'}{1+\frac{1}{2}\mu'}\big(C_{j+1}^{n+1} + C_{j+1}^n\big) + \left(\frac{1-\frac{1}{2}\mu'}{1+\frac{1}{2}\mu'}\right) S_{j+1}^n, \tag{3.5}$$

and they are applied for $j = 0, \ldots, (J-1)$ and $n = 0, \ldots, (N-1)$. This is an explicit scheme when the conditions of (2.3) are used: from (3.5), S_{j+1}^{n+1} can be substituted into (3.4) to obtain an explicit formula to solve for C_{j+1}^{n+1}, by marching away from the boundary.

3.1. Stability analysis. The box scheme is centered in x and t and the trapezoidal scheme is centered in t, so all the odd order terms in a local truncation error analysis cancel and the scheme is second order accurate. Moreover, a Fourier analysis of the method shows that it is Lax–Richtmyer stable for all Δx and Δt. (See [14] for details.) Unfortunately, the scheme suffers from the notorious checkerboard mode $(-1)^{j+n}$. This is clearly a solution of (3.2), and when the reaction equation approximation (3.3) is space-averaged (a form we shall sometimes use in the analysis below), it is a solution of that too. Thus it is clearly a spurious solution of the complete system. However, applying the unaveraged (3.3) along the t-axis provides special boundary data which minimizes its effect, so that it is then mainly initiated by nonsmooth boundary data for c.

A more illuminating analysis of stability is provided by an energy analysis. We consider first the differential equations (2.1) and (2.2): multiplying (2.1) by c and (2.2) by s and integrating in x over $(0, \infty)$ gives

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^\infty \frac{1}{2}c^2\,\mathrm{d}x + V\int_0^\infty \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{1}{2}c^2\right)\mathrm{d}x = -\int_0^\infty c(\lambda c - \mu s)\,\mathrm{d}x, \tag{3.6}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^\infty \frac{1}{2}s^2\,\mathrm{d}x = \int_0^\infty s(\lambda c - \mu s)\,\mathrm{d}x. \tag{3.7}$$

We assume the boundary data is zero for $t > t^*$ and $c(x,t) \to 0$ as $x \to \infty$ for all finite t, so the second integral in (3.6) is then zero; hence if we multiply (3.6) and (3.7) by λ and μ, respectively, and add the results, we deduce that for $t > t^*$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^\infty \frac{1}{2}\left(\lambda c^2 + \mu s^2\right)\,\mathrm{d}x = -\int_0^\infty (\lambda c - \mu s)^2\,\mathrm{d}x \le 0. \tag{3.8}$$

To obtain a similar result for the box scheme we first introduce $R := \lambda C - \mu S$. Then we rewrite the box scheme of (3.2) and (3.3) by substituting for S in the first

and applying the space average to the second to obtain

$$\mu_x\big[C^{n+1}_{j+\frac12}-C^n_{j+\frac12}\big]+\tfrac12 p\delta_x\big[C^{n+1}_{j+\frac12}+C^n_{j+\frac12}\big]=-\tfrac12\Delta t\mu_x\big[R^{n+1}_{j+\frac12}+R^n_{j+\frac12}\big], \tag{3.9}$$

$$\mu_x\big[S^{n+1}_{j+\frac12}-S^n_{j+\frac12}\big]=\tfrac12\Delta t\mu_x\big[R^{n+1}_{j+\frac12}+R^n_{j+\frac12}\big]. \tag{3.10}$$

For convenience, we introduce the notation $\bar C^n := \mu_x C^n_{j+1/2}$, etc. and $\langle \bar C^n, \bar S^n\rangle_2$ for a typical l^2 inner product. To emulate the procedure carried out in the continuous case, we multiply (3.9) and (3.10) by $\bar C^{n+1}+\bar C^n$ and $\bar S^{n+1}+\bar S^n$, respectively, and sum over the mesh. Then from the identity $\langle \mu_x W, \delta_x W\rangle_2 \equiv \frac12(W_J^2-W_0^2)$, and assuming that $t^n > t^*$ so that $C^n_0=0$, we obtain

$$\begin{aligned}\|\bar C^{n+1}\|_2^2-\|\bar C^n\|_2^2+\tfrac12 V\Delta t\big[C^{n+1}_J+C^n_J\big]^2&=-\tfrac12\Delta t\langle \bar C^{n+1}+\bar C^n,\bar R^{n+1}+\bar R^n\rangle_2\\ \|\bar S^{n+1}\|_2^2-\|\bar S^n\|_2^2&=\tfrac12\Delta t\langle \bar S^{n+1}+\bar S^n,\bar R^{n+1}+\bar R^n\rangle_2.\end{aligned}$$

Multiplying the first expression by λ and the second by μ, and adding gives

$$\begin{aligned}&\lambda\|\bar C^{n+1}\|_2^2+\mu\|\bar S^{n+1}\|_2^2-\big[\lambda\|\bar C^n\|_2^2+\mu\|\bar S^n\|_2^2\big]\\ &\qquad=-\tfrac12\Delta t\|\bar R^{n+1}+\bar R^n\|_2^2-\tfrac12 V\Delta t\big[C^{n+1}_J+C^n_J\big]^2\le 0.\end{aligned} \tag{3.11}$$

Hence we have established the strong stability of the [illegible] $\mu_x C$ and $\mu_x S$.

However, this does not cover the spurious mode $(-1)^{j+n}$ oscillations. We could treat these by mapping between the cell averages and the nodal values, which we can do since we have prescribed the boundary value, setting it to zero for $t^n > t^*$: every nodal value can then be obtained from the averages by a recurrence ($C^n_1 = 2\bar C^n_{1/2}$, $C^n_2 = 2(\bar C^n_{3/2}-\bar C^n_{1/2})$, etc.), which defines an oscillation matrix, from the l^2 norm of which we deduce that $\|C^n\|_2 \le \sqrt{2J(J+1)}\|\bar C^n\|_2$. Hence there is a potential linear growth which is consistent with the well-known phenomenon that imposing inappropriate boundary conditions may cause a linear growth in the oscillatory mode [17]. It is more useful, though, to consider these modes by means of a modified equation analysis; by this means we can also show how they can be damped by introducing a θ-weighting in the time averaging for the spatial derivative. This is done next.

3.2. Modified equation analysis. The use of a modified equation analysis is a well-known technique for understanding the properties of a finite difference approximation of a given PDE. Ignoring the highest frequencies, the analysis yields an asymptotic series of PDEs that can provide a more accurate representation of the behavior of the solution of the difference scheme. Such an analysis can be especially helpful in giving a qualitative understanding of features like diffusion and dispersion in a numerical scheme (see [21] and [15]).

The finite difference operators defined by (3.1) can be expanded in terms of differential operators using Taylor series expansions, i.e.,

$$\delta_x = \Delta x\big[1+\tfrac{1}{24}\Delta x^2\partial_x^2+\mathcal{O}(\Delta x^4)\big]\partial_x,\quad \mu_x = 1+\tfrac18\Delta x^2\partial_x^2+\mathcal{O}(\Delta x^4), \tag{3.12}$$

with similar expansions for δ_t and μ_t. Subsequently, we shall use the notation "..." to mean either $\mathcal{O}(\Delta x^4)$ or $\mathcal{O}(\Delta t^4)$. Strictly speaking, we need to replace the discrete variables by suitable prolongations (for example, using Fourier expansions or high order polynomials) before we can apply the differential operators to them, but we shall not make that distinction in our notation here.

If we define the operators

$$\mathcal{D}_x := \frac{\delta_x \mu_x^{-1}}{\Delta x}, \quad \mathcal{D}_t := \frac{\delta_t \mu_t^{-1}}{\Delta t}, \tag{3.13}$$

then we readily obtain

$$\mathcal{D}_x = \left(1 - \tfrac{1}{12}\Delta x^2 \partial_x^2 + \ldots\right) \partial_x, \tag{3.14}$$

while for the corresponding expansion for $\mathcal{D}_t$ it is convenient to invert the difference operator bracket to obtain

$$\partial_t = \left(1 + \tfrac{1}{12}\Delta t^2 \partial_t^2 + \ldots\right) \mathcal{D}_t. \tag{3.15}$$

Moreover, we can rewrite (3.2) and (3.3) as

$$\mathcal{D}_t C + \mathcal{D}_t S + V \mathcal{D}_x C = 0, \quad (\mathcal{D}_t + \mu) S = \lambda C. \tag{3.16}$$

Since all the operators commute, we can eliminate S to obtain a second order equation for C,

$$[\mathcal{D}_t^2 + (\lambda + \mu + V\mathcal{D}_x)\mathcal{D}_t + \mu V \mathcal{D}_x] C = 0 \tag{3.17}$$

(cf. (2.11)). Because the model is linear this equation also holds for S and for the total concentration $C + S$. It can also be interpreted as a quadratic for $\mathcal{D}_t$, with the two roots corresponding to two wave modes.

The solutions to the quadratic equation are functions of the operator $\mathcal{D}_x$: if we expand these as power series, it is easily seen that we get

$$\mathcal{D}_t = -\frac{\mu V}{\lambda + \mu}\mathcal{D}_x + \frac{\mu \lambda V^2}{(\lambda + \mu)^3}\mathcal{D}_x^2 - \frac{\mu\lambda(\lambda - \mu)V^3}{(\lambda + \mu)^5}\mathcal{D}_x^3 + \ldots \tag{3.18}$$

for the positive root, which corresponds to the main advected wave, and

$$\mathcal{D}_t = -(\lambda + \mu) - \frac{\lambda V}{\lambda + \mu}\mathcal{D}_x - \frac{\mu \lambda V^2}{(\lambda + \mu)^3}\mathcal{D}_x^2 + \frac{\mu\lambda(\lambda - \mu)V^3}{(\lambda + \mu)^5}\mathcal{D}_x^3 - \ldots \tag{3.19}$$

for the negative root, which corresponds to a damped wave which is also advected but with a different speed.

The correspondence of the leading terms in (3.18) to those in the improved equilibrium models (2.9) and (2.10) is very clear: it follows, of course, from the consistent way that the box scheme discretizes the differential equation, which is the basis of its multisymplectic properties already referred to. But now we can use the expansion in (3.14) for all the terms in $\mathcal{D}_x$. Moreover, we can apply the similar expansion in (3.15) to $\mathcal{D}_t$ on the left and obtain, after a straightforward calculation and applying the operators to C, the following expansion (to second order in Δx) for C_t:

$$\begin{aligned} C_t = &-\frac{\mu V}{\lambda + \mu} C_x + \frac{\mu \lambda V^2}{(\lambda + \mu)^3} C_{xx} \\ &- \frac{\mu V}{\lambda + \mu}\left(\frac{\lambda(\lambda - \mu)V^2}{(\lambda + \mu)^4} C_{xxx} + \frac{1}{12}\Delta t^2 C_{xtt} - \frac{1}{12}\Delta x^2 C_{xxx}\right) + \ldots . \end{aligned} \tag{3.20}$$

This is in the usual form found in the first stage of a conventional modified equation analysis, namely, with some higher order t-derivatives on the right-hand side;

these can then be eliminated in favor of x-derivatives by successive differentiation and substitution from this equation. Alternatively, we can use finite difference operator calculus to write $\mathcal{D}_t = (\frac{1}{2}\Delta t)^{-1}\tanh(\frac{1}{2}\Delta t\partial_t)$, take the inverse of this relation, and, from the Taylor expansion of $\tanh^{-1}$, obtain the relation

$$\partial_t = \left(1 + \tfrac{1}{12}\Delta t^2\mathcal{D}_t^2 + \ldots\right)\mathcal{D}_t, \tag{3.21}$$

into which we can substitute (3.18). Either procedure produces

$$\begin{aligned} C_t = &-V'C_x + \frac{\lambda\mu V^2}{(\lambda+\mu)^3}C_{xx} \\ &- V'\left(\frac{\lambda(\lambda-\mu)V^2}{(\lambda+\mu)^4} + \frac{1}{12}\left[V'^2\Delta t^2 - \Delta x^2\right]\right)C_{xxx} + \ldots, \end{aligned} \tag{3.22}$$

where $V' = V\mu/(\lambda+\mu)$ is the reduced speed. Similarly and to the same order, from (3.19) we obtain for the damped wave

$$\begin{aligned} C_t = &-(\lambda+\mu)\left(1 + \frac{1}{12}(\lambda+\mu)^2\Delta t^2\right)C - \bar{V}\left(1 + \frac{1}{4}(\lambda+\mu)^2\Delta t^2\right)C_x \\ &- \bar{V}\left(\frac{V'}{\lambda+\mu} + \frac{1}{4}(\lambda+\mu)V\Delta t^2\right)C_{xx} \\ &+ \bar{V}\left(\frac{V'(\bar{V}-V')}{(\lambda+\mu)^2} - \frac{1}{12}(V^2 + VV' + V'^2)\Delta t^2 + \frac{1}{12}\Delta x^2\right)C_{xxx}\ldots, \end{aligned} \tag{3.23}$$

where $\bar{V} = V\lambda/(\lambda+\mu)$ is the basic speed of this wave.

For the main advective wave, the lowest order discretization terms appear in the dispersive term—consistent with the second order accuracy. But the key point to note is that these become zero when the mesh ratio is based on the reduced speed, i.e., when $V'\Delta t = \Delta x$; the dispersive coefficient then reduces to that obtained in section 2.1 for the differential equation. The improved accuracy obtained with this choice is demonstrated in the numerical results given in section 3.5. Note, however, that for the damped wave the dominant effect of the discretization is to increase the damping, although it similarly affects the advective speed and the diffusion term.

The above modified equation analysis, with its emphasis on Taylor expansions of smooth solutions, is unsuitable for studying high frequency modes and in particular the checkerboard mode. The importance of this mode and some of the ways in which it can affect the result of using the box scheme are demonstrated in Figures 2, 3, and 4; there we see that it is greatly enhanced by discontinuous boundary data and that its behavior is markedly different for different values of λ and μ. In the next section we therefore present an extension of the above analysis to describe this behavior, pointing to some of the ways in which the mode may be controlled.

3.3. Separating the smooth and oscillatory numerical solution. Suppose we consider a Fourier mode $C_j = \mathrm{e}^{\mathrm{i}kj\Delta x}$ substituted into the box scheme. The Fourier symbols of the spatial difference operators μ_x and δ_x appearing in (3.2) are given by the relations

$$\mu_x C_{j+\frac{1}{2}} = (\cos\tfrac{1}{2}k\Delta x)C_{j+\frac{1}{2}}, \quad \delta_x C_{j+\frac{1}{2}} = (2\mathrm{i}\sin\tfrac{1}{2}k\Delta x)C_{j+\frac{1}{2}}; \tag{3.24}$$

for small values of $k\Delta x$, Taylor expansions of these symbols correspond to the expansions in (3.12) when the Fourier symbol $\mathrm{i}k$ of ∂_x is substituted in the latter. However,

the checkerboard mode has $k\Delta x = \pi$ and for modes near this frequency the symbol has a very different behavior. Indeed, if we write $k\Delta x = \pi + k'\Delta x$ where $k'\Delta x$ is small, then we get instead

$$\mu_x C_{j+\frac{1}{2}} = (-\sin\tfrac{1}{2}k'\Delta x)C_{j+\frac{1}{2}}, \quad \delta_x C_{j+\frac{1}{2}} = (2\mathrm{i}\cos\tfrac{1}{2}k'\Delta x)C_{j+\frac{1}{2}}. \tag{3.25}$$

These expressions can then be expanded in powers of $k'\Delta x$. Moreover, if we write $C_j = (-1)^j C_j^o$, we get from (3.25) and (3.24)

$$\mu_x C_{j+\frac{1}{2}} = -\sin\tfrac{1}{2}k'\Delta x(-1)^{j+\frac{1}{2}}C^o_{j+\frac{1}{2}} = (-1)^{j+1}\tfrac{1}{2}\delta_x C^o_{j+\frac{1}{2}}, \tag{3.26}$$

and similarly

$$\delta_x C_{j+\frac{1}{2}} = (-1)^{j+1}2\mu_x C^o_{j+\frac{1}{2}}. \tag{3.27}$$

These considerations prompt the following approach.

An arbitrary mesh function can be represented by an expansion in Fourier modes for which $|k\Delta x| \le \pi$. Suppose we split this range at $\pi/2$ for an expansion of both C and S before substitution in the box scheme. Suppose, moreover, that there is little interaction between the two sets of modes. Then the behavior of the smooth set of modes is described by the modified equation analysis of the previous subsection. Now we use the relations (3.26) and (3.27) and an assumption that C^o and S^o are smooth to develop a modified equation analysis of the oscillatory modes and, in particular, the checkerboard mode. Thus we write C_j^n as a sum of smooth and oscillatory components,

$$C_j^n \equiv (C^s)_j^n + (-1)^{j+n}(C^o)_j^n, \tag{3.28}$$

and treat S similarly. Then, with similar relations to (3.26) and (3.27) holding for the time difference operators, (3.2) and (3.3) become, for the oscillatory part, after cancellation of the factor $(-1)^{j+n}$,

$$\delta_x\mu_t(C^o + S^o) + p\delta_t\mu_x C^o \approx 0, \quad -2\mu_t S^o \approx -\tfrac{1}{2}\delta_t\left(\lambda' C^o - \mu' S^o\right). \tag{3.29}$$

We can write these in terms of the operators (3.13) to obtain

$$\mathcal{D}_x(C^o + S^o) + p_2\mathcal{D}_t C^o \approx 0, \quad 4S^o \approx \mathcal{D}_t\left(\lambda'' C^o - \mu'' S^o\right), \tag{3.30}$$

where $p_2 := V(\Delta t/\Delta x)^2$, $\lambda'' := \lambda\Delta t^2$, and $\mu'' := \mu\Delta t^2$.

As in the treatment of the smooth part in the previous subsection, we can eliminate S^o to obtain a second order equation for C^o,

$$\left[\mu'' p_2\mathcal{D}_t^2 + (4p_2 + (\lambda'' + \mu'')\mathcal{D}_x)\,\mathcal{D}_t + 4\mathcal{D}_x\right] C^o = 0. \tag{3.31}$$

Then we rewrite this so that it is precisely of the form of (3.17), namely,

$$\left[\mathcal{D}_t^2 + [(4/\mu'') + ((\lambda+\mu)/\mu p_2)\mathcal{D}_x]\mathcal{D}_t + (4/\mu'' p_2)\mathcal{D}_x\right] C^o = 0. \tag{3.32}$$

To simplify application to the present case of the result proved in section 3.2 for that equation, and also to allow for generalization of the analysis to the general weighted scheme in the next section, we restate the main derivation as a lemma; its proof consists of a straightforward manipulation, which we omit.

LEMMA 3.1. $\mathcal{D}_t$ $\mathcal{D}_x$
(3.13) U $\alpha > \beta$

$$\left[(\mathcal{D}_t+\alpha)(\mathcal{D}_t+\beta)+(a\mathcal{D}_t^2+b\mathcal{D}_t+c)\mathcal{D}_x\right]U=0. \tag{3.33}$$

U

(3.34)

$$D_t=\sum_{j=0}^{3}(-a\mathcal{D}_x)^j\left\{-\beta-\tfrac{1}{2}b\mathcal{D}_x+\tfrac{1}{2}(\alpha-\beta)[d\mathcal{D}_x+\tfrac{1}{2}(e-d^2)(1-d\mathcal{D}_x)\mathcal{D}_x^2]\right\}+\ldots,$$

$$d=[(\alpha+\beta)b-2(\alpha\beta a+c)]/(\alpha-\beta)^2,\quad e=(b^2-4ac)/(\alpha-\beta)^2;$$

α β

Hence, by comparing the coefficients in (3.32) with those in the lemma, we deduce that the main advected wave for the oscillatory mode C^o is described by the following expansion:

$$\begin{aligned}C_t^o=&-\frac{V}{p^2}C_x^o+\left(\frac{\lambda\Delta t^2V^2}{4p^4}\right)C_{xx}^o\\&-\frac{V}{p^2}\left(\frac{\lambda(\lambda-\mu)\Delta t^4V^2}{16p^4}+\frac{\Delta x^2}{12p^2}\left[1-p^2\right]\right)C_{xxx}^o+\ldots.\end{aligned} \tag{3.35}$$

The first coefficient here gives the speed at which the envelope of a set of checkerboard oscillations move. It is easy to check that this speed V/p^2 is in fact just the group velocity of the box scheme applied to the linear advection equation when one sets $k\Delta x=\pi$. Note that it is independent of λ and μ. Also, it is quite small for a reasonably large choice of CFL number. The second, damping coefficient is proportional to λ, and it depends on the mesh through a factor $\Delta x^4/\Delta t^2$. Note that for $\lambda=\mu$ the dispersion coefficient is positive for $p>1$ and negative for $p<1$. We shall see the effect of these terms in the numerical experiments presented in section 3.5.

3.4. The weighted box scheme. The traditional method of controlling the checkerboard mode is to introduce θ-weighting in the time integration (see [6]). In this section we describe this scheme and modify the analysis given above to cover this change. We confine the change to the transport equation and so obtain, instead of (3.2),

$$\mu_x\delta_tC_{j+\frac{1}{2}}^{n+\frac{1}{2}}+\mu_x\delta_tS_{j+\frac{1}{2}}^{n+\frac{1}{2}}+p\theta_t\delta_xC_{j+\frac{1}{2}}^{n+\frac{1}{2}}=0, \tag{3.36}$$

where we define the θ-averaging operator as

$$\theta_tC_j^{n+\frac{1}{2}}=\theta C_j^{n+1}+(1-\theta)C_j^n. \tag{3.37}$$

The scheme is second order accurate provided $\theta=\frac{1}{2}+O(\Delta t)$, and a Fourier analysis shows it is Lax–Richtymer stable for all Δx and Δt provided $\theta\geq\frac{1}{2}$; we shall therefore make this assumption in all that follows.

We first reconsider the energy analysis, making use of the identities and notation used earlier. As implied above, we continue to use the trapezoidal weighting for the source term; thus the equations become

$$\mu_x \delta_t C^{n+\frac{1}{2}} + p\theta_t \delta_x C^{n+\frac{1}{2}} = -\Delta t \mu_x \mu_t R^{n+\frac{1}{2}}, \tag{3.38}$$

$$\mu_x \delta_t S^{n+\frac{1}{2}} = \Delta t \mu_x \mu_t R^{n+\frac{1}{2}}, \tag{3.39}$$

where the discretization of the reaction equation has again been space-averaged.

Following the same procedure as before, we multiply (3.38) and (3.39) by $\theta_t \bar{C}^{n+1/2}$ and $\theta_t \bar{S}^{n+1/2}$, respectively, and then sum the resulting equations over j. The advection inner product collapses and, with the assumption that $t^n > t^*$, the boundary term at $j = 0$ vanishes and we obtain

$$\begin{aligned} &\theta \|\bar{C}^{n+1}\|_2^2 + (1-2\theta)\langle \bar{C}^{n+1}, \bar{C}^n\rangle_2 - (1-\theta)\|\bar{C}^n\|_2^2 \\ &\qquad + \tfrac{1}{2} V \Delta t \big[\theta_t C_J^{n+\frac{1}{2}}\big]^2 = -\Delta t \langle \theta_t \bar{C}^{n+\frac{1}{2}}, \mu_t \bar{R}^{n+\frac{1}{2}}\rangle_2, \end{aligned} \tag{3.40}$$

$$\begin{aligned} &\theta \|\bar{S}^{n+1}\|_2^2 + (1-2\theta)\langle \bar{S}^{n+1}, \bar{S}^n\rangle_2 - (1-\theta)\|\bar{S}^n\|_2^2 \\ &\qquad = \Delta t \langle \theta_t \bar{S}^{n+\frac{1}{2}}, \mu_t \bar{R}^{n+\frac{1}{2}}\rangle_2. \end{aligned} \tag{3.41}$$

Now from the Cauchy–Schwarz inequality we have, for $\theta \geq \frac{1}{2}$,

$$(2\theta - 1)\langle \bar{A}^{n+1}, \bar{A}^n\rangle_2 \leq \big(\theta - \tfrac{1}{2}\big)\big[\|\bar{A}^{n+1}\|_2^2 + \|\bar{A}^n\|_2^2\big].$$

Hence the terms on the left of (3.40) can be greatly simplified to give

$$\tfrac{1}{2}\|\bar{C}^{n+1}\|_2^2 - \tfrac{1}{2}\|\bar{C}^n\|_2^2 \leq -\Delta t \langle \theta_t \bar{C}^{n+\frac{1}{2}}, \mu_t \bar{R}^{n+\frac{1}{2}}\rangle_2. \tag{3.42}$$

Similarly, from (3.41), we get

$$\tfrac{1}{2}\|\bar{S}^{n+1}\|_2^2 - \tfrac{1}{2}\|\bar{S}^n\|_2^2 \leq \Delta t \langle \theta_t \bar{S}^{n+\frac{1}{2}}, \mu_t \bar{R}^{n+\frac{1}{2}}\rangle_2. \tag{3.43}$$

Combining these results as in (3.11) we obtain

$$\begin{aligned} &\lambda \|\bar{C}^{n+1}\|_2^2 + \mu \|\bar{S}^{n+1}\|_2^2 - \lambda \|\bar{C}^n\|_2^2 - \mu \|\bar{S}^n\|_2^2 \\ &\qquad \leq -\Delta t \big[\theta \|\bar{R}^{n+1}\|_2^2 + (1-\theta)\|\bar{R}^n\|_2^2 + \langle \bar{R}^{n+1}, \bar{R}^n\rangle_2\big]. \end{aligned} \tag{3.44}$$

Now we can rewrite the terms on the right by setting $\theta = \frac{1}{2}(1+\xi)$, and apply the Cauchy–Schwarz inequality again, to get

$$\begin{aligned} &\tfrac{1}{2}\big[(1+\xi)\|\bar{R}^{n+1}\|_2^2 + (1-\xi)\|\bar{R}^n\|_2^2 + 2\langle \bar{R}^{n+1}, \bar{R}^n\rangle_2\big] \\ &\qquad \geq \tfrac{1}{2}\xi\big[\|\bar{R}^{n+1}\|_2^2 - \|\bar{R}^n\|_2^2\big]. \end{aligned} \tag{3.45}$$

Substituting this result in (3.44) then yields the required stability of the cell averages, since we have $\xi \geq 0$, in terms of the energy $\lambda\|\bar{C}^n\|_2^2 + \mu\|\bar{S}^n\|_2^2 + \frac{1}{2}\xi\Delta t\|\bar{R}^n\|_2^2$. We should also note at this point how the weighting affects the propagation of the oscillatory mode into the interior from the boundary data—and hence the mapping from the cell averages to the nodal values. In applying a Godunov–Ryabenkii stability analysis, we can substitute a mode of the form $\alpha^n(-1)^j$ into (3.36); only the advection term contributes and yields the result $\alpha = 1 - 1/\theta$. So this mode is damped if $\theta > \frac{1}{2}$.

However, it is the modified equation analysis which gives the most information on the behavior of the weighted box scheme. The averaging operator θ_t can be written as

$$\theta_t = \mu_t + \big(\theta - \tfrac{1}{2}\big)\delta_t, \tag{3.46}$$

so we can introduce the operator

$$\mathcal{M}_t := \mu_t^{-1}\theta_t = 1 + \left(\theta - \tfrac{1}{2}\right)\Delta t\mathcal{D}_t. \tag{3.47}$$

The modified equation analysis can now proceed in exactly the same way as in sections 3.2 and 3.3, with the only changes being the replacement of $\mathcal{D}_x$ by $\mathcal{M}_t\mathcal{D}_x$ and the operator equations for both the smooth waves and the oscillatory waves being in the form covered by Lemma 3.1. For the main advected wave which previously led to the expansion in (3.22), it is a straightforward calculation to find that we get instead

$$\begin{aligned} C_t = &-V'C_x + V'^2\left\{\frac{\lambda}{\mu(\lambda+\mu)} + \left(\theta - \frac{1}{2}\right)\Delta t\right\}C_{xx} \\ &- V'\left\{\frac{\lambda(\lambda-\mu)V^2}{(\lambda+\mu)^4} + \frac{1}{12}\left[V'^2\Delta t^2 - \Delta x^2\right] + 3\frac{\lambda V'^2}{\mu(\lambda+\mu)}\left(\theta - \frac{1}{2}\right)\Delta t\right. \\ &\left. + V'^2\left(\theta - \frac{1}{2}\right)^2\Delta t^2\right\}C_{xxx} + \ldots. \end{aligned} \tag{3.48}$$

On comparing this with the unweighted case, we see that diffusion is enhanced for $\theta > \frac{1}{2}$ as expected, but the advection speed is unaffected; also there are extra terms in the dispersion, which will affect the choice of mesh ratio to minimize the numerical dispersion—note, however, that the term $\left(\theta - \frac{1}{2}\right)^2\Delta t^2$ is negligible if $\theta = \frac{1}{2} + O(\Delta t)$.

The most significant change is in the expansion for the main oscillatory wave, where the leading terms of (3.35) are replaced by

$$\begin{aligned} C_t^o = &-4\frac{\theta - \frac{1}{2}}{\Delta t}(1+\gamma)C^o \\ &- \frac{V}{p^2}\left(\frac{1 - (\lambda+\mu)(\theta - \frac{1}{2})\Delta t}{1 - \mu(\theta - \frac{1}{2})\Delta t}\right)(1+3\gamma)C_x^o + \ldots, \end{aligned} \tag{3.49}$$

where $\gamma = (4/3)(\theta - \frac{1}{2})^2$ and the factors involving this quantity arise from the transformation of an expansion for $\mathcal{D}_t$ into one for ∂_t. Thus even for $\theta = \frac{1}{2} + O(\Delta t)$, exponential damping of the oscillatory modes occurs, while the advective speed is little changed.

3.5. Numerical experiments. In all of the numerical experiments with the linear model problem that we present here, we have taken $V = 1.0$ and, in the early cases, we use the simple scheme with $\theta = \frac{1}{2}$. The plots are generally of the concentration C at a fixed point $x = 1$, as a function of t. Figure 2, in which we take $\Delta x = 0.025$, shows three results for the case of $(\lambda, \mu) = (1, 1)$: the bottom graph shows the smooth result obtained with Gaussian boundary data, while the top two show the corresponding results obtained with a square pulse boundary data (of unit height on the interval $(0, 1)$, which is our standard test data). The exact solution is shown as an unbroken line. These two plots show how devastating the checkerboard mode can be. The top plot is for the CFL number, $p = V\Delta t/\Delta x$, equal to 0.8 and shows the oscillations moving faster than the main pulse; the middle plot is with $p = 1.25$ and shows the oscillations moving more slowly, as predicted by the V^2/p velocity component in (3.35). In Figure 3 corresponding results are shown for the case $\lambda = 90$, $\mu = 10$, the top plot for square pulse data and the bottom plot for Gaussian data, both with $p = 0.8$. The middle plot has $p = 1.25$. The results are quite accurate and show the mollifying effect of increasing λ and μ. In Figure 4 this case is repeated with square pulse initial data, with $\Delta x = 0.04$, and varying values of the CFL number.

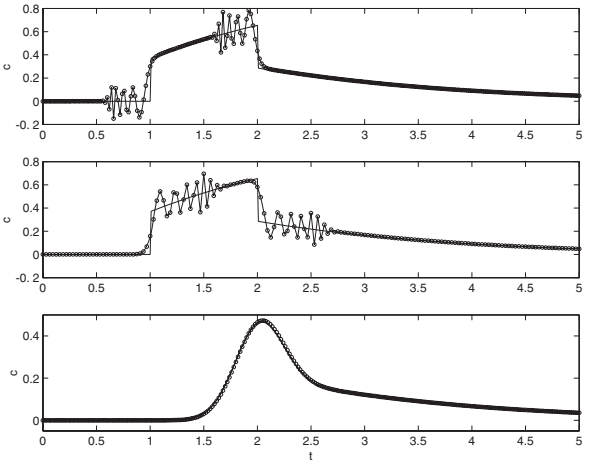


FIG. 2. *Solution c for $V = 1$ and $x = 1$: the thin unbroken line indicates the exact solution, the dots joined by an unbroken line indicate the box scheme. In all cases $(\lambda, \mu) = (1, 1)$, $\Delta x = 0.025$. The top plot has a square pulse initial data with CFL number $p = 0.8$, the middle plot has the same initial data with $p = 1.25$, and the bottom plot has Gaussian initial data with $p = 0.8$.*

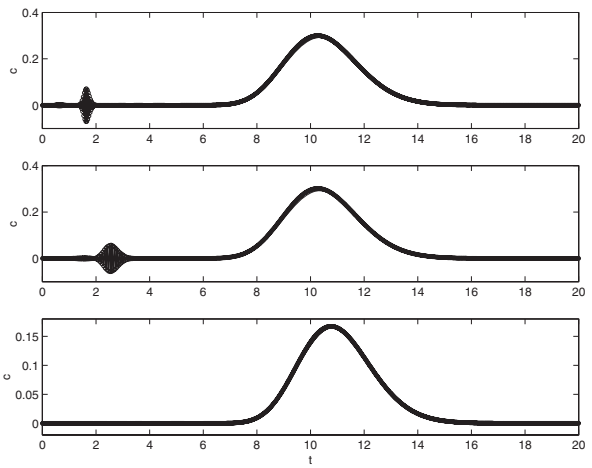


FIG. 3. *As for Figure 2 but with $(\lambda, \mu) = (90, 10)$.*

Choosing p to match the reduced speed (that is, $p = (\lambda + \mu)/\mu = 10$) eliminates the effects of the discretization in the dispersion term in (3.22). However, the best results are obtained by taking $p = 6.4103$, which is calculated to set the coefficient of the dispersion term in (3.22) to zero (which is possible in this model problem but unlikely to hold in applications). These results confirm the modified equation analysis of section 3.2 showing that the accuracy is improved by tuning p to match the reduced speed rather than the speed in the transport equation. They also show how the effect of the oscillations is reduced by such a choice. These experiments are repeated with $\theta = .51$ in Figure 4, showing how such a choice completely eliminates the oscillations, as predicted by the analysis in section 3.4. Finally, we show that the spurious oscillations move with speed V/p^2 independent of λ and μ as predicted by

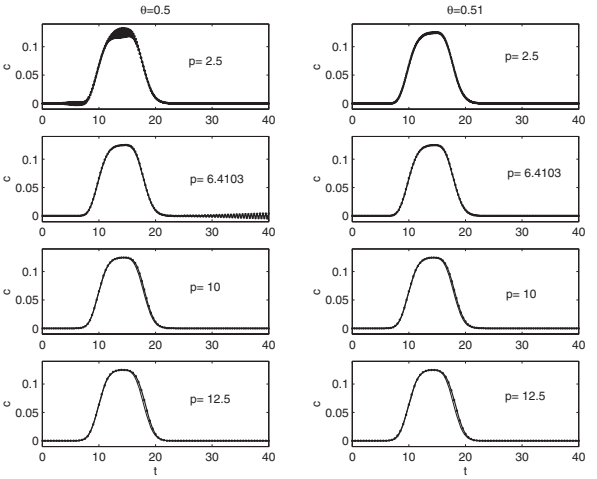


FIG. 4. *Solution c for $V = 1$ at $x = 1$: the thin unbroken line indicates the exact solution and the dotted line the box scheme solution. In all cases $\Delta x = 0.04$ $(\lambda, \mu) = (90, 10)$. The values of p are 2.5, 6.4103, 10, and 12.5. The left-hand column has $\theta = 0.5$ and the right-hand column has $\theta = 0.51$.*

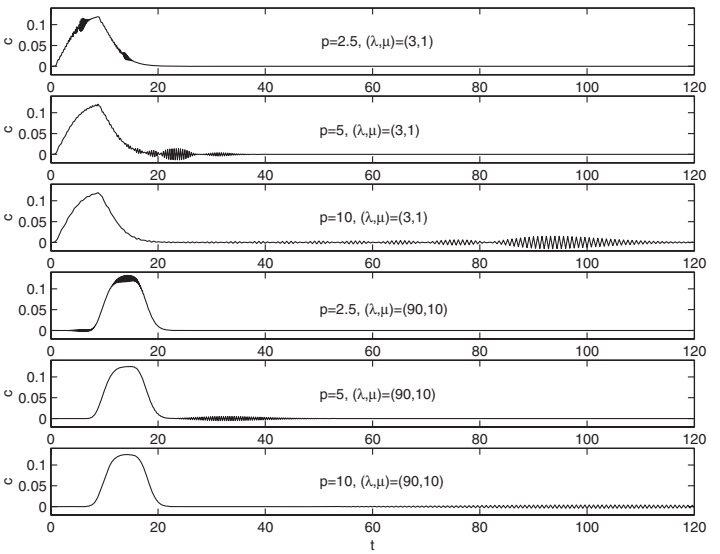


FIG. 5. *Solution c for $V = 1$ at $x = 1$ for $(\lambda, \mu) = (3, 1)$ and $(\lambda, \mu) = (90, 10)$, for three values of $p = \Delta t/\Delta x$: the thin line indicates the box scheme solution. It is easily seen that the spurious oscillations move with speed V/p^2.*

(3.35). This is easily observed in Figure 5 for $(\lambda, \mu) = (90, 10)$ and $(\lambda, \mu) = (3, 1)$, where the spurious oscillations exhibit the same qualitative behavior for the two sets of parameter values.

4. A nonlinear model in one space dimension. We now turn our attention to the following problem with a nonlinear reaction term:

$$c_t + Vc_x = r(c,s), \tag{4.1}$$

$$s_t = -r(c,s). \tag{4.2}$$

If $r(c,s) = -\lambda c + \mu s$, then we recover the linear problem discussed in section 2. Common nonlinear expressions are the Langmuir reaction (see [16], p. 166)

$$r(c,s) = -\lambda c(\mathcal{B} - s) + \mu s, \tag{4.3}$$

where $\mathcal{B}$ represents the maximum capacity for absorption, and thus the rate of absorption of c into the rock reduces as s increases, and the Freundlich reaction

$$r(c,s) = -\lambda c^{1+\beta} + \mu s, \tag{4.4}$$

where the first term on the right-hand side represents a $(1+\beta)$th-order reaction for the adsorption of c into the rock. Often $-1 < \beta < 0$, but we do not impose this restriction here.

Adding (4.1) and (4.2) gives the conservation equation

$$(c+s)_t + Vc_x = 0, \tag{4.5}$$

and, following [13], let us assume that the reaction term $r(c,s)$ in (4.1) and (4.2) can be written as $r(c,s) = (s - S(c))/\epsilon$ for some smooth $S(c)$ and some small parameter ϵ. Thus (4.2) is written as

$$s_t = \frac{1}{\epsilon}(S(c) - s), \tag{4.6}$$

and the equilibrium manifold (see the first paragraph in section 2.1) is $s = S(c)$.

The equilibrium model (see (2.4)) arises by replacing s in (4.5) by $S(c)$ to give the nonlinear conservation law

$$c_x + \frac{1}{V}(c + S(c))_t = 0, \tag{4.7}$$

where the roles of x and t are reversed in the application of standard theory of hyperbolic conservation laws. Consider this equation subject to Riemann boundary data

$$c(0,t) = \begin{cases} c_L, & t \le \tau, \\ c_R, & t > \tau, \end{cases} \tag{4.8}$$

for some $\tau \ge 0$. If $S''(c) > 0$, as is the case for the Freundlich nonlinearity for $\beta > 0$, and $c_L > c_R$, (4.7) develops a shock with speed

$$U_s := \frac{V(c_L - c_R)}{S(c_L) - S(c_R) + c_L - c_R}, \tag{4.9}$$

but if $c_L < c_R$, (4.7) describes a rarefaction wave. If $S''(c) < 0$, as is the case for the Langmuir nonlinearity, the two cases are interchanged. The improved equilibrium model is (cf. (2.9))

$$c_x + \frac{1}{V}(c + S(c))_t = \frac{\epsilon}{V}(S'(c)c_t)_t, \tag{4.10}$$

where we have replaced s_t in (4.5) using $s_t = (S(c))_t - \epsilon s_{tt}$ and have approximated s_{tt} using $s_{tt} \approx (S'(c)c_t)_t$. Here we see that if $S'(c) > 0$, as is the case in both our examples, then the right-hand side of (4.10) provides a "viscous damping" term that prevents shocks from forming. The numerical results presented in section 4.1 show that discontinuities are indeed gradually smoothed and damped.

As in the linear model, we use the predicted pulse speed to choose the time step; this can be done dynamically as the pulse develops. The shock speed U_s, given by (4.9), is our main guide in the case of a data pulse of height $|c_L - c_R|$; but note that as c_L tends to c_R this becomes the [illegible] $V' = V/(1 + S'(c))$, the characteristic speed of the equilibrium model (4.7) which in general differs from the shock speed. We note also that a traveling wave solution of (4.1), (4.2) propagates at the shock speed (cf. [22, pp. 101–102]).

4.1. Numerical results. Here we present numerical results for two examples with nonlinear reactions. We shall see that the weighted box scheme with appropriately chosen mesh ratio produces good results for the problems under consideration. A likely strategy in a practical situation would be to fix Δx and tune Δt based either on the shock speed, so that $U_s \Delta t/\Delta x \approx 1$, or on the reduced speed, so that $\frac{V}{1+S'(c)}\Delta t/\Delta x \approx 1$, for some appropriate choice of c. Here we carry out experiments with fixed Δt and Δx to fully understand the performance of the box scheme, and to compare the nonlinear case with the linear case discussed in section 3.5.

First, let us consider system (4.1), (4.2) with the Freundlich nonlinearity (4.4) where $\beta = 1$, i.e., $r(c, s) = -\lambda c^2 + \mu s$, and hence $S(c) = \lambda c^2/\mu$, the reduced speed is $\mu V/(\mu + 2\lambda c)$, but the shock speed is $\mu V/(\mu + \lambda|c_L - c_R|)$. We take the boundary condition

$$c(0,t) = \begin{cases} 1/3, & 0 \le t \le 1, \\ 0, & \text{otherwise}, \end{cases} \tag{4.11}$$

and since $S'' > 0$ we expect the "switch-off" discontinuity at $t = 1$ to lead to a steep front. Let $V = 1$, $(\lambda, \mu) = (90, 10)$, so the shock speed is $U_s = 0.25$, which suggests the choice $\Delta t = 4\Delta x$. In Figure 6 we present results for several values of $\Delta t/\Delta x$ at $x = 1$, and in Figure 7 we repeat the experiment using the weighted box scheme with $\theta = 0.52$. First, we see in Figure 6 that the spurious oscillations are strongly evident for all values of p shown here, and they move with speed V/p^2 as predicted for the linear theory by (3.35). However, the θ-weighting eliminates the spurious oscillations in all cases. As x increases the pulse diffuses, as indicated by (4.10), so for optimal results we should tune Δt using the reduced speed so that Δt would gradually reduce from the value $4\Delta x$ derived using the shock speed.

Next, let us consider system (4.1), (4.2) with the Langmuir nonlinearity (4.3) with $\mathcal{B} = 1$, i.e., $r(c, s) = -\lambda c(1 - s) + \mu s$, subject to the same boundary condition as the previous example. Now we have $S(c) = \lambda c/(\mu + \lambda c)$, the reduced speed is $V[1 + \lambda\mu(\mu + \lambda c)^{-2}]$, and the shock speed is $V[1 + \lambda\mu\{(\mu + \lambda c_R)(\mu + \lambda c_L)\}^{-1}] = 4/13$ for the data given in (4.11). Since $S''(c) < 0$ we expect the "switch-on" discontinuity at $t = 0$ to give the steep front and we choose Δt based on the shock speed so that $\Delta t = 3.25\Delta x$. Numerical results for the box scheme using $\Delta x = 1/30$ with both $\theta = 0.5$ (i.e., no weighting) and $\theta = 0.51$ are given in Figure 8. Again we see the significant effect of the spurious oscillations in the unweighted scheme, though for $p = \Delta t/\Delta x = 10/3$ rather good results are obtained. As expected, the introduction of a small amount of weighting produces much improved results for all values of p. As in the first example, we see that the spurious oscillations introduced by the discontinuity

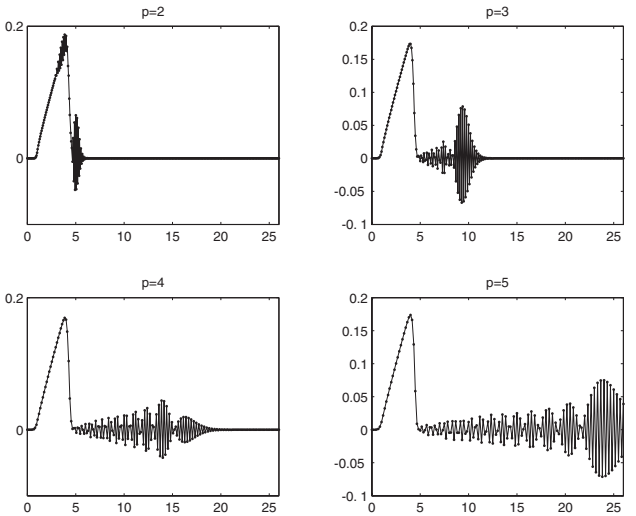


FIG. 6. *The box scheme with no θ-weighting is applied to* (4.1), (4.2) *with Freundlich nonlinearity* (4.4), *and with a square pulse boundary condition* (4.11). *Here* $\beta = 2$, $\lambda = 90$, $\mu = 10$, $V = 1$, $\Delta x = 1/30$, *and* $p := \Delta t/\Delta x$.

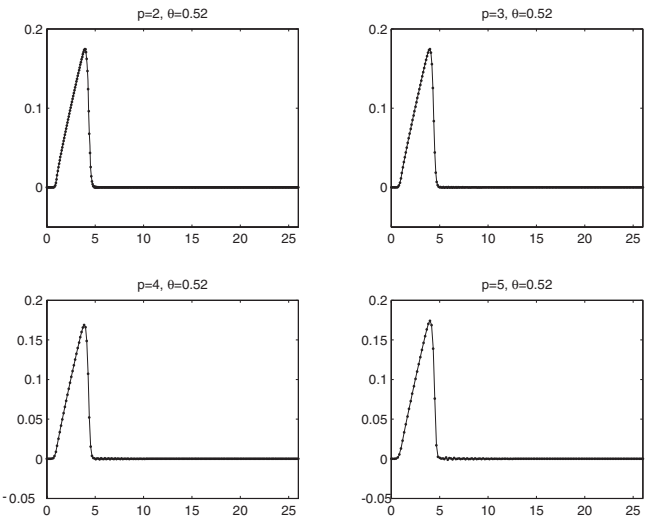


FIG. 7. *As for Figure* 6 *but with* $\theta = 0.52$.

at $x = 0$, $t = 1$ for $\theta = 0.5$ propagate with speed V/p^2, as indicated by (3.35) for the linear problem.

5. The box scheme for two-dimensional problems. Most practical problems are posed in two or three space dimensions with variable velocity profiles. We shall see that the theory of the box scheme in one dimension predicts well its behavior in two dimensions and that the weighted scheme successfully eliminates any spurious oscillations.

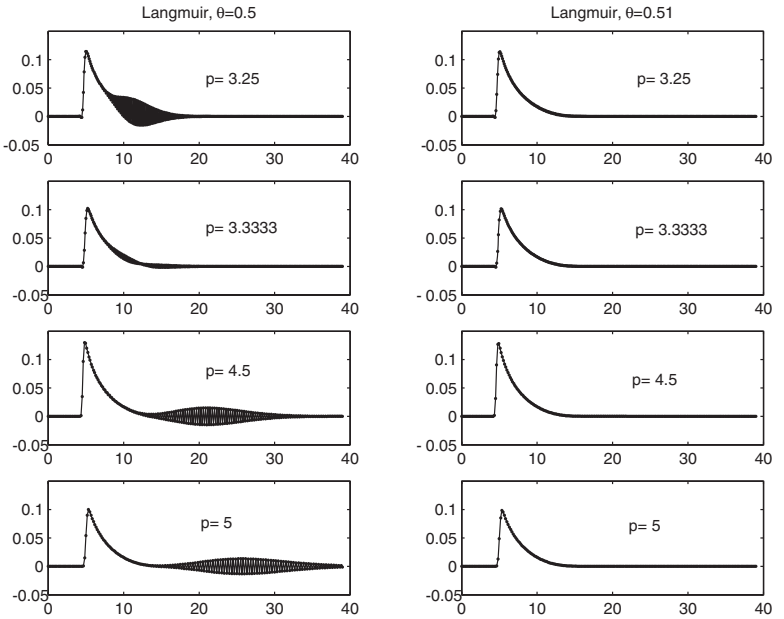


FIG. 8. *The box scheme with* $\theta = 0.5$ *and* 0.51 *is applied to* (4.1), (4.2) *with Langmuir nonlinearity* (4.3), *and with a square pulse boundary condition* (4.11). *Here* $\mathcal{B} = 1$, $(\lambda, \mu) = (90, 10)$, $V = 1$, $x = 1$, *and* $p := \Delta t/\Delta x$.

In [19] and [20] the authors consider the mobility of potentially toxic dissolved metals discharged from a mine tailings source into an aquifer in which the flow is incompressible (see Figure 2 in [20]). We simplify this situation by considering a single chemical pollutant with a reacting term as in the linear model problem. The system to be solved, in conservation form, is

$$c_t + s_t + (U(x,z)c)_x + (V(x,z)c)_z = 0, \tag{5.1}$$

$$s_t = \lambda c - \mu s, \tag{5.2}$$

in the domain $\mathcal{D} := 0.5 \le x \le 2.0$, $0.8 \le z \le 1.0$, for $t \ge 0$, with the incompressible flow field $(U(x,z), V(x,z))$ being given by

$$U(x,z) = \frac{2}{\pi \sinh\left(\frac{\pi Z}{2X}\right)} \cosh\left(\frac{\pi z}{2X}\right) \sin\left(\frac{\pi x}{2X}\right), \tag{5.3}$$

$$V(x,z) = \frac{-2}{\pi \sinh\left(\frac{\pi Z}{2X}\right)} \sinh\left(\frac{\pi z}{2X}\right) \cos\left(\frac{\pi x}{2X}\right). \tag{5.4}$$

(These expressions arise by solving $\nabla^2 \phi(x,z) = 0$ in $0 \le x \le X$, $0 \le z \le Z$, subject to the boundary conditions $\phi_z(x,0) = 0$, $\phi_z(x,Z) = -0.5$, $\phi_x(0,z) = 0$, $\phi(X,z) = 1$, using separation of variables and then taking the first terms in the expressions for $\phi_x(x,z)$ and $\phi_z(x,z)$ as the respective velocity components.)

The initial condition is

$$c(x,z,0) = s(x,z,0) = 0 \quad \text{for } (x,z) \in \mathcal{D}, \tag{5.5}$$

and the boundary condition is

$$c(x,Z,t)=\begin{cases} g(x), & 0\le t\le t^*,\\ 0, & t>t^*. \end{cases} \tag{5.6}$$

In our examples, $t^*=2$ and

$$g(x)=\begin{cases} 0, & x\le\tau,\\ \sin^2\left(\frac{\alpha\pi}{2}(x-\tau)\right), & \tau<x<\frac{1}{\alpha}+\tau,\\ 1, & \frac{1}{\alpha}+\tau\le x\le\beta-\frac{1}{\alpha}+\tau,\\ \sin^2\left(\frac{\alpha\pi}{2}(\beta+\tau-x)\right), & \beta-\frac{1}{\alpha}+\tau\le x\le\beta+\tau,\\ 0, & x>\beta+\tau. \end{cases} \tag{5.7}$$

Here α can be varied, with a small α giving a smooth pulse and a large α making the pulse more "squarelike." In our numerical experiments $\alpha=6$, $\beta=0.8$, and $\tau=0.6$.

In terms of difference operators, the box scheme applied to (5.1), (5.2) produces

$$\begin{aligned} &\frac{\mu_x\mu_z\delta_t}{\Delta t}C^{n+\frac12}_{i+\frac12,j+\frac12}+\frac{\mu_x\mu_z\delta_t}{\Delta t}S^{n+\frac12}_{i+\frac12,j+\frac12}\\ &\quad+\frac{\mu_z\mu_t\delta_x}{\Delta x}(U\ C)^{n+\frac12}_{i+\frac12,j+\frac12}+\frac{\mu_x\mu_t\delta_z}{\Delta z}(V\ C)^{n+\frac12}_{i+\frac12,j+\frac12}=0,\\ &\frac{\delta_t}{\Delta t}S^{n+\frac12}_{i+1,j+1}=\lambda\ \mu_t C^{n+\frac12}_{i+1,j+1}-\mu\ \mu_t S^{n+\frac12}_{i+1,j+1}. \end{aligned}$$

The box scheme is an implicit scheme, but in this linear problem, with the fluxes flowing in the same general direction in all the cells, an efficient explicit scheme is obtained by sweeping from the top left-hand corner. We use an averaged mesh in the x and z direction calculated as follows. Set

$$\hat U(x)=\frac{1}{0.2}\int_{0.8}^{1}U(x,z)\,\mathrm{d}z,\qquad \hat V(z)=\frac{1}{1.5}\int_{0.5}^{2}V(x,z)\,\mathrm{d}x$$

and choose variable step sizes Δx_{i+1} and Δz_{j+1} such that

$$\frac{\hat U(x_{i+1})+\hat U(x_i)}{2\Delta x_{i+1}}=\frac{\hat V(z_{j+1})+\hat V(z_j)}{2\Delta z_{j+1}}.$$

In our numerical experiments we take $\lambda=90$, $\mu=10$. We expect the solution to be close to that for the equilibrium model, and hence the solution speed is reduced by a factor of $\frac{\mu}{\lambda+\mu}$. Thus we expect the optimal choice for Δt to be given by

$$\Delta t=\frac{\lambda+\mu}{\mu}\frac{2\Delta x_{i+1}}{\hat U(x_{i+1})+\hat U(x_i)}=\frac{\lambda+\mu}{\mu}\frac{2\Delta z_{i+1}}{\hat V(x_{i+1})+\hat V(x_i)}.$$

If we introduce the quantity p given by

$$p=\frac{\hat U(x_{i+1})+\hat U(x_i)}{2}\frac{\Delta t}{\Delta x_{i+1}} \tag{5.8}$$

(cf. (3.2) for the one-dimensional problem), then to tune the mesh for optimal results we take p to be $\frac{\lambda+\mu}{\mu}$. Numerical results obtained using the box scheme with $p=\frac{\lambda+\mu}{\mu}=10$ are shown in Figure 9, and for the θ-weighted scheme in Figure 10. The unweighted scheme clearly shows the spurious mode moving at a different speed from the pulse, and the weighted scheme has again successfully removed the oscillations. To illustrate the behavior of the spurious oscillations, we show in Figures 11 and 12 the

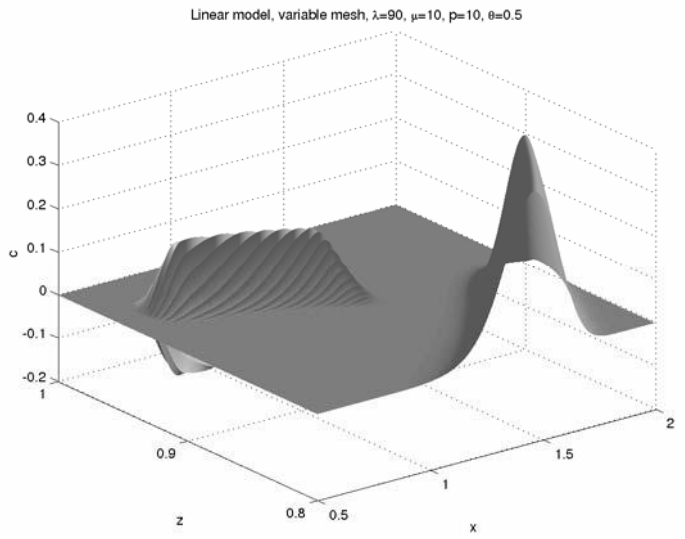


FIG. 9. *Box scheme applied to* (5.1), (5.2) *with* $\theta = 0.5$. *Here the solution pulse moves independently of the spurious oscillations.*

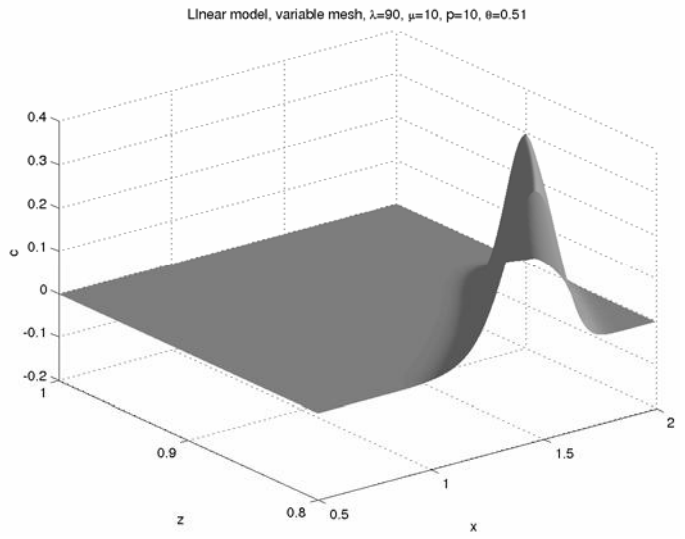


FIG. 10. *Box scheme applied to* (5.1), (5.2) *with* $\theta = 0.51$ *has removed the spurious oscillations.*

computed values of $c(1, 0.95, t)$ for $0 \leq t \leq 25$ for four values of p, namely, $p = 1.25$, 2.5, 5, and 10. Figure 11 shows the results from the unweighted box scheme. As p doubles we see the spurious oscillations move at a speed determined by the mesh and not the parameters in the differential equation. In fact, they move at a speed roughly like $1/p^2$, as predicted by (3.35) for one-dimensional problems (cf. also Figure 5). Figure 12 shows improved results obtained by the weighted box scheme with $\theta = 0.51$, and in particular the spurious oscillations have disappeared for the optimal value $p = 10$, as expected.

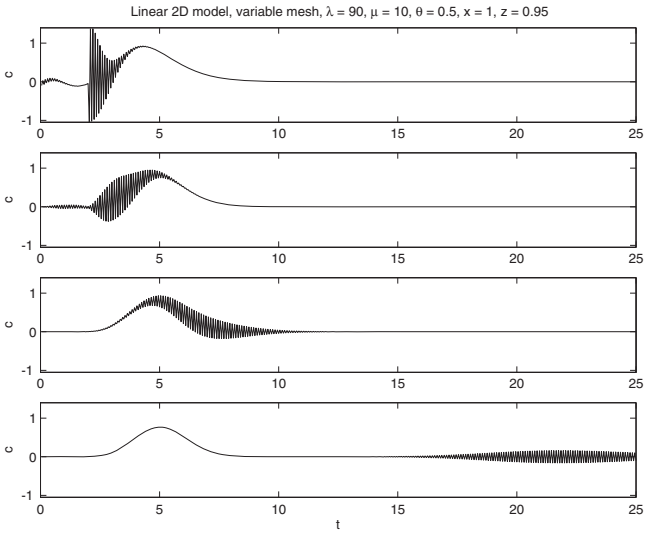


FIG. 11. *Plot of the computed value of* $c(1, 0.95, t)$ *for* $0 \le t \le 25$ *for* $\Delta t = 0.05$ *and* $p = 1.25, 2.5, 5, 10$*, respectively. The pulse of spurious oscillations moves like* p^{-2} *just as predicted in the one-dimensional theory.*

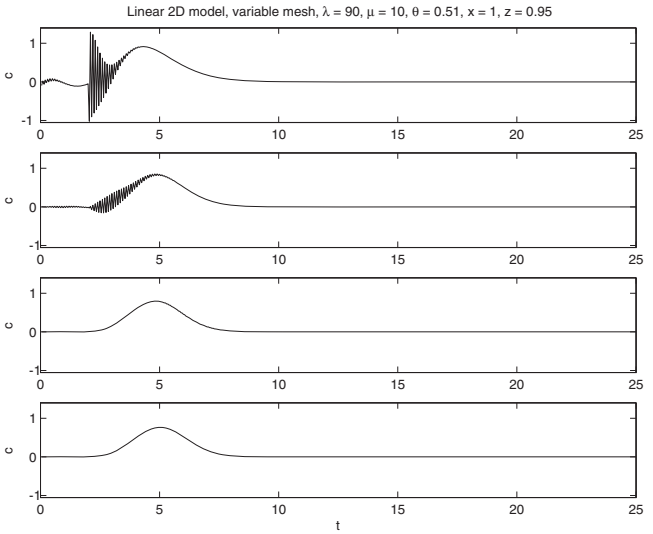


FIG. 12. *As for Figure* 11 *but with* $\theta = 0.51$*. The spurious oscillations have been greatly reduced and for* $p = 10$ *have disappeared.*

6. Conclusions. In this paper we discuss the application of the box scheme to hyperbolic conservation laws with reactive source terms. Our discussion centers initially on a model problem with linear reaction terms. First, we provide a straightforward asymptotic analysis that explains the phenomena of reduced speed, enhanced diffusion, and dispersion. An energy analysis for the box scheme shows the stability of cell averages but also indicates how the notorious checkerboard mode may grow linearly. A detailed modified equation analysis shows how to tune the mesh parameters

in the box scheme to capture the main advected wave with minimal dispersion. A novel modified equation analysis, based on separating smooth and oscillatory solutions, shows how the checkerboard mode moves at a nonphysical speed determined by the mesh parameters. The θ-weighted box scheme is also discussed and, in particular, the modified equation analysis clearly shows how the θ-weighting exponentially damps the checkerboard mode. We show how the deductions from the analysis for the linear problem apply to problems with nonlinear reactions; this has applications in other physically important problems, such as the St. Venant equations for open channel flow. Numerical results for linear and nonlinear reactions in one dimension and linear reactions in two dimensions illustrate the applicability of the theory, including the nonphysical speed of the checkerboard mode and the effectiveness of θ-weighting, across a variety of problems.

REFERENCES

[1] T. Arbogast, S. Bryant, C. Dawson, F. Saaf, C. Wang, and M. Wheeler, *Computational methods for multiphase flow and reactive transport problems arising in subsurface contaminant remediation*, J. Comput. Appl. Math., 74 (1996), pp. 19–32.

[2] D. A. Barry, K. Bajracharya, and C. T. Miller, *Alternative split-operator approach for solving chemical reaction/groundwater transport models*, Advances in Water Resources, 19 (1996), pp. 261–275.

[3] D. A. Barry, C. T. Miller, P. J. Culligan, and K. Bajracharya, *Analysis of split operator methods for nonlinear and multispecies groundwater chemical transport models*, Math. Comput. Simulation, 43 (1997), pp. 331–341.

[4] G.-Q. Chen, D. Levermore, and T.-P. Liu, *Hyperbolic conservation laws with stiff relaxation terms and entropy*, Commun. Pure Appl. Math., 47 (1994), pp. 787–830.

[5] P. I. Crumpton, J. A. MacKenzie, and K. W. Morton, *Cell vertex algorithms for the compressible Navier-Stokes equations*, J. Comput. Phys., 109 (1993), pp. 1–15.

[6] J. A. Cunge, F. M. Holly, Jr., and A. Verwey, *Practical Aspects of Computational River Hydraulics*, Pitman, Boston, 1980.

[7] J. Herzer and W. Kinzelbach, *Coupling of transport and chemical processes in numerical transport models*, Geoderma, 44 (1989), pp. 115–127.

[8] A. A. Jennings, D. J. Kirkner, and T. L. Theis, *Multicomponent equilibrium chemistry in groundwater quality models*, Water Resources Research, 18 (1982), pp. 1089–1096.

[9] H. B. Keller, *Numerical Methods for Two-Point Boundary-Value Problems*, Blaisdell, London, 1968.

[10] H. B. Keller, *A new finite difference scheme for parabolic problems*, in Numerical Solution of Partial Differential Equations II, SYNSPADE 1970, B. Hubbard, ed., Academic Press, New York, 1971, pp. 327–350.

[11] M. J. Lighthill and G. B. Whitham, *On kinematic waves:* I. *Flood movement in long rivers*, Proc. Roy. Soc. A, 229 (1955), pp. 281–316.

[12] M. J. Lighthill and G. B. Whitham, *On kinematic waves:* II. *A theory of traffic flow on long crowded roads*, Proc. Roy. Soc. A, 229 (1955), pp. 317–345.

[13] T.-P. Liu, *Hyperbolic conservation laws with relaxation*, Commun. Math. Phys., 108 (1987), pp. 153–175.

[14] S. L. Mitchell, *Coupling Transport and Chemistry: Numerics, Analysis and Applications*, Ph.D. thesis, University of Bath, UK, 2003.

[15] K. W. Morton and D. F. Mayers, *Numerical Solution of Partial Differential Equations*, 2nd ed., Cambridge University Press, Cambridge, UK, 2005.

[16] H.-K. Rhee, R. Aris, and N. R. Amundson, *First-Order Partial Differential Equations, Volume* I*, Theory and Application of Single Equations*, Prentice Hall, Englewood Cliffs, NJ, 1986.

[17] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial Value Problems*, Interscience, New York, 1967.

[18] V. Thomée, *A stable difference scheme for the mixed boundary value problem for a hyperbolic first order system in two dimensions*, J. Soc. Indust. Appl. Math., 10 (1962), pp. 229–245.

[19] A. L. Walter, E. O. Frind, D. W. Blowes, C. J. Ptacek, and J. W. Molson, *Modeling of multicomponent reactive transport in groundwater* 1. *Model development and evaluation*,

Water Resources Research, 30 (1994), pp. 3117–3148.
[20] A. L. Walter, E. O. Frind, D. W. Blowes, C. J. Ptacek, and J. W. Molson, *Modeling of multicomponent reactive transport in groundwater* 2. *Metal mobility in aquifers impacted by acidic mine tailings discharge*, Water Resources Research, 30 (1994), pp. 3149–3158.
[21] R. F. Warming and B. J. Hyett, *The modified equation approach to the stability and accuracy analysis of finite-difference methods*, J. Comput. Phys., 14 (1974), pp. 159–179.
[22] G. B. Whitham, *Linear and Nonlinear Waves*, John Wiley, New York, 1974.
[23] P. F. Zhao and M. Z. Qin, *Multisymplectic geometry and multisymplectic Preissmann scheme for the KdV equation*, J. Phys. A, 33 (2000), pp. 3613–3626.
[24] A. Zysset, F. Stauffer, and T. Dracos, *Modeling of chemically reactive groundwater transport*, Water Resources Research, 30 (1994), pp. 2217–2228.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1226–1260
© 2006 Society for Industrial and Applied Mathematics

AVERAGING TECHNIQUES FOR THE EFFECTIVE NUMERICAL SOLUTION OF SYMM'S INTEGRAL EQUATION OF THE FIRST KIND*

CARSTEN CARSTENSEN† AND DIRK PRAETORIUS‡

Abstract. Averaging techniques for finite element error control, occasionally called *ZZ estimators* for the gradient recovery, enjoy a high popularity in engineering because of their striking simplicity and universality: One does not even require a PDE to apply the nonexpensive postprocessing routines. Recently, averaging techniques have been mathematically proved to be reliable and efficient for various applications of the finite element method. This paper establishes a class of averaging error estimators for boundary integral methods. Symm's integral equation of the first kind with a nonlocal single-layer integral operator serves as a model equation studied both theoretically and numerically. We introduce four new error estimators which are proven to be reliable and efficient up to terms of higher order. The higher-order terms depend on the regularity of the exact solution. Several numerical experiments illustrate the theoretical results and show that the [normally unknown] error is sharply estimated by the proposed estimators, i.e., error and estimators almost coincide.

Key words. integral equations, boundary element method, a posteriori error estimates, reliability, efficiency, adaptive algorithm

AMS subject classifications. 65N38, 65R20, 65N50

DOI. 10.1137/040609033

1. Introduction.

Reliable error control and efficient meshdesign in today's boundary element analysis are usually based on a posteriori error estimates. Let Ω be a bounded domain in $\mathbb{R}^d$, $d = 2, 3$, with Lipschitz boundary $\partial\Omega$, and let $\Gamma \subset \partial\Omega$ be an open surface. Suppose we are given the right-hand side f and an approximation u_h for the unknown exact solution u of the operator equation

$$Vu = f \qquad \text{in } \widetilde{H}^{-1/2}(\Gamma) \tag{1.1}$$

for the single-layer potential [ds_y denotes surface integration on $\Gamma \subseteq \mathbb{R}^d$ with respect to the variable y] defined by

$$(Vu)(x) = \int_\Gamma u(y)\kappa(x-y)\, ds_y \quad \text{for } x \in \Gamma \tag{1.2}$$

and interpreted in a weak sense for the kernel

$$\kappa(x) := \begin{cases} -\frac{1}{\pi}\log|x| & \text{for } d = 2, \\ \frac{1}{2\pi}|x|^{-1} & \text{for } d = 3. \end{cases} \tag{1.3}$$

A posteriori error estimators $\eta = \eta(u_h, f, \mathcal{T})$ are computable quantities in terms of the right-hand side f, a computed approximate solution u_h, and the given underlying

*Received by the editors May 26, 2004; accepted for publication (in revised form) March 8, 2005; published electronically January 6, 2006. The support of the Austrian Science Fund FWF under grant P15274 is thankfully acknowledged.

http://www.siam.org/journals/sisc/27-4/60903.html

†Department of Mathematics, Humboldt-Universität zu Berlin, Unter den Linden 6, D-10099 Berlin, Germany (cc@math.hu-berlin.de).

‡Institute for Analysis and Scientific Computing, Vienna University of Technology, Wiedner Hauptstrasse 8-10, A-1040 Vienna, Austria (Dirk.Praetorius@tuwien.ac.at).

mesh $\mathcal{T} = \{\Gamma_1, \ldots, \Gamma_n\}$ which bound the exact error from below or above [so-called efficiency and reliability of η, resp.]. So far, for boundary element methods, the following four groups (i)–(iv) of a posteriori estimates have been introduced; see [CFa] for more details.

(i) Weighted residual error estimators in 2D, established in [C1, C2, C3, CES, CS1, CS2],

$$\eta_{R,j} := h_j^{\alpha} \, \|\partial R/\partial s\|_{L^2(\Gamma_j)} \tag{1.4}$$

with the derivative $\partial R/\partial s$ of the residual $R := f - Vu_h$ along Γ and $h_j := |\Gamma_j|$ the size of Γ_j. For the *hp*-method, see [CFS] and, recently, for 3D [CMS, CMPS].

(ii) Local double-integral seminorms in [F2, F3, CP1] with a double integration over overlapping domains $\omega_j := \Gamma_{j-1} \cup \Gamma_j$ in 2D and 3D,

$$\varrho_{F,j}^2 := \int_{\omega_j} \int_{\omega_j} \frac{|R(x) - R(y)|^2}{|x - y|^{1+2\alpha}} \, ds_x \, ds_y. \tag{1.5}$$

(iii) Based on an idea in the finite element literature [BR], Babuška–Rheinboldt-type error estimators are suggested in [F1], so far only for hypersingular integral equations.

(iv) Multilevel error estimators involve a hierarchy of grids and, usually, a disputable saturation assumption [MMS, MSW, CMPS].

Other suggested error estimators employ the notion of an influence index and strengthened Cauchy inequalities [R1, R2, WY, Y1, Y2], localize the outer integration in the Sobolev–Slobodeckij norm (1.5) of the residual $R := f - Vu_h$ [FHK], try to recover gradients [SW, SSW], or employ corrections with another integral equation [MPM, S, SSt].

The nonlocal character of the involved pseudodifferential operator V and the nonlocal Sobolev spaces [of functions on Γ] cause severe difficulties in the mathematical derivation of computable lower and upper error bounds for a discrete (known) approximation u_h to the (unknown) exact solution u. A comparison [F2, F3] shows that the Faermann error estimator $\eta_F := \big(\sum_{j=1}^n \eta_{F,j}^2\big)^{1/2}$ and its modification μ_F from [CP1] were the , , , proven reliable and efficient estimators for unstructured grids in the sense that

$$C_{\text{eff}} \|u - u_h\|_{\widetilde{H}^{-1/2}(\Gamma)} \le \eta_F \le \mu_F \le C_{\text{rel}} \, \|u - u_h\|_{\widetilde{H}^{-1/2}(\Gamma)} \tag{1.6}$$

for the energy norm $\|u - u_h\|_{\widetilde{H}^{-1/2}(\Gamma)}$ of the error in a Galerkin boundary element method.

In this paper we introduce a new class of error estimators η_M, μ_M, η_A, and μ_A for Symm's integral equation based on averaging techniques. For finite element methods it has recently been shown that any averaging technique yields in fact reliable error estimators [ZZ, CB, AC, CFu]. Our results in this paper establish this concept for the Galerkin boundary element method.

In the simplest case, let $\mathcal{T}_H$ be a given mesh with mesh size H, and let $\mathcal{T}_h$ be obtained by uniform refinements of $\mathcal{T}_H$. Let $u_h \in \mathcal{P}_0(\mathcal{T}_h)$ be a $\mathcal{T}_h$-piecewise constant Galerkin approximation of the exact solution $u \in \widetilde{H}^{-1/2}(\Gamma)$ of (1.1). If $\mathbb{G}_H : \widetilde{H}^{-1/2}(\Gamma) \to \mathcal{P}_1(\mathcal{T}_H)$ and $\mathcal{A}_H : \widetilde{H}^{-1/2}(\Gamma) \to \mathcal{P}_1(\mathcal{T}_H)$ denote the Galerkin projection, respectively, the L^2-projection onto the $\mathcal{T}_H$-piecewise affine [not necessarily continuous] functions and if the mesh size h is small enough compared with H, then the error estimator [with $|\!|\!| \cdot |\!|\!|$ the energy norm of section 2.2]

$$\eta_M := |\!|\!| u_h - \mathbb{G}_H u_h |\!|\!| := \langle V(u_h - \mathbb{G}_H u_h)\,; u_h - \mathbb{G}_H u_h\rangle^{1/2} \tag{1.7}$$

is always reliable and efficient up to terms of higher order; cf. Theorem 5.2. The higher-order terms depend only on the smoothness of the exact solution $u \in \widetilde{H}^{-1/2}$. For the lowest-order ansatz $u_h \in \mathcal{P}_0(\mathcal{T}_h)$ it suffices that u is $\mathcal{T}_h$-piecewise in $H^{1+\varepsilon}$ for some $\varepsilon > 0$ [in fact, Γ is therefore required to be piecewise smoother than only Lipschitz].

Since $\mathbb{G}_H$ is the best approximation operator with respect to the energy norm, there holds $|\!|\!| u_h - \mathbb{G}_H u_h |\!|\!| \le |\!|\!| u_h - \mathcal{A}_H u_h |\!|\!|$. In particular, the error estimator

$$\eta_A := |\!|\!| u_h - \mathcal{A}_H u_h |\!|\!| \tag{1.8}$$

is reliable. Theorem 5.6 states the efficiency of η_A provided H^1-stability of $\mathcal{A}_H$ [i.e., $\mathcal{A}_H$ is continuous as operator from $H^1(\Gamma)$ to $H^1(\Gamma)$]. By interpolation and inverse estimates we show that the error estimators

$$\mu_M := \|H^{1/2}(\mathbb{1} - \mathbb{G}_H)u_h\|_{L^2(\Gamma)} \quad \text{and} \quad \mu_A := \|H^{1/2}(\mathbb{1} - \mathcal{A}_H)u_h\|_{L^2(\Gamma)} \tag{1.9}$$

are equivalent to η_M and η_A, respectively, i.e.,

$$C_1^{-1}\mu_M \le \eta_M \le C_2\mu_M \quad \text{and} \quad C_1^{-1}\mu_A \le \eta_A \le C_2\mu_A \tag{1.10}$$

with constants C_1, C_2 that do not depend on the size or the number of the elements in $\mathcal{T}_h$ and $\mathcal{T}_H$; cf. Corollaries 5.4 and 5.5. Since the L^2 norm is local in the sense that $\|\cdot\|^2_{L^2(\Gamma)} = \sum_{\gamma\in\mathcal{T}_H} \|\cdot\|^2_{L^2(\gamma)}$, μ_M and μ_A can be employed for an indicator-based adaptive mesh refinement introduced in section 6.3 below.

The paper is organized as follows: section 2 recalls the definition of the Sobolev spaces $\widetilde{H}^\alpha(\Gamma)$ and $H^\alpha(\Gamma)$ of fractional order $-1 \le \alpha \le 1$. Section 3 displays preliminaries on the mesh geometry and finite elements. A local first-order approximation result for the L^2-projection in scales of $H^\alpha(\Gamma)$ is proven in section 4. Section 5 establishes the a posteriori error estimates and introduces four error estimators (1.7)–(1.9) based on averaging techniques. In section 6 we give implementational details following the spirit of [CP1]. Finally, section 7 reports on five experiments, from which conclusions are drawn in section 8.

2. Preliminaries on the functional analytic setting. This section aims to recall the definition of the fractional-order Sobolev spaces on open screens $\Gamma \subseteq \partial\Omega$, where Ω is a bounded Lipschitz domain in $\mathbb{R}^d$, $d = 2, 3$, with boundary $\partial\Omega$. We provide the interpolation results we are going to use in section 4 and recall the mapping properties of the single layer potential (1.2) in scales of Sobolev spaces $H^\alpha(\Gamma)$.

2.1. Fractional order Sobolev spaces and interpolation. For any (relatively) open set $\omega \subseteq \partial\Omega$ and $0 \le \alpha \le 1$, we define Sobolev spaces of fractional order by complex interpolation

$$\widetilde{H}^\alpha(\omega) = [L^2(\omega); H^1_0(\omega)]_\alpha \quad \text{and} \quad H^\alpha(\omega) = [L^2(\omega); H^1(\omega)]_\alpha, \tag{2.1}$$

where $[X_0; X_1]_\alpha$ denotes the complex interpolation of X_0 and $X_1 \subseteq X_0$ [BL, McL]. The norm $\|\cdot\|_{H^1(\omega)}$ is given by the surface gradient ∇ as $\|u\|^2_{H^1(\omega)} = \|u\|^2_{L^2(\omega)} + \|\nabla u\|^2_{L^2(\omega)}$, and $H^1(\omega)$ and $H^1_0(\omega)$ are defined as the completions of $\mathrm{Lip}(\omega)$ and $\{v \in \mathrm{Lip}(\omega) : v|_{\partial\omega} = 0\}$, respectively. Sobolev spaces with negative index are defined by duality,

$$H^{-\alpha}(\Gamma) := \widetilde{H}^\alpha(\Gamma)^* \quad \text{and} \quad \widetilde{H}^{-\alpha}(\Gamma) := H^\alpha(\Gamma)^* \tag{2.2}$$

with corresponding norms and duality brackets

$$\langle \cdot\,;\cdot \rangle \quad \text{in } \widetilde{H}^{-\alpha}(\Gamma) \times H^{\alpha}(\Gamma) \tag{2.3}$$

which extend the $L^2(\Gamma)$ scalar product.

2.1. Let X_0, X_1 be normed spaces with $X_1 \subseteq X_0$ and $0 \le \alpha \le 1$. Then, the norm of the interpolation space $X := [X_0; X_1]_\alpha$ satisfies

$$\|\cdot\|_{[X_0;X_1]_\alpha} \le \|\cdot\|_{X_0}^{1-\alpha} \|\cdot\|_{X_1}^{\alpha}. \tag{2.4}$$

Furthermore, let $Y := [Y_0; Y_1]_\alpha$ be the complex interpolation of normed spaces $Y_1 \subseteq Y_0$. If $T \in L(X_0; Y_0)$ can also be viewed as an operator $T \in L(X_1; Y_1)$, then $T : X \to Y$ is well defined and continuous and the corresponding operator norms satisfy

$$\|T\|_{L(X;Y)} \le \|T\|_{L(X_0;Y_0)}^{1-\alpha} \|T\|_{L(X_1;Y_1)}^{\alpha}. \qquad \square \tag{2.5}$$

LEMMA 2.1 (see [P, StS]). Let $\mathcal{T} = \{\Gamma_1, \ldots, \Gamma_n\}$ be a partition of Γ into relatively open pieces $\Gamma_1, \ldots, \Gamma_n$ of Γ, $0 \le \alpha \le 1$, and $u \in H^\alpha(\Gamma)$. Then $u|_{\Gamma_j} \in H^\alpha(\Gamma_j)$ for $j = 1, \ldots, n$ and

$$\sum_{j=1}^{n} \|u|_{\Gamma_j}\|_{H^\alpha(\Gamma_j)}^2 \le \|u\|_{H^\alpha(\Gamma)}^2. \qquad \square \tag{2.6}$$

2.2. (i) The constant factor 1 on the right-hand side of (2.6) [not displayed explicitly] holds for complex interpolation and needs to be replaced by an n-independent constant in case of real interpolation as well as in case of alternative definitions by extension or by Sobolev–Slobodeckij norms.

(ii) The converse inequality in (2.6) fails to hold in general [F2, F3].

2.2. Single-layer potential and energy norm. The single-layer potential (1.2) defines a continuous linear operator

$$V : \widetilde{H}^{\alpha-1}(\Gamma) \to H^{\alpha}(\Gamma) \tag{2.7}$$

for all $0 \le \alpha \le 1$ [Co, McL]. For $d = 3$, V always is an isomorphism [i.e., V is bijective and V and V^{-1} are continuous]. Moreover,

$$\langle\!\langle u\,; v \rangle\!\rangle := \langle Vu\,; v\rangle \quad \text{for } u, v \in \widetilde{H}^{-1/2}(\Gamma) \tag{2.8}$$

defines a scalar product on $\widetilde{H}^{-1/2}(\Gamma)$. For $d = 2$, V is bijective if the capacity of Γ is not 1, and (2.8) defines a scalar product on $\widetilde{H}^{-1/2}(\Gamma)$ provided the capacity is strictly less than 1 [e.g., Ω is contained in the open unit disk]; cf. [McL].

In what follows we assume for $d = 2$ that the capacity of Ω is strictly less than 1 so that all results of this paper hold for $d = 2$ and $d = 3$ simultaneously. The induced

$$|\!|\!| u |\!|\!| := \langle\!\langle u\,; u \rangle\!\rangle^{1/2} \tag{2.9}$$

is an equivalent norm on $\widetilde{H}^{-1/2}(\Gamma)$. According to the Lax–Milgram lemma, given $f \in H^{1/2}(\Gamma)$ there is a unique solution $u := V^{-1} f \in \widetilde{H}^{-1/2}(\Gamma)$ of (1.1).

3. Preliminaries on finite element approximation.

3.1. Galerkin discretization of $\widetilde{H}^{-1/2}(\Gamma)$. Let $\mathcal{T} = \{\Gamma_1, \dots, \Gamma_n\}$ be a triangulation of Γ. Each element Γ_j of the triangulation $\mathcal{T}$ is supposed to be a connected (affine) boundary piece for $d = 2$ and a (flat) triangle for $d = 3$, respectively. For $d = 3$, we assume that $\mathcal{T}$ is a regular triangulation: Two distinct and intersecting Γ_j and Γ_k share either a common edge or a vertex. The set of all nodes of the triangulation $\mathcal{T}$ is denoted with $\mathcal{N}$. Let $h \in L^\infty(\Gamma)$ denote the local mesh size $h|_{\Gamma_j} := h_j := \operatorname{diam}(\Gamma_j)$.

For an integer $p \geq 0$, $\mathcal{P}_p(\mathcal{T})$ denotes the space of all isoparametric polynomials of total degree $\leq p$ [defined on reference elements $\Gamma_{\mathrm{ref}} = [0,1]$ and $\Gamma_{\mathrm{ref}} = \operatorname{conv}\{(0,0),(0,1),(1,0)\}$ for $d = 2, 3$, resp.].

If $\mathcal{S}$ is a finite dimensional subspace of $\widetilde{H}^{-1/2}(\Gamma)$ [e.g., $\mathcal{S} = \mathcal{P}_p(\mathcal{T})$], the discrete Galerkin approximation $u_h \in \mathcal{S}$ is uniquely determined by the linear system of equations

$$\langle\!\langle u_h \,;\, v_h \rangle\!\rangle = \langle f \,;\, v_h \rangle \quad \text{for all } v_h \in \mathcal{S}. \tag{3.1}$$

The Galerkin projection $\mathbb{G} : \widetilde{H}^{-1/2}(\Gamma) \to \mathcal{S}$ defined by

$$\langle\!\langle \mathbb{G}u \,;\, v_h \rangle\!\rangle = \langle\!\langle u \,;\, v_h \rangle\!\rangle \quad \text{for all } v_h \in \mathcal{S}$$

is the orthogonal projection onto $\mathcal{S} \subseteq \widetilde{H}^{-1/2}(\Gamma)$ with respect to the energy norm.

3.2. Inverse estimate for the energy norm. Given $\mathcal{T}$ and $\mathcal{S} = \mathcal{P}_p(\mathcal{T})$, we assume that there is a mesh size independent constant $C_{\mathrm{inv}} > 0$ which depends only on Γ, p, and the shape of the elements of $\mathcal{T}$ such that, for all $v_h \in \mathcal{P}_p(\mathcal{T})$, there holds

$$\|v_h\|_{L^2(\Gamma)} \leq C_{\mathrm{inv}} |\!|\!| h^{-1/2} v_h |\!|\!|. \tag{3.2}$$

3.1. The estimate (3.2) is proven in [GHS, Theorem 3.6] for $p \in \mathbb{N}_0$ and Γ a 2D manifold in $\mathbb{R}^3$, but the arguments work for a 1D boundary as well. We stress that here, with a focus on adapted meshes, the mesh size enters the estimates. Well-established inverse estimates

$$\|v_h\|_{L^2(\Gamma)} \lesssim h_{\min}^{-1/2} |\!|\!| v_h |\!|\!| \qquad \text{for } v_h \in \mathcal{P}_p(\mathcal{T}) \tag{3.3}$$

involve the global quantity $h_{\min} := \min_{1 \leq j \leq n} h_j$.

3.3. Standard approximation estimate. Given a regular triangulation $\mathcal{T}$, real numbers $\alpha, m \in \mathbb{R}$ with $m \geq \alpha$, and an integer $p \geq 0$, define $\widehat{p} := \min\{p+1, m\} - \alpha$. Moreover, let $\mathcal{S}_p(\mathcal{T})$ denote either $\mathcal{P}_0(\mathcal{T})$ or $\mathcal{P}_p(\mathcal{T}) \cap \mathcal{C}(\Gamma)$ for $p = 0$ and $p > 0$, respectively. Define the $\mathcal{T}$-piecewise Sobolev space

$$H^m(\mathcal{T}) := \{u \in L^2(\Gamma) \,:\, u|_{\Gamma_j} \in H^m(\Gamma_j) \text{ for all } \Gamma_j \in \mathcal{T}\}$$

with norm $\|u\|^2_{H^m(\mathcal{T})} = \sum_{j=1}^n \|u|_{\Gamma_j}\|^2_{H^m(\Gamma_j)}$. Then, there is constant $C_3 > 0$ depending only on Γ, α, m, p, and the shape of the elements in $\mathcal{T}$ such that

$$\min_{v_h \in \mathcal{S}_p(\mathcal{T})} \|u - v_h\|_{H^\alpha(\Gamma)} \leq C_3 h_{\max}^{\widehat{p}} \, \|u\|_{H^m(\mathcal{T})} \tag{3.4}$$

for all $u \in H^\alpha(\Gamma) \cap H^m(\mathcal{T})$ and $h_{\max} := \max_{1 \leq j \leq n} h_j$; cf. [SaS].

3.4. Local first-order approximation operator. The following definitions adapt [CB] for fixed $0 \le \alpha \le 1$ to obtain an approximation operator:

$$\mathcal{J}_h : H^\alpha(\Gamma) \to \mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma) \subseteq L^2(\Gamma). \tag{3.5}$$

For each node $z \in \mathcal{N}$ let $\varphi_z \in \mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma)$ denote the nodal basis functions with $\varphi_z(z) = 1$ and $\varphi_z(\tilde{z}) = 0$ for $\tilde{z} \in \mathcal{N}\backslash\{z\}$. Note that $\{\varphi_z : z \in \mathcal{N}\}$ is a partition of unity. The support of φ_z is denoted with $\omega_z := \operatorname{supp}(\varphi_z)$ and has the diameter $h_z := \operatorname{diam}(\omega_z)$. For each $u \in L^1(\Gamma)$ define

$$\mathcal{J}_h(u) := \sum_{z\in\mathcal{N}} \lambda_z(u)\varphi_z \in \mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma) \quad \text{with} \quad \lambda_z(u) := \frac{\int_{\omega_z} u\varphi_z\, ds}{\int_{\omega_z} \varphi_z\, ds} \quad \text{for } z \in \mathcal{N}. \tag{3.6}$$

LEMMA 3.1. [illegible] $C_4 > 0$ [illegible] Γ [illegible] [illegible] $z \in \mathcal{N}$ $0 \le \alpha \le 1$ [illegible] $u \in H^\alpha(\Gamma)$ [illegible]

$$\|(u - \lambda_z(u))\varphi_z\|_{L^2(\omega_z)} \le C_4 h_z^\alpha \|u\|_{H^\alpha(\omega_z)}. \tag{3.7}$$

[illegible]. We derive from [CB, Theorem 2.1] the existence of $C_4 > 0$ such that

$$\|(u - \lambda_z(u))\varphi_z\|_{L^2(\omega_z)} \le C_4 \min\{\|u\|_{L^2(\omega_z)}, h_z\|\nabla u\|_{L^2(\omega_z)}\}$$

for all $z \in \mathcal{N}$ and $u \in H^1(\Gamma)$. [The proof therein is formulated for a domain $\Gamma \subseteq \mathbb{R}^2$ but applies to the present situation.] For $\alpha = 0$ and $\alpha = 1$, the linear operator

$$T_\alpha : H^\alpha(\omega_z) \to L^2(\omega_z), u \mapsto (u - \lambda_z(u))\varphi_z$$

is well defined and continuous with operator norms $\|T_0\| \le C_4$ and $\|T_1\| \le C_4 h_z$, respectively. By interpolation, T_α is well defined and continuous with $\|T_\alpha\| \le C_4 h_z^\alpha$. This proves (3.7). □

4. Local first-order approximation property. An interpolation argument shows that the L^2-projection has a [illegible] first-order approximation property with respect to the energy norm [equivalent to the $\widetilde{H}^{-1/2}$norm].

DEFINITION 1. [illegible] $\mathcal{S}$ [illegible] $\mathcal{P}_p(\mathcal{T})$ $p \ge 0$ [illegible] $0 \le \alpha \le 1$ [illegible] [illegible] [illegible] $\mathcal{A} : L^2(\Gamma) \to \mathcal{S}$ [illegible] local first-order approximation property with respect to the $\widetilde{H}^{-\alpha}$-norm [illegible] $C_{\text{apx}} > 0$ [illegible] Γ α p [illegible] $\mathcal{T}$ [illegible]

$$\|(\mathbb{1} - \mathcal{A})v\|_{\widetilde{H}^{-\alpha}(\Gamma)} \le C_{\text{apx}} \|h^\alpha v\|_{L^2(\Gamma)} \quad \text{[illegible]} \quad v \in L^2(\Gamma). \tag{4.1}$$

[illegible] C_{apx} [illegible]

The following theorem shows that the L^2-projection has the first-order approximation property provided the space $\mathcal{S}$ is rich enough. It will be used first to prove reliability and efficiency of η_M in Theorem 5.1 and second to show reliability of μ_A and μ_M in Corollary 5.4.

THEOREM 4.1. [illegible] $\mathcal{S}$ [illegible] $\mathcal{P}_0(\mathcal{T})$ [illegible] $\mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma)$ [illegible] L^2 [illegible] $\Pi : L^2(\Gamma) \to \mathcal{S}$ [illegible] $\mathcal{S}$ [illegible] (4.1)

The proof of Theorem 4.1 follows ideas from [CMPS, Theorem 4.1] and involves Lemmas 2.1 and 3.1. We shall consider the cases $\mathcal{P}_0(\mathcal{T}) \subseteq \mathcal{S}$ and $\mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma) \subseteq \mathcal{S}$ separately.

Proof of Theorem 4.1 for $\mathcal{P}_1(\mathcal{T}) \cap \mathcal{C}(\Gamma) \subseteq \mathcal{S}$ Notice that, for $v \in L^2(\Gamma)$, orthogonality of the projection $(\mathbb{1} - \Pi)$ and $\varphi_z \in \mathcal{S}$ lead to

$$\langle (\mathbb{1} - \Pi)v \,;\, \varphi_z \rangle = 0 \quad \text{for all } z \in \mathcal{N}. \tag{4.2}$$

For $u \in H^\alpha(\Gamma)$, we have $u = \sum_{z\in\mathcal{N}} u\varphi_z$ and therefore $\varrho := (\mathbb{1} - \Pi)v$ satisfies

$$\begin{aligned} \langle \varrho \,;\, u \rangle &= \sum_{z\in\mathcal{N}} \langle \varrho \,;\, u\varphi_z \rangle = \sum_{z\in\mathcal{N}} \langle \varrho \,;\, (u - \lambda_z(u))\varphi_z \rangle \\ &\le \sum_{z\in\mathcal{N}} \|\varrho\|_{L^2(\omega_z)} \|(u - \lambda_z(u))\varphi_z\|_{L^2(\omega_z)}. \end{aligned} \tag{4.3}$$

The combination of (4.3) and (3.7) yields

$$\begin{aligned} \langle \varrho \,;\, u \rangle &\le C_4 \sum_{z\in\mathcal{N}} \|h_z^\alpha \varrho\|_{L^2(\omega_z)} \|u\|_{H^\alpha(\omega_z)} \\ &\le C_4 \left(\sum_{z\in\mathcal{N}} \|h_z^\alpha \varrho\|^2_{L^2(\omega_z)} \right)^{1/2} \left(\sum_{z\in\mathcal{N}} \|u\|^2_{H^\alpha(\omega_z)} \right)^{1/2}. \end{aligned} \tag{4.4}$$

A coloring argument in [CMS, section 3] and [CMPS, Theorem 4.1] shows that we can find a finite number of index sets $J_1, J_2, \ldots, J_M$ such that for each k the sets ω_z in $\{\omega_z \,:\, z \in J_k\}$ are pairwise disjoint. The number M depends on the overlap of the patches ω_z, $z \in \mathcal{N}$, and thus on the aspect ratios of the elements and on Γ. Lemma 2.1 can be applied for each set J_k and so yields eventually

$$\sum_{z\in\mathcal{N}} \|u\|^2_{H^\alpha(\omega_z)} \le M \, \|u\|^2_{H^\alpha(\Gamma)}. \tag{4.5}$$

With the constant $C_5 := \max\{h_z/h_T \,:\, z \in \mathcal{K}, T \in \mathcal{T} \text{ with } T \subseteq \omega_z\}$, there holds

$$\sum_{z\in\mathcal{N}} \|h_z^\alpha \varrho\|^2_{L^2(\omega_z)} \le C_5^{2\alpha} \sum_{z\in\mathcal{N}} \|h^\alpha \varrho\|^2_{L^2(\omega_z)} \le M C_5^{2\alpha} \, \|h^\alpha \varrho\|^2_{L^2(\Gamma)}. \tag{4.6}$$

The combination of (4.4), (4.6) yields

$$\|\varrho\|_{\widetilde{H}^{-\alpha}(\Gamma)} = \sup_{\substack{u\in H^\alpha(\Gamma) \\ u\neq 0}} \frac{\langle \varrho \,;\, u \rangle}{\|u\|_{H^\alpha(\Gamma)}} \le M \, C_4 C_5^\alpha \|h^\alpha \varrho\|_{L^2(\Gamma)}. \qquad \square \tag{4.7}$$

Proof of Theorem 4.1 for $\mathcal{P}_0(\mathcal{T}) \subseteq \mathcal{S}$ With $\Pi_0 : L^2(\Gamma) \to \mathcal{P}_0(\mathcal{T})$ the L^2-projection onto $\mathcal{P}_0(\mathcal{T})$, there holds for any $u \in H^1(\Gamma_j)$

$$\|(\mathbb{1}-\Pi)u\|_{L^2(\Gamma_j)} = \inf_{v_h\in\mathcal{S}} \|u - v_h\|_{L^2(\Gamma_j)} \le \inf_{v_h\in\mathcal{P}_0(\mathcal{T})} \|u - v_h\|_{L^2(\Gamma_j)} = \|(\mathbb{1}-\Pi_0)u\|_{L^2(\Gamma_j)}.$$

An application of the Poincaré inequality proves $\|(\mathbb{1}-\Pi_0)u\|_{L^2(\Gamma_j)} \le C_6 h_j \|\nabla u\|_{L^2(\Gamma_j)}$ with $C_6 = 1/\pi$. Hence, the operator $T_\alpha : H^\alpha(\Gamma_j) \to L^2(\Gamma_j), u \mapsto u - \Pi u$ has operator norm $||T_0|| = 1$ and $||T_1|| \le C_6 h_j$ for $\alpha = 0, 1$, respectively. With the interpolation estimate (2.5), we infer $||T_\alpha|| \le C_6^\alpha h_j^\alpha$ and therefore

$$\|(\mathbb{1} - \Pi)u\|_{L^2(\Gamma_j)} \le C_6^\alpha h_j^\alpha \|u\|_{H^\alpha(\Gamma_j)}, \quad \text{for } u \in H^\alpha(\Gamma_j). \tag{4.8}$$

Now, let $v \in L^2(\Gamma)$ and $u \in H^\alpha(\Gamma)$. The symmetry of orthogonal projections yields

$$\langle (\mathbb{1} - \Pi)v\,;u\rangle = \langle v\,;(\mathbb{1} - \Pi)u\rangle \le \sum_{j=1}^{n} \|v\|_{L^2(\Gamma_j)} \|(\mathbb{1} - \Pi)u\|_{L^2(\Gamma_j)}. \tag{4.9}$$

The combination of (4.8)-(4.9) with Lemma 2.1 concludes the proof,

$$\langle (\mathbb{1} - \Pi)v\,;u\rangle \le C_6^\alpha \sum_{j=1}^{n} \|h^\alpha v\|_{L^2(\Gamma_j)} \|u\|_{H^\alpha(\Gamma_j)} \le C_6^\alpha \|h^\alpha v\|_{L^2(\Gamma)} \|u\|_{H^\alpha(\Gamma)}. \qquad \square$$

COROLLARY 4.2. $\alpha = 1/2$ 4.1 $\mathbb{G} : \widetilde{H}^{-1/2}(\Gamma) \to \mathcal{S}$ $\mathcal{S}$. With the best approximation property $\|(\mathbb{1} - \mathbb{G})v\| \le \|(\mathbb{1} - \Pi)v\|$, for all $v \in L^2(\Gamma)$, the proof follows from the equivalence of $\|\cdot\|$ and the $\widetilde{H}^{-1/2}(\Gamma)$-norm. $\square$

The following elementary lemma sharpens the local first-order approximation property. It will be applied for Π and $\mathbb{G}$ to obtain reliability of μ_A and μ_M, respectively.

LEMMA 4.3. $\mathcal{A} : L^2(\Gamma) \to \mathcal{P}_p(\mathcal{T})$ [$\mathcal{A}^2 = \mathcal{A}$] $\mathcal{A}$ (4.1) $v \in L^2(\Gamma)$

$$\|(\mathbb{1} - \mathcal{A})v\|_{\widetilde{H}^{-\alpha}(\Gamma)} \le C_{\rm apx} \min\big\{\|h^\alpha v\|_{L^2(\Gamma)}, \|h^\alpha (\mathbb{1} - \mathcal{A})v\|_{L^2(\Gamma)}\big\}. \tag{4.10}$$

Since $(\mathbb{1} - \mathcal{A})$ is idempotent, we obtain from (4.1) for $w := (\mathbb{1} - \mathcal{A})v \in L^2(\Gamma)$

$$\|w\|_{\widetilde{H}^{-\alpha}(\Gamma)} = \|(\mathbb{1} - \mathcal{A})v\|_{\widetilde{H}^{-\alpha}(\Gamma)} = \|(\mathbb{1} - \mathcal{A})w\|_{\widetilde{H}^{-\alpha}(\Gamma)} \le C_{\rm apx}\, \|h^\alpha w\|_{L^2(\Gamma)}. \qquad \square$$

5. A posteriori error control by averaging techniques. This section aims to provide a new class of error estimators and states their reliability and efficiency of which. We need two discrete finite element spaces $\mathcal{S}_h$ and $\mathcal{S}_H$, where $\mathcal{S}_h$ belongs to a finer mesh $\mathcal{T}_h$ but lower polynomial degree as compared to $\mathcal{S}_H$.

5.1. Assumptions and notations. Fix two regular triangulations $\mathcal{T}_h = \{\Gamma_1, \ldots, \Gamma_n\}$ and $\mathcal{T}_H = \{\gamma_1, \ldots, \gamma_N\}$ with mesh sizes $h \ll H$ [cf. (5.4) for a precise statement]. For integers $0 \le p < q$, let $\mathcal{S}_h$ and $\mathcal{S}_H$ be subspaces of $\mathcal{P}_p(\mathcal{T}_h)$ and $\mathcal{P}_q(\mathcal{T}_H)$, respectively. With respect to Theorem 4.1 we suppose that $\mathcal{S}_h$ contains $\mathcal{P}_0(\mathcal{T}_h)$ or $\mathcal{P}_1(\mathcal{T}_h) \cap \mathcal{C}(\Gamma)$.

Let $u \in \widetilde{H}^{-1/2}(\Gamma)$ denote the unique solution of (1.1), and let $u_h \in \mathcal{S}_h$ be its Galerkin approximation with respect to $\mathcal{S} = \mathcal{S}_h$ in (3.1). The Galerkin projection

$$\mathbb{G}_H : \widetilde{H}^{-1/2}(\Gamma) \to \mathcal{S}_H \subseteq \widetilde{H}^{-1/2}(\Gamma) \tag{5.1}$$

onto $\mathcal{S}_H$ [i.e., the orthogonal projection onto $\mathcal{S}_H$ with respect to the energy norm] is compared with an [not necessarily linear or continuous] operator

$$\mathcal{A}_H : L^2(\Gamma) \to \mathcal{S}_H. \tag{5.2}$$

We consider the following four error estimators:

$$\begin{aligned} \eta_M &:= \|(\mathbb{1} - \mathbb{G}_H)u_h\|, & \mu_M &:= \|H^{1/2}(\mathbb{1} - \mathbb{G}_H)u_h\|_{L^2(\Gamma)}, \\ \eta_A &:= \|(\mathbb{1} - \mathcal{A}_H)u_h\|, & \mu_A &:= \|H^{1/2}(\mathbb{1} - \mathcal{A}_H)u_h\|_{L^2(\Gamma)}, \end{aligned} \tag{5.3}$$

i.e., we estimate the difference of the (low-order) discrete solution u_h on a finer mesh $\mathcal{T}_h$ and a higher-order approximation of which on a coarser mesh $\mathcal{T}_H$.

In what follows, we use the inverse estimate (3.2) for [subspaces of] $\mathcal{P}_p(\mathcal{T}_h)$, $\mathcal{P}_q(\mathcal{T}_h)$, and $\mathcal{P}_q(\mathcal{T}_H)$. To be precise, we therefore write, for instance, $C_{\rm inv}^{h,p} = C_{\rm inv}(\Gamma, p, \mathcal{T}_h)$. The analogous notation is used for the first-order approximation property of the L^2-projection [resp., Galerkin-projection] and we write, for instance, $C_{\rm apx}^{h,p}$. Let $\Pi_h : L^2(\Gamma) \to \mathcal{S}_h$ denote the L^2-projection onto $\mathcal{S}_h$, and let $C_{\rm apx}^{h,p} > 0$ be defined as in Theorem 4.1 with $\mathcal{S} = \mathcal{S}_h$. With the constant $C_{\rm inv}^{H,q} > 0$ in the inverse estimate (3.2) for $\mathcal{S}_H$, we assume

$$C_{\rm apx}^{h,p} C_{\rm inv}^{H,q} \max_{\gamma_j \in \mathcal{T}_H} (\|h\|_{L^\infty(\gamma_j)}/H_j)^{1/2} =: L < 1. \tag{5.4}$$

The analysis in the subsequent sections requires some additional regularity assumption on the exact solution, namely

$$u \in H^m(\mathcal{T}_H) \quad \text{for some } m > p + 1. \tag{5.5}$$

[illegible] 5.1. (i) The authors are aware that Assumption (5.5) is not reasonable for arbitrary Lipschitz screens Γ but requires some further (piecewise) smoothness of Γ so that $H^m(\mathcal{T}_H)$ is well defined [SaS].

(ii) Even for slit problems [HMS] with smooth right-hand side f in (1.1), the solution u does not satisfy (5.5).

5.2. The results. The main theorem in section 5.3 states the reliability and efficiency of η_M [up to terms of higher order]. The perhaps surprising result is that η_A is always reliable; cf. Corollary 5.3. The explicit estimator μ_A appears to be less costly compared with the other three since these involve the (approximate) computation of large full matrices; cf. section 8. The L^2-norm based estimators μ_M and μ_A immediately allow for local error indication for adaptive mesh-refining algorithms. Under weak additional assumptions [cf. section 5.4], we prove that η_M and μ_M, respectively, η_A and μ_A are equivalent, i.e., there are constants $C_7, C_8 > 0$ such that there holds

$$C_7\mu_M \le \eta_M \le C_8\mu_M \quad \text{and} \quad C_7\mu_A \le \eta_A \le C_8\mu_A.$$

In the case that $\mathcal{A}_H$ is the L^2-projection onto $\mathcal{S}_H$ and provided that $\mathcal{A}_H$ is H^1-stable, Theorem 5.6 states the efficiency of η_A and, in particular, of μ_A.

5.3. Reliability and efficiency of the error estimator η_M. The idea of the following argument goes back at least to an Oberwolfach conference in the eighties as the authors learned from L.B. Wahlbin, but it has not been applied to integral equations before.

THEOREM 5.1. [illegible] (5.4) [illegible]

$$|||u - u_h||| \le \frac{1}{1-L} |||(\mathbb{1} - \mathbb{G}_H)(u - u_h)|||. \tag{5.6}$$

[illegible]. Define $\widetilde{h} \in \mathcal{P}_0(\mathcal{T}_H)$ by $\widetilde{h}|_{\gamma_j} := \|h\|_{L^\infty(\gamma_j)}$ for each element $\gamma_j \in \mathcal{T}_H$. For the Galerkin error $e := u - u_h$, the approximation property of Π_h yields

$$|||(\mathbb{1} - \Pi_h)\mathbb{G}_H e||| \le C_{\rm apx}^{h,p} \|h^{1/2}\mathbb{G}_H e\|_{L^2(\Gamma)} \le C_{\rm apx}^{h,p} \|\widetilde{h}^{1/2}\mathbb{G}_H e\|_{L^2(\Gamma)}.$$

Now, we use the inverse estimate (3.2) for $H^{1/2}\mathbb{G}_H e \in \mathcal{P}_q(\mathcal{T}_H)$ and infer

$$\|\widetilde{h}^{1/2}\mathbb{G}_H e\|_{L^2(\Gamma)} \leq \|(\widetilde{h}/H)^{1/2}\|_{L^\infty(\Gamma)}\|H^{1/2}\mathbb{G}_H e\|_{L^2(\Gamma)} \leq C_{\mathrm{inv}}^{H,q}\|(\widetilde{h}/H)^{1/2}\|_{L^\infty(\Gamma)}|||\mathbb{G}_H e|||.$$

The combination with the best approximation property $|||\mathbb{G}_H e||| \leq |||e|||$ shows

$$|||(\mathbb{1} - \Pi_h)\mathbb{G}_H e||| \leq L|||e|||.$$

Using the Galerkin orthogonality and a Cauchy inequality, we obtain for $\Pi_h\mathbb{G}_H e \in \mathcal{S}_h$

$$\langle\!\langle e\,;\mathbb{G}_H e\rangle\!\rangle = \langle\!\langle e\,;\mathbb{G}_H e - \Pi_h\mathbb{G}_H e\rangle\!\rangle \leq |||e|||\,|||(\mathbb{1} - \Pi_h)\mathbb{G}_H e||| \leq L|||e|||^2.$$

Now, another Cauchy inequality leads to

$$|||e|||^2 = \langle\!\langle e\,;\mathbb{G}_H e\rangle\!\rangle + \langle\!\langle e\,;e - \mathbb{G}_H e\rangle\!\rangle \leq |||e|||\,\big(L|||e||| + |||(\mathbb{1} - \mathbb{G}_H)e|||\big).$$

This concludes the proof. □

5.2. The proof of Theorem 5.1 relies solely on the existence of an approximation operator $P_h : L^2(\Gamma) \to \mathcal{S}_h$ with $|||(\mathbb{1} - P_h)v_H||| \leq C_{\mathrm{apx}}\|h^{1/2}v_H\|_{L^2(\Gamma)}$ for all $v_H \in \mathcal{S}_H$. If Π_h is replaced by such an operator P_h, the richness assumption on $\mathcal{S}_h$ could be dropped.

5.3. Under the additional assumption $u \in L^2(\Gamma)$, $\mathbb{G}_H$ in Theorem 5.1 may be replaced by $\mathcal{A}_H$ since the proof only needs that $\mathcal{A}_H(u - u_h)$ is well defined. No further properties of $\mathbb{G}_H$ enter.

5.4. In contrast to residual-based techniques, we could not derive analogous results for collocation or qualocation schemes since the proof of Theorem 5.1 makes explicit use of the Galerkin orthogonality.

THEOREM 5.2. [illegible] (5.4) [illegible] (5.5) [illegible] η_M [illegible]

[illegible]

$$|||u - u_h||| \leq \frac{1}{1 - L}\big(\eta_M + |||(\mathbb{1} - \mathbb{G}_H)u|||\big) \tag{5.7}$$

[illegible] $0 < L < 1$ [illegible] 5.1 [illegible]

$$\eta_M \leq |||u - u_h||| + |||(\mathbb{1} - \mathbb{G}_H)u|||. \tag{5.8}$$

[illegible] $|||u - u_h|||$ [illegible] η_M [illegible] $|||(\mathbb{1} - \mathbb{G}_H)u|||$ [illegible]

[illegible]. According to (3.4), the error in the energy norm is of order

$$|||u - u_h||| = O(h_{\max}^{\widehat{p}}) \quad \text{with} \quad \widehat{p} := \min\{p+1, m\} + 1/2 = p + 3/2.$$

Furthermore, for smooth u we have

$$|||(\mathbb{1} - \mathbb{G}_H)u||| = O(H_{\max}^{\widehat{q}}) \quad \text{with} \quad \widehat{q} := \min\{q+1, m\} + 1/2 \geq \min\{p+2, m\} + 1/2.$$

Since $m > p + 1$, $|||(\mathbb{1} - \mathbb{G}_H)u|||$ is of higher order. Theorem 5.1 combined with a triangle inequality shows

$$|||u - u_h||| \leq \frac{1}{1 - L}\big(|||(\mathbb{1} - \mathbb{G}_H)u||| + |||(\mathbb{1} - \mathbb{G}_H)u_h|||\big)$$

and proves the reliability. A simple triangle inequality shows the efficiency

$$\eta_M \leq |||(\mathbb{1} - \mathbb{G}_H)(u - u_h)||| + |||(\mathbb{1} - \mathbb{G}_H)u|||,$$

since $(\mathbb{1} - \mathbb{G}_H)$ is an orthogonal projection with respect to the energy norm. □

5.4. Reliability of the error estimators η_A, μ_A, and μ_M. Replacing the best approximation operator $\mathbb{G}_H$ by another operator $\mathcal{A}_H$, we immediately obtain the following corollary.

COROLLARY 5.3. $\eta_M \le \eta_A$ (5.4) (5.5)
η_A □

This and an application of Lemma 4.3 proves the reliability of μ_A and μ_M.

COROLLARY 5.4. $\mathcal{A}_H : L^2(\Gamma) \to \mathcal{S}_H \subseteq L^2(\Gamma)$ (4.1)

$$\eta_A \le C_{\rm apx}^{H,q}\mu_A. \tag{5.9}$$

μ_M (5.4) (5.5)
$\mathcal{P}_0(\mathcal{T}_H) \subseteq \mathcal{S}_H$ $\mathcal{P}_1(\mathcal{T}_H) \cap \mathcal{C}(\Gamma) \subseteq \mathcal{S}_H$ □

An application of the inverse estimate (3.2) shows that μ_A is efficient if and only if η_A is efficient.

COROLLARY 5.5. $C_9 := \max_{\Gamma_j \in \mathcal{T}_h} (\|H\|_{L^\infty(\Gamma_j)}/h_j)^{1/2}$ $\mathcal{S}_H \subseteq \mathcal{P}_q(\mathcal{T}_h)$

$$\mu_A \le C_9 C_{\rm inv}^{h,q}\eta_A. \tag{5.10}$$

μ_M □

5.5. Note that the assumption $\mathcal{S}_H \subseteq \mathcal{P}_q(\mathcal{T}_h)$ is quite weak and, in particular, satisfied if $\mathcal{T}_h$ is obtained by refinements of $\mathcal{T}_H$.

5.5. Efficiency of η_A and μ_A for the L^2-projection. For the remaining part of this section, we suppose that the interpolation operator $\mathcal{A}_H = \Pi_H$ is just the L^2-projection onto $\mathcal{S}_H \subseteq H^1(\Gamma)$.

DEFINITION 2. L^2 $\mathcal{A}_H : L^2(\Gamma) \to \mathcal{S}_H$ $\mathcal{S}_H$ H^1-stable

$$\|\mathcal{A}_H v\|_{H^1(\Gamma)} \le C_{10}\|v\|_{H^1(\Gamma)} \quad v \in H^1(\Gamma), \tag{5.11}$$

$C_{10} > 0$ Γ, q $\mathcal{T}_H$

THEOREM 5.6. (5.4) (5.5) H^1 L^2 $\mathcal{A}_H : L^2(\Gamma) \to \mathcal{S}_H$ $\mathcal{S}_H$ η_A

. Define the operator $T_\alpha = (\mathbb{1} - \mathcal{A}_H) : H^\alpha(\Gamma) \to H^\alpha(\Gamma)$ which is linear and continuous for $\alpha = 0$ and $\alpha = 1$, respectively. Interpolation yields continuity of $T_{1/2}$ with operator norm $\|T_{1/2}\| \le \|T_1\|^{1/2}$ since $\|T_0\| = 1$. We consider the adjoint operator $T_{1/2}^* : \widetilde{H}^{-1/2}(\Gamma) \to \widetilde{H}^{-1/2}(\Gamma)$ defined formally by

$$\langle T_{1/2}^* w\,; v\rangle = \langle w\,; T_{1/2} v\rangle \quad \text{for } w \in \widetilde{H}^{-1/2}(\Gamma), v \in H^{1/2}(\Gamma). \tag{5.12}$$

Notice that, for $w \in L^2(\Gamma)$, we have $T_{1/2}^* w = T_{1/2} w$ according to the symmetry of the orthogonal projection T_0 and the L^2 scalar product on the right-hand side of (5.12) in case $w \in L^2(\Gamma)$. Since $T_{1/2}^*$ is continuous, let $C_{11} < \infty$ denote the operator norm of $T_{1/2}^*$ with respect to the energy norm. As above, a simple triangle inequality shows

$$\eta_A \le |\!|\!|(\mathbb{1} - \mathcal{A}_H)(u - u_h)|\!|\!| + |\!|\!|(\mathbb{1} - \mathcal{A}_H)u|\!|\!| \le C_{11}|\!|\!|u - u_h|\!|\!| + |\!|\!|(\mathbb{1} - \mathcal{A}_H)u|\!|\!|, \tag{5.13}$$

where we have used $u-u_h \in L^2(\Gamma)$ and $T^*_{1/2}(u-u_h) = (\mathbb{1}-\mathcal{A}_H)(u-u_h)$. Theorem 4.1 and a standard approximation result in L^2 yield

$$|||(\mathbb{1}-\mathcal{A}_H)u||| \le C_{\mathrm{apx}} H_{\max}^{1/2} \|(\mathbb{1}-\mathcal{A}_H)u\|_{L^2(\Gamma)} = O(H_{\max}^{\widehat{q}+1/2}) \quad \text{with} \quad \widehat{q} := \min\{q+1, m\}.$$

Therefore, the last term in (5.13) is of higher order; cf. the proof of Theorem 5.2. ∎

The efficiency of μ_A follows from the previous section.

COROLLARY 5.7. $\mathcal{P}_1(\mathcal{T}_H) \cap \mathcal{C}(\Gamma) \subseteq \mathcal{S}_H \subseteq \mathcal{P}_q(\mathcal{T}_h)$

$$\mu_A \le C_9 C_{\mathrm{inv}}^{h,q} \eta_A \qquad \eta_A \le C_{\mathrm{apx}}^{H,q} \mu_A, \tag{5.14}$$

μ_A 5.6 ∎

5.6. For special cases it is easy to derive the efficiency of η_A although $\mathcal{S}_H \not\subseteq H^1(\Gamma)$. For instance, assume $\mathcal{S}_h = \mathcal{P}_0(\mathcal{T}_h)$ and $\mathcal{S}_H = \mathcal{P}_1(\mathcal{T}_H) \subseteq \mathcal{P}_1(\mathcal{T}_h)$. Suppose that $\mathcal{A}_H$ is the L^2-projection onto $\mathcal{S}_H$ and Π_H is the L^2-projection onto $\mathcal{P}_1(\mathcal{T}_H) \cap \mathcal{C}(\Gamma)$. An application of the derived results shows

$$\eta_A \lesssim \|H^{1/2}(\mathbb{1}-\mathcal{A}_H)u_h\|_{L^2(\Gamma)} \le \|H^{1/2}(\mathbb{1}-\Pi_H)u_h\|_{L^2(\Gamma)} \lesssim |||u-u_h|||,$$

where $\lesssim$ denotes $\le$ up to a mesh size independent constant.

6. Numerical realization in two dimensions. For the numerical experiments, we choose $\mathcal{S}_h = \mathcal{P}_0(\mathcal{T}_h)$ and $\mathcal{S}_H = \mathcal{P}_1(\mathcal{T}_H)$. The finer mesh $\mathcal{T}_h$ is obtained from the coarser $\mathcal{T}_H$ by uniform refinements; cf. section 6.3 for details. This ensures $\mathcal{S}_H = \mathcal{P}_1(\mathcal{T}_H) \subseteq \mathcal{P}_1(\mathcal{T}_h)$. The approximation $u_h \in \mathcal{S}_h$ is then defined by $u_h|_{\Gamma_j} = \mathbf{x}_j$, where $\mathbf{x} \in \mathbb{R}^n$ is the solution of a linear system

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{6.1}$$

In all numerical experiments we consider the error estimators η_M, μ_M based on the Galerkin projection $\mathbb{G}_H$ and the error estimators η_A, μ_A based on the L^2-projection $\mathcal{A}_H = \Pi_H$.

6.1. Poisson problem and Symm's integral equation. In the numerical experiments we consider three examples where the right-hand side f in (1.1) comes from a Poisson problem

$$\Delta U = 0 \text{ in } \Omega \quad \text{and} \quad U = g \text{ on } \Gamma = \partial\Omega \tag{6.2}$$

with given Dirichlet data g on Γ [plus boundary conditions at infinity if Ω is unbounded]. This problem is equivalent to Symm's integral equation (1.1), where f takes the form $f = (K+\mathbb{1})g$ with the double-layer potential operator K, defined as Cauchy principal value by

$$(Kg)(x) := -\frac{1}{\pi} \int\!\!\!\!\!\!- _{\Gamma} g(y) \frac{(y-x)\cdot n(y)}{|x-y|^2}\, ds_y \quad \text{for } x \in \Gamma. \tag{6.3}$$

Then, the exact solution of (1.1) is just the normal derivative $u = \partial U/\partial n$ of U on the boundary Γ. Notice that Kg vanishes on Γ whenever Γ is a slit.

6.2. Computation of the discrete solution. The coefficients of the stiffness matrix $\mathbf{A} \in \mathbb{R}^{n\times n}_{sym}$ are computed by

$$\mathbf{A}_{jk} := \langle\!\langle \chi_j \,;\, \chi_k \rangle\!\rangle = -\frac{1}{\pi} \int_{\Gamma_j} \int_{\Gamma_k} \log|x-y| \, ds_y ds_x. \tag{6.4}$$

Here, χ_j denotes the characteristic function of the set $\Gamma_j \subseteq \mathbb{R}^2$ [i.e., $\chi_j(x) = 1$ if $x \in \Gamma_j$ and $\chi_j(x) = 0$ else]. The right-hand side $\mathbf{b} \in \mathbb{R}^n$ is given by

$$\mathbf{b}_j := \int_{\Gamma_j} f(x) \, ds_x \tag{6.5}$$

with f from (1.1). While $\mathbf{A}_{jk}$ can be computed analytically [Ma], the computation of $\mathbf{b}_j$ involves proper quadrature rules—in particular, if the right-hand side is induced by a Poisson problem; cf. [CP1, section 6].

6.3. Adaptive algorithm. All mesh refinements are performed with the following adaptive algorithm based on the refinement indicators $\mu_{M,j}$ or $\mu_{A,j}$ defined as follows: Given the coarse mesh $\mathcal{T}_H = \{\gamma_1, \dots, \Gamma_N\}$ and the L^2-projection $\Pi_H : \mathcal{P}_0(\mathcal{T}_h) \to \mathcal{P}_1(\mathcal{T}_H)$, we define

(6.6)

$$\mu_{M,j} := H_j^{1/2} \|(\mathbb{1} - \mathbb{G}_H) u_h\|_{L^2(\gamma_j)}, \quad \text{respectively,} \quad \mu_{A,j} := H_j^{1/2} \|(\mathbb{1} - \Pi_H) u_h\|_{L^2(\gamma_j)} \quad \text{for } j = 1, \dots, N,$$

i.e., $\mu_M = \big(\sum_{j=1}^N \mu_{M,j}^2\big)^{1/2}$ and $\mu_A = \big(\sum_{j=1}^N \mu_{A,j}^2\big)^{1/2}$, respectively.

ALGORITHM 6.1. [illegible] $\mathcal{T}_H^{(0)}$ $k = 0$ $\ell \in \mathbb{N}_{\geq 2}$ $0 \leq \theta \leq 1$

(i) [illegible] $\mathcal{T}_h^{(k)} = \{\Gamma_1, \dots, \Gamma_n\}$ [illegible] $\mathcal{T}_H^{(k)} = \{\gamma_1, \dots, \gamma_N\}$ [illegible] $\gamma_j \in \mathcal{T}_H^{(k)}$ [illegible] ℓ [illegible]

(ii) [illegible] $u_h^{(k)}$ [illegible] $\mathcal{T}_h^{(k)}$

(iii) [illegible] η_M η_A [illegible] $\mu_{M,j}$ $\mu_{A,j}$

(iv) [illegible] γ_j [illegible] $\mu_{M,j} \geq \theta \max\{\mu_{M,1}, \dots, \mu_{M,N}\}$ $\mu_{A,j} \geq \theta \max\{\mu_{A,1}, \dots, \mu_{A,N}\}$ [illegible]

(v) [illegible] $\gamma_j \in \mathcal{T}_H^{(k)}$ [illegible] $\mathcal{T}_H^{(k+1)}$ [illegible] (i)

7. Numerical experiments in two dimensions. This section reports on some numerical experiments to study the accuracy of the introduced error estimators and the performance of the proposed adaptive strategy. All computations are done with MATLAB. Example 7.1 corresponds to a Dirichlet problem with smooth solution such that the smoothness assumptions in section 5 are satisfied and guarantee reliability and efficiency of our error estimators. Examples 7.2–7.4 are taken from [CP1] and [ChS], respectively. They have been realized and studied in [CP1] for adaptive mesh-refinement with respect to the residual-based error estimator introduced by Faermann [F1, F2, F3]; cf. (7.3). Example 7.2 considers Symm's integral equation corresponding to a Dirichlet problem on the L-shaped domain. The exact solution is known and has a singularity at the reentrant corner. Therefore, adaptivity is necessary to retain the optimal convergence rate. Example 7.3 deals with a constant

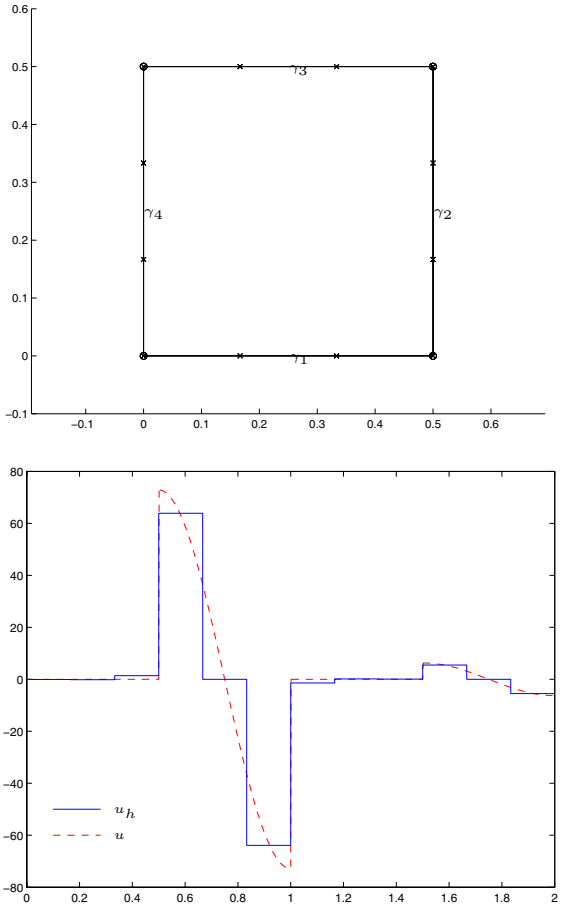


FIG. 1. *Initial coarse mesh* $\mathcal{T}_H^{(0)}$ *in Poisson problem* 7.1 *with* $N = 4$ *elements* [*nodes indicated by* o] (*top*) *and the corresponding refined mesh* $\mathcal{T}_h^{(0)}$ *for* $\ell = 3$ *in Algorithm* 6.1, *i.e.* $n = 12$ [*nodes indicated by* x]. *The related discrete solution* u_h *on* $\mathcal{T}_h^{(0)}$ (*bottom*) *is plotted over the arclength* $s = 0, \dots, 2$ *of* Γ. *The exact solution* u *from* (7.6) *is shown for comparison* (*bottom*); u *is piecewise smooth and jumps together with the outer normal in the corners of* Γ [*for arclength parameters* $s = 1/2, 1, 3/2,$ *and* 2].

right-hand side to exclude positive and negative effects due to quadrature errors. The exact solution is unknown. The sequence of discrete solutions shows singularities at the five rectangular corners of the L-shape. Example 7.4 taken from [ChS] considers a slit problem, where the known exact solution u lacks almost any smoothness, more precisely $u \notin L^2(\Gamma)$. Finally, Example 7.5 involves the approximation of a smooth boundary by a polygonal boundary. We approximate a smooth eigenfunction of the single-layer potential on the sphere with radius 1/2.

Preliminaries. Uniform and adaptive meshes as well as plots of discrete and exact solutions u_h and u are shown in Figures 1, 4, and 10 as functions of its arclength parametrization s, $0 \le s \le \text{length}(\Gamma) = 2$. Furthermore, errors and estimators are considered for the energy norm. Various numbers are provided in Tables 1, 2, and 3 and even more convergence results are visualized in Figures 2, 5–7, 9, 11, 13, and

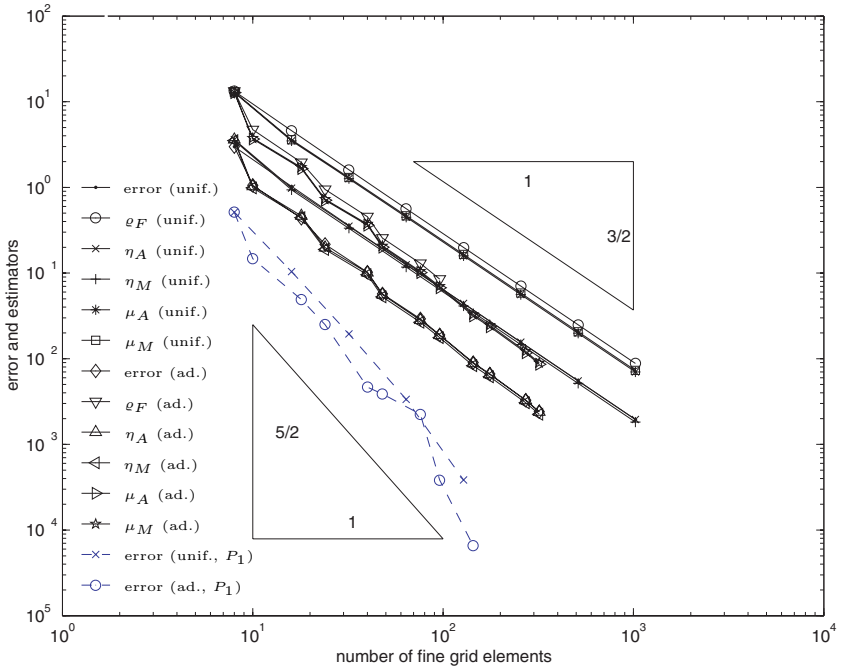


FIG. 2. *Error and error estimators η_M, η_A, μ_M, and μ_A for uniform* [*indicated by* unif.] *and μ_A-adaptive* [*indicated by* ad.] *mesh refinement in Poisson problem* 7.1 *and $\ell = 2$ in Algorithm* 6.1. *The $\mathcal{P}_0$ boundary element method leads to the optimal convergence rate $h^{3/2}$. For comparison, also the error for the $\mathcal{P}_1$ boundary element method* [*dashed lines*] *on the generated meshes* [*i.e., uniform, resp., μ_A-adaptive*] *is shown. As expected, one again observes the optimal experimental convergence rate $h^{5/2}$.*

16. Some adaptively generated meshes are shown in Figures 3, 12, and 14.

We compute the experimental convergence rate of the error

$$E^{(k)} := |||u - u_h^{(k)}||| = \big(|||u|||^2 - |||u_h^{(k)}|||^2\big)^{1/2} \tag{7.1}$$

by the formula

$$\kappa^{(k)} = \log(E^{(k-1)}/E^{(k)}) \,/\, \log(n^{(k)}/n^{(k-1)}). \tag{7.2}$$

Here, $n^{(k)} = \mathrm{card}(\mathcal{T}_h^{(k)})$ denotes the number of elements. The energy norm of the discrete solution u_h reads $|||u_h|||^2 = \mathbf{x}\cdot\mathbf{A}\mathbf{x}$ with the stiffness matrix $\mathbf{A}$ and the coefficient vector $\mathbf{x}$ introduced in section 6. The energy norm $|||u|||$ of the exact solution is either computed exactly or obtained by Aitkin's Δ^2 extrapolation of the sequence of values for discrete solutions on uniformly refined meshes.

For comparison, in Figures 2, 5–7, 11, 13, and 16 we also show the residual-based error estimator ϱ_F introduced by Faermann,

$$\varrho_F := \left(\sum_{j=1}^{n} \varrho_{F,j}^2\right)^{1/2}, \tag{7.3}$$

where the refinement indicators

TABLE 1

Experimental error, error estimator η_M and η_A, and convergence rates for Poisson problem 7.1, *$\ell = 2$, and uniform (top), respectively, μ_A-adaptive mesh refinement (bottom).*

Uniform mesh refinement							
k	n	E	η_M	η_A	η_M/E	η_M/η_A	κ
0	8	2.9820e+00	3.5112e+00	3.6134e+00	1.18	0.97	
1	16	1.0276e+00	9.3703e-01	9.8030e-01	0.91	0.96	1.54
2	32	3.5630e-01	3.3380e-01	3.5308e-01	0.94	0.95	1.53
3	64	1.2519e-01	1.1730e-01	1.2496e-01	0.94	0.94	1.51
4	128	4.4187e-02	4.1356e-02	4.4168e-02	0.94	0.94	1.50
5	256	1.5615e-02	1.4609e-02	1.5614e-02	0.94	0.94	1.50
6	512	5.5202e-03	5.1640e-03	5.5201e-03	0.94	0.94	1.50
7	1024	1.9514e-03	1.8256e-03	1.9516e-03	0.94	0.94	1.50
Adaptive mesh refinement							
k	n	E	η_M	η_A	η_M/E	η_M/η_A	κ
0	8	2.9820e+00	3.5112e+00	3.6134e+00	1.18	0.97	
1	10	1.0570e+00	9.8846e-01	1.0292e+00	0.94	0.96	4.65
2	18	4.3445e-01	4.4375e-01	4.6590e-01	1.02	0.95	1.51
3	24	2.1601e-01	1.8876e-01	1.9815e-01	0.87	0.95	2.43
4	40	1.0226e-01	9.5542e-02	1.0109e-01	0.93	0.95	1.46
5	48	5.7202e-02	5.2676e-02	5.5940e-02	0.92	0.94	3.19
6	76	2.9243e-02	2.6660e-02	2.8243e-02	0.91	0.94	1.46
7	96	1.8958e-02	1.7670e-02	1.8807e-02	0.93	0.94	1.86
8	144	9.0153e-03	8.2982e-03	8.8322e-03	0.92	0.94	1.83
9	176	6.6515e-03	6.1626e-03	6.5680e-03	0.93	0.94	1.52
10	272	3.2840e-03	3.0642e-03	3.2651e-03	0.93	0.94	1.62

TABLE 2

Experimental error and error estimator μ_M, μ_A, and ϱ_F for Poisson problem 7.1, *$\ell = 2$, and uniform (top), respectively, μ_A-adaptive mesh refinement (bottom).*

Uniform mesh refinement									
k	n	E	μ_M	μ_A	ϱ_F	μ_M/E	μ_A/E	μ_M/μ_A	ϱ_F/E
0	8	2.982e+00	1.302e+01	1.278e+01	1.328e+01	4.37	4.29	1.02	4.45
1	16	1.028e+00	3.592e+00	3.496e+00	4.593e+00	3.50	3.40	1.03	4.47
2	32	3.563e-01	1.307e+00	1.264e+00	1.602e+00	3.67	3.55	1.03	4.50
3	64	1.252e-01	4.655e-01	4.484e-01	5.638e-01	3.72	3.58	1.04	4.50
4	128	4.419e-02	1.649e-01	1.586e-01	1.991e-01	3.73	3.59	1.04	4.51
5	256	1.562e-02	5.835e-02	5.607e-02	7.036e-02	3.74	3.59	1.04	4.51
6	512	5.520e-03	2.063e-02	1.982e-02	2.487e-02	3.74	3.59	1.04	4.51
7	1024	1.951e-03	7.295e-03	7.009e-03	8.794e-03	3.74	3.59	1.04	4.51
		Adaptive mesh refinement							
k	n	E	μ_M	μ_A	ϱ_F	μ_M/E	μ_A/E	μ_M/μ_A	ϱ_F/E
0	8	2.982e+00	1.302e+01	1.278e+01	1.328e+01	4.37	4.29	1.02	4.45
1	10	1.057e+00	3.755e+00	3.667e+00	4.721e+00	3.55	3.47	1.02	4.47
2	18	4.344e-01	1.710e+00	1.660e+00	1.952e+00	3.94	3.82	1.03	4.49
3	24	2.160e-01	7.278e-01	7.072e-01	9.458e-01	3.37	3.27	1.03	4.38
4	40	1.023e-01	3.738e-01	3.616e-01	4.546e-01	3.66	3.54	1.03	4.45
5	48	5.720e-02	2.078e-01	2.005e-01	2.560e-01	3.63	3.51	1.04	4.47
6	76	2.924e-02	1.046e-01	1.011e-01	1.297e-01	3.58	3.46	1.03	4.43
7	96	1.896e-02	6.998e-02	6.742e-02	8.484e-02	3.69	3.56	1.04	4.48
8	144	9.015e-03	3.287e-02	3.167e-02	4.026e-02	3.65	3.51	1.04	4.47
9	176	6.651e-03	2.448e-02	2.356e-02	2.979e-02	3.68	3.54	1.04	4.48
10	272	3.284e-03	1.216e-02	1.171e-02	1.472e-02	3.70	3.57	1.04	4.48

FIG. 3. *Adaptive mesh refinement in Poisson problem* 7.1 *related to* $\ell = 2$ *and error estimator* μ_A. *Adaptive mesh refinement with respect to* μ_M *leads to the same meshes.*

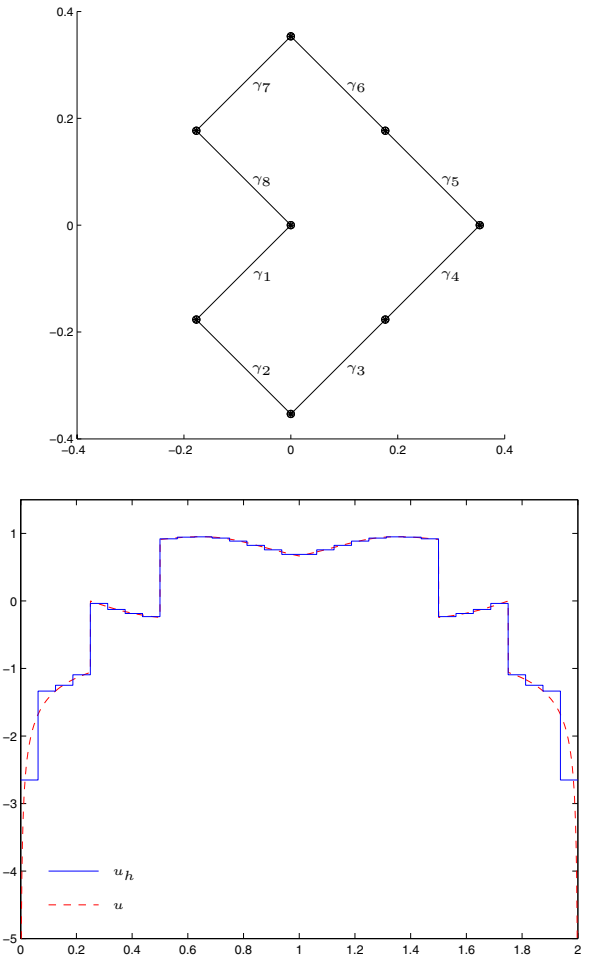


FIG. 4. *Initial coarse mesh* $\mathcal{T}_H^{(0)}$ *in Poisson problems* 7.2 *and* 7.3 *with* $N = 8$ *elements* (*top*) *and the corresponding discrete solution* u_h *on* $\mathcal{T}_h^{(0)}$ *for* $\ell = 4$, *i.e.*, $n = 32$ *fine grid elements, in Poisson problem* 7.2 (*bottom*). *The exact solution* u *from* (7.8) *is shown for comparison* (*bottom*).

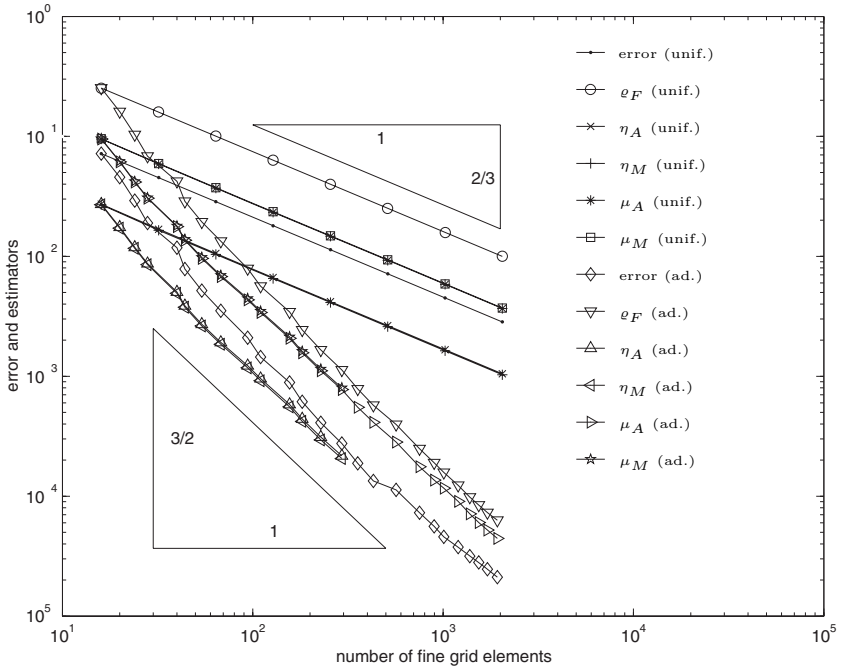


FIG. 5. *Error and error estimators* η_M, η_A, μ_M, *and* μ_A *for uniform* [*indicated by* unif.] *and* μ_A*-adaptive* [*indicated by* ad.] *mesh refinement in Poisson problem* 7.2 *and* $\ell = 2$ *in Algorithm* 6.1.

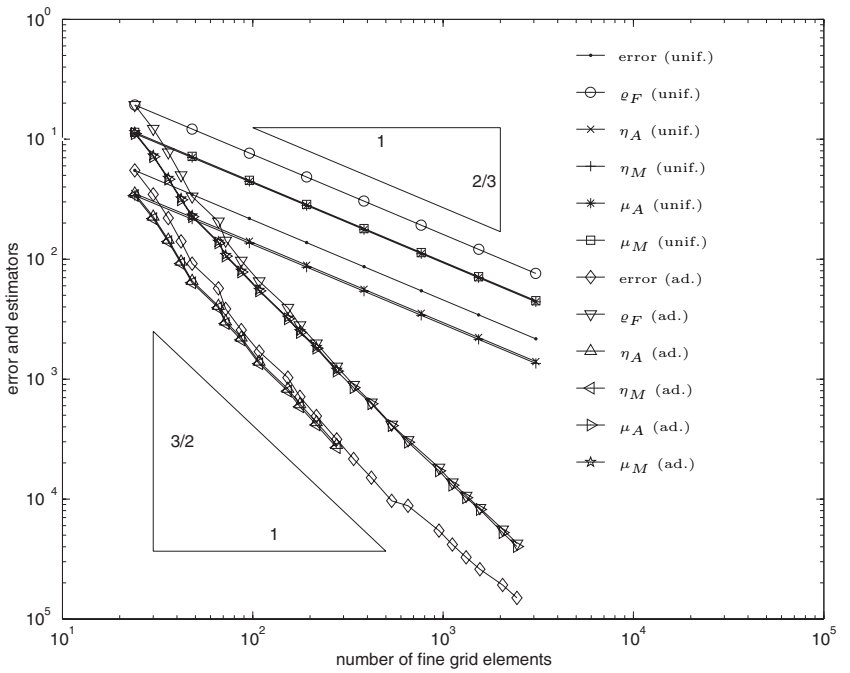


FIG. 6. *Error and error estimators* η_M, η_A, μ_M, *and* μ_A *for uniform* [*indicated by* unif.] *and* μ_A*-adaptive* [*indicated by* ad.] *mesh refinement in Poisson problem* 7.2 *and* $\ell = 3$ *in Algorithm* 6.1.

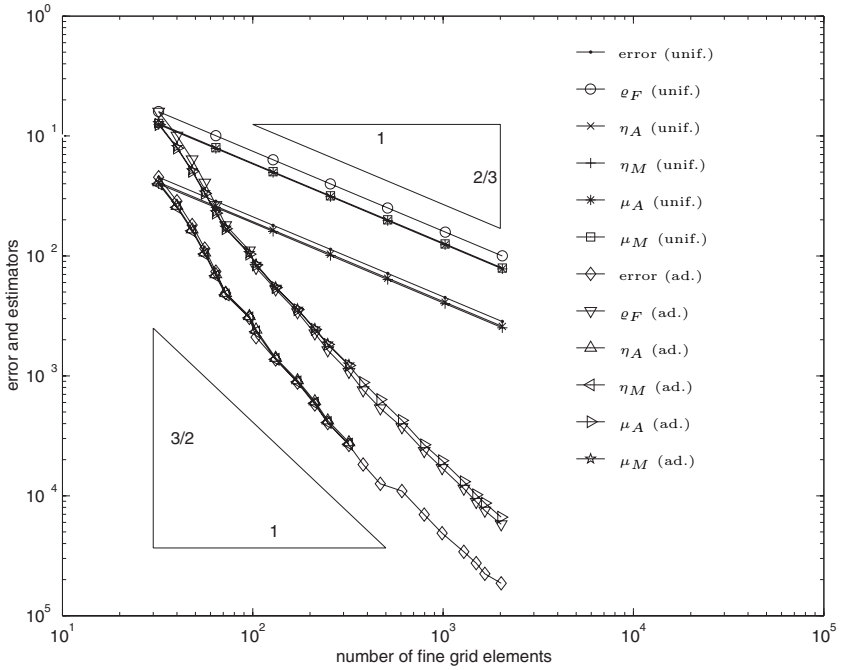


FIG. 7. *Error and error estimators η_M, η_A, μ_M, and μ_A for uniform* [*indicated by* unif.] *and μ_A-adaptive* [*indicated by* ad.] *mesh refinement in Poisson problem* 7.2 *and $\ell = 4$ in Algorithm* 6.1.

$$\varrho_{F,j} := \int_{\Gamma_j \cup \Gamma_{j+1}} \int_{\Gamma_j \cup \Gamma_{j+1}} \frac{|r_h(x) - r_h(y)|^2}{|x-y|^2}\, ds_y ds_x \tag{7.4}$$

are local Sobolev–Slobodeckij seminorms of the residual $r_h := f - Vu_h \in H^{1/2}(\Gamma)$. This estimator is known to be reliable and efficient for the Galerkin method. Details on the numerical realization of these double boundary integrals are provided in [CP1]. Since the reliability and efficiency of this estimator depends in 2D on the local mesh ratio

$$\kappa(\mathcal{T}_h) = \max\{h_{T_j}/h_{T_k} \,:\, T_j, T_k \in \mathcal{T}_h \text{ neighbors}\}, \tag{7.5}$$

we extend the marking strategy in (iv) of Algorithm 6.1 to ensure that there holds $\kappa(\mathcal{T}_H) \le 2$, i.e., that the mesh size of two neighboring coarse elements varies at most by 2.

7.1. Poisson-problem with smooth solution. We consider problem (6.2) on the halved unit square $\Omega = [0, 1/2]^2$ with exact solution $U(x, y) = \sinh(2\pi\, x)\, \cos(2\pi\, y)$ and solve the corresponding Symm's integral equation $Vu = f$ with right-hand side $f = (K + 1)g$ and $g = U|_\Gamma$. Then, $u = \partial U/\partial n$ is smooth on each affine boundary piece,

$$u(x, y) = 2\pi \begin{pmatrix} \cosh(2\pi x)\, \cos(2\pi y) \\ -\sinh(2\pi x)\, \sin(2\pi y) \end{pmatrix} \cdot n(x, y) \tag{7.6}$$

with the outer normal $n(x, y)$ of Ω on $\Gamma = \partial\Omega$, and we expect to obtain optimal convergence rate $\mathcal{O}(n^{3/2})$ even for uniform mesh refinement. Notice that u vanishes on $[0, 1/2] \times \{0, 1/2\}$, i.e., on the lower and upper boundary of Ω.

Figure 1 shows the initial coarse mesh $\mathcal{T}_H^{(0)}$, the exact solution u from (7.6) plotted over the arclength $s = 0, \dots, 2$, and a discrete solution u_h corresponding to $\mathcal{T}_h^{(0)}$ with $\ell = 3$ in Algorithm 6.1. As can be seen, u vanishes on the parameter intervals $0 \le s \le 1/2$ and $1 \le s \le 3/2$. On the other two affine boundary pieces, u is smooth and jumps with the normal vector in the corners of Γ.

Figure 2 shows errors and error estimators for both uniform and μ_A-adaptive mesh refinement and $\ell = 2$ in Algorithm 6.1. All values are shown on a log-log scale so that the experimental convergence rates (7.2) of both the error and the estimators are visible as the slope of the corresponding curves. The error $E = |||u - u_h|||$ is computed via (7.1), where the energy norm of the exact solution is obtained by extrapolation, $|||u|||^2 = 162.1448097$. As is to be expected from the regularity of u, uniform mesh refinement leads to the optimal experimental convergence rate $h^{3/2}$. The curve for the error on adaptively generated meshes is parallel, but the absolute values are improved by a factor of 5. This improvement can be explained by the fact that the adaptive algorithm mainly resolves the strong growth of the function within the arclength parameter intervals $1/2 \le s \le 1$ and $3/2 \le s \le 2$, where u does not vanish; cf. Figure 1. The corresponding adaptive meshes are shown in Figure 3. For comparison, Figure 2 also shows the error for the $\mathcal{P}_1$ boundary element methods on the same meshes, since this term corresponds to the higher-order terms of our error estimates. As expected, the $\mathcal{P}_1$-error shows the optimal experimental convergence rate of $h^{5/2}$.

For both $\mathcal{P}_0$ and $\mathcal{P}_1$ boundary element methods, the error shows no preasymptotic behavior but converges with optimal convergence order from the very start. This is reflected by the error estimation of the $\mathcal{P}_0$ error $E = |||u - u_h|||$ by the error estimators η_M and η_A. The error is very sharply estimated: The values of both estimators almost coincide with the corresponding error value. This is also underlined by several numbers in Table 1: The quotients η_M/E, respectively, η_M/η_A converge to approximately 0.94 and 0.93 for both uniform and adaptive mesh refinement. The performance of the L^2-error estimators μ_M and μ_A is examined in Table 2. The error estimator μ_M and μ_A almost coincide. The quotient μ_M/μ_A stays bounded by approximately 1.04. The error is overestimated by a factor 3.5–3.7. For comparison, note that the Faermann error estimator ϱ_F overestimates the error by a factor 4.4, 4.5.

7.2. Poisson-problem on L-shaped domain. For a fixed parameter $\alpha > 0$ we consider Problem (6.2) on the L-shaped domain Ω shown in Figure 4 with exact solution

$$U(x) = r^\alpha \cos(\alpha\varphi) \quad \text{in polar coordinates} \quad x = r\,(\cos\varphi, \sin\varphi). \tag{7.7}$$

Then, the exact solution u of Symm's integral equation reads in polar coordinates

$$u(x) = (w \cdot n(x))\,\alpha r^{\alpha-1} \quad \text{with } w := \begin{pmatrix} \cos(\varphi)\cos(\alpha\varphi) + \sin(\varphi)\sin(\alpha\varphi) \\ \sin(\varphi)\cos(\alpha\varphi) - \cos(\varphi)\sin(\alpha\varphi) \end{pmatrix} \in \mathbb{R}^2. \tag{7.8}$$

For the numerical experiment we choose $\alpha = 2/3$. The Poisson problem then leads to $U \notin H^2(\Omega)$. Aitkin's Δ^2 method gives $|||u|||^2 = 0.4041161973$.

The initial coarse mesh $\mathcal{T}_H^{(0)}$ with $N = 8$ equisized elements and the corresponding discrete solution u_h for $\ell = 4$ in Algorithm 6.1 as well as the exact solution are shown in Figure 4. Here, u and u_h are shown as plots over the arc length. The singularity of u at $(0, 0)$ is visible at arc-length parameter $s = 0$ and $s = 2$ by periodicity.

To illustrate the performance of the proposed adaptive algorithm, we run Algorithm 6.1 with $\mathcal{T}_0$ from Figure 4, $\ell = 2, 3, 4$, and $\theta = 0, 1/2$. The values of the error and the error estimators are visualized in Figures 5–7 and 9. The error $E = \|\!|u - u_h|\!\|$ is computed via (7.1), where the energy norm of the exact solution is obtained by extrapolation with Aitkin's Δ^2 method, $\|\!|u|\!\|^2 = 0.4041161973$. For any choice of ℓ, the uniform mesh refinement, i.e., $\theta = 0$, leads to suboptimal convergence rate $2/3$ for the error caused by a corner singularity of u which can be predicted theoretically. The fact that the slope of the corresponding error estimators even is $2/3$ gives empirical evidence that the estimators are reliable and efficient although the solution lacks the regularity assumed in section 5. The error estimators η_M and η_A, respectively, μ_M and μ_A almost coincide. Due to numerical instabilities in computing the stiffness matrix for the $\mathcal{P}_1$ boundary element method, the error estimators μ_M, η_M, and η_A are only computed up to an error of $10^{-7/2}$. Up to this breakdown of the computation, the μ_M- and μ_A-adaptive meshes coincide. Thus, for large numbers n of fine grid elements, Figures 5–7 only show the error and the error estimators ϱ_F and μ_A.

The proposed adaptive mesh-refining strategy retains the optimal convergence rate $3/2$ expected for smooth solutions; cf. section 7.1. This observation is independent of the choice of ℓ in Algorithm 6.1. The larger ℓ, the better is the estimation of the error by the error estimators η_A and η_M which can be expected from the involved constant L in Theorems 5.1 and 5.2; cf. (5.4) for the definition. We always have $\eta_M \lesssim \|\!|u - u_h|\!\| \lesssim 1/(1 - L)\,\eta_M$, where $\lesssim$ denotes an inequality up to terms of higher order. By definition of our mesh strategy in Algorithm 6.1, there holds $\|h\|_{L^\infty(\gamma_j)}/H_j = 1/\ell$, whence $L \to 0$ for $\ell \to \infty$. On the other hand, ℓ should not be chosen too large since the asymptotic behavior of $h^{3/2}$ and $H^{5/2}$ then might not be visible for the number of elements considered. However, the effect is visible for $\ell = 2, 3, 4$ as well; cf. Figures 5–7. For $\ell = 2$, the curves of η_M, η_A, and E are getting closer for $H, h \to 0$, and it is expected that the curves almost coincide for larger values of n. For $\ell = 3$, the error and the error estimators η_M and η_A seem to coincide for $n \geq 200$. Finally, for $\ell = 4$, the error is sharply estimated by η_M and η_A even for lower degrees of freedom; cf. Figure 7.

Figure 8 shows the sequence of (μ_A-) adaptively generated meshes $\mathcal{T}_H$. We observe the expected mesh refinement towards the reentrant corner at the endpoints 0 and 2 and a moderate refinement elsewhere. The sequence of adaptive meshes for $\ell = 2, 3$ looks similar. Figure 9 shows the error and the error estimator μ_A for uniform and μ_A-adaptive mesh refinement and, for comparison, $\ell = 2, 3, 4$. Note that not only the slope of the errors for different choices of ℓ but even the absolute values coincide asymptotically.

Finally, Table 3 shows the calculated values of the error $E = \|\!|u - u_h|\!\|$, the error estimator, and the corresponding experimental convergence rate κ for uniform and μ_A-adaptive mesh refinement and $\ell = 4$. The values underline the given interpretation of the figures. For uniform mesh refinement, μ_A overestimates the error by a factor 2.75 independent of the number n of fine grid elements. For adaptively generated meshes, the quotient μ_A/E varies between 2.75 and 5 and seems to converge to a value about 3 for large n. This underlines the empirical efficiency and reliability of μ_A as is proven for smooth solutions in section 5. For the error estimators μ_M and μ_A, the quotient μ_M/μ_A is about 1 and both error estimators almost coincide. This underlines why both refinement strategies lead to the same adaptive mesh refinement.

7.3. Symm's integral equation with constant right-hand side. The boundary integral equation $V\phi = 1$ with constant right-hand side is considered for the L-

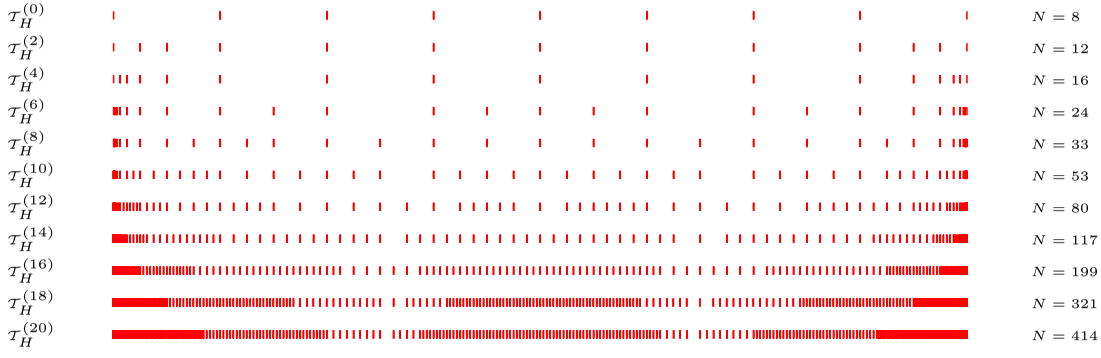


FIG. 8. *Adaptive mesh refinement in Poisson problem* 7.2 *related to* $\ell = 4$ *and error estimator* μ_A. *Adaptive mesh-refinement with respect to* μ_M *leads to the same meshes up to* $\mathcal{T}_H^{(11)}$ *[with* $N = 64$ *coarse grid elements] when the* $P1$ *boundary element method becomes unstable. The adaptively generated meshes are highly adapted towards the reentrant corner of the L-shape.*

TABLE 3

Experimental error, error estimator μ_A, and convergence rates for Poisson problem 7.2, $\ell = 4$, and uniform (top), respectively, μ_A-adaptive mesh refinement (bottom).

Uniform mesh refinement						
k	n	E	μ_A	μ_A/E	μ_A/μ_M	κ
0	32	4.5341e-02	1.2451e-01	2.75	0.979	
1	64	2.8567e-02	7.8630e-02	2.75	0.984	0.67
2	128	1.7999e-02	4.9539e-02	2.75	0.984	0.67
3	256	1.1339e-02	3.1211e-02	2.75	0.984	0.67
4	512	7.1434e-03	1.9662e-02	2.75	0.984	0.67
5	1024	4.5001e-03	1.2387e-02	2.75	0.984	0.67
6	2048	2.8349e-03	7.8032e-03	2.75	n/a	0.67
Adaptive mesh refinement						
k	n	E	μ_A	μ_A/E	μ_A/μ_M	κ
0	32	4.5341e-02	1.2451e-01	2.75	0.979	
2	48	1.8094e-02	5.0731e-02	2.80	0.984	2.51
4	64	7.4020e-03	2.2723e-02	3.07	0.984	3.30
6	96	3.0431e-03	1.0351e-02	3.40	0.983	1.66
8	132	1.3706e-03	5.4421e-03	3.97	0.988	1.79
10	212	5.7738e-04	2.4396e-03	4.23	0.986	2.01
12	320	2.6589e-04	1.2230e-03	4.60	n/a	1.65
14	468	1.2698e-04	6.3510e-04	5.00	n/a	1.76
16	796	7.1773e-05	2.6601e-04	3.71	n/a	1.62
18	1284	3.8295e-05	1.3110e-04	3.42	n/a	1.15
20	1656	2.8272e-05	8.6995e-05	3.08	n/a	1.33

shaped domain Ω of Figure 4. To compute the error in the energy norm, we used the extrapolated value $\|\!|u|\!\|^2 = 2.40769127$ in (7.1). The exact solution u is unknown. Figure 10 shows discrete solutions u_h related to the initial mesh and an adaptively generated mesh [with $n = 32$ and $n = 656$ fine grid elements, resp.]. The discrete solutions show singularities at the five rectangular corners of the L-shape corresponding to the arc-length parameters $s = 1/4, 1/2, 1, 3/2$, and $7/4$. Figure 11 shows the errors and the error estimators for uniform and μ_A-adaptive mesh refinement and $\ell = 3, 4$ in Algorithm 6.1. As in Poisson problem 7.2, the error is sharply estimated by the error estimators η_M and η_A and the error estimators μ_M and μ_A coincide: The larger ℓ, the better is the error estimation. Again, the $\mathcal{P}_1$ boundary element stiffness matrix showed instabilities for the error $E \approx 10^{-7/2}$. The corresponding adaptive meshes are shown in Figure 12.

7.4. Exterior crack problem with nonsmooth solution. The fourth example from [ChS] represents a typical endpoint singularity for open curves and concerns the Poisson problem (6.2) exterior to a straight slit $\Gamma := [-1, 1] \times \{0\}$, $\Omega := \mathbb{R}^2 \backslash \Gamma$. For $g(x, 0) := -x$, the exact solution u of the corresponding Symm's integral equation reads

$$u(x, 0) = -x/\sqrt{1 - x^2} \quad \text{for } -1 < x < 1. \tag{7.9}$$

There holds $u \in H^\alpha(\Gamma)$ for all $\alpha < 0$, but $u \notin L^2(\Gamma)$ since u is singular at the tips of the domain, $u(x, 0) \to \pm\infty$ for $x \to \pm 1$. Since Kg vanishes on the straight slit Γ, Symm's integral equation (1.1) simplifies to

$$Vu = g \quad \text{on } \Gamma. \tag{7.10}$$

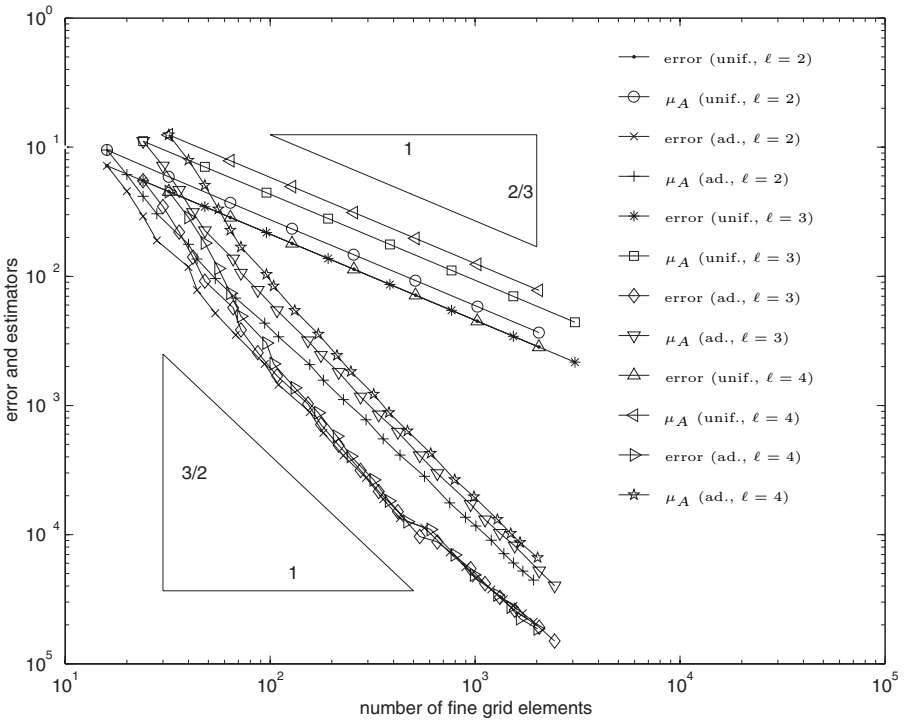


FIG. 9. *Comparison of error and error estimator μ_A in Poisson problem* 7.2 *for uniform and μ_A-adaptive mesh refinement and different choices of $\ell = 2, 3, 4$ in Algorithm* 6.1.

The energy norm of the exact solution can be computed exactly,

$$\|u\|^2 = \langle Vu\,;u\rangle = \int_{-1}^{1} \frac{x^2}{\sqrt{1-x^2}}\,dx = \frac{\pi}{2}.$$

Figure 13 shows the error and error estimators for $\ell = 3, 4$ and both uniform and adaptive mesh refinement. Some adaptively generated meshes are provided in Figure 14. As can be expected, we observe a high mesh refinement towards the ends ± 1 of the slit and coarse local mesh sizes inbetween.

7.5. Smooth eigenfunction of the single-layer potential. Finally, we consider an example on a smooth boundary with the discrete scheme involving the approximation of the boundary Γ by a piecewise-affine boundary Γ_h. Note that the error due to the boundary approximation is not included in the analysis given above.

We consider the sphere $\Gamma = \partial B(0, r)$ with radius $r < 1$. Then, for fixed $k \in \mathbb{Z}\backslash\{0\}$,

$$\phi(x) := \cos(kt) \quad \text{with } x = r\,(\cos t, \sin t) \tag{7.11}$$

is an eigenfunction of the single-layer potential V corresponding to the eigenvalue $\lambda = r/|k|$. For the numerical experiment, we used $r = 1/2$ and $k = 3$. Note that this example satisfies the smoothness assumptions of section 5. For the implementational realization, we approximate Γ by a (convex) polygonal Γ_H with vertices on the sphere. We therefore modify the adaptive Algorithm 6.1 as follows.

ALGORITHM 7.1. [illegible] $\mathcal{T}_H^{(0)}$ [illegible] $\mathcal{T}_H^{(0)}$ [illegible] $k = 0$ $\ell \in \mathbb{N}_{\geq 2}$ [illegible] $0 \leq \theta \leq 1$

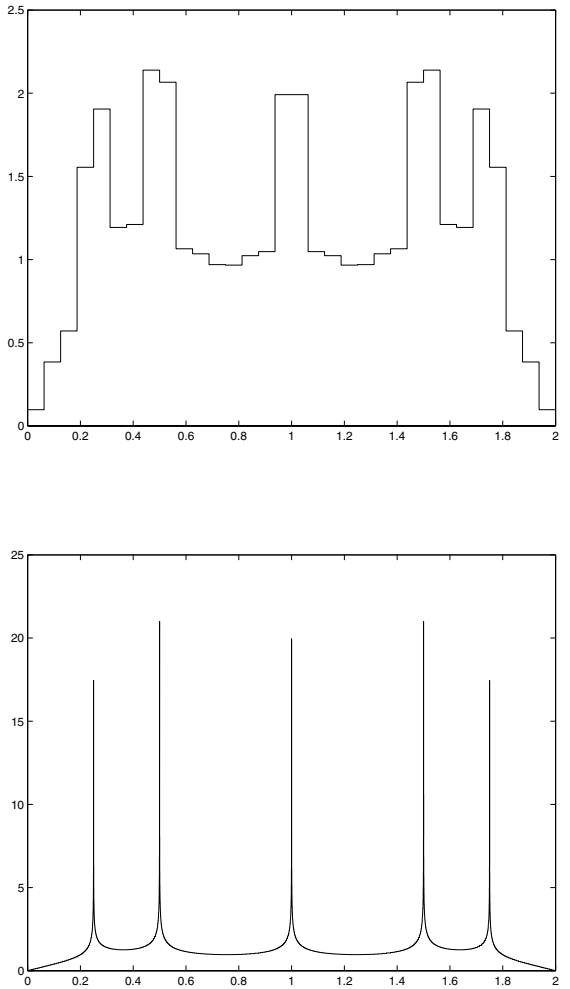


FIG. 10. *Discrete solution* u_h $(n = 32)$ *in Example* 7.3 *related to* $\ell = 4$ *and the initial mesh* $\mathcal{T}_H^{(0)}$ *with* $N = 8$ *elements shown in Figure* 4 (*top*) *and discrete solution* u_h $(n = 656)$ *related to* μ_A*-adaptive generated coarse mesh* $\mathcal{T}_H^{(10)}$ *with* $N = 164$ *elements* (*bottom*). *Note the different scalings on the y-axis of the plots.*

(i) $\mathcal{T}_h^{(k)} = \{\Gamma_1, \ldots, \Gamma_n\}$ $\mathcal{T}_H^{(k)} = \{\gamma_1, \ldots, \gamma_N\}$ $\gamma_j \in \mathcal{T}_H^{(k)}$ ℓ

(ii) $\widetilde{\mathcal{T}}_h^{(k)}$ $\mathcal{T}_h^{(k)}$

(iii) $\widetilde{u}_h^{(k)}$ $\widetilde{\mathcal{T}}_h^{(k)}$

(iv) $\Gamma_j \in \mathcal{T}_h^{(k)}$ $\widetilde{\Gamma}_j \in \mathcal{T}_h^{(k)}$ $u_h \in \mathcal{P}_0(\mathcal{T}_h^{(k)})$ $u_h|_{\Gamma_j} := \widetilde{u}_h|_{\widetilde{\Gamma}_j}$ $\Gamma_j \in \mathcal{T}_h^{(k)}$

(v) η_M η_A $\mu_{M,j}$ $\mu_{A,j}$

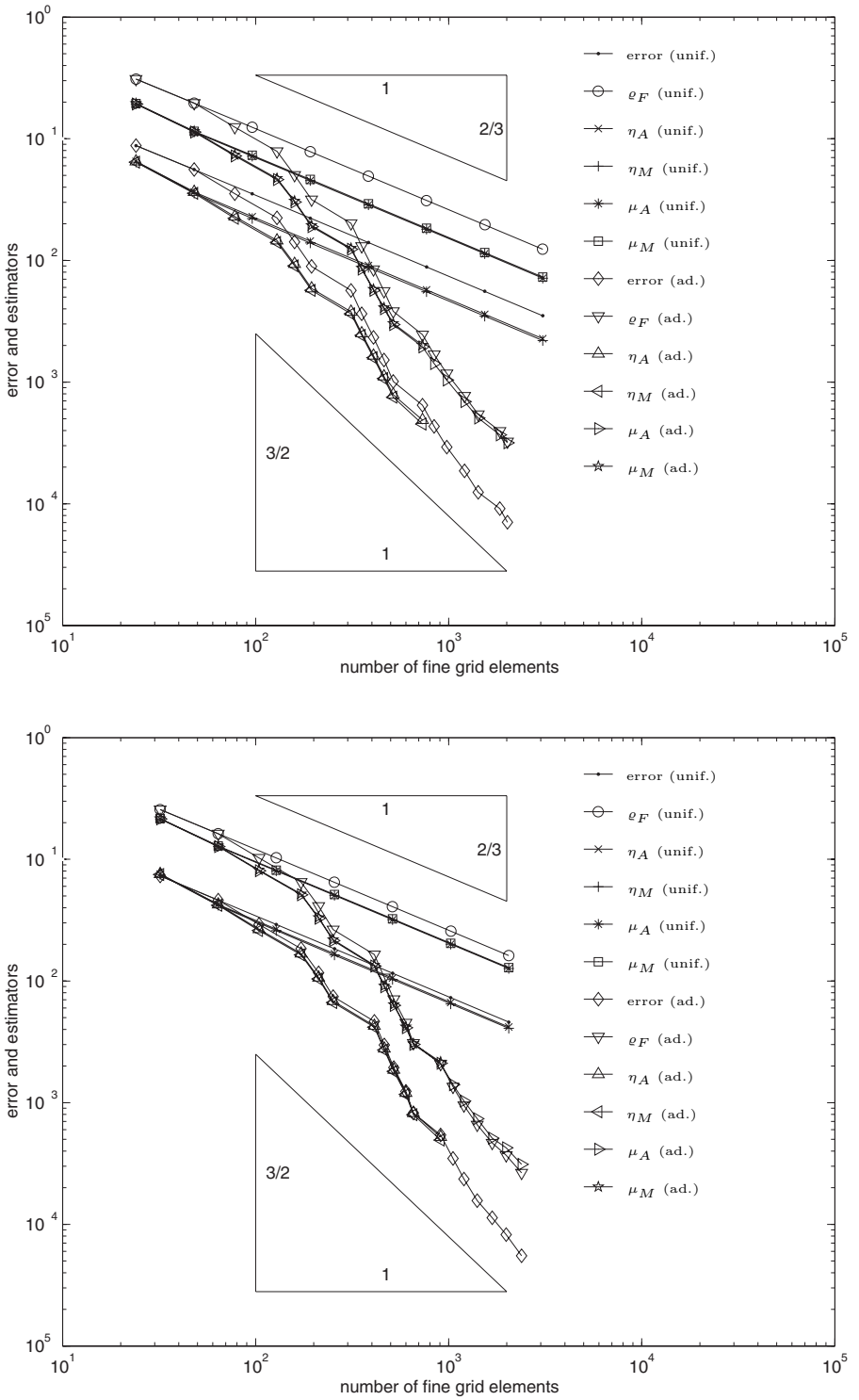


FIG. 11. *Error and error estimators η_M, η_A, μ_M, and μ_A for uniform* [*indicated by* unif.] *and μ_A-adaptive* [*indicated by* ad.] *mesh refinement in Example* 7.3 *and $\ell = 3$ (top), respectively, $\ell = 4$ (bottom) in Algorithm* 6.1. *As for Poisson problem* 7.2, *the larger ℓ, the sharper is the error estimated by the error estimators η_M and η_A. On the other hand, the larger ℓ, the larger is the preasymptotic range before the optimal convergence order* 3/2 *is visible for adaptive mesh refinement.*

FIG. 12. *Adaptive mesh refinement in Example 7.3 related to $\ell = 4$ and error estimator μ_A. Adaptive mesh refinement with respect to μ_M leads to the same meshes up to $\mathcal{T}_H^{(11)}$ [with $N = 64$ coarse grid elements] when the P1 boundary element method becomes unstable. The adaptively generated meshes are highly adapted towards the five rectangular corners of the L-shape, where the discrete solutions show singularities; cf. Figure* 10.

(vi) $\gamma_j \in \mathcal{T}_H^{(k)}$ $\mu_{M,j} \geq \theta \max\{\mu_{M,1},\dots,\mu_{M,N}\}$ $\mu_{A,j} \geq \theta \max\{\mu_{A,1},\dots,\mu_{A,N}\}$

(vii) $\gamma_j \in \mathcal{T}_H^{(k)}$ $\mathcal{T}_H^{(k+1)}$ (i)

To clarify the mesh organization, Figure 15 shows the initial coarse mesh with $N = 6$ elements and the refined meshes $\mathcal{T}_h^{(0)}$ and $\widetilde{\mathcal{T}}_h^{(0)}$ for $\ell = 3$ with $n = 18$ elements. Experimental results are shown in Figure 16 for $\ell = 4$ and both uniform and adaptive mesh refinement. As can be expected from the smoothness of u, the adaptive mesh refining strategy leads to almost uniform meshes. The experimental convergence orders for uniform and adaptive mesh refinement stay optimal. The curves of the error and error estimators almost coincide.

8. Conclusions. In this paper we introduced a new class of error estimators based on averaging techniques. We gave the analytical fundament that these error estimators estimate the (unknown) error $|||u - u_h|||$ both reliably and efficiently, under weak assumptions on the boundary elements used. The strongest assumption is a (piecewise) high regularity of the exact solution u. However, this regularity assumption might be nonsatisfied in practice. We introduced an adaptive algorithm which steers the mesh refinement with respect to the localized error estimators μ_M and μ_A, respectively. In the numerical experiments we treated examples with different regularity. In all experiments the introduced adaptive strategy retains the optimal convergence rate $\mathcal{O}(h^{3/2})$ and is therefore superior to uniform mesh refinement.

Finally, we highlight some of our computational and analytical results.

8.1. Error estimation with η_M, η_A and choice of ℓ. The numerical experiments underline the good performance of the introduced error estimators. The values of the error estimators η_M and η_A almostcoincide with the error $|||u-u_h|||$ provided ℓ is chose large enough. The explicit choice of ℓ is not discussed in this paper and is part of a generalized eigenvalue problem to compute $C_\Pi C_{\rm inv}^{H,q}$ in (5.4). For the numerical experiments treated, the choice of $\ell = 3, 4$ was sufficient to ensure that the error is sharply estimated by η_M and η_A. The efficiency and reliability constants, i.e.,

$$C_{\rm eff}^{-1}\eta \leq |||u - u_h||| \leq C_{\rm rel}\eta$$

for an error estimator η, are unknown except that we know $C_{\rm eff} = 1$ [up to higher order terms] for η_M. At least in the numerical experiments in section 7 we observed $C_{\rm eff}, C_{\rm rel} \to 1$ for $h \to 0$.

8.2. Adaptive mesh-refinement with respect to μ_M, μ_A. The examples 7.2–7.4 show that the proposed adaptive strategy can retain the optimal convergence rate $\mathcal{O}(h^{3/2})$ even for examples with nonsmooth exact solution. Poisson problem 7.2, also a benchmark for 2D finite element schemes, has exactly one corner singularity. Therefore the asymptotic convergence rates are visible from the beginning. This is not typical and so we addressed a generic example with smooth right-hand side, namely $f = 1$, which is [according to the authors' knowledge] not explicitly accompanied with an equivalent Poisson problem. Figure 11 shows a larger preasymptotic range for $N \leq 20$ and adaptive mesh refinement. The final Example 7.5 shows how the averaging based error estimation can be employed in combination with the approximation of a smooth boundary.

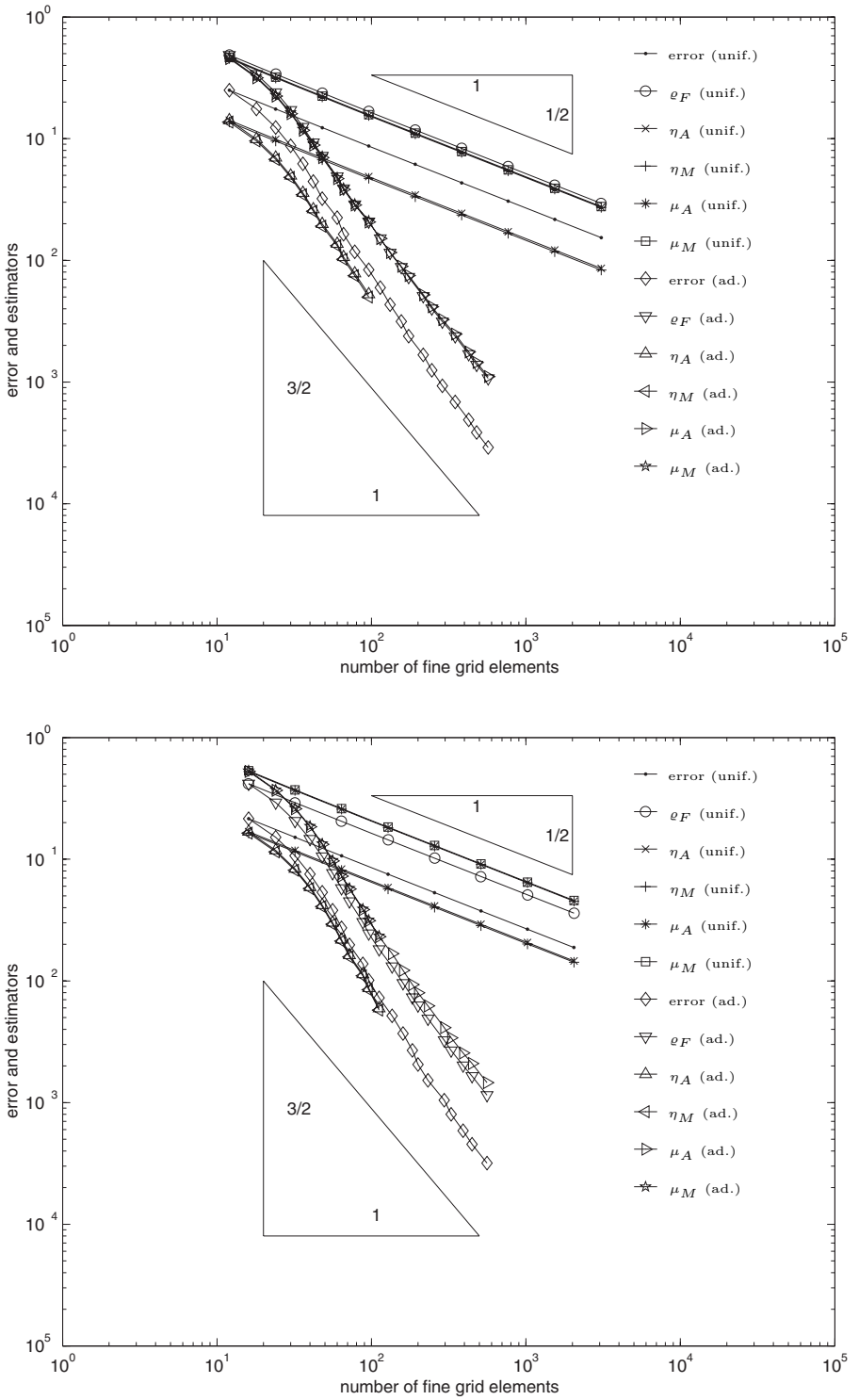


FIG. 13. *Error and error estimators* η_M, η_A, μ_M, *and* μ_A *for uniform* [*indicated by* unif.] *and* μ_A*-adaptive* [*indicated by* ad.] *mesh refinement in Example* 7.4 *and* $\ell = 3$ (*top*), *respectively,* $\ell = 4$ (*bottom*) *in Algorithm* 6.1.

8.3. Computational cost of the introduced error estimators. The naive computational cost for the estimators μ_M, η_M, η_A is quadratic in the number of unknowns since we have to compute the stiffness matrix for the $\mathcal{P}_1$ boundary element methods to compute $\mathbb{G}_H u_h$ for μ_M and η_M, respectively, $\|\mathbb{G}_H u_h\|$ and $\|\mathcal{A}_H u_h\|$ for η_M and η_A. This can be overcome by use of approximation techniques like panel clustering or $\mathcal{H}$-matrices which compute, store, and evaluate the stiffness matrix in almost linear complexity. Nevertheless, the computational cost for computing μ_A seems to be striking: The computation of μ_A for the L^2-projection $\mathcal{A}_H$ only needs the assembly of the usual L^2 mass matrix $\mathbf{M}$. This matrix is sparse and therefore assembled and stored in linear complexity. Moreover, the computation of $\mathcal{A}_H u_h$ involves the solution of $\mathbf{My} = \mathbf{c}$ for a given right-hand side $\mathbf{c}$, and the condition number of $\mathbf{M}$ is bounded, i.e., $\mathcal{O}(1)$ for $h \to 0$.

8.4. Applicability to 3D problems. We stress that the presented analysis works for 3D problems, i.e., 2D boundary pieces, as well. In the numerical experiments we restricted ourselves to 2D problems for ease of presentation, since the implementation of the boundary element schemes and even the data structures are much more involved for the 3D boundary element method.

8.5. Further developments and open problems. From an analytical point of view, Theorem 5.1 is the core result for providing the averaging error estimators. The proof is only based on the validity of a local inverse estimate; cf. section 3.2, and the existence of an appropriate (local) first order approximation operator. In a forthcoming paper, we will provide both the hypersingular integral equation as well [CP2, FP]. Is is expected that the ideas carry over to a quite general class of integral equations.

The analytical verification of the introduced error estimators needs high regularity assumptions on u. Since our numerical experiments indicate that these assumptions can be weakened, it would be desirable to have a refined analysis that covers these cases as well, i.e. which either avoids the regularity assumptions on u or explains the good performance of the indicator-based strategy analytically.

REFERENCES

[AC] J. Alberty and C. Carstensen, *Averaging techniques for reliable a posteriori FE-error control in elastoplasticity with hardening*, Comput. Methods Appl. Mech. Engrg., 192 (2003), pp. 1435–1450.

[BL] J. Bergh and J. Löfström, *Interpolation Spaces*, Grundlehren der Mathematischen Wissenschaften 223, Springer-Verlag, Berlin, 1976.

[BR] I. Babuška and W. C. Rheinboldt, *Error estimates for adaptive finite element computations*, SIAM J. Numer. Anal., 15 (1978), pp. 736–754.

[C1] C. Carstensen, *Adaptive boundary element methods and adaptive finite element and boundary element coupling*, In *Boundary value problems and integral equations on non-smooth domains*. Lecture Notes in Pure and Applied Mathematics 167, M. Costabel, M. Dauge, S. Nicaise, eds., Marcel Decker, New York (1995), pp. 47–58.

[C2] C. Carstensen, *Efficiency of a posteriori BEM-error estimates for first kind integral equations on quasi-uniform meshes*, Math. Comp., 65 (1996), pp. 69–84.

[C3] C. Carstensen, *An a posteriori error estimate for a first-kind integral equation*, Math. Comp., 66 (1997), pp. 139–155.

[CB] C. Carstensen and S. Bartels, *Each averaging technique yields reliable a posteriori error control in FEM on unstructured grids. I. Low order conforming, nonconforming, and mixed FEM*, Math. Comp., 71 (2002), pp. 945–969.

[CES] C. Carstensen, D. Estep, and E. P. Stephan, *h-adaptive boundary element schemes*, Comput. Mech., 15 (1995), pp. 372–383.

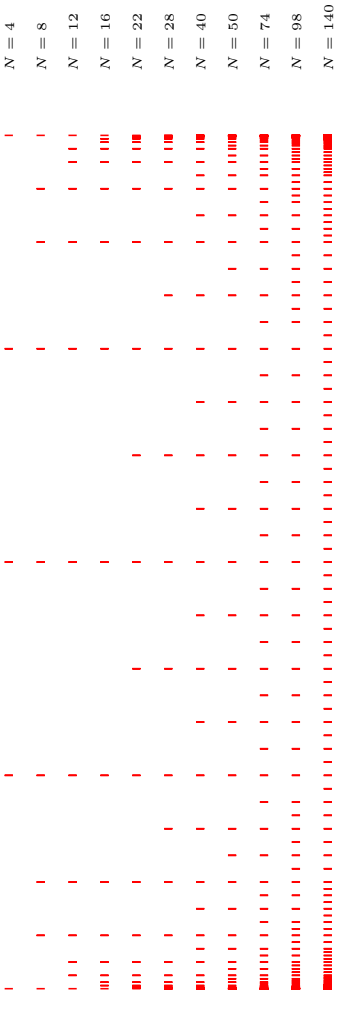


FIG. 14. *Adaptive mesh refinement in slit Problem 7.4 related to $\ell = 4$ and error estimator μ_A. Adaptive mesh refinement with respect to μ_M leads to the same meshes up to $\mathcal{T}_H^{(11)}$ [with $N = 64$ coarse grid elements] when the P1 boundary element method becomes unstable. The adaptively generated meshes are highly adapted towards the endpoints $(\pm 1, 0)$ of the slit.*

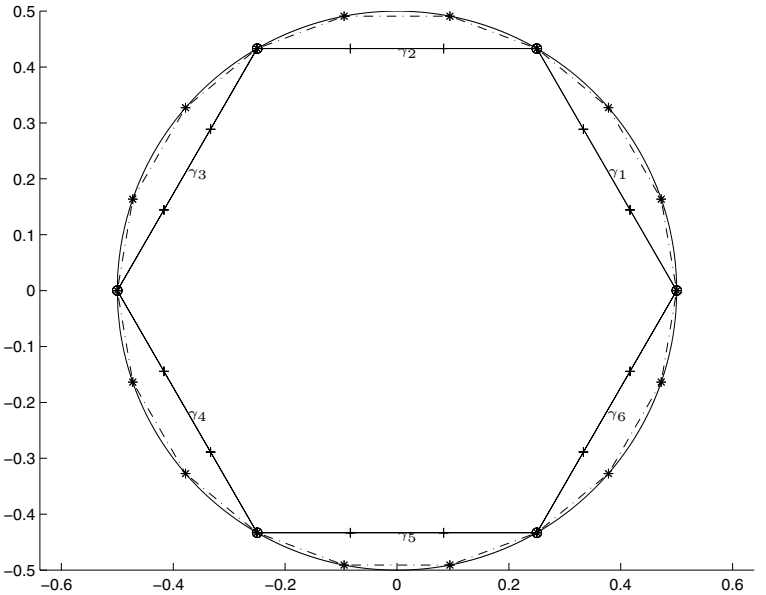


FIG. 15. *Initial meshes in Example* 7.5*: The initial coarse mesh* $\mathcal{T}_H^{(0)}$ *consists of* $N = 6$ *elements* [*nodes marked by* o]. *For* $\ell = 3$, $\mathcal{T}_h^{(0)}$ *is a refinement of* $\mathcal{T}_H^{(0)}$ *with* $n = 18$ *elements* [*nodes marked by* +] *and* $\widetilde{\mathcal{T}}_h^{(0)}$ *is obtained by projecting the nodes onto the sphere* [*marked by* ∗].

[CFa] C. CARSTENSEN AND B. FAERMANN, *Mathematical foundation of a posteriori error estimates and adaptive mesh-refining algorithms for boundary integral equations of the first kind*, Eng. Anal. Bound. Elem., 25 (2001), pp. 497–509.

[CFu] C. CARSTENSEN AND S. A. FUNKEN, *Averaging technique for FE—a posteriori error control in elasticity. I. Conforming FEM*, Comput. Methods Appl. Mech. Eng., 190 (2001), pp. 2483–2498. II. λ*-independent estimates*, Comput. Methods Appl. Mech. Eng., 190 (2001), pp. 4663–4675. III. *Locking-free nonconforming FEM*, Comput. Methods Appl. Mech. Eng., 191 (2001), pp. 861–877.

[CFS] C. CARSTENSEN, S. A. FUNKEN, AND E. P. STEPHAN, *A posteriori error estimates for hp-boundary element methods*, Appl. Anal., 61 (1996), pp. 233–253.

[CMS] C. CARSTENSEN, M. MAISCHAK, AND E. P. STEPHAN, *A posteriori error estimate and h-adaptive algorithm on surfaces for Symm's integral equation*, Numer. Math., 90 (2001), pp. 197–213.

[CMPS] C. CARSTENSEN, M. MAISCHAK, D. PRAETORIUS, AND E. P. STEPHAN, *Residual-based a posteriori error estimate for hypersingular equation on surfaces*, Numer. Math., 97 (2004), pp. 397–425.

[CP1] C. CARSTENSEN AND D. PRAETORIUS, *A posteriori error control in adaptive qualocation boundary element analysis for a logarithmic-kernel integral equation of the first kind*, SIAM J. Sci. Comput., 25 (2003), pp. 259–283.

[CP2] C. CARSTENSEN AND D. PRAETORIUS, *Averaging techniques for the a posteriori BEM error control for a hypersingular integral equation in* 2*d*, SIAM J. Sci. Comp., accepted, 2005.

[CS1] C. CARSTENSEN AND E. P. STEPHAN, *A posteriori error estimates for boundary element methods*, Math. Comput., 64 (1995), pp. 483–500.

[CS2] C. CARSTENSEN AND E. P. STEPHAN, *Adaptive boundary element methods for some first kind integral equations*, SIAM J. Numer. Anal., 33 (1996), pp. 2166–2183.

[ChS] G. A. CHANDLER AND I. H. SLOAN, *Spline qualocation methods for boundary integral equations*, Numer. Math., 58 (1990), pp. 537–567.

[Co] M. COSTABEL, *Boundary integral operators on Lipschitz domains: Elementary results*, SIAM J. Math. Anal., 19 (1988), pp. 613–626.

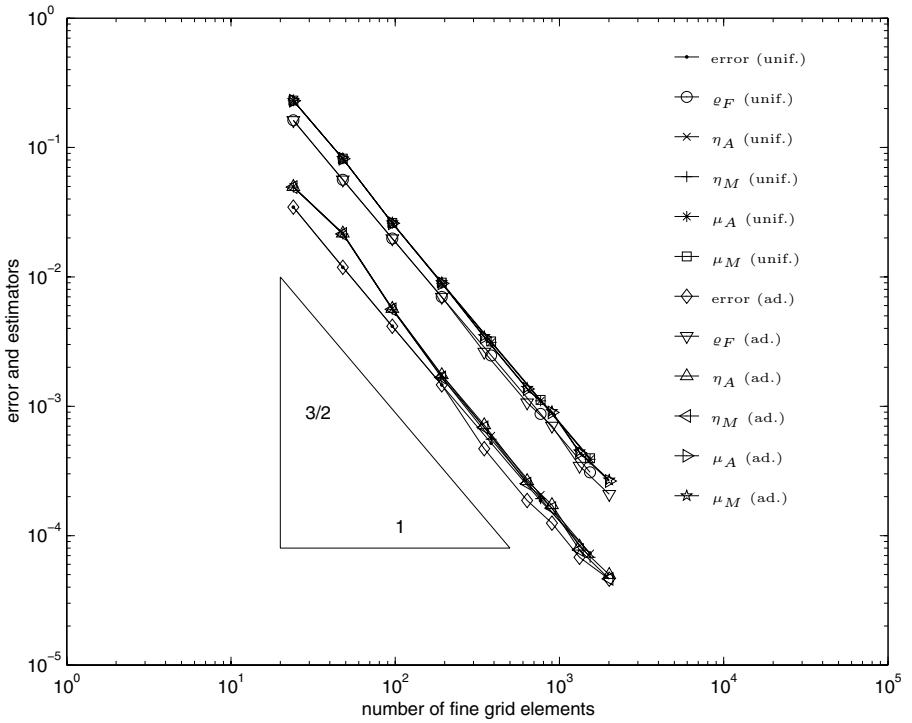


FIG. 16. *Error and error estimators* η_M, η_A, μ_M, *and* μ_A *for uniform* [*indicated by* unif.] *and* μ_A*-adaptive* [*indicated by* ad.] *mesh refinement in Example* 7.5 *and* $\ell = 4$ *in Algorithm* 6.1.

[F1] B. FAERMANN, *Local a-posteriori error indicators for the Galerkin discretization of boundary integral equations*, Numer. Math., 79 (1998), pp. 43–76.

[F2] B. FAERMANN, *Localization of the Aronszajn-Slobodeckij norm and application to adaptive boundary element methods.* I. *The two-dimensional case*, IMA J. Numer. Anal., 20 (2000), pp. 203–234.

[F3] B. FAERMANN, *Localization of the Aronszajn-Slobodeckij norm and application to adaptive boundary element methods.* II. *The three-dimensional case*, Numer. Math., 92 (2002), pp. 467–499.

[FHK] M. FEISTAUER, G. C. HSIAO, AND R. E. KLEINMAN, *Asymptotic and a posteriori error estimates for boundary element solutions of hypersingular integral equations*, SIAM J. Numer. Anal., 33 (1996), pp. 666–685.

[FP] S. FUNKEN AND D. PRAETORIUS, *Averaging on large patches for first kind integral equations in* 3*D*, work in progress, 2005.

[GHS] I. G. GRAHAM, W. HACKBUSCH, AND S. A. SAUTER, *Finite elements on degenerate meshes: Inverse-type inequalities and applications*, IMA J. Numer. Anal., 25 (2005), pp. 379–407.

[HMS] N. HEUER, M. MAISCHAK, AND E. P. STEPHAN, *Exponential convergence of the hp-version for the boundary element method on open surfaces*, Numer. Math., 83 (1999), pp. 641–666.

[Ma] M. MAISCHAK, *The Analytical Computation of the Galerkin Elements for the Laplace, Lamé and Helmholtz Equation in BEM*, Preprint (1999): 2*D BEM*, Preprint (2000): 3*D BEM*, Institut für Angewandte Mathematik, Universität Hannover.

[MMS] M. MAISCHAK, P. MUND, AND E. P. STEPHAN, *Adaptive multilevel BEM for acoustic scattering*, Comput. Methods Appl. Mech. Engrg., 150 (1997), pp. 351–367.

[McL] W. MCLEAN, *Strongly Elliptic Systems and Boundary Integral Equations*, Cambridge University Press, Cambridge, UK, 2000.

[MPM] G. Menon, G. H. Paulino, and S. Mukherjee, *Analysis of hypersingular residual error estimates in boundary element methods for potential problems*, Comput. Methods Appl. Mech. Engrg., 73 (1999), pp. 449–473.

[MSW] P. Mund, E. P. Stephan, and J. Weisse, *Two-level methods for the single-layer potential in* $\mathbb{R}^3$, Computing, 60 (1998), pp. 243–266.

[P] T. von Petersdorff, *Randwertprobleme der Elastizitätstheorie für Polyeder–Singularitäten und Approximation mit Randelementmethoden*, PhD thesis, Darmstadt, Germany, 1989.

[R1] E. Rank, *A-posteriori error estimates and adaptive refinement for some boundary integral element method*, in Proc. Int. Conf. on Accuracy Estimates and Adaptive Refinements in FE Computations ARFEC, Lisbon, Portugal, 1984, pp. 55–64.

[R2] E. Rank, *Adaptivity and accuracy estimation for FEM and BEM*, in *Accuracy Estimates and Adaptive Refinements in FE Computations*, I. Babuška, O. C. Zienkiewicz, J. Gago, E. R. Oliveira, eds., Wiley, Chichester, UK, 1986.

[SaS] S. A. Sauter and C. Schwab, *Randelemente, Analyse und Implementierung Schneller Algorithmen*, Teubner, Stuttgart, Germany, 2004.

[SSW] H. Schulz, C. Schwab, and W. L. Wendland, *Extraction, higher order boundary element methods, and adaptivity*, in *Mathematical aspects of boundary element methods* (Palaiseau, 1998), Chapman & Hall/CRC Res. Notes Math., 414, Chapman & Hall/CRC, Boca Raton, FL 2000, pp. 263–274.

[SSt] H. Schulz and O. Steinbach, *A new a posteriori error estimator in adaptive direct boundary element methods. The Dirichlet problem*, Calcolo, 37 (2000), pp. 79–96.

[SW] C. Schwab and W. L. Wendland, *On the extraction technique in boundary integral equations*, Math. Comp., 68 (1999), pp. 91–122.

[S] O. Steinbach, *Adaptive boundary element methods based on computational schemes for Sobolev norms*, SIAM J. Sci. Comput., 22 (2000), pp. 604–616.

[StS] E. P. Stephan and M. Suri, *The h-p version of the boundary element method on polygonal domains with quasiuniform meshes*, RAIRO Model. Math. Anal. Numér., 25 (1991), pp. 783–807.

[WY] W. L. Wendland and D. Yu, *Adaptive boundary element methods for strongly elliptic integral equations*, Numer. Math., 53 (1988), pp. 539–558.

[Y1] D. Yu, *A posteriori error estimates and adaptive approaches for some boundary element methods*, in *Boundary Elements* 9, C. A. Brebbia, W. L. Wendland, G. Kuhn, eds., Berlin, Stuttgart, (1987), pp. 241–256.

[Y2] D. Yu, *Self-adaptive boundary element methods*, ZAMM, 68 (1988), pp. T435–T437.

[ZZ] O. C. Zienkiewicz and J. Z. Zhu, *A simple error estimator and adaptive procedure for practical engineering analysis*, Internat. J. Numer. Methods Engrg., 24 (1987), pp. 337–357.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1261–1286
© 2006 Society for Industrial and Applied Mathematics

ADAPTIVE ALGEBRAIC MULTIGRID*

M. BREZINA†, R. FALGOUT‡, S. MACLACHLAN†,
T. MANTEUFFEL†, S. MCCORMICK†, AND J. RUGE†

Abstract. Efficient numerical simulation of physical processes is constrained by our ability to solve the resulting linear systems, prompting substantial research into the development of multiscale iterative methods capable of solving these linear systems with an optimal amount of effort. Overcoming the limitations of geometric multigrid methods to simple geometries and differential equations, algebraic multigrid methods construct the multigrid hierarchy based only on the given matrix. While this allows for efficient black-box solution of the linear systems associated with discretizations of many elliptic differential equations, it also results in a lack of robustness due to unsatisfied assumptions made on the near null spaces of these matrices. This paper introduces an extension to algebraic multigrid methods that removes the need to make such assumptions by utilizing an adaptive process. Emphasis is on the principles that guide the adaptivity and their application to algebraic multigrid solution of certain symmetric positive-definite linear systems.

Key words. iterative methods, algebraic multigrid (AMG), adaptive method

AMS subject classifications. 65F10, 65N55, 65F30

DOI. 10.1137/040614402

1. Introduction.

While the original development of algebraic multigrid (AMG) began over twenty years ago (cf. [8, 24]), the current high level of interest and research activity is fairly recent. This dramatic increase is largely due to the potential of these methods for solving very large problems arising from partial differential equations with irregular grids and varying coefficients. See, for example, [1, 3, 13, 28, 33].

By the term AMG, we mean the class of solvers based on multigrid principles that depend little or not at all on geometric information about the problem, but instead attempt to use basic concepts of "algebraic smoothness" to determine effective coarsening and/or relaxation processes. Solvers of this type typically assume some defining characteristic of algebraic smoothness, specifying error components that are not quickly eliminated by the relaxation that is being used. For example, all such components are assumed, in standard AMG (cf. [27] and section 2), to vary slowly along so-called strong connections in the matrix, or, in smoothed aggregation (SA; cf. [29]), to be represented locally by a few prototype vectors supplied by the user. While appropriate use of the characteristic of algebraic smoothness seems essential

*Received by the editors September 3, 2004; accepted for publication (in revised form) April 25, 2005; published electronically January 6, 2006. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/27-4/61440.html

†Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309–0526 (mbrezina@math.cudenver.edu, maclachl@colorado.edu, tmanteuf@colorado.edu, stevem@colorado.edu, jruge@colorado.edu). This work was sponsored by the National Science Foundation under grant DMS-0420873, Lawrence Livermore National Laboratory under contract B541045, and the Department of Energy under grants DE-FC02-01ER25479 and DE-FG02-03ER25574.

‡Center for Applied Scientific Computation, Lawrence Livermore National Lab, P.O. Box 808, L-561, Livermore, CA 94551 (rfalgout@llnl.gov). This work was performed under the auspices of the U. S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

for obtaining effective solvers, these additional assumptions limit the scope of applicability of the resulting methods. In many important cases, errors missed by standard relaxation processes can vary substantially along strong matrix connections, and, in many cases, even the concept of strength of connection is not well understood. Moreover, supplying a fully representative set of prototypical smooth components is not always easy nor possible in practice.

The principal aim of the adaptive approach developed here is to eliminate, or substantially reduce, this reliance on the additional assumptions usually present in related methods. The basic idea is to test the initial version of the given solver on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, to determine its performance and expose whatever types of errors it does not effectively reduce. The resulting prototypical errors that these tests produce are then used to improve the AMG process.

The concept of using a multigrid algorithm to improve itself is not new. Using prototypical algebraically smooth vectors in the coarsening process was first developed in the early stages of the AMG project of Brandt, McCormick, and Ruge (documented in [24]), where interpolation was defined to fit vectors obtained by relaxation on the homogeneous problem. In [8], a variation of this idea was used for recovering typical AMG convergence rates for a badly scaled scalar elliptic problem. While this initial approach was very basic and used only one prototype of algebraically smooth error, it contained many of the ingredients of the adaptive process described here. These concepts were developed further in [23, 26, 25, 27]. The concept of fitting eigenvectors corresponding to the smallest eigenvalues was advocated in [23] and [27], where an AMG algorithm determining these eigenvectors through Rayleigh quotient minimization was outlined. These vectors were, in turn, used to update the AMG interpolation and coarse-level operators. Adaptivity of linear solvers has a long history, including adaptive choice of the overrelaxation parameter in SOR [18] or the parameters in a Chebyshev iteration [21], and has seen renewed interest of late as a way to improve robustness of existing linear solvers, as in the work of Brandt [7] and Wagner and Wittum [32, 31].

In many ways, using algebraically smooth vectors in the definition of interpolation represents the next logical step in improving the interpolation operators used in robust geometric and algebraic multigrid methods. Alcouffe et al. introduced the concept of operator-induced interpolation in [2]. This improvement on the previously used geometric interpolation approach opened up a much larger class of problems to black-box multigrid solution.

In the present paper, we use an operator-induced interpolation approach as well, but also rely on an automatic process that supplies representative algebraically smooth components to ensure optimal performance. By integrating information regarding algebraically smooth vectors into the definition of interpolation, we develop multigrid schemes that are hopefully optimal for elliptic problems, where the discrete system is not necessarily in the form of an M-matrix. These operator- and relaxation-induced interpolation approaches can, if properly implemented, greatly enlarge the class of problems that admits optimal performance by a black-box multigrid technique.

We also introduce this form of adaptivity into AMG. While classical multigrid methods can be viewed as stationary iterative methods [4], the methods presented here are dynamic in their setup phase. In fact, we propose using the method itself to drive its own iterative improvement. A "bootstrap" AMG method that is similar to the approach developed here was recently proposed for the classical AMG setting by Brandt [7] and by Brandt and Ron [9].

Several other attempts have been made to allow the solver itself to determine from the discrete problem the information required to solve it successfully, without a priori assumptions on the form of the smooth error, including the methods of [11, 10, 14, 17]. All of these methods, however, have in common their requirement that the local finite element matrices of the problem be available, and they construct the multigrid transfer operators based on the algebraically smooth eigenvectors corresponding to local stiffness matrices assembled over element agglomerates. Although they can achieve encouraging convergence rates, their need to construct, store, and manipulate the coarse-level element information typically leads to increased storage requirements compared to those of classical AMG or standard SA. The methods described below attempt to achieve the good convergence properties of the element-based methods without requiring access to or manipulation of element information that may not even be available.

We refer to the approach developed here as adaptive because it involves self-testing to expose slowly converging error components and adaptation of the schemes' components to improve themselves. The acronym αMG is used to refer to the general class of multigrid methods of this type, to suggest their primal algebraic nature: we have in mind methods that use only the defining characteristic of algebraic smoothness and must use an automatic algebraic process to determine additional characteristics that enable effective determination of the full MG algorithm. The additional abbreviations αAMG and αSA, respectively, are used to refer to the specific AMG and SA versions of αMG.

In the next section, we give a brief description of standard AMG to set the stage for the development of αAMG in the following sections. Section 3 provides an introduction to the adaptive framework and develops some fundamental principles that guide our construction of the adaptive process. In section 4, we discuss details of the interpolation scheme used as well as some of its theoretical properties. Some important details of implementation are discussed in section 5, and numerical performance is illustrated in section 6.

2. Standard AMG. The key to the efficiency of any multigrid process lies in the complementarity of the relaxation and coarse-grid correction processes. For this reason, any discussion of the classical AMG algorithm begins with the defining property of algebraic smoothness that the residual is, on average, small after a few sweeps of relaxation: $(A\mathbf{e})_i \approx 0$ for each point i. Considering pointwise relaxation, such as Gauss–Seidel, and a symmetric positive-definite operator, such errors are typically associated with the small eigenvalues of the operator, and this is often what is meant when discussing algebraically smooth errors. Also central is the assumed property that such errors vary slowly along strong connections in the matrix. This additional assumption is generally applicable for discretizations of scalar elliptic PDEs, for which AMG was originally designed. Other problems, such as systems of PDEs, may require other assumptions on the nature of algebraically smooth errors. In any case, awareness of this assumption is important in analyzing the performance of AMG, particularly when this performance is poor.

To construct interpolation to approximate a general algebraically smooth error, $\mathbf{e}$, we use the premise of a small residual (that, for example, $\mathbf{e}^T A^2 \mathbf{e} \ll \rho(A)\mathbf{e}^T A\mathbf{e}$, where $\rho(A)$ is the spectral radius of A) to conclude that, for a given row, i, $(A\mathbf{e})_i \approx 0$ or

$$a_{ii}e_i \approx -\sum_{j\neq i} a_{ij}e_j. \tag{2.1}$$

Now, suppose that a coarse set of points that forms a subset of the fine degrees of freedom (DOFs) has been chosen (strategies for which are discussed below). Then, the fine-level DOFs can be represented as $\{1, 2, \ldots, N\} = C \cup F$, where C is the set of coarse-level points and F is the set of remaining fine-level points (so $C \cap F = \emptyset$). Since A is sparse, we introduce "neighborhood" notation: $N_i = \{j \neq i : a_{ij} \neq 0\}$, $C_i = C \cap N_i$, and $F_i = F \cap N_i$. Equation (2.1) can then be rewritten as

$$a_{ii}e_i \approx -\sum_{j \in C_i} a_{ij}e_j - \sum_{k \in F_i} a_{ik}e_k. \tag{2.2}$$

Were the last sum not present in (2.2), this expression could be used to define interpolation, because it would give the F-point value, e_i, approximately as a sum of the C_i-point values. The aim is therefore to "collapse" the connections from point i to points $k \in F_i$ onto the points $\{j \in C_i\} \cup \{i\}$. That is, we want to set a_{ik}, $k \in F_i$, to 0 while adjusting a_{ij}, for $j \in C_i$, and a_{ii} in some way to compensate for the inaccuracies this elimination introduces. The main assumption needed to collapse the stencil is that the values of algebraically smooth $\mathbf{e}$ at F_i points can be written in terms of its values at points in $C_i \cup \{i\}$:

$$e_k \approx \sum_{j \in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i \qquad \text{for } k \in F_i. \tag{2.3}$$

We could then substitute this expression into the latter sum in (2.2) to obtain an expression for e_i in terms of $e_j, j \in C_i$, which is exactly the aim. Note that (2.3) is a special form of interpolation from $C_i \cup \{i\}$ to F_i. This special interpolation formula is used in reducing the stencil connections to determine the final interpolation formula, so this overall process is sometimes called "twice-removed" or "iterated" interpolation.

Now, in classical AMG algorithms, the $\{\omega_{kj}^i\}$ in (2.3) are determined from $\{a_{kj}\}$ based on the additional assumption that $\mathbf{e}$ is constant along strong connections. Thus, first we must ask the question as to whether the connection between i and k is important. To do this, we define the concept of strength of connection between two nodes. Considering the connections in the matrix to represent weights on the edges of the matrix graph (if $a_{ij} = 0$, there is no edge between nodes i and j), node i is said to strongly depend on node j if $-a_{ij} \geq \theta \max_{k \neq i}, \{-a_{ik}\}$. Likewise, node i strongly influences node j if $-a_{ji} \geq \theta \max_{k \neq j}, \{-a_{jk}\}$. Here, θ is a predetermined threshold value, $0 \leq \theta \leq 1$, often chosen to be 0.25.

Now, if k does not strongly influence i, then its value is not helpful in interpolation, as it is not a strong interpolation point. The set of such k is denoted F_i^w and referred to as the set of weak connections. These connections are not simply discarded; rather, their values are collapsed onto the diagonal. That is, $\omega_{kj}^i = 0$ for $j \neq i$, and $\omega_{ki}^i = 1$. Notice that this approach is, in fact, quite stable. If the classification of $k \in F_i$ misidentifies a strong influence as a weak one, it replaces the value e_k with e_i, which (because of the assumption about smooth error and strong connections) are, in fact, approximately equal.

The remaining $k \in F_i \setminus F_i^w \equiv F_i^s$ all strongly influence i. Adding the requirement that a node $k \in F_i^s$ be also strongly dependent on some $j \in C_i$, then we can hope to determine the value at point k based on the coarse-grid neighbors it has in common with i. Since the strength of k's connection to a point $j \in C_i$ is proportional to the matrix coefficient a_{kj}, the value e_k can be approximated by the average of these e_j

weighted by their coefficients. That is,

$$e_k = \frac{\sum_{j \in C_i} a_{kj} e_j}{\sum_{l \in C_i} a_{kl}}. \tag{2.4}$$

An important property of this intermediate interpolation formula is that it is an average; if $\mathbf{e}$ is constant for all $j \in C_i$, then it takes the same value at k. For problems such as the Laplacian, where the smoothest mode is the constant vector (up to boundary conditions), it is quite important to ensure the accuracy of interpolation for this mode.

Combining these approximations yields an overall interpolation formula for $i \in F$: $e_i = \sum_{j \in C_i} w_{ij} e_j$, with

$$w_{ij} = -\frac{a_{ij} + \sum_{k \in F_i^s} \left(\frac{a_{ik} a_{kj}}{\sum_{l \in C_i} a_{kl}} \right)}{a_{ii} + \sum_{m \in F_i^w} a_{im}}.$$

The effectiveness of this interpolation operator is now very dependent on the quality of the coarse-grid points selected. This is apparent from the assumption that each point $k \in F_i^s$ has a strong dependence on some $j \in C_i$. In addition, we need the coarse grid to satisfy an often contradictory condition that it must be small enough that there is a real benefit to coarsening. There are many options for choosing the coarse-grid points, such as the approach of [27], where the number of strongly connected neighbors of each point is considered. Also of interest is the recent work on compatible relaxation; cf. [16, 19]. We do not discuss the selection process further, except to say that it is of primary importance to the efficiency and success of an AMG: the coarse level must support accurate approximation of algebraically smooth error using only a small fraction of the fine-level points.

Once a set of coarse points, C, and an interpolation operator, P, have been chosen, we must still choose a restriction operator, R (for transferring the residuals to the coarse level), and a coarse-level operator, A_c (for defining the coarse-level correction equation). Assuming that A is a symmetric and positive-definite matrix, it is natural to define these operators by the so-called Galerkin conditions (cf. [27]):

$$R = P^T \qquad \text{and} \qquad A_c = RAP.$$

These definitions arise from minimization principles and are a result of choosing R and A_c so that the coarse-level correction minimizes the fine-level A-norm of the error over all such corrections.

AMG is a generalization of classical geometric multigrid. It is an efficient solver for many problems, including those involving discretizations on stretched or irregular grids, or discretizations of many problems with anisotropic or variable coefficients. There are, however, still many problems for which classical AMG is not effective, including problems with highly anisotropic or highly variable coefficients and those coming from the discretization of certain systems of PDEs such as linear elasticity. Simply put, the further the algebraically smooth components of a problem are from being locally constant, the more the performance of classical AMG suffers.

3. The adaptive AMG framework. The details of the αAMG and αSA [12] algorithms are quite intricate. We arrived upon them by careful consideration of the basic principles and methodologies of an adaptive algorithm. For this reason, our

discussion focuses on these basic principles before the particular details. We restrict our attention for the remainder of the paper to the case that the $N \times N$ matrix, $A = (a_{ij})$, is symmetric and positive-definite, although most of what is developed applies to more general cases. Our aim is to develop an AMG process to solve the matrix equation,

$$A\mathbf{x} = \mathbf{b},$$

without a priori knowledge of the character of algebraically smooth error.

3.1. The adaptive MG algorithm. An efficient multigrid process for solving $A\mathbf{x} = \mathbf{b}$ relies on the appropriate complementarity of relaxation and coarse-grid correction. Thus, we view the goal of the adaptive process as the development of a representative collection of vectors for which the chosen relaxation process is inefficient. In its most basic form, the adaptive process would be quite simple: relax on a significant number of vectors to expose slow-to-converge components (which act as prototypes of algebraically smooth error), and then choose interpolation to fit these vectors. Such an approach may, however, be quite inefficient and a multiscale viewpoint often proves more useful.

Suppose we know the matrix, A, but nothing more. Relaxation is then the only feasible way to expose algebraically smooth error components. However, with no further knowledge, there is no way of predicting, a priori, how many distinct components are needed to achieve good results. Experience (and, in the case of SA, theory [30]) has shown that, for discretizations of second-order scalar elliptic PDEs, a single component is sufficient, whereas six components may be needed for a problem such as three-dimensional (3D) linear elasticity. Thus, to arrive at an optimal solver with a minimal amount of work, it seems necessary to start with a single vector and introduce new prototypes as the evolving method proves inadequate.

The situation is then that we have a given matrix and seek to find a single algebraically smooth vector upon which to base interpolation. Relaxation alone can achieve this, simply by iteration on the homogeneous problem. However, it typically requires a huge number of relaxations to expose a global algebraically smooth vector, and so we seek to expose such components through multiscale development instead. We first relax only a few times, which should be sufficient to expose errors that are smooth enough to be handled on the first coarse grid. Smoother errors may be exposed by relaxation on the homogeneous problem on this coarse grid, particularly since the injected fine-grid prototype is a good initial guess for the algebraically smooth error we seek. The coarse-grid algebraically smooth error may then be used to create a still coarser grid in the usual multigrid fashion. Just a few steps of relaxation on the homogeneous problem on the finest grid quickly reduces a significant portion of a random initial guess, leaving error that can then be said to be locally algebraically smooth. If this prototype is used locally to define interpolation from some preselected coarse grid, then a coarse-grid problem that adequately represents the algebraically smooth error on the fine grid can be created. We can now iterate to an appropriate coarsest grid and interpolate a prototype of the smooth error to all grids. Proceeding recursively, this resembles a full approximation scheme (FAS; cf. [5]) multigrid process for the algebraically smooth component, rather than the usual correction scheme method.

In this manner, a good prototype of a single algebraically smooth component can be determined and the resulting solver tested by applying it to the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, with a random initial guess. If the solver proves sufficient, then

the adaptive stage is complete. Inefficiency in the resulting solver indicates that relaxation and coarse-grid correction are not yet appropriately complementary, and that there are distinct algebraically smooth components that are not being well enough approximated on the coarser levels. Since these components are being reduced neither by relaxation nor coarse-grid correction, they can be exposed by an iteration as above, with the current solver taking the place of relaxation. This may be repeated until acceptable convergence rates are attained.

Thus, we sketch the adaptive procedure as follows.

ALGORITHM 1 (abstract adaptive process).

1. $k = 1$ $\mathbf{x}^{(1)}$ l $_l(\mathbf{x}^{(l)}, \mathbf{b}^{(l)})$ ν $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$
2. $_k(\mathbf{x}^{(k)}, \mathbf{0})$
3. I_{k+1}^k $\mathbf{x}^{(k+1)} = (\mathbf{x}^{(k)})_c$ $\mathbf{x}^{(k)}$ $k+1$ $k = k+1$ 2
4. $k > 1$ $k = k-1$ $\mathbf{x}^{(k)} = I_{k+1}^k \mathbf{x}^{(k+1)}$ $_k(\mathbf{x}^{(k)}, \mathbf{0})$
5. $k = 1$ $\mathbf{x}^{(1)}$ l $_l(\mathbf{x}^{(l)}, \mathbf{b}^{(l)})$ ν $A^{(l)}\mathbf{x}^{(l)} = \mathbf{b}^{(l)}$ $_1(\mathbf{x}^{(1)}, \mathbf{0})$ 2

3.2. Principles. Perhaps the easiest way to understand the adaptive methodology is to begin with the principles upon which it is based. Here, we list the core ideas that motivate and provide a foundation for αMG, with the primary focus on the αAMG scheme. The pragmatic reader may prefer to defer reading this discussion until after section 4 or to see [12].

Algebraic smoothness. The concept of algebraic smoothness is of utmost importance in achieving an optimally efficient AMG method. Since we allow reduction of the error only through the processes of relaxation and coarse-grid correction, the algebraically smooth error (which, by definition, is slow to be resolved by relaxation) must be accurately corrected from the coarse grid. That is, interpolation must be very accurate for algebraically smooth components. In fact, a stronger requirement is imposed by the eigenvector approximation criterion that, for a given eigenvector, $\mathbf{u}$, of A, interpolation must reconstruct $\mathbf{u}$ to an accuracy proportional to its eigenvalue [6, 22].

The AMG methods considered here are based on some defining characteristic of what algebraic smoothness means. This definition generally amounts to articulating an algebraic property of the errors that the given (fixed) relaxation process cannot reduce effectively. For example, classical AMG is developed based on properties of a polynomial iterative method such as the Richardson iteration:

$$\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} - \frac{1}{\|A\|}(A\tilde{\mathbf{x}} - \mathbf{b}).$$

For this iteration, the error, $\mathbf{e} = A^{-1}\mathbf{b} - \tilde{\mathbf{x}}$, converges slowly in the A-norm,

$$\|\mathbf{e}\|_A = \sqrt{\langle A\mathbf{e}, \mathbf{e}\rangle},$$

if and only if $\mathbf{e}$ yields a small generalized Rayleigh quotient:

$$RQ_A(\mathbf{e}) = \frac{\langle A\mathbf{e}, A\mathbf{e}\rangle}{\|A\|\langle A\mathbf{e}, \mathbf{e}\rangle}.$$

For this reason, the set of algebraically smooth vectors is often also referred to as the near null space of A; we use these two terms interchangeably, with the prevailing meaning to be that of vectors that the chosen relaxation scheme is slow to reduce. Proper use of this defining property of algebraic smoothness gives AMG its potential for optimal performance over a wide range of problems. It enables coarsening processes that rightfully depend on the matrix and hopefully capture the errors that relaxation cannot eliminate.

Most AMG methods, however, require additional assumptions regarding algebraically smooth error that allow them to capitalize on the special nature of the assumed algebraic smoothness. For example, classical AMG rests on two related main assumptions: that the constant vector, $\mathbf{1}$, must be interpolated exactly, and that algebraically smooth errors vary slowly along strong connections. While this enables effective treatment of many problems, it also restricts the class to which these algorithms apply. Many discrete systems exhibit algebraically smooth errors that vary dramatically across strong connections, and many others offer no clear understanding of what strength of connection even means. Also, as we discuss further in the next section, $\mathbf{1}$ is not necessarily a good representative of algebraically smooth error. A major goal of the adaptive process is to capitalize on the definition of algebraically smooth error without making additional specific assumptions about its character.

Prototypes. A central idea in the development of αAMG methods is the use of prototypes that serve as representatives of algebraically smooth error. In fact, prototypes are used in the development of nearly all multigrid methods, often implicitly. As mentioned above, for example, classical AMG uses $\mathbf{1}$ to build its matrix-based interpolation coefficients (see (2.4) and the discussion in section 2). αAMG differs in that it attempts to generate these prototypes automatically.

Given a set of these prototypes, it is important to recognize that they should only be used locally as representatives of algebraically smooth error. Otherwise, it would not be possible to achieve optimality. As an illustration of this point, consider the fact that the preconditioned conjugate gradient method can be interpreted as a two-level αMG method that uses the generated prototypes globally as representatives of slow-to-converge error components. (Here, the smoother plays the role of the preconditioner, and the coarse-grid corrections are the Krylov subspace projections; see [15] for details.) In general, errors left by relaxation consist of a large fraction of the spectrum of the matrix, so that coarsening must effectively approximate $O(N)$ components with varying degrees of accuracy. The goal in designing a multigrid interpolation operator is to achieve this approximation property by using only a small, $O(1)$, number of computed prototypes.

This use of a small number of prototypes to achieve approximation of a much larger space is a cornerstone of multigrid methods. It is essential, then, that each prototype be used effectively as a representative of many components with similar local character. Remember that the constant vector, $\mathbf{1}$, is used in classical AMG to define an interpolation whose range not only includes $\mathbf{1}$ but also approximates all smooth errors. This is analogous to how local basis functions are used in finite elements: piecewise polynomial basis functions are used locally not only because they can reconstruct their global counterparts, but also because they can approximate all smooth components of the solution of the PDE. The global prototype is a representative of many algebraically smooth components and thus is used locally to determine an interpolation operator that has many such smooth components in its range.

Self-testing. Computation of a rich supply of prototypes can be done by carefully testing the algorithm as it evolves. These self-tests should be done on a problem with known solution. The homogeneous problem, $A\mathbf{x} = \mathbf{0}$, is especially appropriate because it avoids trouble with machine representation when the approximation is very close to that solution. For our αMG schemes, we can test the current version of the algorithm on $A\mathbf{x} = \mathbf{0}$ by measuring the A-norm of the error of successive iterates. This test serves a dual role: it signals when the algorithm is performing well enough, and it produces a good prototype when it is not. Assuming that enough iterations are used, the prototype must be appropriate because it is algebraically smooth (relaxation is not eliminating it), yet poorly represented by whatever current coarsening process is being used (if any). This prototype can then be used in the underlying algorithm precisely where the additional smoothness assumptions were used. For classical AMG, this means that the prototype would provide information on the correct coefficients to use in eliminating the matrix connections to points that are only on the fine level.

While the homogeneous problem is important as a measure of performance because it has a known solution, other measures can be useful in monitoring the evolving behavior and improving the prototypes. This issue is most clearly exposed when a direct solver is used on the coarsest level, where solving the homogeneous problem seems paradoxical: why solve $A\mathbf{x} = \mathbf{0}$ when all you presumably get is $\mathbf{x} = \mathbf{0}$? At this point, a simple approach is to just accept the prototype computed on the next finer level so that the coarsest level is never really used in the adaptive process. This means that the prototype is never really improved there either. It may be better to enhance the coarsest-level prototype by using a more precise measure of smoothness. For our αMG schemes, we could choose to improve the prototype, $\mathbf{x}$, on the coarsest level by minimizing the generalized Rayleigh quotient, $RQ_A(\mathbf{x})$. The situation is more complicated in the case of multiple prototypes, however; in neither the scalar PDE case considered in section 4 nor the αSA method in [12] have we found any need for improving the coarsest-level prototype in this manner, so we choose not to address this issue further. The Rayleigh quotient can, however, be useful as a diagnostic tool in assessing how the adaptive process is behaving, as in [20].

Range of interpolation. The primary aim of coarsening in any multigrid process is to appropriately complement relaxation, allowing fast solution of the given system of equations. To accomplish this, all algebraically smooth errors must be approximated well by vectors in the range of interpolation. Within the adaptive process, such errors are represented only by the prototypes, and therefore the primary aim of this process is to develop an interpolation operator that accurately approximates the prototypes in such a way that good multigrid convergence is obtained by the resulting cycle.

Achieving the desired multigrid performance requires more than simply fitting the set of given prototypes exactly. Error in the direction of a single prototype can easily be corrected from a single coarse DOF, but this generally does not provide enough of an acceleration to the relaxation process to yield a scalable algorithm. Indeed, such an algorithm resembles a preconditioned conjugate gradient iteration [15] that would require many steps to converge for the matrices that typically arise from discretized PDEs. To develop an effective interpolation operator, it is necessary to consider each prototype as a [illegible] of a class of errors that relaxation is slow to reduce. Since pointwise relaxation is a local process, any error that has a similar character to the prototype, measured locally, is also expected to be slow to converge with relaxation. We look for all such errors to be approximated by the

range of interpolation and, thus, employ a localization (partition of unity) strategy in defining our multigrid interpolation operator. This strategy, unfortunately, is also generally not sufficient to achieve good multigrid performance: piecewise constant interpolation fits the prototypical constant vector exactly, and many locally similar errors well, yet leads to poor V-cycle performance for solving Laplace's equation [5]. Interpolation must also be chosen so that a variational coarse-grid correction is not distracted by non–algebraically smooth errors that are also in the range of interpolation. Classical AMG interpolation (as discussed in section 2) is known to attain all of these properties for typical scalar elliptic equations based on the single prototype of the constant vector, and we aim to mimic these properties here in the more general adaptive AMG setting.

Constructing interpolation in this way (described in detail in section 4) has important consequences. Not only is each prototype fit closely by the range of interpolation, but so are all locally similar vectors are as well. Thus, the range of interpolation includes many algebraically smooth vectors, including, if possible, many vectors that are significantly smoother (measured, for example, by their Rayleigh quotients) than the prototype used to construct interpolation in the first place. Given the lower cost of relaxation on the coarser grid, we look to improve the prototype through iteration on the homogeneous coarse-grid problem. Thus, we coarsen in the adaptive process, to quickly expose better prototypes of the fine-level algebraically smooth error. As the fine-level prototype is, itself, well represented on the coarse grid, the adaptive process attempts to improve the prototype directly on this level, instead of through a more cumbersome correction scheme. Success of this process rests on our ability to find components in the range of the constructed interpolation operator that are significantly smoother than the current prototype. Again, the keys to this success are in AMG's local operator-induced interpolation approach, which ensures that the range of P is rich in components that are locally similar to the prototypes, and the variational construction of the coarse-grid operators through the Galerkin condition [27], which allows iteration on the coarse grid to find a significantly smoother prototype than the one upon which interpolation was based (if such a vector exists).

Optimality. Multigrid methods are useful solution techniques because they exhibit optimal traits, such as $O(N)$ or $O(N \log N)$ scaling in both number of operations and storage. As such, any adaptive multigrid process should also retain this optimality. In particular, the adaptive process must not make requirements of the solver that compromise the optimality of the overall process and must itself scale optimally in operation count and storage.

Classical AMG controls complexity by its intricate way of determining the coarse points and its careful use of the matrix entries. The adaptive AMG approach assumes that a suitable coarsening process is available (such as compatible relaxation [7, 16, 19]), with the attendant assumption that there is sufficient reduction in grid sizes from fine to coarse levels. When fitting multiple prototypes, however, it is tempting to abandon the tight control on the stencil of interpolation (such as to the nonzero pattern of A_{fc}, the submatrix of A linking fine- to coarse-grid nodes, as is used in classical AMG) to allow for exact fitting of more prototypes. This must be done with utmost care, as each new nonzero entry in the interpolation operator can lead to new nonzero connections in the coarse-grid matrix. Care must also be taken to ensure that the stencil of interpolation is not too small: early experiments limited the size of C_i, the set of coarse-grid points from which the fine-grid point i is interpolated, to the number of prototypes being fit, which led to an extremely inefficient algorithm

because a single prototype could be used to define only one-sided interpolation, and so multiple prototypes were needed for good convergence even for second-order scalar PDEs.

Constructing a prototype set of minimal size is important to practical success and control over the complexity of the algorithm. While the same near null space may be well represented by a small number of very precise vectors or a larger number of less-resolved prototypes, the costs of the adaptive process and, possibly, the resulting method increase with the number of prototypes. For this reason, it is often more efficient to consider improvement of the existing prototype(s) than to add a new one. As each prototype emerges, we can consider improvement of its representation by running the setup cycle again, with the prototype as an initial guess for the algebraically smooth error. This either improves the prototype as a representative of the algebraically smooth components that are not well represented by the rest of the prototype set, or signals that the prototype is a strong representative of such components. In either case, valuable information about the adaptive process is gained, either through an improved prototype or through the knowledge that further improvement in the resulting algorithm must be gained through additional prototypes.

To ensure that the adaptive process is also optimal, adaptations are made whenever sufficient new information becomes known, but also only when the change is expected to improve the overall algorithm. For example, we develop the algebraically smooth prototype in a manner similar to the full approximation scheme, representing the prototype on all levels and looking for the algebraically smoothest component on each level. The prototype on a given level is discarded when it can be improved from a coarser grid, as we expect relaxation on coarser levels to better expose the algebraically smooth components that we seek. We do not, however, update interpolation or coarse-grid operators on the upward traverse of the setup V-cycle. Such an adaptation would be wasted because operators at higher levels also change as the cycle moves toward the finest grid. For this reason, while we allow for multiple V-cycles to be performed in the setup phase, the last cycle always terminates at the coarsest grid.

Termination of the adaptive process must also be properly implemented in order to maintain optimality. Experience has shown that improvement in the resulting multigrid process becomes less cost-effective with the number of setup phases and the total amount of relaxation in the adaptive step. A method with an acceptable convergence factor may be attained after even a single adaptive step, and a second adaptive step improves this factor by only a fraction of a percent. This may be addressed by reducing the amount of relaxation per adaptive step to a single sweep on each level and monitoring the convergence of the prototype vector between sweeps (for example, measuring its Rayleigh quotient). Unfortunately, the majority of the cost of an adaptive step is in the computation of interpolation and coarse-grid operators and not relaxation, so performing many adaptive steps is undesirable. For this reason, we choose a strategy attempting to minimize the number of setup cycles, motivated in [20], with enough relaxations in these cycles to quickly achieve convergence factors within a few percent of the apparent optimal performance.

4. Interpolation for adaptive AMG. Our goal in developing a new type of AMG method is to extend the applicability of classical AMG schemes. Thus, a primary concern is the generalization of the definition of interpolation in AMG. The guiding principles for this generalization come from basic properties of all multigrid algorithms:

- simple relaxation is inefficient for solving $A\mathbf{x} = \mathbf{b}$ on error components, $\mathbf{e}$, whose residuals, $A\mathbf{e}$, are small relative to $\mathbf{e}$ in some sense;
- efficient multigrid performance depends on effective complementarity of relaxation and coarsening so that they efficiently cooperate to eliminate all error components.

In this section, we propose a method for choosing interpolation that is appropriate in the case of M-matrices for which the algebraically smoothest mode is not close to the constant vector. As such, this interpolation may be viewed as an extension of the classical AMG interpolation [27]. In developing this new interpolation procedure, we consider the case of pure algebraic coarsening; however, for practical reasons, we choose to first implement the algorithm in the case of regular geometric coarsening. We do not address the important question of coarse-grid selection here, but expect that in a practical situation, the coarsening techniques of compatible relaxation [7, 16, 19] would be necessary in choosing an adequate coarse grid. The numerical results presented in section 6 are for this interpolation on geometrically chosen grids in the case of a scalar PDE.

4.1. Definition of interpolation. Since the success of our methods depends on the complementarity of relaxation and coarse-grid correction, a good starting point for defining interpolation is to consider a vector, $\mathbf{e}$, that is not quickly reduced by relaxation. Using a simple (pointwise) relaxation scheme, such as Gauss–Seidel, this also means that $A\mathbf{e} \approx \mathbf{0}$ or

$$a_{ii}e_i \approx -\sum_{j\in C_i} a_{ij}e_j - \sum_{k\in F_i} a_{ik}e_k, \tag{4.1}$$

where, as in (2.2), we assume a splitting of N_i into C_i and F_i. Again writing error e_k, $k \in F_i$, as

$$e_k \approx \sum_{j\in C_i} \omega_{kj}^i e_j + \omega_{ki}^i e_i$$

and using this expression in the second sum of (4.1) yields a general interpolation formula for point $i \in F$,

$$e_i = -\sum_{j\in C_i} \left(\frac{a_{ij} + \sum_{k\in F_i} a_{ik}\omega_{kj}^i}{a_{ii} + \sum_{k\in F_i} a_{ik}\omega_{ki}^i} \right) e_j. \tag{4.2}$$

The αAMG interpolation is different from that used in classical AMG [27] in that the $\{\omega_{kj}^i\}$ are chosen to depend on both the entries in A and a (computed) prototype, $\mathbf{x}^{(1)}$, that represents many algebraically smooth components. How this prototype is computed is the subject of section 5.

To be specific about the choice of $\{\omega_{kj}^i\}$, consider the idea of twice-removed interpolation [8]. Suppose we have a point, i, whose neighbors have been partitioned into the two sets, C_i and F_i. The problem of collapsing the $F - F$ connections is equivalent to that of determining a way to interpolate to point $k \in F_i$ from points $j \in C_i$ (or, more generally, $j \in C_i \cup \{i\}$). That is, we seek to write (as before)

$$e_k = \sum_{j\in C_i} \omega_{kj}^i e_j, \tag{4.3}$$

dropping the term $\omega_{ki}^i e_i$, under the assumption that k is as strongly connected to some point (or points) in C_i as it is to i. If there is a particular vector, $\mathbf{x}^{(1)}$, that we want to be in the range of interpolation, then we ask that (4.3) hold when $\mathbf{e}$ is replaced by $\mathbf{x}^{(1)}$. This constraint with one vector, $\mathbf{x}^{(1)}$, fixes one DOF of the possibly many for set $\{\omega_{kj}^i\}$, but leads to a unique F_i-interpolation formula if it is restricted to be of the form $D^{-1}A_{fc}$, where D is a diagonal matrix. (This choice is motivated by the discussion in [11].) The matrix D is determined by

$$d_{kk}^i x_k^{(1)} = -\sum_{j\in C_i} a_{kj} x_j^{(1)}$$

or

$$d_{kk}^i = \frac{-\sum_{j\in C_i} a_{kj} x_j^{(1)}}{x_k^{(1)}}. \tag{4.4}$$

Thus, choosing $\omega_{kj}^i = (d_{kk}^i)^{-1} a_{kj}$ in (4.3), the F_i-interpolation formula is

$$e_k = -\sum_{j\in C_i} \frac{a_{kj}}{d_{kk}^i} e_j.$$

Interpolation to $i \in F$, given by (4.2), then has the particular form

$$e_i = -\sum_{j\in C_i} \frac{1}{a_{ii}} \left(a_{ij} + \sum_{k\in F_i} a_{ik} \frac{a_{kj} x_k^{(1)}}{\sum_{j'\in C_i} a_{kj'} x_{j'}^{(1)}} \right) e_j. \tag{4.5}$$

Note that the interpolation operator, P, as a mapping from C to $F \cup C$, then has the form

$$P = \begin{bmatrix} W \\ I \end{bmatrix},$$

where W is the matrix of coefficients determined by (4.5).

This αAMG interpolation formula is a simple generalization of the classical AMG formula that allows for a sense of smoothness that may differ from what AMG conventionally uses. The primary assumption used in standard AMG to collapse $F - F$ connections is that the smoothest error component is constant [27]. Thus, classical AMG interpolation is recovered from the formula in (4.5) by choosing $\mathbf{x}^{(1)} \equiv \mathbf{1}$.

The iterated interpolation of (4.3) and (4.4) was chosen to exactly match the near null space approximation, $\mathbf{x}^{(1)}$. The final interpolation in (4.5), however, does not necessarily match this vector exactly. For a fine-grid point, i, the misfit is easily calculated as

$$x_i^{(1)} - (P\mathbf{x}_c^{(1)})_i = \frac{1}{a_{ii}} (A\mathbf{x}^{(1)})_i. \tag{4.6}$$

This is in accord with the classical AMG point of view that interpolation must be more accurate for errors that yield smaller residuals. In fact, it may be directly compared with the eigenvector approximation criterion, as described by Brandt [6, Theorem 4.1], which relies on the existence of a constant, C_0, bounded away from zero such that

$$C_0 \sum_i a_{ii} (e_i - (P\mathbf{e}_c)_i)^2 \le \mathbf{e}^T A \mathbf{e}$$

for all $\mathbf{e} \in \mathbb{R}^N$. Squaring (4.6), multiplying through by a_{ii}, and summing over both fine- and coarse-grid points, we have

$$\sum_i a_{ii}(x_i^{(1)} - (P\mathbf{x}_c^{(1)})_i)^2 \leq \sum_i \frac{1}{a_{ii}}(A\mathbf{x}^{(1)})_i^2 = (\mathbf{x}^{(1)})^T A \left(\operatorname{diag}\left(\frac{1}{a_{ii}}\right)\right) A\mathbf{x}^{(1)}$$
$$\leq \rho\left(A^{\frac{1}{2}}\left(\operatorname{diag}\left(\frac{1}{a_{ii}}\right)\right)A^{\frac{1}{2}}\right)(\mathbf{x}^{(1)})^T A\mathbf{x}^{(1)}.$$

For a diagonally dominant operator with constant diagonal, such as the finite-element Laplacian, $\rho(A^{\frac{1}{2}}(\operatorname{diag}(\frac{1}{a_{ii}}))A^{\frac{1}{2}})$ is easily bounded by 2, and so the constant, C_0, in Brandt's bound is not made unduly small due to the misfit in the interpolation of $\mathbf{x}^{(1)}$. While such a bound is only for the prototype, $\mathbf{x}^{(1)}$, and not for any arbitrary fine-grid vector (as required by the eigenvector approximation criteria), we consider $\mathbf{x}^{(1)}$ to be an appropriate prototype of the algebraically smooth error for which this bound is most difficult to achieve and, thus, indicative of a good interpolation scheme.

. . . . 1. While this paper considers methods involving just one prototype vector, $\mathbf{x}^{(1)}$, appropriate for scalar PDEs, these concepts can also be generalized to systems. Consider discretizing a system so that its DOFs are located on the same grid; i.e., there are d DOFs colocated at each node. Since we seek to generalize the ideas from the scalar case, we start by generalizing the notation: A_{kj} becomes the $d \times d$ matrix of connections between the DOFs located at nodes k and those located at node j, the diagonal entries of D (D_{kk}^i) become $d \times d$ matrices, and $\mathbf{x}^{(1)}$ becomes the matrix, $X^{(1)}$, composed of d columns of distinct prototypes. The restriction of $X^{(1)}$ to the d DOFs at node k is denoted by $X_k^{(1)}$. The analogue of (4.4) is then

$$D_{kk}^i = -\Big(\sum_{j \in C_i} A_{kj} X_j^{(1)}\Big)\big(X_k^{(1)}\big)^{-1}.$$

The F_i-interpolation formula for systems thus becomes

$$\mathbf{e}_k = -\sum_{j \in C_i} (D_{kk}^i)^{-1} A_{kj} \mathbf{e}_j,$$

which yields the final nodal interpolation formula

$$\mathbf{e}_i = -A_{ii}^{-1}\left(\sum_{j \in C_i}\left(A_{ij} + \sum_{k \in F_i} A_{ik}(D_{kk}^i)^{-1}A_{kj}\right)\right)\mathbf{e}_j.$$

4.2. Theoretical properties. One situation that can cause difficulty for classical AMG is when the matrix is rescaled. For example, if A is the discretization of a Poisson-like problem, then it is generally true that A applied to $\mathbf{1}$ yields a relatively small residual: $A\mathbf{1} \approx \mathbf{0}$. This means that constant vectors are indeed algebraically smooth, as classical AMG presumes. Rescaling A by multiplying it on both left and right (to preserve symmetry) by a positive diagonal matrix can dramatically change this property. Thus, if A is replaced by $\hat{A} = SAS$ for some positive diagonal matrix S, then $\hat{A}(S^{-1}\mathbf{1}) \approx 0$, and the new near null space component is actually $S^{-1}\mathbf{1}$. If the diagonal entries of S have significant variation in them, then $S^{-1}\mathbf{1}$ has a significantly different character than does $\mathbf{1}$. For classical AMG, this can cause a dramatic deterioration in convergence rates, although it can be prevented if the scaling is supplied to AMG so that the original matrix can essentially be recovered (as in [8]), but this is

not always possible in practice. Fortunately, as the following result shows, such scaling causes no problem for αAMG, provided the scaled prototype can be accurately computed.

THEOREM 4.1. [illegible] S [illegible] $\mathbf{x}^{(1)}$ $\hat{\mathbf{x}}^{(1)} = S^{-1}\mathbf{x}^{(1)}$ $\mathbf{b}$ [illegible] $\hat{\mathbf{b}} = S\mathbf{b}$ [illegible] α [illegible] $\hat{A}\hat{\mathbf{x}} = \hat{\mathbf{b}}$ [illegible] $\hat{\mathbf{x}}^{(1)}$ [illegible] $\hat{A}$ [illegible] α [illegible] $A\mathbf{x} = \mathbf{b}$ [illegible] $\mathbf{x}^{(1)}$ [illegible] A [illegible].

PROOF. Given a coarse-grid set, C, and the complementary fine-grid set, F, partition A and S to have the forms

$$A = \begin{bmatrix} A_{ff} & A_{fc} \\ A_{cf} & A_{cc} \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} S_f & 0 \\ 0 & S_c \end{bmatrix},$$

so that

$$\hat{A} = \begin{bmatrix} S_f A_{ff} S_f & S_f A_{fc} S_c \\ S_c A_{cf} S_f & S_c A_{cc} S_c \end{bmatrix}.$$

The weights, $\hat{\omega}^i_{kj}$, for matrix $\hat{A}$ are given by

$$\begin{aligned} \hat{\omega}^i_{kj} &= \frac{\hat{a}_{kj}\hat{x}^{(1)}_k}{\sum_{j' \in C_i} \hat{a}_{kj'}\hat{x}^{(1)}_{j'}} \\ &= \frac{s_k a_{kj} s_j s_k^{-1} x^{(1)}_k}{\sum_{j' \in C_i} s_k a_{kj'} s_{j'} s_{j'}^{-1} x^{(1)}_{j'}} = s_k^{-1}\omega^i_{kj} s_j, \end{aligned}$$

where the weights, ω^i_{kj}, are chosen as for matrix A. Equation (4.5) then gives (with some algebra)

$$e_i = -\sum_{j \in C_i} s_i^{-1} \left(\frac{a_{ij} + \sum_{k \in F_i} a_{ik}\omega^i_{kj}}{a_{ii}} \right) s_j e_j, \qquad i \in F.$$

For $i \in C$, again simply take the value from the coarse grid and assign it as the value on the fine grid. Thus, the interpolation operator, $\hat{P}$, is of the form

$$\hat{P} = \begin{bmatrix} \hat{W} \\ I \end{bmatrix} = \begin{bmatrix} S_f^{-1} W S_c \\ I \end{bmatrix} = S^{-1} P S_c,$$

where P is the interpolation operator from the unscaled case. Further, considering the coarse-grid operator, $\hat{A}_c$, note that $\hat{A}_c = S_c P^T A P S_c = S_c A_c S_c$, where A_c is the coarse-grid operator from αAMG on A.

Thus, the coarse-grid operator for the scaled problem is simply the scaled version of the coarse-grid operator for the unscaled problem. Since standard relaxation techniques such as Gauss–Seidel or Jacobi (both pointwise and block forms) are scaling-invariant (that is, if A is scaled to SAS as above, initial guess $\mathbf{x}^{(0)}$ to $S^{-1}\mathbf{x}^{(0)}$, and initial right side $\mathbf{b}$ to $S\mathbf{b}$, then the approximation generated changes from $\mathbf{x}^{(1)}$ to $S^{-1}\mathbf{x}^{(1)}$), we see that the entire process is independent of any diagonal scaling. □

THEOREM 4.2. [illegible] 4.1 [illegible].

PROOF. The proof is identical in form to that for the scalar case and is, thus, omitted. □

5. Determining $\mathbf{x^{(1)}}$. Successful implementation of this interpolation scheme relies on having an appropriate prototype vector, $\mathbf{x}^{(1)}$ (or set of vectors, $X^{(1)}$). Since we rely on the complementarity of relaxation and coarsening, the best choice for $\mathbf{x}^{(1)}$ would be a representative of the vectors for which relaxation is inefficient. Thus, a straightforward method for generating this prototype would be to start with a vector that is hopefully rich in all components (i.e., all eigenvectors of symmetric A), relax on $A\mathbf{x} = \mathbf{b}$ for some $\mathbf{b}$, and then determine the error in the approximate solution after a sufficient number of relaxations.

We typically make use of relaxation schemes whose error-propagation matrices have the form $I - BA$. While, for general B, it is possible that the slow-to-converge modes of the relaxation iteration, $I - BA$, are not modes for which $A\mathbf{e} \approx \mathbf{0}$, in many practical situations they are. In particular, for the pointwise relaxation schemes considered here, the two descriptions of algebraically smooth error are equivalent. In fact, for many choices of B, the true near null space of A is accurately reflected in the vectors for which relaxation is inefficient. Knowledge of this space could be used as it is with standard AMG to determine an effective coarsening process. Our focus, however, is on the case where this knowledge is misleading, inadequate, or even unavailable. We thus concentrate on the case that a good prototype of errors that are slow to converge under the given relaxation scheme, $\mathbf{x}^{(1)}$, is not known.

Start with a vector generated randomly from a uniform distribution on $(0, 1)$. (Consideration of positive vectors is motivated by the case of scalar second-order operators, which tend to discretize as M-matrices and so have positive near null space vectors. A more general choice is appropriate when considering problems such as linear elasticity, but care must be taken because the definition of interpolation for αAMG presented here breaks down if $\sum_{j\in C_i} a_{kj} x_j^{(1)} = 0$ for any $i \in F$, $k \in F_i$.) Such a vector is, in general, not equally rich in all error components. However, in the scalar PDE case, it tends to be rich enough that a few relaxation sweeps on the homogeneous problem, $A\mathbf{x} = \mathbf{0}$, produce a good representative of the slow-to-converge components. Note that the homogeneous problem is advantageous to use here because the prototype is simply the error in approximating the exact solution, $\mathbf{x} = \mathbf{0}$. Thus, starting with a random initial guess and performing relaxation on $A\mathbf{x} = \mathbf{0}$ generates a prototype vector, $\mathbf{x}^{(1)}$, that represents the slow-to-converge components and that can then be used in the interpolation formula developed in section 4.1.

Unfortunately, generating the prototype by fine-grid relaxation alone is effective only in terms of the first coarse level and is, in general, quite inefficient in a multilevel setting. To produce a prototype that is smooth enough to represent the components associated with smoothness on very coarse levels, a multilevel scheme is needed. Here, we again measure smoothness by the eigenvector approximation criterion. Basing every interpolation operator on a single component, whose Rayleigh quotient is near the minimal eigenvalue on the finest grid, requires significant algebraic smoothness in that component, as the eigenvector corresponding to this eigenvalue must be interpolated with accuracy proportional to this small value. For coarser grids, such smoothness is much more efficiently represented by calculation on these grids. Thus, we start with a random guess on the fine level and perform a few (ν_0) relaxation sweeps there to generate a tentative $\mathbf{x}^{(1)}$. Using this current prototype, an interpolation operator is computed (as in section 4.1), and the coarse-level and restriction operators are formed using the Galerkin condition. We use injection of $\mathbf{x}^{(1)}$ (direct restriction of the values on the C-points) to form a coarse-level initial guess, relax ν_1 times, and recurse to the coarsest level. From this coarsest level, interpolate and relax ν_2 times on the vector

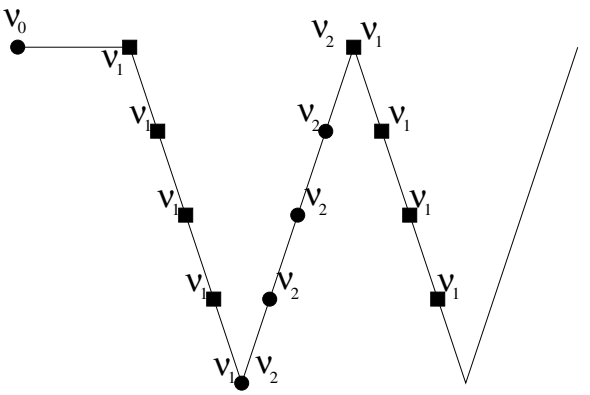


FIG. 5.1. *The setup scheme for determining* $x^{(1)}$.

all the way to the finest level, but do not recompute the coarse-level and grid-transfer operators.

The cycle can then be repeated, using the resulting vector as an overall initial guess in an attempt to further improve the prototype. This cycling strategy is illustrated in Figure 5.1, where boxes indicate stages where coarse-level operators are computed and circles indicate stages where only application of the current solver is necessary. Note that since the multigrid operators are computed only on the downward part of the cycle, no relaxation is necessary on the upward part of the final setup cycle. As is discussed in section 6, this procedure yields multigrid solvers with convergence factors bounded independently of mesh size (tested up to 1024×1024 grids) for many scalar problems.

One important benefit of generating the initial prototype vector in a multilevel fashion is the ability to implement a proper transition to simplicity in the algorithm. That is, since we begin by relaxing on a random vector (assumed to be rich in all components), it is easy to tell whether relaxation alone is sufficient to solve either the fine-grid problem or one of the generated coarse-level problems. If this is indeed the case, then no additional labor is needed in designing an algorithm because an efficient solver already exists.

For systems of PDEs and higher-order problems, an added wrinkle is the need to generate multiple prototype vectors. The cycling scheme described above can easily be generalized to account for multiple prototypes, by replacing the role of relaxation with the current solver. That is, once a single prototype is known, additional prototypes need to be generated that both are algebraically smooth and that represent distinct slow-to-converge components from the existing prototype. One way to generate such components is to apply the current solver to the homogeneous problem, in much the same way as we currently use relaxation to expose its own slow-to-converge components. Further investigation is necessary, however, to ensure that the generated prototypes represent a full set of algebraically smooth errors, without being redundant. This is the subject of current research.

6. Numerical results. To examine the feasibility of this approach, we implemented a solver for the special case of a rectangular grid in two dimensions with full coarsening. This restriction in generality has a notable effect on the range of problems that are able to be reasonably considered (anisotropy, for example, becomes

much more difficult to account for in this setting). However, examining problems without such difficulties, we feel that we can obtain a good initial indication of the performance of this approach. In particular, the aim here is to test the quality of the interpolation operator, not that of the coarsening procedure. Indeed, given current research into new coarsening techniques [7, 16, 19], it is difficult to say which coarsening method would be most appropriate for comparison.

We consider several measures of the effectiveness of the algorithm. Of primary importance is the total time to solution, given only the matrix equation, $A\mathbf{x} = \mathbf{b}$. Here, solving a problem is defined as reducing the residual by 10 orders of magnitude, and the wall-clock time to solution on a modern desktop workstation (2.66 GHz Intel Pentium 4) is measured. Another relevant measure is the asymptotic convergence factor of the resulting cycle. While setup costs may form a significant portion of the cost of solving a linear system with a single right-hand side, many problems require repeated solution with multiple right-hand sides (such as in implicit time stepping). In these cases, the (possibly large) setup cost can be amortized over the number of solutions, and the cost of only the solution stage is important. The asymptotic convergence factor reflects this cost, as a lower factor requires fewer iterations in the solution phase. We discard the usual AMG measures of grid and operator complexity because, in the structured coarsening framework considered here, these measures are constant for all fine-grid operators of the same mesh and stencil sizes.

As discussed in section 5, the setup may be performed in an iterative fashion. This is an appealing feature because, in practice, convergence of the prototype vector can be measured as an indicator of convergence of the method. This iteration is, however, quite expensive, as it involves computation of new interpolation and new Galerkin coarse-grid operators at each iteration. Testing indicates that it is usually significantly more efficient to perform more relaxation sweeps (i.e., increase ν_0 and ν_1) in a single setup cycle than it is to perform multiple setup cycles with fewer iterations per cycle to generate a single prototype [20]. Thus, in the results that follow, we first consider performing only a single setup cycle and track the number of relaxations (values of ν_0 and ν_1) necessary to achieve highly efficient solver performance. For the problems considered here, this calibrated AMG method is typically more efficient than the adaptive AMG method, whose results appear later. In the adaptive procedure, we fix the number of relaxations used in each setup phase and perform setup iterations until an acceptable solver is determined (measured by the error reduction in the solver). Here, we use the adaptive approach described in section 5 and Figure 5.1, where a single prototype is iteratively improved until the resulting solver is deemed to perform well.

We consider four PDEs as test problems, all discretized using bilinear finite elements on the unit square. Problem 1 is Laplace's equation with pure Dirichlet boundary conditions. Problem 2 is Laplace's equation with pure Neumann boundary conditions. Problems 3 and 4 are discretizations of

$$-\nabla \cdot \mathcal{K}(\mathbf{x}) \nabla p(\mathbf{x}) = 0,$$

with Dirichlet boundary conditions on the East and West boundaries and Neumann boundary conditions along the North and South boundaries. For Problem 3, $\mathcal{K}(\mathbf{x})$ is chosen as

$$\mathcal{K}(\mathbf{x}) = \begin{cases} 10^{-8}, & \mathbf{x} \in [\frac{1}{3}, \frac{2}{3}]^2, \\ 1, & \text{otherwise.} \end{cases}$$

TABLE 6.1

Standard AMG: wall-clock time in seconds (and iteration count) to reduce residuals by 10^{10}. *The notation "*> 200*" indicates that the residual convergence criterion was not met after* 200 *iterations.*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (9)	0.22 (9)	0.91 (9)	3.32 (9)	13.13 (9)
Problem 1u	0.05 (9)	0.24 (9)	0.86 (9)	3.33 (9)	13.15 (9)
Problem 1r	> 200	> 200	> 200	> 200	> 200
Problem 2	0.04 (8)	0.20 (8)	0.81 (8)	3.12 (8)	12.30 (8)
Problem 2u	0.09 (27)	0.93 (52)	5.64 (90)	36.94 (158)	> 200
Problem 2r	> 200	> 200	> 200	> 200	> 200
Problem 3	0.04 (9)	0.25 (9)	0.89 (9)	3.41 (9)	13.23 (9)
Problem 3u	0.11 (26)	0.78 (41)	4.73 (69)	29.81 (124)	> 200
Problem 3r	> 200	> 200	> 200	> 200	> 200
Problem 4	0.05 (12)	0.28 (12)	1.04 (11)	4.40 (13)	17.64 (14)
Problem 4u	0.18 (64)	3.14 (179)	> 200	> 200	> 200
Problem 4r	> 200	> 200	> 200	> 200	> 200

For Problem 4, $\mathcal{K}(\mathbf{x})$ is assumed to be constant on each element and chosen to have value 10^{-8} on 20% of the elements (chosen randomly) and value 1 everywhere else.

A significant advantage of the αAMG method is its invariance to diagonal scaling, as shown in section 4.2. Thus, for each of these problems, we consider the results of such scaling. A common scaling is to make the diagonal entries of A all have value 1, using the diagonal matrix given by $s_{ii} = \frac{1}{\sqrt{a_{ii}}}$. For Problems 1–4 above, the problems with matrices thus scaled are referred to as Problems 1u–4u (where the u refers to the unit diagonal of the matrix). We also consider a more drastic scaling given by $s_{ii} = 10^{5r_i}$, where again r_i is chosen from a uniform distribution on $[0, 1]$ for each i. We call these Problems 1r–4r (where the r refers to the random scaling). Note that this scaling is dependent on the mesh size, and so it is expected that the overall work (measured relative to the cost of a fine-scale matrix-vector product), including both setup and solution phases, will grow as the mesh size is reduced.

6.1. AMG benchmarks. A baseline for these problems is established by considering the performance of standard AMG under the same assumptions (primarily that coarsening is performed geometrically). Since we expect the scalings employed to destroy any sense of strong connection in the matrix coefficients, we consider here a "strong-connection-only" version of AMG, equivalent to setting $\theta = 0$ in the definitions of strong influence and dependence in section 2. Wall-clock times and iteration counts are shown in Table 6.1, while convergence factors are shown in Table 6.2.

These results show that classical AMG interpolation gives a scalable solver for Problems 1, 2, and 3, coming directly from discretization. If, however, the discretization matrices are scaled, then there is a significant increase in the work needed for solution, especially noticeable as the problem grows (but also present with smaller fine grids). Improving on the results from these unscaled problems is difficult, so our aim should be to determine a balance where significant improvement on the results from the scaled problems is found, while not excessively increasing the cost of solution for problems such as 1, 2, and 3. The results for the unscaled Problem 4 are typical of the situation where, as h decreases, the problem becomes more difficult to solve due to the increase in the number of internal material interfaces.

6.2. Calibrated AMG results. For the adaptive AMG methodology, we consider several questions around the same problems. Experience has shown that a single

TABLE 6.2
Standard AMG: asymptotic convergence factors (measured after at most 200 *iterations).*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.104	0.115	0.124	0.131	0.137
Problem 1u	0.104	0.115	0.124	0.131	0.137
Problem 1r	0.991	0.997	0.996	0.996	0.996
Problem 2	0.068	0.069	0.070	0.071	0.071
Problem 2u	0.594	0.754	0.864	0.929	0.964
Problem 2r	0.991	0.993	0.993	0.993	0.992
Problem 3	0.111	0.122	0.130	0.136	0.141
Problem 3u	0.488	0.656	0.793	0.886	0.940
Problem 3r	0.995	0.997	0.996	0.996	0.996
Problem 4	0.209	0.212	0.233	0.290	0.375
Problem 4u	0.760	0.914	0.976	0.994	0.998
Problem 4r	0.996	0.996	0.996	0.996	0.995

setup cycle is often most efficient if parameters ν_0 and ν_1 can be chosen such that this cycle yields an effective solver. How to best choose these parameters depends on our interests. For solving the matrix equation, $A\mathbf{x} = \mathbf{b}$, for a single vector, $\mathbf{b}$, we would like to choose ν_0 and ν_1 such that the total time to solution is smallest. This may mean sacrificing performance of the solver to save cost in the setup stage. For solving the matrix equation for many right-hand sides, the parameters should be chosen such that the solver performs optimally. Here, we primarily consider the latter situation, and demonstrate that doing so does not severely impact the time to solution for a single right-hand side.

We first present what might best be described as a calibrated AMG approach, reflecting that parameters ν_0 and ν_1 are chosen to calibrate the AMG performance, rather than a truly adaptive approach, where ν_0 and ν_1 remain fixed and setup is performed until an efficient solver is exposed. Thus, these results indicate the theoretical best performance of an adaptive AMG algorithm, with the minimal amount of total setup work, but do not represent a realistic adaptive algorithm, where adaptation is performed until a termination criterion is met. Such an algorithm is presented in the next section.

Our experiments were performed with the goal of (approximately) minimizing the asymptotic convergence factors of the resulting methods. To do this, we computed the asymptotic convergence factors for very large ν_0 and ν_1, corresponding to the optimal case where $\mathbf{x}^{(1)}$ is the slowest-to-converge mode of relaxation, then chose the smallest values for ν_0 and ν_1 such that the convergence factor of the resulting method was within 0.005 of the optimal factor when that factor was less than 0.1, and within 0.01 of the optimal factor otherwise. We found that, for Problems 1, 2, and 3, good asymptotic convergence factors could be achieved with relatively small values of ν_0 and ν_1. For Problem 4, the same performance as for standard AMG on the unscaled system can be recovered with small values of ν_0 and ν_1. Table 6.3 shows the time required for the setup phase for given values of these parameters and different grid sizes. As always happens when measuring computational performance, the timings are accurate only to within a few hundredths of a second, and so there is some variation in the timings of program stages that require the same number and ordering of operations.

Table 6.4 presents the asymptotic convergence factors for the methods resulting from the setup stages as outlined in Table 6.3. Note that, for the first three problems and for all grid sizes, αAMG achieves convergence factors bounded well below 1, with very small growth as the mesh size decreases. For Problem 4, we see growth like that

TABLE 6.3
*Calibrated AMG: wall-clock time in seconds (and values of ν_0, ν_1) for setup phase. The notation * indicates that two setup cycles were more efficient.*

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.02 (2,2)	0.09 (2,2)	0.38 (3,3)	1.67 (4,5)	7.21 (7,7)
Problem 1u	0.02 (2,2)	0.09 (2,2)	0.37 (3,3)	1.63 (4,5)	7.27 (7,7)
Problem 1r	0.02 (4,4)	0.11 (5,5)	0.47 (8,7)	2.13 (11,11)	9.52 (16,17)
Problem 2	0.02 (3,2)	0.10 (4,4)	0.42 (6,6)	1.88 (9,9)	8.57 (13,13)
Problem 2u	0.04 (3,2)	0.09 (4,4)	0.43 (7,6)	1.83 (10,9)	8.03 (14,13)
Problem 2r	0.03 (7,7)	0.11 (9,9)	0.51 (14,14)	2.50 (21,21)	11.84 (31,31)
Problem 3	0.02 (2,2)	0.11 (4,4)	0.45 (4,4)	1.69 (6,6)	6.86 (7,7)
Problem 3u	0.02 (2,2)	0.10 (4,4)	0.41 (4,4)	1.66 (6,6)	6.94 (7,7)
Problem 3r	0.03 (5,5)	0.10 (6,6)	0.45 (9,8)	1.90 (10,11)	8.50 (17,16)
Problem 4	0.02 (2,2)	0.06 (3,2)	0.35 (4,4)	1.60 (6,6)	6.80 (8,8)
Problem 4u	0.02 (2,2)	0.09 (3,2)	0.39 (5,5)	1.60 (6,6)	6.98 (9,9)
Problem 4r	0.02 (6,5)	0.12 (9,9)	0.50 (13,14)	2.37 (22,21)	17.09(22,20*)

TABLE 6.4
Calibrated AMG: asymptotic convergence factors.

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.067	0.073	0.079	0.080	0.079
Problem 1u	0.067	0.073	0.079	0.080	0.079
Problem 1r	0.069	0.078	0.077	0.078	0.079
Problem 2	0.069	0.069	0.071	0.071	0.073
Problem 2u	0.069	0.071	0.071	0.071	0.072
Problem 2r	0.072	0.071	0.071	0.072	0.073
Problem 3	0.070	0.097	0.081	0.110	0.103
Problem 3u	0.072	0.097	0.080	0.109	0.106
Problem 3r	0.070	0.100	0.084	0.111	0.108
Problem 4	0.194	0.202	0.243	0.288	0.376
Problem 4u	0.189	0.212	0.231	0.294	0.374
Problem 4r	0.187	0.212	0.235	0.292	0.383

TABLE 6.5
Calibrated AMG: total solution wall-clock time in seconds (and iteration count) to reduce residuals by 10^{10}.

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.04 (8)	0.25 (8)	0.89 (8)	3.52 (8)	14.70 (8)
Problem 1u	0.06 (8)	0.21 (8)	0.90 (8)	3.55 (8)	14.73 (8)
Problem 1r	0.03 (7)	0.22 (7)	0.91 (7)	3.64 (7)	15.64 (7)
Problem 2	0.05 (8)	0.22 (8)	0.92 (8)	3.83 (8)	15.85 (8)
Problem 2u	0.04 (8)	0.24 (8)	0.95 (8)	3.83 (8)	15.94 (8)
Problem 2r	0.05 (7)	0.24 (7)	0.97 (7)	4.23 (7)	18.60 (7)
Problem 3	0.05 (8)	0.20 (8)	0.93 (8)	3.63 (8)	14.58 (8)
Problem 3u	0.03 (8)	0.21 (8)	0.92 (8)	3.66 (8)	14.43 (8)
Problem 3r	0.04 (7)	0.24 (8)	0.94 (8)	3.81 (8)	16.32 (8)
Problem 4	0.03 (11)	0.31 (11)	1.09 (11)	4.84 (13)	22.06 (16)
Problem 4u	0.04 (11)	0.32 (11)	1.10 (11)	4.79 (13)	20.40 (14)
Problem 4r	0.05 (10)	0.27 (10)	1.22 (11)	5.35 (12)	28.27 (12)

in the standard AMG results. Note also that scaling the matrices has no effect on our ability to determine an efficient solver for these problems.

Finally, in Table 6.5, we consider the total cost of solving $A\mathbf{x} = \mathbf{0}$ a single time, with random initial guess, using the near-optimal solver. Calibrated AMG did not (and could not be expected to) beat the overall performance of standard AMG on

the four unscaled problems. However, the setup costs of calibrated AMG were not significantly higher than those of AMG. On a 1024×1024 mesh, AMG setup required approximately 5 seconds of CPU, and so the adaptive setup needed between 30% and 240% more time, but tended to produce a better solver than classical AMG. For these reasons, the overall cost of calibrated AMG is close to that of standard AMG in the cases where standard AMG works well.

When standard AMG fails, there is no contest. Calibrated AMG was able to solve those problems that caused difficulty for standard AMG in a small fraction of the time. For the first three randomly scaled problems, the calibrated AMG setup time requires at most 12 seconds on the 1024×1024 grid, equivalent in cost to approximately 15 AMG V-cycles. However, the resulting convergence factors of at most 0.108 are much lower than the AMG convergence factors, the smallest of which was 0.992. Thus, after setup and a single cycle of the calibrated AMG approach, the error will be smaller than if the equivalent amount of work, 16 classical AMG cycles, was performed, which would reduce the error by a factor of approximately 0.879. Even for the fourth randomly scaled problem, where the setup cost is equivalent to approximately 20 AMG V-cycles, the error reduction after a single cycle will be significantly better than the equivalent in cost of 21 classical AMG cycles, which would yield a total reduction of approximately 0.900.

6.3. Adaptive AMG (αAMG) results. While the calibrated AMG approach yields solvers with optimal performance characteristics, its results are achieved at a significant cost of user time in tuning the setup parameters of ν_0 and ν_1. Details of this tuning process are documented in [20]. In practice, an adaptive approach may be used instead, moving the burden of calibration from the user to the solver itself.

In the αAMG approach, a small fixed number of relaxation sweeps are initially performed on the finest grid to locally expose algebraically smooth error. This error is used to create an interpolation operator and (through the Galerkin conditions) a coarse-grid problem. Relaxation is then performed on this grid, and the problem is further coarsened. After the initial multigrid hierarchy is created in this fashion, another identical setup cycle is performed with the current prototype as an initial guess. This is done to further improve the prototype as a representative of algebraically smooth error.

Alternately, we could look for a representative of the error missed by the multigrid cycle that results from the first setup cycle and use a prototype of this error to further improve interpolation and generate a more robust solver. Here, however, we consider problems similar to those for which classical AMG is effective. In particular, we consider those problems where the near null space can be represented by a single vector. The question of how to improve an existing AMG interpolation operator with information from a second prototype vector is not easily addressed. Instead, here we redefine the interpolation operator based on improving the prototype vector and not multiple distinct algebraically smooth components.

The results in Tables 6.6, 6.7, and 6.8 reflect performing 8 iterations of Gauss–Seidel relaxation on the homogeneous problem at each level of the setup phase. After each setup cycle, the resulting solver is tested by running 8 iterations of the solver on the homogeneous problem with a random initial guess. The fine-grid multigrid cycle is accepted when the error reduction factor (measured in the A-norm) of the last of these iterations is less than 0.4. Performing fewer iterations of relaxation on each level in the setup phase results in less exposure of slow-to-converge error and typically increases

TABLE 6.6
Adaptive AMG: wall-clock time in seconds (and number of setup cycles) for setup phase.

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.03 (1)	0.19 (1)	0.79 (1)	3.31 (1)	13.33 (1)
Problem 1u	0.04 (1)	0.20 (1)	0.82 (1)	3.25 (1)	13.41 (1)
Problem 1r	0.04 (1)	0.19 (1)	0.83 (1)	6.30 (2)	25.70 (2)
Problem 2	0.04 (1)	0.20 (1)	0.89 (1)	3.35 (1)	13.59 (1)
Problem 2u	0.04 (1)	0.20 (1)	0.83 (1)	3.37 (1)	13.48 (1)
Problem 2r	0.04 (1)	0.20 (1)	0.84 (1)	6.44 (2)	51.62 (4)
Problem 3	0.04 (1)	0.20 (1)	0.82 (1)	3.36 (1)	13.20 (1)
Problem 3u	0.04 (1)	0.19 (1)	0.82 (1)	3.33 (1)	13.26 (1)
Problem 3r	0.04 (1)	0.21 (1)	0.82 (1)	9.53 (3)	38.57 (3)
Problem 4	0.04 (1)	0.20 (1)	0.84 (1)	3.25 (1)	25.81 (2)
Problem 4u	0.04 (1)	0.19 (1)	0.82 (1)	6.36 (2)	25.67 (2)
Problem 4r	0.04 (1)	0.39 (2)	1.56 (2)	31.06 (10)	229.11 (18)

TABLE 6.7
Adaptive AMG: asymptotic convergence factors.

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.065	0.069	0.070	0.086	0.201
Problem 1u	0.065	0.069	0.070	0.086	0.201
Problem 1r	0.068	0.085	0.210	0.071	0.071
Problem 2	0.067	0.069	0.089	0.156	0.335
Problem 2u	0.068	0.069	0.091	0.159	0.338
Problem 2r	0.099	0.160	0.355	0.075	0.338
Problem 3	0.067	0.097	0.080	0.118	0.294
Problem 3u	0.068	0.097	0.080	0.121	0.298
Problem 3r	0.075	0.113	0.293	0.110	0.867
Problem 4	0.186	0.195	0.243	0.395	0.384
Problem 4u	0.185	0.195	0.231	0.282	0.382
Problem 4r	0.227	0.202	0.235	0.282	0.385

TABLE 6.8
Adaptive AMG: total solution wall-clock time in seconds (and iteration count) to reduce residuals by 10^{10}.

	64×64	128×128	256×256	512×512	1024×1024
Problem 1	0.06 (8)	0.32 (8)	1.29 (8)	5.15 (8)	23.48 (11)
Problem 1u	0.06 (8)	0.32 (8)	1.29 (8)	5.13 (8)	23.58 (11)
Problem 1r	0.06 (7)	0.30 (7)	1.41 (10)	8.14 (7)	32.38 (7)
Problem 2	0.06 (8)	0.35 (8)	1.37 (8)	5.84 (9)	26.54 (13)
Problem 2u	0.06 (8)	0.34 (8)	1.34 (8)	5.67 (9)	24.58 (11)
Problem 2r	0.07 (8)	0.33 (8)	1.64 (13)	8.33 (7)	64.48 (12)
Problem 3	0.06 (8)	0.33 (8)	1.31 (8)	5.30 (8)	25.92 (13)
Problem 3u	0.06 (8)	0.34 (8)	1.33 (8)	5.31 (8)	25.83 (13)
Problem 3r	0.06 (7)	0.33 (8)	1.61 (13)	11.37 (7)	46.88 (8)
Problem 4	0.07 (11)	0.38 (11)	1.55 (11)	7.69 (18)	38.63 (13)
Problem 4u	0.07 (11)	0.37 (10)	1.53 (11)	9.22 (11)	38.67 (13)
Problem 4r	0.07 (10)	0.55 (10)	2.20 (10)	34.72 (11)	241.56 (13)

the number of setup cycles needed. Considering the high cost in recomputation of the interpolation and coarse-grid operators, we prefer to avoid this. Performing more iterations of relaxation on the homogeneous problem on each level better exposes the error sought, but at the cost of potentially significant unnecessary computation. Another additional cost in the adaptive setup phase is that of the "test drive" that

performs 8 iterations of the current V-cycle on the homogeneous problem (to ensure adequate performance) after each setup cycle. Here, fewer iterations may give a poor indication of asymptotic cycle performance, while the cost of more iterations to better determine the true cycle performance is wasted if the cycle is to be further improved by another setup phase. Choosing the threshold for accepting the solver is less ambiguous, as setting a high threshold reduces the setup time but admits poor solvers, whereas setting the threshold to be small improves the solver but potentially requires more setup cycles.

Table 6.6 demonstrates the typical higher computational cost of the adaptive approach as compared to the calibrated approach. Some of this cost is certainly unavoidable. Fixing the number of iterations performed on the homogeneous problem undoubtedly results in some extraneous relaxations being performed, particularly on coarser grids, where few relaxation sweeps are "necessary." Likewise, the cost of testing the solver can be avoided only if we know, a priori, whether the solver will be acceptable. While these costs are significantly higher than those for calibrated AMG (as in Table 6.3), they do remain relatively low, still under one minute of CPU time for a 1024×1024 grid in all but the most difficult case.

Convergence factors for the adaptive AMG approach (Table 6.7) demonstrate its robustness. While the requirement for a successful setup was a single-step convergence factor less than 0.4 when measured after 8 iterations, we see that the asymptotic convergence factors are generally bounded by 0.4 as well. The asymptotic performance of the cycle for Problem 3r is relatively poor, but, in our tests, the first 15 steps of the resulting method reduced the residuals by factors better than 0.3 at each step. This example, however, does indicate that a more reliable performance indicator would be of use. Note that some of the factors are worse than the corresponding convergence factors for the calibrated AMG approach; they could also be improved at the cost of more expensive setup cycles. A constant convergence factor of 0.4 is still sufficient, however, to reduce the error by a factor of 10^{10} in 25 iterations.

The overall time to solution of the adaptive approach, as in Table 6.8, reflects the expected behavior. Iteration counts are low—at most 18 iterations are required for residual reduction by a factor of 10^{10}—and so the time to solution, ignoring the setup phase, is quite low. The increased setup costs due to multiple adaptations are reflected in the total time to solution. Comparing these results to the calibrated AMG approach is somewhat disappointing; however, in all cases, the solution was still obtained quickly, with total time for the αAMG approach within a factor of at most 8.5 (and often only 2) of the total time for the calibrated AMG results. Comparing this with the standard AMG results shows that the overall time to solution for the unscaled problems is higher than that for standard AMG, whereas, for the scaled problems, adaptive AMG always yields a quickly converging solver and so yields lower overall solution times in all cases where standard AMG performance suffered.

The calibrated and adaptive AMG results are very encouraging. For the scaled problems, there is a tremendous improvement in the amount of effort required for solution of the linear systems when compared to the results for standard AMG interpolation in Table 6.2. The solution phase of the algorithm scales well across all five grid sizes, and the actual costs are quite reasonable. It must be acknowledged, however, that once we account for the cost of the setup phase of our algorithm, classical AMG is still a slightly more efficient solver for the unscaled matrices when we consider solving with only a single right-hand side. Put simply, if the algebraically smoothest component of an elliptic PDE is known exactly, αAMG can do no better than design-

ing multigrid interpolation based on that component. Indeed, if this component is given as input to the adaptive AMG method for creating the multigrid hierarchy, we can solve the problem with the same cost as classical AMG on the unscaled problem, simply by using this prototype as $\mathbf{x}^{(1)}$ in (4.5), as discussed in section 4.2.

7. Conclusions. The adaptive multigrid strategies outlined here provide an opportunity for the recovery of classical multigrid performance in cases where the near null space components of the matrix are not locally constant. This can be done in both the case of a known nonconstant near null space component or of an unknown near null space component. The interpolation presented is a generalization of the classical Ruge–Stüben scheme, and, when coupled with the multilevel prototype of the near null space, it provides a scalable algorithm for matrix systems that the classical interpolation does not.

REFERENCES

[1] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, *Parallel multigrid smoothing: Polynomial versus Gauss–Seidel*, J. Comput. Phys., 188 (2003), pp. 593–610.

[2] R. E. Alcouffe, A. Brandt, J. E. Dendy, Jr., and J. W. Painter, *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comput., 2 (1981), pp. 430–454.

[3] P. B. Bochev, C. J. Garasi, J. J. Hu, A. C. Robinson, and R. S. Tuminaro, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[4] J. H. Bramble, *Multigrid Methods*, Pitman Res. Notes Math. Sci. 294, Longman Scientific & Technical, Essex, UK, 1993.

[5] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[6] A. Brandt, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[7] A. Brandt, *General highly accurate algebraic coarsening*, Elect. Trans. Numer. Anal., 10 (2000), pp. 1–20.

[8] A. Brandt, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. J. Evans, ed., Cambridge University Press, Cambridge, UK, 1984, pp. 257–284.

[9] A. Brandt and D. Ron, *Multigrid solvers and multilevel optimization strategies*, in Multilevel Optimization in VLSICAD, J. Cong and J. R. Shinnerl, eds., Comb. Optim. 14, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2003, pp. 1–69.

[10] M. Brezina, C. I. Heberton, J. Mandel, and P. Vaněk, *An Iterative Method with Convergence Rate Chosen a Priori*, UCD/CCM Report 140, Center for Computational Mathematics, University of Colorado at Denver, Denver, CO, 1999; available online from http://www.math.cudenver. edu/ccmreports/rep140.ps.gz.

[11] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[12] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA)*, SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[13] M. Brezina, C. Tong, and R. Becker, *Parallel algebraic multigrids for structural mechanics*, SIAM J. Sci. Comput., to appear.

[14] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., (2003), pp. 1–26.

[15] R. D. Falgout, *A Note on the Relationship Between Adaptive AMG and PCG*, technical report UCRL-TR-205838, Lawrence Livermore National Laboratory, Livermore, CA, 2004.

[16] R. D. Falgout and P. S. Vassilevski, *On generalizing the algebraic multigrid framework*, SIAM J. Numer. Anal., 42 (2004), pp. 1669–1693.

[17] J. Fish and V. Belsky, *Generalized aggregation multilevel solver*, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 4341–4361.

[18] L. A. Hageman and R. B. Kellogg, *Estimating optimum overrelaxation parameters*, Math.

Comp., 22 (1968), pp. 60–68.
[19] O. Livne, *Coarsening by compatible relaxation*, Numer. Linear Algebra Appl., 11 (2004), pp. 205–227.
[20] S. MacLachlan, *Improving Robustness in Multiscale Methods*, Ph.D. thesis, University of Colorado at Boulder, Boulder, CO, 2004.
[21] T. A. Manteuffel, *Adaptive procedure for estimating parameters for the nonsymmetric Tchebychev iteration*, Numer. Math., 31 (1978), pp. 183–208.
[22] S. F. McCormick and J. W. Ruge, *Multigrid methods for variational problems*, SIAM J. Numer. Anal., 19 (1982), pp. 924–929.
[23] S. F. McCormick and J. W. Ruge, *Algebraic multigrid methods applied to problems in computational structural mechanics*, in State-of-the-Art Surveys on Computational Mechanics, ASME, New York, 1989, pp. 237–270.
[24] J. W. Ruge, *Algebraic multigrid (AMG) for geodetic survey problems*, in Preliminary Proceedings of the International Multigrid Conference, Fort Collins, CO, 1983, Institute for Computational Studies at Colorado State University.
[25] J. W. Ruge, *Final Report on amg*02, technical report, Gesellschaft für Mathematik und Darenverarbeitung, St. Augustin, 1985.
[26] J. Ruge, *AMG for problems of elasticity*, Appl. Math. Comp., 19 (1986), pp. 293–309.
[27] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, S. F. McCormick, ed., Frontiers in Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[28] K. Stüben, *An introduction to algebraic multigrid*, in Multigrid, U. Trottenberg, C. Oosterlee, and A. Schüller, eds., Academic Press, San Diego, CA, 2001, pp. 413–528.
[29] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[30] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.
[31] C. Wagner and G. Wittum, *Filtering decompositions with respect to adaptive test vectors*, in Multigrid Methods V, Proceedings of the Fifth European Multigrid Conference, Stuttgart, Germany, 1996, W. Hackbusch and G. Wittum, eds., Lecture Notes in Comput. Sci. Engrg. 3, Springer, Berlin, 1998, pp. 320–334.
[32] C. Wagner and G. Wittum, *Adaptive filtering*, Numer. Math., 78 (1997), pp. 305–328.
[33] U. M. Yang, *On the use of relaxation parameters in hybrid smoothers*, Numer. Linear Algebra Appl., 11 (2004), pp. 155–172.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1287–1304
© 2006 Society for Industrial and Applied Mathematics

A TWO-GRID DISCRETIZATION SCHEME FOR SEMILINEAR ELLIPTIC EIGENVALUE PROBLEMS*

C.-S. CHIEN† AND B.-W. JENG†

Dedicated to Professor E. L. Allgower on the occasion of his seventieth birthday

Abstract. We present a two-grid finite element discretization scheme with a two-loop continuation algorithm for tracing solution branches of semilinear elliptic eigenvalue problems. First we use the predictor-corrector continuation method to compute an approximating point for the solution curve on the coarse grid. Then we use this approximating point as a predicted point for the solution curve on the fine grid. In the corrector step we solve the first and the second order approximations of the nonlinear PDE to obtain corrections for the state variable on the fine grid and the coarse grid, respectively. The continuation parameter is updated by computing the Rayleigh quotient on the fine space. To guarantee the approximating point we just obtained lies on the solution curve, we perform Newton's method. We repeat the process described above until the solution curve on the fine space is obtained. We show how the singular points, such as folds and bifurcation points, can be well approximated. Comprehensive numerical experiments show that the two-grid finite element discretization scheme with a two-loop continuation algorithm is efficient and robust for solving second order semilinear elliptic eigenvalue problems.

Key words. two-grid scheme, linear eigenvalue problems, nonlinear eigenvalue problems, continuation method, finite element method, singular points

AMS subject classifications. 65N55, 65N25, 65P30, 65N30, 65N06

DOI. 10.1137/030602447

1. Introduction.

During the 1980s, some researchers studied multigrid methods in the context of continuation methods for second order semilinear elliptic eigenvalue problems. See, e.g., Bank and Chan [3], Bolstad and Keller [4], Chan and Keller [5], Mittelmann and Weber [22], Ta'asan [26], and Weber [28, 29]. Other numerical studies of nonlinear elliptic eigenvalue problems can be found, e.g., in [12, 15, 20, 21].

Almost simultaneously, the mesh independence principle (MIP) for Newton's method was investigated by various researchers. See, e.g., Allgower and Böhmer [1] and the further references cited therein. The MIP means that the number of Newton iterations required in solving the discretized nonlinear differential equation is asymptotically independent of the asymptotic parameters such as the mesh size. In [30, 31] Xu studied two-grid discretization methods for semilinear elliptic boundary value problems based on finite element approximations. It is shown in [30] that one first uses a coarse space to produce a rough approximate solution. A fine grid correction is made on this approximate solution by solving a residual equation of the PDE. The desired solution on the fine space is obtained by performing a further coarse grid correction, which can improve the accuracy of the approximate solution.

Later, Xu and Zhou [32] proposed a two-grid discretization scheme for solving linear eigenvalue problems, including both PDEs and integral equations. If this procedure is implemented in the context of a finite element method, one needs to solve a generalized linear eigenvalue problem to obtain an eigenpair on a coarse space. Next,

*Received by the editors December 15, 2003; accepted for publication (in revised form) June 1, 2005; published electronically January 6, 2006.
http://www.siam.org/journals/sisc/27-4/60244.html
†Department of Applied Mathematics, National Chung-Hsing University, Taichung 402, Taiwan (cschien@amath.nchu.edu.tw, d9053401@mail.nchu.edu.tw). The work of the first author was supported by the National Science Council of R.O.C. (Taiwan) through project NSC 93-2115-M-005-005.

one computes the desired eigenvector on the fine space. The corresponding eigenvalue is obtained by computing the Rayleigh quotient on the fine space. The eigenpairs can be computed up to the dimension of the coarse space.

In this paper, we will combine the ideas of these two schemes described in [30, 32] and develop a two-grid discretization scheme with a two-loop continuation algorithm for solving semilinear elliptic eigenvalue problems of the following form:

$$\begin{aligned} -\Delta u + \lambda f(u) &= 0 \qquad \text{in } \Omega = (0,1)^2, \\ u &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{1.1}$$

where $f(u)$ is a smooth function of u. For bifurcation problems further assumptions are necessary, e.g., f must satisfy $f(0) = 0$, $f'(0) \cdot f'''(0) \neq 0$. For simplicity we rewrite (1.1) as an operator form

$$F(u, \lambda) = 0, \tag{1.2}$$

where $F : B_1 \times \mathbb{R} \to B_2$ with $u \in B_1$, $\lambda \in \mathbb{R}$ is a smooth mapping, B_1 and B_2 are two real Banach spaces, and $\mathbb{R}$ is the real line. We call u the state variable and λ the parameter variable. To solve (1.2) numerically by the two-grid discretization scheme with a two-loop continuation method, we first discretize it, e.g., by a finite element method or a finite difference method on a coarse space. The predictor-corrector continuation method [2, 17] is then used to compute an approximating point on the coarse space.

Let $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ be a point which has been accepted as an approximating point for the solution curve $\widetilde{c}$ on the coarse grid. We use this approximating point as a predicted point for the solution curve c on the fine grid. We solve the first order approximation of (1.2) on the fine grid to obtain a correction for the state variable $u_{\tilde{h}}^{(i)}$. To improve the accuracy of the state variable, we make a further correction on the coarse space by solving the second order approximation of (1.2), and thus obtain the state variable $u_h^{(i)}$ on the fine grid. In order to obtain an accurate parameter $\lambda_h^{(i)}$ corresponding to $u_h^{(i)}$, we compute the Rayleigh quotient as in [32]. To guarantee the approximating point is accurate enough on the fine grid, we use $(u_h^{(i)}, \lambda_h^{(i)})$ as an initial guess and perform Newton's method. This process constitutes the corrector step of the continuation algorithm on the fine space. Then we go back to the coarse grid to find the next approximating point $(u_{\tilde{h}}^{(i+1)}, \lambda_{\tilde{h}}^{(i+1)})$ for $\widetilde{c}$ by using the predictor-corrector continuation method. We repeat the process described above until the solution curve c on the fine space is traced.

Two main issues must be addressed here. First, are the solution branches obtained by implementing the two-grid discretization scheme accurate enough on the fine space? Next, are the singular points on the solution branches well approximated? The accuracy of the approximating solution curve is guaranteed by implementing the error estimators developed by Liu and Rheinboldt [19]. An alternative is to use the solution curve obtained by implementing, e.g., the multigrid-continuation algorithm [9] as the known data, and construct an interpolation polynomial for the approximating points obtained on the fine grid. The error bounds can be easily obtained. However, this process has to be implemented pointwise and is tedious. For simplicity we use sample checking on some interior nodes of the domain at each continuation step. In order to get better approximations of singular points in curve tracking, we perform Newton's method near the singular point after the two-grid solution is obtained. The effectiveness of this viewpoint has been verified by our numerical experiments.

This paper is organized as follows. In section 2 we briefly review the predictor-corrector continuation method. Section 3 describes a two-grid finite element discretization scheme for computing the first few eigenpairs of a second order linear elliptic eigenvalue problem. We indicate that the main work of this algorithm lies in solving symmetric positive definite linear systems with multiple right-hand sides. We propose to use the single seed method [25] or the preconditioned Lanczos method with iterative refinement [8] to solve the linear systems. In section 4 we derive the two-grid discretization schemes with two-loop continuation algorithms for semilinear elliptic eigenvalue problems. The two algorithms proposed in section 4 were implemented to trace solution curves of some test problems. These problems have been widely used as numerical examples in the literature of continuation methods [[3, 4, 5, 6, 7, 9, 14, 15, 19, 20, 21, 22, 26, 28, 29]. Our numerical results in section 5 show that the two-grid finite element discretization scheme is quite robust for second order semilinear elliptic eigenvalue problems. Finally, some concluding remarks are given in section 6.

2. A brief review of the predictor-corrector continuation method. To start with, we briefly review the predictor continuation method for tracing solution curves of semilinear elliptic eigenvalue problems. We refer to [2, 17] and the further references cited therein for a comprehensive study of numerical continuation methods.

We consider parameter-dependent problems of the form

$$F(x, \lambda) = 0, \tag{2.1}$$

where $F : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ is a continuously differentiable function of $x \in \mathbb{R}^N$ and $\lambda \in \mathbb{R}$. We denote the Jacobian of F by $DF = [D_x F, D_\lambda F]$ and a solution curve of (2.1) by

$$c = \{y(s) = (x(s), \lambda(s)) \mid F(y(s)) = 0, s \in I\}.$$

Here I is any interval in $\mathbb{R}$. Assume that a parameterization via arc length is available on c. We will trace the solution curve c by the predictor-corrector continuation methods. If the assumption that 0 is a regular value of F is not satisfied, then $F^{-1}(0)$ may contain bifurcation points and the augmented matrix of DF becomes singular there. On the other hand, if the solution curve c contains a simple fold, then the matrix $D_x F$ becomes singular there. In either case we have to solve nearly singular systems in numerical curve tracking if we are close to a fold or a bifurcation point. Differentiating (2.1) with respect to s, we obtain

$$DF(y(s)) \cdot \dot{y}(s) = 0, \tag{2.2}$$

where $\dot{y}(s) = (\dot{x}(s), \dot{\lambda}(s))^T$ denotes a tangent vector to c at $y(s)$ and $DF(y(s)) = (D_x F(y(s)), D_\lambda F(y(s)))$ is the $N \times (N+1)$ Jacobian matrix.

Let $y_i = (x_i, \lambda_i) \in \mathbb{R}^{N+1}$ be a point which has been accepted as an approximating point for c. A new point $z_{i+1,1}$ is predicted by the Euler predictor, i.e.,

$$z_{i+1,1} = y_i + \delta_i v_i. \tag{2.3}$$

Here $\delta_i > 0$ is the step length and v_i is the unit tangent vector at y_i, which is obtained by solving the linear system

$$A(y_i) \cdot v_i = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{2.4}$$

where $A(y(s)) = \begin{bmatrix} DF(y(s)) \\ v_{i-1}^T \end{bmatrix}$ is the augmented Jacobian matrix and v_{i-1} is the unit tangent vector of the previous continuation step. To maintain orientation and control of the local curvature, we also require that

$$v_i^T \cdot v_{i-1} > 1 - \alpha > 0 \quad \text{for some } \alpha \in (0,1). \tag{2.5}$$

The accuracy of approximation to the solution curve must generally be improved via a corrector process. This may be done by choosing a hyperplane, which is orthogonal to $\dot{y}(s)$ at $z_{i+1,1}$ and performing Newton's iteration constrained to the hyperplane. The continuation algorithm described above will be used to trace solution curves on the coarse grid in section 4.

3. Two-grid discretization schemes for linear eigenvalue problems. A two-grid discretization scheme is proposed in [32] to approximate the first few eigenpairs of the following linear eigenvalue problem:

$$\begin{aligned} -\Delta u &= \lambda u \qquad \text{in } \Omega = (0,1)^2, \\ u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.1}$$

Let $H^k(\Omega)$ be the Sobolev space endowed with Sobolev norm and $H_0^k(\Omega) = \{v \in H^k(\Omega) \mid v|_{\partial\Omega} = 0\}$. For $0 < h < \widetilde{h} < 1$, let $V_{\tilde{h}}(\Omega) \subset V_h(\Omega) \subset H_0^1(\Omega)$ be two finite element subspaces. The finite element approximation of (3.1) on the coarse space $V_{\tilde{h}}(\Omega)$ is to find $\lambda_{\tilde{h}} \in \mathbb{R}^1$, $u_{\tilde{h}} \in V_{\tilde{h}}(\Omega)$ with $\|\nabla u_{\tilde{h}}\|_{L^2} = 1$ such that

$$\int_\Omega \nabla u_{\tilde{h}} \cdot \nabla v = \lambda_{\tilde{h}} \int_\Omega u_{\tilde{h}} v \quad \text{holds } \forall v \in V_{\tilde{h}}(\Omega). \tag{3.2}$$

For simplicity, we denote $V_{\tilde{h}}(\Omega)$ and $V_h(\Omega)$ by $V_{\tilde{h}}$ and V_h, respectively. Note that the discrete version of (3.2) is a generalized symmetric eigenvalue problem

$$A_{\tilde{h}}\{u_{\tilde{h}}\} = \lambda_{\tilde{h}} B_{\tilde{h}}\{u_{\tilde{h}}\}, \tag{3.3}$$

where $A_{\tilde{h}}$ and $B_{\tilde{h}}$ are the coefficient matrices associated with $\int_\Omega \nabla u_{\tilde{h}} \cdot \nabla v$ and $\int_\Omega u_{\tilde{h}} v$, respectively, and $\{u_{\tilde{h}}\}$ is a vector whose entries are the values of $u_{\tilde{h}}$ at each coarse grid point. For simplicity in what follows we will omit the symbol "$\{\cdot\}$" in (3.4) and use $u_{\tilde{h}}$ to represent a vector in the Euclidean space as well. The Rayleigh quotient iteration (RQI) is widely used to solve generalized symmetric eigenvalue problems, where the preconditioned Lanczos method or the MINRES [23] can be used as a linear solver. See, e.g., [13] and the further references cited therein.

The accuracy of the approximate eigenvectors we obtained on the coarse grid can be improved by solving symmetric positive definite linear systems on the fine space. Suppose that we wish to compute the first k eigenpairs of (3.1). The main work of the two-grid scheme will involve solving linear systems with multiple right-hand sides of the form

$$AX = B \tag{3.4}$$

on the fine grid, where $A \in \mathbb{R}^{N\times N}$ is symmetric positive definite, $X = [x^{(1)}, \ldots, x^{(k)}] \in \mathbb{R}^{N\times k}$ is the matrix of unknowns, and $B = [b^{(1)}, \ldots, b^{(k)}] \in \mathbb{R}^{N\times k}$ is the matrix of right-hand sides. Smith, Peterson, and Mittra [25] developed a specific conjugate gradient (CG) algorithm, called the single seed method, for solving (3.4). Recently Chien and Chang [8] proposed to use the preconditioned Lanczos method with iterative refinement for solving (3.4). The numerical results reported therein show that

the two algorithms mentioned above are quite competitive for solving (3.4). We also let A_h and B_h be the coefficient matrices corresponding to $\int_\Omega \nabla u_h \cdot \nabla v$ and $\int_\Omega u_h v$ on the fine space, respectively. In order to improve the accuracy of the eigenpairs we obtained on the fine space V_h, we propose to perform the RQI again. The two-grid finite element scheme of Xu and Zhou [32] can be modified as follows.

ALGORITHM 3.1. A two-grid finite element discretization scheme for (3.1).

1. Use the RQI to find the first k eigenvalues $\lambda_{\tilde{h}}^1, \ldots, \lambda_{\tilde{h}}^k$ and the corresponding eigenfunctions $u_{\tilde{h}}^1, \ldots, u_{\tilde{h}}^k \in V_{\tilde{h}}$ with $\|\nabla u_{\tilde{h}}^1\|_{L^2} = \cdots = \|\nabla u_{\tilde{h}}^k\|_{L^2} = 1$ on the coarse space $V_{\tilde{h}}$: Solve
$$\int_\Omega \nabla u_{\tilde{h}}^i \cdot \nabla v = \lambda_{\tilde{h}}^i \int_\Omega \nabla u_{\tilde{h}}^i \nabla v \quad \forall v \in V_{\tilde{h}}, \quad i = 1, \ldots, k.$$
2. Use the single seed method or the preconditioned Lanczos method with iterative refinement to solve the s.p.d. linear systems with multiple right-hand sides on the fine space V_h: Find $u_h^i \in V_h$ such that
$$\int_\Omega \nabla u_h^i \cdot \nabla v = \lambda_{\tilde{h}}^i \int_\Omega u_{\tilde{h}}^i v \quad \forall v \in V_h, \quad i = 1, 2, \ldots, k.$$
3. Compute the Rayleigh quotient
$$\lambda_h^i = \frac{\|\nabla u_h^i\|_{L^2}^2}{\|u_h^i\|_{L^2}^2}, \quad i = 1, 2, \ldots, k.$$
4. Perform the RQI:
   (i) Normalize the approximate eigenvectors obtained in Step 2,
   $$u_0^i = \frac{u_h^i}{\|\nabla u_h^i\|_{L^2}}, \quad \text{and set} \quad \mu_0^i = \lambda_h^i, \quad i = 1, 2, \ldots, k.$$
   (ii) For $i = 1 : k$
   For $j = 0, 1, 2, \ldots$
   Solve $(A_h - \mu_j^i B_h) z_{j+1}^i = B_h u_j^i$ for z_{j+1}^i
   $$u_{j+1}^i = \frac{z_{j+1}^i}{\sqrt{(z_{j+1}^i)^T A_h z_{j+1}^i}}$$
   $$\mu_{j+1}^i = \frac{(u_{j+1}^i)^T A_h u_{j+1}^i}{(u_{j+1}^i)^T B_h u_{j+1}^i}$$
   End
   End

Let u_h be the approximate eigenvector obtained on the fine grid as in [32]. The following results hold:

$$\|\nabla(u - u_h)\| = O(h + \widetilde{h}^2) \quad \text{and} \quad |\lambda_h - \lambda| = O(h^2 + \widetilde{h}^4). \tag{3.5}$$

These estimates show that we can choose $\widetilde{h} = O(\sqrt{h})$. We refer to [32] for details.

It is straightforward to describe a two-grid centered difference discretization scheme corresponding to Algorithm 3.1. The only difference is that we use the block Lanczos method to compute the first k eigenpairs of (3.1) on the coarse grid. Recently, the authors have developed some two-grid centered difference discretization schemes for solving the Schrödinger eigenvalue problem [10] and the Schrödinger–Poisson system [11, 16, 24]. The details will be given elsewhere.

4. Two-grid continuation algorithms for nonlinear eigenvalue problems. In this section we will present some two-grid discretization schemes with two-loop continuation algorithms for tracing solution branches of (1.2). We consider the finite element approximation of (1.2) on the coarse space by finding an approximating point $(u_{\tilde{h}}, \lambda_{\tilde{h}}) \in V_{\tilde{h}} \times \mathbb{R}$ for the solution curve $\widetilde{c}$ such that

$$\langle F(u_{\tilde{h}}, \lambda_{\tilde{h}}), v\rangle = \langle \nabla u_{\tilde{h}}, \nabla v\rangle + \lambda_{\tilde{h}} \langle f(u_{\tilde{h}}), v\rangle = 0 \qquad \forall v \in V_{\tilde{h}}. \tag{4.1}$$

Let $(u, \lambda_{\tilde{h}})$ be an exact solution of (1.2). We fix the parameter $\lambda_{\tilde{h}}$ here in order to make a correction for the state variable $u_{\tilde{h}}$ on the fine space as in [30]. The linear approximation of the mapping $F(u, \lambda_{\tilde{h}})$ at $u_{\tilde{h}}$ is

$$F(u, \lambda_{\tilde{h}}) \approx F(u_{\tilde{h}}, \lambda_{\tilde{h}}) + D_u F(u_{\tilde{h}}, \lambda_{\tilde{h}})(u - u_{\tilde{h}}). \tag{4.2}$$

Setting $e = u - u_{\tilde{h}}$, (4.2) is equivalent to

$$D_u F(u_{\tilde{h}}, \lambda_{\tilde{h}})e \approx -F(u_{\tilde{h}}, \lambda_{\tilde{h}}),$$

where the Fréchet derivative $D_u F(u_{\tilde{h}}, \lambda_{\tilde{h}}) = -\Delta + \lambda_{\tilde{h}} f'(u_{\tilde{h}})$. The approximate solution $u_{\tilde{h}}$ can be corrected by solving the following equation:

$$\begin{aligned} -\Delta e + \lambda_{\tilde{h}} f'(u_{\tilde{h}})e &= \Delta u_{\tilde{h}} - \lambda_{\tilde{h}} f(u_{\tilde{h}}) && \text{in } \Omega = (0,1)^2, \\ e &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{4.3}$$

Since $-\langle \nabla u_{\tilde{h}}, \nabla v\rangle - \lambda_{\tilde{h}} \langle f(u_{\tilde{h}}), v\rangle = 0$ for all $v \in V_{\tilde{h}}$, a finite element discretization of (4.3) on the coarse space $V_{\tilde{h}}$ produces only the trivial solution, and (4.3) must be solved on the fine space V_h. The finite element approximation of (4.3) is to find $e_h \in V_h$ such that

$$\langle \nabla e_h, \nabla v\rangle + \lambda_{\tilde{h}} \langle f'(u_{\tilde{h}})e_h, v\rangle = -\langle \nabla u_{\tilde{h}}, \nabla v\rangle - \lambda_{\tilde{h}} \langle f(u_{\tilde{h}}), v\rangle \qquad \forall v \in V_h.$$

For the further correction on the coarse space $V_{\tilde{h}}$, we use the quadratic approximation of $F(u, \lambda_{\tilde{h}})$ at $u_{\tilde{h}}$, i.e.,

$$F(u, \lambda_{\tilde{h}}) \approx F(u_{\tilde{h}}, \lambda_{\tilde{h}}) + D_u F(u_{\tilde{h}}, \lambda_{\tilde{h}})(u - u_{\tilde{h}}) + \frac{1}{2} D_{uu} F(u_{\tilde{h}}, \lambda_{\tilde{h}})(u - u_{\tilde{h}})^2.$$

Then

$$D_u F(u_{\tilde{h}}, \lambda_{\tilde{h}})(\widetilde{e} + e_h) \approx -F(u_{\tilde{h}}, \lambda_{\tilde{h}}) - \frac{1}{2} D_{uu} F(u_{\tilde{h}}, \lambda_{\tilde{h}})(\widetilde{e} + e_h)^2, \tag{4.4}$$

where $\widetilde{e} = u - u_{\tilde{h}} - e_h$. Note that $D_{uu} F(u_{\tilde{h}}, \lambda_{\tilde{h}})(\widetilde{e} + e_h)^2 = \lambda_{\tilde{h}} f''(u_{\tilde{h}})(\widetilde{e} + e_h)^2$. Equation (4.4) can be expressed as

$$-\Delta(\widetilde{e} + e_h) + \lambda_{\tilde{h}} f'(u_{\tilde{h}})(\widetilde{e} + e_h) \approx \Delta u_{\tilde{h}} - \lambda_{\tilde{h}} f(u_{\tilde{h}}) - \frac{1}{2} \lambda_{\tilde{h}} f''(u_{\tilde{h}})(\widetilde{e} + e_h)^2. \tag{4.5}$$

Since e_h is the approximate solution of (4.3), (4.5) becomes

$$-\Delta \widetilde{e} + \lambda_{\tilde{h}} f'(u_{\tilde{h}})\widetilde{e} \approx -\frac{1}{2} \lambda_{\tilde{h}} f''(u_{\tilde{h}})(\widetilde{e} + e_h)^2 \approx -\frac{1}{2} \lambda_{\tilde{h}} f''(u_{\tilde{h}}) e_h^2.$$

Thus the further coarse space correction is obtained by solving the following equation:

$$\begin{aligned} -\Delta \widetilde{e} + \lambda_{\tilde{h}} f'(u_{\tilde{h}})\widetilde{e} &= -\frac{1}{2} \lambda_{\tilde{h}} f''(u_{\tilde{h}}) e_h^2 && \text{in } \Omega = (0,1)^2, \\ \widetilde{e} &= 0 && \text{on } \partial\Omega. \end{aligned} \tag{4.6}$$

The finite element approximation of (4.6) is to find $e_{\tilde{h}} \in V_{\tilde{h}}$ such that

$$\langle \nabla e_{\tilde{h}}, \nabla v \rangle + \lambda_{\tilde{h}} \langle f'(u_{\tilde{h}}) e_{\tilde{h}}, v \rangle = -\frac{1}{2} \lambda_{\tilde{h}} \langle f''(u_{\tilde{h}}) e_{\tilde{h}}^2, v \rangle \qquad \forall v \in V_{\tilde{h}}. \tag{4.7}$$

In order to correct the parameter $\lambda_{\tilde{h}}$, we set $u_h = u_{\tilde{h}} + e_h + e_{\tilde{h}}$ and compute the Rayleigh quotient

$$\lambda_h = -\frac{\displaystyle\int_\Omega \nabla u_h \cdot \nabla u_h}{\displaystyle\int_\Omega f(u_h) u_h}.$$

From the discussion given above, we will describe a two-grid finite element discretization scheme with a two-loop continuation algorithm for solving second order semilinear elliptic eigenvalue problems. If $(u_{\tilde{h}}^{(i-1)}, \lambda_{\tilde{h}}^{(i-1)})$ has been accepted as an approximating point for $\widetilde{c}$ on the coarse space, then $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ can be obtained by solving (1.2) with the predictor-corrector continuation method, which is called the outer-loop continuation. To obtain the corresponding approximating point $(u_h^{(i)}, \lambda_h^{(i)})$ for the solution curve c on the fine space, we use $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ as a predicted point for the inner-loop continuation. In practice, we have to interpolate $u_{\tilde{h}}^{(i)}$ on the fine space. In the corrector step, we solve the first order approximation of $F(u, \lambda) = 0$ on the fine space to obtain a correction e_h for the state variable $u_{\tilde{h}}^{(i)}$. The linearization of (1.2) shows that this process is equivalent to performing one Newton iteration on the fine space; see [30]. To improve the accuracy of the state variable, we make a further correction on the coarse space by solving the second order approximation of (1.2), and thus obtain the state variable $u_h^{(i)}$ on the fine grid. We compute the Rayleigh quotient as in [32] for the corresponding $\lambda_h^{(i)}$. In order to guarantee that the approximating point is accurate enough, we use $(u_h^{(i)}, \lambda_h^{(i)})$ as an initial guess and perform Newton's method until it converges to the desired solution. Then we finish the corrector step of the inner continuation and go back to the coarse space to find the next approximating point for $\widetilde{c}$. We repeat this process until the solution curve c on the fine space is traced. We denote the restriction operator from the fine space to the coarse space by $I_h^{\tilde{h}}$ and the interpolation operator from the coarse space to the fine space by $I_{\tilde{h}}^h$. That is, $I_h^{\tilde{h}} : \mathbb{R}^{N^2} \to \mathbb{R}^{\widetilde{N}^2}$ and $I_{\tilde{h}}^h : \mathbb{R}^{\widetilde{N}^2} \to \mathbb{R}^{N^2}$.

ALGORITHM 4.1. A two-grid finite element discretization scheme with a two-loop continuation algorithm for tracing solution curves of (1.1).

Input:

$\varepsilon :=$ accuracy tolerance of approximating points for both solution curves $\widetilde{c}$ and c on the coarse and fine spaces, respectively

$i_{max} :=$ maximum number of continuation steps

$\left(u_{\tilde{h}}^{(0)}, \lambda_{\tilde{h}}^{(0)}\right) :=$ starting approximating point for $\widetilde{c}$

$i = 1$

1. **Outer continuation.**
   (i) Use the predictor-corrector continuation algorithm to find the approxi-

mating point $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)}) \in V_{\tilde{h}} \times \mathbb{R}$ for $\widetilde{c}$ such that

$$\int_\Omega \nabla u_{\tilde{h}}^{(i)} \cdot \nabla v + \lambda_{\tilde{h}}^{(i)} \int_\Omega f\left(u_{\tilde{h}}^{(i)}\right) v = 0 \quad \forall v \in V_{\tilde{h}}.$$

(ii) Compute the a posteriori error estimate for $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$.

2. **Inner continuation.**

(i) Predictor. Set $(I_{\tilde{h}}^h u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ as the predicted point.

(ii) Corrector.

(a) Make a correction on the fine space V_h: Find $e_h \in V_h$ such that

$$\int_\Omega \nabla e_h \cdot \nabla v + \lambda_{\tilde{h}}^{(i)} \int_\Omega f'\left(u_{\tilde{h}}^{(i)}\right) e_h v = -\int_\Omega \nabla u_{\tilde{h}}^{(i)} \cdot \nabla v - \lambda_{\tilde{h}}^{(i)} \int_\Omega f\left(u_{\tilde{h}}^{(i)}\right) v \quad \forall v \in V_h.$$

(b) Make a further correction on the coarse space $V_{\tilde{h}}$: Find $e_{\tilde{h}} \in V_{\tilde{h}}$ such that

$$\int_\Omega \nabla e_{\tilde{h}} \cdot \nabla v + \lambda_{\tilde{h}}^{(i)} \int_\Omega f'(u_{\tilde{h}}^{(i)}) e_{\tilde{h}} v = -\frac{1}{2} \lambda_{\tilde{h}}^{(i)} \int_\Omega f''\left(u_{\tilde{h}}^{(i)}\right) e_h^2 v \quad \forall v \in V_{\tilde{h}}.$$

(c) Set $u_h^{(i)} = u_{\tilde{h}}^{(i)} + e_h + e_{\tilde{h}}$.

(d) Compute $\lambda_h^{(i)} = -\dfrac{\displaystyle\int_\Omega \nabla u_h^{(i)} \cdot \nabla u_h^{(i)}}{\displaystyle\int_\Omega f(u_h^{(i)}) u_h^{(i)}}$.

(e) If $\|F(u_h^{(i)}, \lambda_h^{(i)})\| > \varepsilon$, then
use $(u_h^{(i)}, \lambda_h^{(i)})$ as an initial guess and perform Newton's method. If Newton's method does not converge, then stop.
End if

(f) Compute an posteriori error estimate for the accepted approximating point.

3. If $i = i_{max}$, exit.
Else
set $i = i + 1$ and go to step 1.
End if

Next, we will describe a centered difference counterpart of Algorithm 4.1. Let $(u_{\tilde{h}}, \lambda_{\tilde{h}}) \in \mathbb{R}^{\widetilde{N}^2} \times \mathbb{R}$ be the centered-difference approximate solution of (1.1) on the coarse grid with uniform mesh size $\widetilde{h} = \frac{1}{\widetilde{N}+1}$, i.e.,

$$A_{\tilde{h}} u_{\tilde{h}} + \lambda_{\tilde{h}} f(u_{\tilde{h}}) = 0, \tag{4.8}$$

where $A_{\tilde{h}} \in \mathbb{R}^{\widetilde{N}^2 \times \widetilde{N}^2}$ is the coefficient matrix corresponding to the Laplace operator $-\Delta$ and $f(u_{\tilde{h}}) = [f((u_{\tilde{h}})_1), f((u_{\tilde{h}})_2), \ldots, f((u_{\tilde{h}})_{\widetilde{N}^2})]^T \in \mathbb{R}^{\widetilde{N}^2}$. Similar to the two-grid finite element discretization scheme, we correct $(u_{\tilde{h}}, \lambda_{\tilde{h}})$ by solving

$$\begin{aligned} -\Delta e + \lambda_{\tilde{h}} f'(u_{\tilde{h}}) e &= \Delta u_{\tilde{h}} - \lambda_{\tilde{h}} f(u_{\tilde{h}}) && \text{in } \Omega = (0,1)^2, \\ e &= 0 && \text{on } \partial\Omega \end{aligned} \tag{4.9}$$

on the fine grid with uniform mesh size $h = \frac{1}{N+1}$. Let

$$\begin{aligned} D_h &= \operatorname{diag}\left(f'((I_{\tilde{h}}^h u_{\tilde{h}})_1), f'((I_{\tilde{h}}^h u_{\tilde{h}})_2), \ldots, f'((I_{\tilde{h}}^h u_{\tilde{h}})_{N^2})\right) \in \mathbb{R}^{N^2 \times N^2}, \\ f(I_{\tilde{h}}^h u_{\tilde{h}}) &= \left[f((I_{\tilde{h}}^h u_{\tilde{h}})_1), f((I_{\tilde{h}}^h u_{\tilde{h}})_2), \ldots, f((I_{\tilde{h}}^h u_{\tilde{h}})_{N^2})\right]^T \in \mathbb{R}^{N^2}. \end{aligned} \tag{4.10}$$

The centered-difference analogue of (4.9) is of the form

$$(A_h + \lambda_{\tilde{h}} D_h) e_h = -A_h (I_{\tilde{h}}^h u_{\tilde{h}}) - \lambda_{\tilde{h}} f(I_{\tilde{h}}^h u_{\tilde{h}}). \tag{4.11}$$

Denote the solution of (4.11) by e_h. The further coarse grid correction is obtained by solving

$$\begin{aligned} -\Delta \widetilde{e} + \lambda_{\tilde{h}} f'(u_{\tilde{h}}) \widetilde{e} &= -\frac{1}{2} \lambda_{\tilde{h}} f''(u_{\tilde{h}}) e_{\tilde{h}}^2 \qquad \text{in } \Omega = (0,1)^2, \\ \widetilde{e} &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{4.12}$$

on the coarse grid. Denote

$$\begin{aligned} D_{\tilde{h}} &= \operatorname{diag}\left(f'((u_{\tilde{h}})_1), f'((u_{\tilde{h}})_2), \ldots, f'((u_{\tilde{h}})_{\widetilde{N}^2})\right) \in \mathbb{R}^{\widetilde{N}^2 \times \widetilde{N}^2}, \\ E_{\tilde{h}} &= \operatorname{diag}\left((I_h^{\tilde{h}} e_h)_1, (I_h^{\tilde{h}} e_h)_2, \ldots, (I_h^{\tilde{h}} e_h)_{\widetilde{N}^2}\right) \in \mathbb{R}^{\widetilde{N}^2 \times \widetilde{N}^2}, \\ f''(u_{\tilde{h}}) &= \left[f''((u_{\tilde{h}})_1), f''((u_{\tilde{h}})_2), \ldots, f''((u_{\tilde{h}})_{\widetilde{N}^2})\right]^T \in \mathbb{R}^{\widetilde{N}^2}. \end{aligned} \tag{4.13}$$

The centered-difference counterpart of (4.12) is of the form

$$(A_{\tilde{h}} + \lambda_{\tilde{h}} D_{\tilde{h}}) e_{\tilde{h}} = -\frac{1}{2} \lambda_{\tilde{h}} E_{\tilde{h}}^2 f''(u_{\tilde{h}}). \tag{4.14}$$

Denote the solution of (4.14) by $e_{\tilde{h}}$. In order to obtain an accurate parameter λ_h on the fine grid, we set $u_h = I_{\tilde{h}}^h u_{\tilde{h}} + e_h + I_{\tilde{h}}^h e_{\tilde{h}}$ and compute the Rayleigh quotient

$$\lambda_h = -\frac{(u_h)^T A_h u_h}{(u_h)^T f(u_h)},$$

where $f(u_h) = [f((u_h)_1), f((u_h)_2), \ldots, f((u_h)_{N^2})]^T \in \mathbb{R}^{N^2}$. Therefore, we have the following result.

ALGORITHM 4.2. A two-grid centered-difference discretization scheme with a two-loop continuation algorithm for tracing solution curves of (1.1).

Input:

$\varepsilon :=$ accuracy tolerance of approximating points for both solution curves $\widetilde{c}$ and c on the coarse and fine spaces, respectively

$i_{max} :=$ maximum number of continuation steps

$(u_{\tilde{h}}^{(0)}, \lambda_{\tilde{h}}^{(0)}) :=$ accepted approximating point for $\widetilde{c}$

$i = 1$

1. **Outer continuation.**
   (i) Use the predictor-corrector continuation algorithm to find the approximating point $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ for $\widetilde{c}$ on the coarse grid: Solve the nonlinear system

$$A_{\tilde{h}} u_{\tilde{h}}^{(i)} + \lambda_{\tilde{h}}^{(i)} f(u_{\tilde{h}}^{(i)}) = 0.$$

   (ii) Compute the a posteriori error estimate for $(u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$.

2. **Inner continuation.**
   (i) Predictor. Set $(I_{\tilde{h}}^{h} u_{\tilde{h}}^{(i)}, \lambda_{\tilde{h}}^{(i)})$ as the predicted point.
   (ii) Corrector.
      (a) Make a correction on the fine grid: Solve the linear system

$$\left(A_h + \lambda_{\tilde{h}}^{(i)} D_h\right) e_h = -A_h \left(I_{\tilde{h}}^{h} u_{\tilde{h}}^{(i)}\right) - \lambda_{\tilde{h}}^{(i)} f\left(I_{\tilde{h}}^{h} u_{\tilde{h}}^{(i)}\right).$$

      (b) Make a further correction on the coarse grid: Solve the linear system

$$\left(A_{\tilde{h}} + \lambda_{\tilde{h}}^{(i)} D_{\tilde{h}}\right) e_{\tilde{h}} = -\frac{1}{2} \lambda_{\tilde{h}}^{(i)} E_{\tilde{h}}^{2} f''\left(u_{\tilde{h}}^{(i)}\right).$$

      (c) Set $u_h^{(i)} = I_{\tilde{h}}^{h} u_{\tilde{h}}^{(i)} + e_h + I_{\tilde{h}}^{h} e_{\tilde{h}}$.
      (d) Compute $\lambda_h^{(i)} = -\dfrac{(u_h^{(i)})^T A_h u_h^{(i)}}{(u_h^{(i)})^T f(u_h^{(i)})}$.
      (e) If $\|F(u_h^{(i)}, \lambda_h^{(i)})\| > \varepsilon$, then
          use $(u_h^{(i)}, \lambda_h^{(i)})$ as an initial guess and perform Newton's method. If Newton's method does not converge, then stop.
          End if
      (f) Compute an posteriori error estimate for the accepted approximating point.
3. If $i = i_{max}$, exit.
   Else
      set $i = i + 1$ and go to step 1.
   End if

We remark here that Algorithms 4.1 and 4.2 provide a mechanism which guarantees the accuracy of the approximate solution curves and singular points we obtained if the algorithms converge. We explain this mechanism as follows.

(1) In steps 1 and 2 of Algorithm 4.1 we use the error estimators in [19] to compute an posteriori error estimate for the approximate solution $(u_h^{(i)}, \lambda_h^{(i)})$. For comparison the multigrid-continuation algorithm [9] was also implemented to trace solution branches of some test problems. More precisely, let $(v_h^{(i)}(x_0, y_0), \mu_h^{(i)})$ be the approximate solution at a point $(x_0, y_0) \in \Omega$ which is obtained by implementing the multigrid-continuation algorithm. We use the data $\{(v_h^{(j)}(x_0, y_0), \mu_h^{(j)})\}_{j=i-k}^{j=i+k}$ to construct an interpolation polynomial $P_{2k}(\mu)$ of degree $2k$. It was verified by our numerical experiments that the inequality $|u_h^{(i)}(x_0, y_0) - P_{2k}(\lambda_h^{(i)})| < 10^{-5}$ holds.
(2) The Newton iteration in step 2 of Algorithms 4.1 and 4.2 serves as a criterion to check how close the two-grid approximating point we obtained is to the solution curve on the fine grid. Moreover, if the Newton iteration converges at each continuation step, then the algorithms converge eventually. As well, it is evident that we can compute accurate singular points.

At this moment it does not seem adequate to us to give a rigorous proof for the convergence of Algorithm 4.1. However, our numerical results show that the algorithm is indeed robust and efficient and converges for various second order semilinear elliptic eigenvalue problems. Note that the PLTMGC [3] also uses a continuation procedure to follow the solution curve on the coarsest grid and a multigrid algorithm to refine the solution at selected points using an adaptive mesh refinement strategy. The

drawback of this strategy is that the approximating points on the solution curve without refinement would become poor approximations for the exact solution curve. See, e.g., Tables 2 and 5.

5. Numerical results. The algorithms described in sections 3 and 4 were implemented to compute the first few eigenpairs of (3.1) and to trace solution branches of various second order semilinear elliptic eigenvalue problems with Dirichlet boundary conditions, respectively. These problems have been widely used as test problems in the literature of continuation methods. Table 1 lists the numerical algorithms we have implemented during the past years. However, for consistency we will discuss only the numerical results concerning the implementations of Algorithms 4.1 and 4.2. The numerical results of implementing Methods 1 and 4 can be found in [6] and [9], respectively.

Equations (5.1)–(5.8) were discretized by the six-node triangular elements method. The domain Ω was divided into two families of triangles, $\mathcal{T}_{\tilde{h}} = \{\tau_{\tilde{h}}\}$ and $\mathcal{T}_h = \{\tau_h\}$, as shown in Figure 1. We chose $V_{\tilde{h}} \subset V_h \subset H_0^1$ as the quadratic spaces of piecewise continuous quadratic functions defined on $\mathcal{T}_{\tilde{h}}$ and $\mathcal{T}_h$, respectively. Here $\widetilde{h}$ and h denote the distance of two adjacent nodes on the coarse and fine grids, respectively. In Examples 1–6 we chose $\widetilde{h} = \frac{1}{16}$ and $h = \frac{1}{128}$, while in Examples 7–8, we set $\widetilde{h} = \frac{1}{16}$ and $h = \frac{1}{256}$. For the two-grid centered difference discretization scheme, $\widetilde{h}$ and h are defined in the same way. All of our computations were executed on an IBM xSeries 335 machine using FORTRAN 95 language with double precision arithmetic. The following notations are used in Tables 2–7.

NOMENCLATURE

ε	: accuracy tolerance in Newton corrector
tol	: stopping criterion for solving linear systems
$\|u_{\tilde{h}}\|_\infty$	: maximum norm of the approximate solution $u_{\tilde{h}}$
Error$(u_{\tilde{h}}, \lambda_{\tilde{h}})$	: the a posteriori error estimate for the approximate solution $(u_{\tilde{h}}, \lambda_{\tilde{h}})$
$\|u_h\|_\infty$	: maximum norm of the approximate solution u_h
Error(u_h, λ_h)	: the a posteriori error estimate for the approximate solution (u_h, λ_h)
Time	: total execution time (in seconds)
NCS	: number of the continuation steps

' . , 1.

$$\begin{aligned} \Delta u + \lambda e^u &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.1}$$

Equation (5.1) is the well-known Gelfand–Bratu problem and is widely used as a test problem in the literature of continuation methods; see, e.g., [3, 5, 9, 14, 15, 19]. Our numerical experiments show that we have to choose a proper step size on the outer-loop continuation if Algorithm 4.2 was implemented to follow the solution curve of (5.1).

' . , 2.

$$\begin{aligned} \Delta u + \lambda \exp\left(\frac{u}{1+\alpha u}\right) &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.2}$$

Equation (5.2) is the two-parameter Bratu problem [4].

TABLE 1
Numerical algorithms for semilinear elliptic eigenvalue problems.

Method 1 := multigrid-continuation algorithm, finite element approximation
Method 2 := Algorithm 4.1
Method 3 := Algorithm 4.2
Method 4 := Algorithm 4.1 with symmetry reduction on the domain

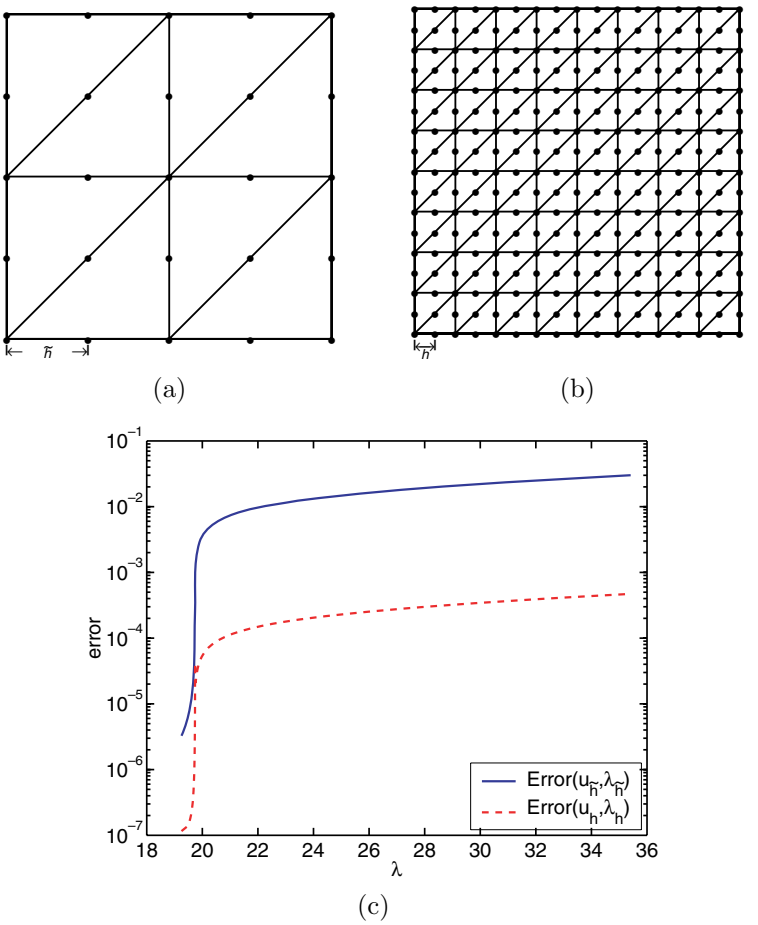


FIG. 1. (a) *Triangulation* $\mathcal{T}_{\tilde{h}} = \{\tau_{\tilde{h}}\}$. (b) *Triangulation* $\mathcal{T}_h = \{\tau_h\}$. (c) *The a posteriori error estimates for the numerical solution branches of* (5.6) *bifurcating at* $(0, \lambda_{1,1})$.

, , 3.

$$\begin{aligned} \Delta u + \lambda u + f(u,\lambda) &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.3}$$

Here f satisfies $f(0,\lambda) = f_u(0,\lambda) = 0$ for all $\lambda \in \mathbb{R}$. We chose $f(u,\lambda) = \frac{\lambda}{2}u^2$. See [15].

, , 4.

$$\begin{aligned} \Delta u + \lambda \sinh u &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.4}$$

TABLE 2
The solution curve of (5.6) *branching from* $(0, \lambda_{1,1}) \approx (0, 19.7392088)$, $\widetilde{h} = 1/16$, $h = 1/128$, $\varepsilon = 5 \times 10^{-8}$, $tol = 5 \times 10^{-10}$, *using Algorithm* 4.1.

$\lambda_{\tilde{h}}$	$\|u_{\tilde{h}}\|_\infty$	Error$(u_{\tilde{h}}, \lambda_{\tilde{h}})$	λ_h	$\|u_h\|_\infty$	Error(u_h, λ_h)
19.6843308	0.3279019D−02	0.2789457D−04	19.6843311	0.3542039D−02	0.4778044D−06
19.7364078	0.2355281D−01	0.2571973D−03	19.7366224	0.3730704D−01	0.9701814D−05
19.7395094	0.3217089D−01	0.4119890D−03	19.7392089	0.4595079D−01	0.3108281D−04
19.7402543	0.3466964D−01	0.4540608D−03	19.7397414	0.4769844D−01	0.3666674D−04
19.7930029	0.1632023D+00	0.1848991D−02	19.7929801	0.1702689D+00	0.2782277D−04
19.9706649	0.3487274D+00	0.3382880D−02	19.9706657	0.3519709D+00	0.5116206D−04
21.7451746	0.9995050D+00	0.9150861D−02	21.7451781	0.1000270D+01	0.1406428D−03
22.9714078	0.1240284D+01	0.1152408D−01	22.9714125	0.1240768D+01	0.1777489D−03
23.9193413	0.1386652D+01	0.1312081D−01	23.9193469	0.1387021D+01	0.2027532D−03
24.9578112	0.1521533D+01	0.1474845D−01	24.9578179	0.1521826D+01	0.2282735D−03
26.6598995	0.1702960D+01	0.1727919D−01	26.6599082	0.1703189D+01	0.2680191D−03
30.8766139	0.2024594D+01	0.2338915D−01	30.8766294	0.2024807D+01	0.3643140D−03

TABLE 3
(a) *The total execution time for tracing the solution branch of* (5.6) *bifurcating at* $(0, \lambda_{1,1})$ *by implementing Algorithm* 4.1 *and the multigrid-continuation algorithm until* $\lambda_h \approx 79.4$.

	Method 1	Algorithm 4.1
Time (sec)	1800.844	491.9531
NCS	243	100
$(\|u_h\|_\infty, \lambda_h)$	$(2.9547348, 79.4243429)$	$(2.9545366, 79.3873102)$

(b) *The total execution time for tracing the solution branch of* (5.6) *bifurcating at* $(0, \lambda_{1,1})$ *by implementing Algorithm* 4.1 *and the multigrid-continuation algorithm with* 100 *continuation steps.*

	Method 1	Algorithm 4.1
Time (sec)	715.0938	491.9531
$(\|u_h\|_\infty, \lambda_h)$	$(1.1561320, 22.4964732)$	$(2.9545366, 79.3873102)$

Equation (5.4) describes a model for the equilibrium of a set of point vortices or a guiding-center plasma in the domain Ω. See [28, 29].

' . ,' 5.

$$\begin{aligned} \Delta u + \lambda u - u^3 &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.5}$$

See [29] for the numerical study of (5.5).

' . ,' 6.

$$\begin{aligned} \Delta u + \lambda \sin u &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.6}$$

The numerical study of (5.6) can be found, e.g., in [20, 22] and some references cited in [20]. Both Algorithms 4.1 and 4.2 were used to trace the solution curve of (5.6) bifurcating at $(0, \lambda_{1,1}) \approx (0, 19.7392088)$. Our numerical result is shown in Table 2. Figure 1 displays the error estimates for the solution curves on both grids. By comparing our result with the multigrid solution we find that the maximum norm of u_h is correct up to six decimal digits. The total execution time of implementing Algorithm 4.1 and the multigrid-continuation algorithm is shown in Table 3.

' . ,' 7.

$$\begin{aligned} \Delta u + \lambda \sin u - u^4 &= 0 \quad \text{in} \quad \Omega = (0,1)^2, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned} \tag{5.7}$$

TABLE 4

The positive solution curve of (5.7) *branching from* $(0, \lambda_{1,1}) \approx (0, 19.7392088)$, $\widetilde{h} = 1/16$, $h = 1/256$, $\varepsilon = 5 \times 10^{-8}$, $tol = 5 \times 10^{-10}$, *using Algorithm* 4.2.

$\lambda_{\tilde{h}}$	$u_{\tilde{h}}(0.5, 0.5)$	λ_h	$u_h(0.5, 0.5)$
19.6518910	0.834033118D−01	19.6326416	0.288352028D−01
19.7340228	0.195303437D+00	19.7408253	0.120508559D+00
19.8444760	0.290887470D+00	19.8473108	0.234671350D+00
20.0276840	0.412808135D+00	20.0276234	0.375472856D+00
21.2991341	0.827219415D+00	21.2981236	0.812145863D+00
22.6720583	0.106983733D+01	22.6711671	0.105985707D+01
23.8238891	0.121683190D+01	23.8231217	0.120906681D+01
24.9076795	0.132854605D+01	24.9070257	0.132221414D+01
26.4594188	0.145909291D+01	26.4589147	0.145425090D+01
28.1033343	0.157148730D+01	28.1029698	0.156781362D+01
30.2519752	0.169098021D+01	30.2517638	0.168846209D+01
32.4754514	0.179171533D+01	32.4753657	0.179011343D+01

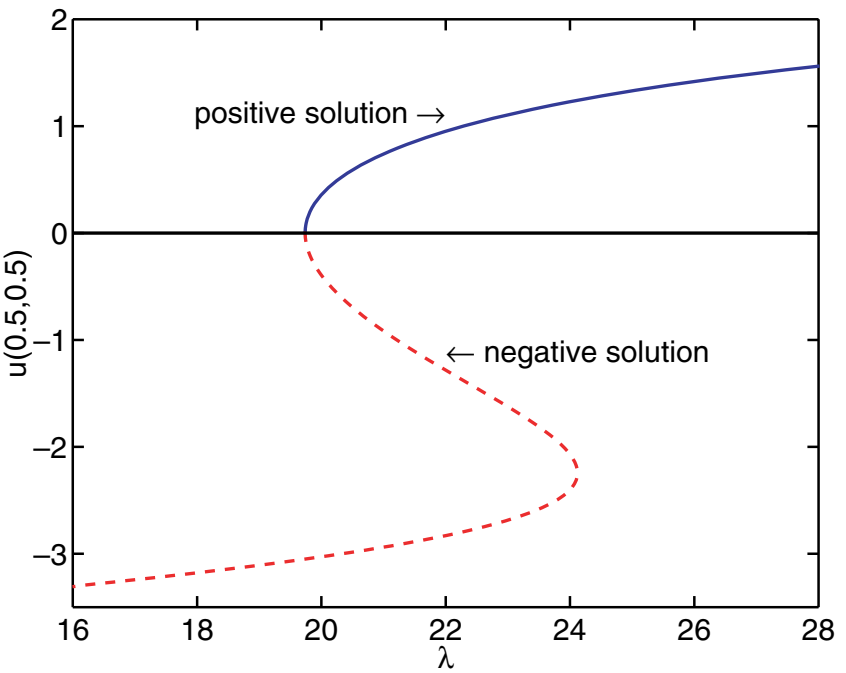


FIG. 2. *The solution curve of* (5.7) *bifurcating at* $(0, \lambda_{1,1})$.

Equation (5.7) was used as a test problem in [29]. Algorithms 4.1 and 4.2 were implemented to trace the solution curve of (5.7) with $\widetilde{h} = \frac{1}{16}$ and $h = \frac{1}{256}$. Table 4 lists some sample approximating points for the solution curve of (5.7). Figure 2 shows the solution curve.

8. Let Ω be a bounded regular domain in $\mathbb{R}^N$. In [18, Remark 1.7] Lions claimed that the equation

$$(5.8) \qquad \begin{aligned} -\Delta u - \lambda(u(1 - \sin u) + u^p) &= 0 \quad \text{in} \quad \Omega = (0,1)^N, \quad 1 < p < N^\star, \\ u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

has exactly one fold on the $(\lambda, \|u\|_\infty)$ plane if $N = 2$. Here $N^* = \infty$ if $N = 1, 2$, and

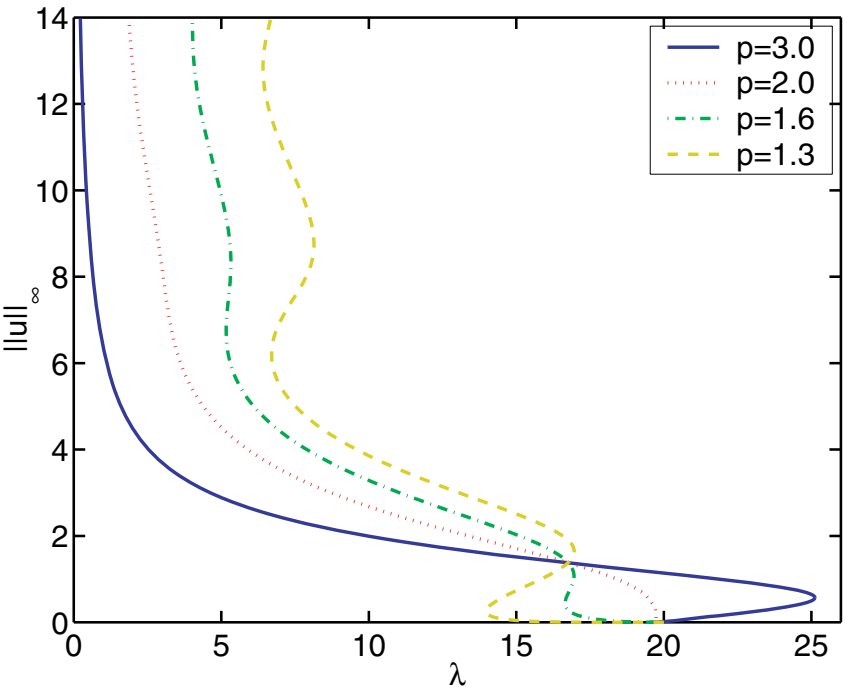


FIG. 3. *The solution curves of* (5.8) *with* $p = 1.3, 1.6, 2,$ *and* 3.

$N^* = (N+2)/(N-2)$ if $N \geq 3$. The numerical study of (5.8) can be found, e.g., in [21, 29], where only the case $p = 3$ was considered. Recently, Wang [27] showed that for the one-dimensional problem of (5.8) the number of folds on each solution curve varies with respect to the values of p. In particular, for $p = 1$ the solution curve has infinitely many folds. Therefore, it is getting tough to trace the solution curves of (5.8) as p is close to 1. A detailed numerical investigation of (5.8) for $N = 1, 2,$ and 3 can be found in [7]. We used Algorithm 4.1 to follow some solution curves of (5.8). Figure 3 shows the solution curves of (5.8) with $p = 1.3, 1.6, 2,$ and 3. Table 5 lists some sample approximating points for the solution curve of (5.8) with $p = 1.3$, where the first four folds are detected at $(\|u\|_\infty, \lambda) \approx (0.2523042, 13.9777120)$, $(1.6399861, 16.9622642)$, $(6.2075371, 6.7043602)$, and $(8.7675950, 8.1421997)$. For $p = 1.6$, the first four folds are detected at $(0.4404359, 16.6538055)$, $(1.0762562, 16.9532893)$, $(6.7520017, 5.1649507)$, and $(8.2972692, 5.3251472)$. For $p = 2.0$, there is no fold on the solution curve, while for $p = 3.0$ the only fold is detected at $(0.5650960, 25.1183419)$. We refer to [7] for a comparison.

Table 6 lists the performance results of Methods 1, 2, 3, and 4 on Examples 1, 2, 3, 4, 5, 6, 7, and 8.

9. The eigenvalues of (3.1) are

$$\lambda_{p,q} = (p^2 + q^2)\pi^2, \quad p, q = 1, 2, 3, \ldots.$$

Equation (3.1) was discretized via six-node triangular elements. We chose $\widetilde{h} = \frac{1}{8}$ on the coarse grid and $h = \frac{1}{128}$ on the fine grid. The coefficient matrices on the coarse grid and on the fine grid are of order 49×49 and 16129×16129, respectively. We used Algorithm 3.1 to find the first six eigenpairs of the discrete problem. Table 7 shows the numerical results.

TABLE 5

The solution curve of (5.8) *with* $p = 1.3$ *branching from* $(0, \lambda_{1,1}) \approx (0, 19.7392088)$, $\widetilde{h} = 1/16$, $h = 1/256$, $\varepsilon = 5 \times 10^{-8}$, $tol = 5 \times 10^{-10}$, *using Algorithm* 4.1.

$\lambda_{\tilde{h}}$	$\|u_{\tilde{h}}\|_\infty$	Error$(u_{\tilde{h}}, \lambda_{\tilde{h}})$	λ_h	$\|u_h\|_\infty$	Error(u_h, λ_h)
19.7400000	0.6615081D−06	0.5824778D−08	19.7374345	0.6611850D−06	0.5733167D−10
18.9825001	0.3194642D−04	0.2721934D−06	18.9824436	0.3129806D−04	0.1031793D−08
17.0827460	0.3083873D−02	0.2626539D−04	17.0827462	0.3064743D−02	0.1008200D−06
15.1656976	0.3485314D−01	0.2982262D−03	15.1656976	0.3469748D−01	0.1144231D−05
14.0068879	0.2024069D+00	0.1864059D−02	14.0068826	0.1996069D+00	0.6938069D−05
13.9815036	0.2606921D+00	0.3098978D−02	13.9777120	0.2523042D+00	0.1062710D−04
14.0159459	0.3186641D+00	0.3456459D−02	14.0159441	0.3216593D+00	0.1267115D−04
15.0668721	0.7547041D+00	0.6978355D−02	15.0668741	0.7554925D+00	0.2662709D−04
16.4174872	0.1211507D+01	0.1153721D−01	16.4174890	0.1212559D+01	0.4403549D−04
16.9411884	0.1552548D+01	0.1659600D−01	16.9411739	0.1558846D+01	0.6201316D−04
16.9634193	0.1612429D+01	0.2048482D−01	16.9622642	0.1639986D+01	0.7484689D−04
16.9393355	0.1733619D+01	0.2705462D−01	16.9393327	0.1725836D+01	0.9285853D−04
15.5447116	0.2361841D+01	0.2187175D−01	15.5447043	0.2358756D+01	0.8320662D−04
13.1216769	0.2987208D+01	0.3113458D−01	13.1216539	0.2983048D+01	0.1184269D−03
10.7840627	0.3615562D+01	0.4240436D−01	10.7840371	0.3610490D+01	0.1608645D−03
8.0262974	0.4684141D+01	0.5950526D−01	8.0262863	0.4679324D+01	0.2220307D−03

TABLE 6

Problem	(5.1)	(5.2)	(5.3)	(5.4)	(5.5)	(5.6)	(5.7)	(5.8)
Method 1	success	success	—	success	success	success	success	success
Method 2	success	success	success	success	success	success	success	success
Method 3	success	success†	success	failed	—	success	success	—
Method 4	—	success	success	success	—	—	—	—

†For $\alpha = 0.1$, Method 3 failed to converge at the fold.
— No test.

TABLE 7

The first six eigenvalues and residuals of the corresponding eigenvectors of the discretized generalized eigenvalue problem in two dimensions with Dirichlet boundary conditions.

Vector	Coarse grid $\lambda_{\tilde{h}}$	Fine grid λ_h	Residual	RQI μ_1	Residual
1	19.805118628	19.739260864	1.076337239D−05	19.739209915	2.499124043D−10
2	50.383506082	49.351779995	9.558794252D−05	49.348041011	1.760607123D−08
3	50.383506082	49.351779995	9.558794252D−05	49.348041011	1.760607133D−08
4	82.142640415	79.076580633	2.702660953D−04	78.956910426	8.509756273D−07
5	103.696686497	98.547414332	4.242685159D−04	98.696132374	1.373508247D−06
6	103.696686497	98.547414332	4.242685159D−04	98.696132374	1.373508246D−06

The first six exact eigenvalues:
19.739208802 49.348022005 49.348022005 78.956835208 98.696044010 98.696044010

6. Conclusions. In this paper, we present two-grid discretization schemes with two-loop continuation algorithms for tracing solution branches of second order semilinear elliptic eigenvalue problems. Of particular interest here are the computations of singular points such as folds and bifurcations. Based on the results of our numerical implementations, we wish to state some conclusions concerning the performance of the two schemes we propose. We will also compare the differences between the two-grid, the single grid, and the multigrid schemes for semilinear elliptic eigenvalue problems.

(1) In the two-grid schemes, two solution branches are numerically traced; one is on the coarse grid, and the other is on the fine grid. Therefore, we need to

perform two-loop continuation algorithms. Besides, the shape of approximating points computed on both grids looks like a saw. Tracing a solution branch on the coarse grid is quite inexpensive because the grid size is relatively small compared to the one on the fine grid. Moreover, the approximating points on the coarse grid serve as predicted points for computing approximating points on the fine space. This means that we don't have to solve linear systems for tangent vectors. Furthermore, it is unnecessary to use any step size control strategy on the inner-loop continuation. The purpose of multigrid method is to reduce the computational costs for solving linear systems on the finest grid. However, we have to solve additional linear systems on each level between the coarsest and finest grids. Both the single grid and most of the multigrid methods require step size strategy and the determination of tangent vectors on the fine grid except the PLTMGC.

(2) In the two-grid schemes, exploiting symmetry of the Laplace operator is manifest on both coarse and fine spaces. In the case of single grid or multigrid schemes, however, in order to exploit symmetry of the Laplacian, one either has to solve bordered linear systems by using the block elimination algorithm or choose a proper constraint vector to make the augmented matrices symmetric.

(3) From Table 3 we see that Algorithm 4.1 runs faster and more efficiently than the multigrid-continuation algorithm. We obtained similar results for Example 1 which are not shown here.

(4) Finally, our numerical results show that the two-grid centered difference discretization scheme may fail to converge near folds of semilinear elliptic eigenvalue problems. We suspect that it is probably the discretization error that makes this scheme less robust than the finite element one.

Acknowledgments. The authors would like to thank the two referees for their valuable comments that have improved the original version of this paper.

REFERENCES

[1] E. L. Allgower and K. Böhmer, *Application of the mesh independence principle to mesh refinement strategies*, SIAM J. Numer. Anal., 24 (1987), pp. 1335–1351.

[2] E. L. Allgower and K. Georg, *Introduction to Numerical Continuation Methods*, SIAM, Philadelphia, PA, 2003.

[3] R. E. Bank and T. F. Chan, *PLTMGC: A multi-grid continuation program for parameterized nonlinear elliptic systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 540–559.

[4] J. H. Bolstad and H. B. Keller, *A multigrid continuation method for elliptic problems with folds*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 1081–1104.

[5] T. F. Chan and H. B. Keller, *Arc-length continuation and multi-grid techniques for nonlinear elliptic eigenvalue problems*, SIAM J. Sci. Stat. Comput., 3 (1982), pp. 173–194.

[6] S.-L. Chang and C.-S. Chien, *A multigrid-Lanczos algorithm for the numerical solutions of nonlinear eigenvalue problems*, Int. J. Bifurcation and Chaos, 13 (2003), pp. 1217–1228.

[7] S.-L. Chang, C.-S. Chien, and B.-W. Jeng, *Tracing the solution surface with folds of a two parameter system*, Int. J. Bifurcation and Chaos, 15 (2005), pp. 2689–2700.

[8] C.-S. Chien and S.-L. Chang, *Application of the Lanczos algorithm for solving the linear systems that occur in continuation problems*, Numer. Linear Algebra Appl., 10 (2003), pp. 335–355.

[9] C.-S. Chien and B.-W. Jeng, *Symmetry reductions and a posteriori finite element error estimators for bifurcation problems*, Int. J. Bifurcation and Chaos., 15 (2005), pp. 2091–2107.

[10] S. Costiner and S. Ta'asan, *Adaptive multigrid techniques for large-scale eigenvalue problems: Solutions of the Schrödinger problem in two and three dimensions*, Phys. Rev. E, 51 (1995), pp. 3704–3717.

[11] S. Costiner and S. Ta'asan, *Simultaneous multigrid techniques for nonlinear eigenvalue problems: Solutions of the nonlinear Schrödinger-Poisson eigenvalue problem in two and three dimensions*, Phys. Rev. E, 52 (1995), pp. 1181–1192.
[12] E. J. Doedel and H. Sharifi, *Collocation methods for continuation problems in nonlinear elliptic PDEs*, Notes Numer. Fluid Mech., 74 (2000), pp. 105–118.
[13] F. A. Dul, *MINRES and MINERR are better than SYMMLQ in eigenpair computations*, SIAM J. Sci. Comput., 19 (1998), pp. 1767–1782.
[14] A. I. Fedoseyev, M. J. Friedman, and E. J. Kansa, *Continuation for nonlinear elliptic partial differential equations discretized by the multiquadric method*, Int. J. Bifurcation and Chaos, 10 (2000), pp. 481–492.
[15] R. Glowinski, H. B. Keller, and L. Reinhart, *Continuation-conjugate gradient methods for the least squares solution of nonlinear boundary value problems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 793–832.
[16] R. Illner, O. Kavian, and H. Lange, *Stationary solutions of quasi-linear Schrödinger-Poisson systems*, J. Differential Equations, 145 (1998), pp. 1–16.
[17] H. B. Keller, *Lectures on Numerical Methods in Bifurcation Problems*, Springer-Verlag, Berlin, 1987.
[18] P. L. Lions, *On the existence of positive solutions of semilinear elliptic equations*, SIAM Rev., 24 (1982), pp. 441–467.
[19] J.-L. Liu and W. C. Rheinboldt, *A posteriori finite element error estimators for parametrized nonlinear boundary value problems*, Numer. Funct. Anal. Optim., 17 (1996), pp. 605–637.
[20] Z. Mei, *Path following around corank-2 bifurcation points of a semilinear elliptic problem with symmetry*, Computing, 47 (1991), pp. 69–85.
[21] Z. Mei, *Solution branches of a semilinear elliptic problem at corank-2 bifurcation points*, Manuscripta Math., 75 (1992), pp. 395–409.
[22] H. D. Mittelmann and H. Weber, *Multi-grid solution of bifurcation problems*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 49–60.
[23] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[24] O. Sánchez and J. Soler, *Long-time dynamics of the Schrödinger-Poisson-Slater system*, J. Statist. Phys., 114 (2004), pp. 179–204.
[25] C. F. Smith, A. F. Peterson, and R. Mittra, *A conjugate gradient algorithm for the treatment of multiple incident electromagnetic fields*, IEEE Trans. Antennas and Propagation, 37 (1989), pp. 1490–1493.
[26] S. Ta'asan, *Multigrid methods for locating singularities in bifurcation problems*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 51–62.
[27] S.-H. Wang, *On the Evolution and Qualitative Behaviors of Bifurcation Curve for a Boundary Value Problem*, preprint, National Tsing Hua University, Hsinchu, Taiwan, 2004.
[28] H. Weber, *An efficient technique for the computation of stable bifurcation branches*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 332–348.
[29] H. Weber, *Multigrid bifurcation iteration*, SIAM J. Numer. Anal., 22 (1985), pp. 262–279.
[30] J. Xu, *A novel two-grid method for semilinear elliptic equations*, SIAM J. Sci. Comput., 15 (1994), pp. 231–237.
[31] J. Xu, *Two-grid discretization techniques for linear and nonlinear PDEs*, SIAM J. Numer. Anal., 33 (1996), pp. 1759–1777.
[32] J. Xu and A. Zhou, *A two-grid discretization scheme for eigenvalue problems*, Math. Comp., 70 (1999), pp. 17–25.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1305–1328
© 2006 Society for Industrial and Applied Mathematics

PARALLEL FULL SPACE SQP LAGRANGE–NEWTON–KRYLOV–SCHWARZ ALGORITHMS FOR PDE-CONSTRAINED OPTIMIZATION PROBLEMS*

ERNESTO E. PRUDENCIO†, RICHARD BYRD‡, AND XIAO-CHUAN CAI‡

Abstract. Optimization problems constrained by nonlinear partial differential equations have been the focus of intense research in scientific computing lately. Current methods for the parallel numerical solution of such problems involve sequential quadratic programming (SQP), with either reduced or full space approaches. In this paper we propose and investigate a class of parallel full space SQP Lagrange–Newton–Krylov–Schwarz (LNKSz) algorithms. In LNKSz, a Lagrangian functional is formed and differentiated to obtain a Karush–Kuhn–Tucker (KKT) system of nonlinear equations. An inexact Newton method with line search is then applied. At each Newton iteration the linearized KKT system is solved with a Schwarz preconditioned Krylov subspace method. We apply LNKSz to the parallel numerical solution of some boundary control problems of two-dimensional incompressible Navier–Stokes equations. Numerical results are reported for different combinations of Reynolds number, mesh size, and number of parallel processors. We also compare the application of the LNKSz method to flow control problems against the application of the Newton–Krylov–Schwarz (NKSz) method to flow simulation problems.

Key words. nonlinear partial differential equations constrained optimization, inexact Newton method, domain decomposition, Schwarz preconditioners, parallel computing, flow control, incompressible Navier–Stokes equations

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65N22, 65N55, 65Y05, 76N05, 76D55, 90C06, 90C90, 93C20

DOI. 10.1137/040602997

1. Introduction.

In this paper we describe a general framework for solving optimization problems in interaction with nonlinear partial differential equations (PDEs). The focus is on how to adapt state-of-the-art PDE solvers to the requirements of optimization methods, while allowing for an efficient parallel implementation. Our method treats the differential equations as equality constraints. In order to demonstrate its effectiveness, the problem of optimizing fluid flows modeled by incompressible Navier–Stokes equations on two-dimensional domains is considered. Among optimization problems with constraints, those constrained by nonlinear PDEs are very challenging, both mathematically and computationally. Examples of such problems are inverse, optimal design, and optimal control problems. In inverse problems some parameters of the governing equations of the system behavior are not known and must be estimated by analysis of experimental system output data [3, 34, 45]. Optimal design problems usually refer to problems where the variable is the shape of a domain and one has to find the best shape that minimizes or maximizes an objective function

*Received by the editors January 8, 2004; accepted for publication (in revised form) June 8, 2005; published electronically January 6, 2006.
http://www.siam.org/journals/sisc/27-4/60299.html

†Advanced Computations Department, Stanford Linear Accelerator Center, Menlo Park, CA 94025 (prudenci@slac.stanford.edu). The research of this author was supported in part by the National Science Foundation, CCR-0219190, ACI-0072089, and ACI-0305666.

‡Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (richard@cs.colorado.edu, cai@cs.colorado.edu). The research of the second author was supported in part by the National Science Foundation, CCR-0219190, and by Army Research Office contract DAAD19-02-1-0407. The research of the third author was supported in part by the Department of Energy, DE-FC02-01ER25479, and in part by the National Science Foundation, CCR-0219190, ACI-0072089, and ACI-0305666.

[35]. In optimal control problems one usually searches for the best feasible control variables, such as boundary values or external forces that minimize or maximize a certain system behavior, such as turbulence [4, 6, 10, 23, 43]. In this paper, we consider only boundary control problems, which refer to the control of the system through boundary conditions. In general, a control problem can be described as

$$\begin{cases} \text{Find controls } \mathbf{u} \in \mathbf{U} \text{ and states } \mathbf{s} \in \mathbf{S} \\ \text{such that } \mathcal{F}(\mathbf{s}, \mathbf{u}) \text{ is minimized,} \\ \text{subject to } \mathbf{C}(\mathbf{s}, \mathbf{u}) \geqslant \mathbf{0}. \end{cases}$$

Here $\mathbf{S}$ and $\mathbf{U}$ are called state and control spaces, respectively; the state variables $\mathbf{s}$ represent the state of the system being controlled; the control variables $\mathbf{u}$ represent the means one has to control the system; the objective or cost functional $\mathcal{F} : \mathbf{S} \times \mathbf{U} \to \mathbb{R}$ to be minimized (or maximized) represents the reason why one wants to control the system; and constraints $\mathbf{C}(\mathbf{s}, \mathbf{u}) \geqslant \mathbf{0}$ represent the system behavior and other constraints imposed on state and control variables.

In flow control problems [16, 26, 28, 29], typical state variables are velocity, pressure, temperature, and internal energy. Control variables might be defined on the boundary or inside the flow domain and can be the temperature, velocity, external force, etc. The objective can be, for instance, minimization of turbulence or maximization of mixing. The constraints are given by the physical equations that must be obeyed by the system being controlled, e.g., Navier–Stokes equations, and by eventual bounds on the state and control variables. The discipline of flow control comprises many other disciplines, e.g., fluid mechanics, nonlinear PDEs, optimization algorithms, discretization techniques, numerical methods, high-performance computing, control engineering, microelectromechanical systems, and nanoelectromechanical systems [7]. In addition, the control of fluid flows has many applications as well, e.g., internal combustion engines (efficiency and emissions), heat transfer, chemical reactors, and aerodynamic surfaces [40].

For finite dimensional equality constrained optimization problems there are two major families of techniques: reduced space sequential quadratic programming (SQP) methods and full space SQP methods. Reduced space methods have been the method of choice until recently since they require much less memory, even though many subiterations are needed to converge the outer iterations and the parallel scalability is less ideal. As more powerful computer systems with lots of memory and many processors become available, full space methods seem to be more appropriate due to their increased scalability. One such method, Lagrange–Newton–Krylov–Schur (LNKSr), was introduced in [8, 9], where four block factorization based preconditioners, as well as globalization techniques and heuristics, are proposed and tested. In this paper we replace the Schur-type preconditioner with an overlapping Schwarz method, which has a better asymptotic convergence rate and is easier to convert to a nonlinear preconditioner [13, 14, 31] for highly nonlinear problems.

The rest of the paper is organized as follows. Section 2 discusses the numerical solution of optimal control problems with equality constraints. Section 3, the core of this paper, presents the full space SQP [illegible] (LNKSz) method for the parallel numerical solution of such problems and also briefly describes some current methods that have already appeared in the literature. Section 4 presents some boundary flow control problems, which are then solved in section 5 with LNKSz. Numerical experiments are performed and analyzed for different combinations of Reynolds number, mesh size, and number of parallel processors. Final conclusions are given in section 6.

2. Optimal control with equality constraints. In this paper we focus on optimal control problems with equality constraints:

$$\left\{ \begin{array}{ll} \min\limits_{(\mathbf{s},\mathbf{u})\in\mathbf{S}\times\mathbf{U}} & \mathcal{F}(\mathbf{s},\mathbf{u}) \\ \text{subject to} & \mathbf{C}(\mathbf{s},\mathbf{u}) = \mathbf{0} \in \mathbf{Y}, \end{array} \right. \tag{2.1}$$

where $\mathbf{S}$, $\mathbf{U}$, and $\mathbf{Y}$ are normed spaces, $\mathcal{F} : \mathbf{S} \times \mathbf{U} \to \mathbb{R}$ is the objective function, $\mathbf{C} : \mathbf{S} \times \mathbf{U} \to \mathbf{Y}$ represents the constraints, and $\mathbf{C}(\cdot,\cdot) = \mathbf{0}$ is referred to as state equations or a forward problem. The Lagrangian functional $\mathcal{L} : \mathbf{S} \times \mathbf{U} \times \mathbf{Y}^* \to \mathbb{R}$ associated with (2.1) is defined by

$$\mathcal{L}(\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \equiv \mathcal{F}(\mathbf{s},\mathbf{u}) + \langle \boldsymbol{\lambda}, \mathbf{C}(\mathbf{s},\mathbf{u})\rangle_{\mathbf{Y}} \quad \forall\ (\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \in \mathbf{S} \times \mathbf{U} \times \mathbf{Y}^*, \tag{2.2}$$

where $\mathbf{Y}^*$ is the adjoint space of $\mathbf{Y}$, $\langle\cdot,\cdot\rangle_{\mathbf{Y}}$ denotes the duality pairing, and variables $\boldsymbol{\lambda}$ are called Lagrange multipliers or adjoint variables.

When the constraints are PDEs over a domain Ω, the discretization necessary for the solution of (2.1) can occur at two different points in the logical development of an algorithm. In the first case, the [illegible] (OTD) approach, one demonstrates that if $(\hat{\mathbf{s}},\hat{\mathbf{u}})$ is a (local) solution of (2.1), then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ such that $(\hat{\mathbf{s}},\hat{\mathbf{u}},\hat{\boldsymbol{\lambda}})$ is a critical point of $\mathcal{L}$. So, under sufficient smoothness assumptions, one obtains, as necessary condition for a solution, a system of equations, called the Karush–Kuhn–Tucker (KKT) or optimality system, which is then discretized, generating a finite dimensional system of nonlinear equations [27, 32, 41].

In the second case, the [illegible] (DTO) approach, one begins by creating a mesh Ω_h of characteristic size $h > 0$ and then discretizing problem (2.1), obtaining a finite dimensional equality constrained optimization problem with $\mathbf{S} = \mathbb{R}^{n_{s,h}}$, $\mathbf{U} = \mathbb{R}^{n_{u,h}}$, and $\mathbf{Y} = \mathbb{R}^{m_h} = \mathbf{Y}^*$. Under sufficient smoothness conditions the KKT system becomes $\nabla\mathcal{L}_h(\hat{\mathbf{s}},\hat{\mathbf{u}},\hat{\boldsymbol{\lambda}}) = \mathbf{0} \in \mathbb{R}^{n_{s,h}+n_{u,h}+m_h}$. The theory for finite dimensional constrained optimization problems guarantees, under appropriate assumptions, the existence of such Lagrange multipliers $\hat{\boldsymbol{\lambda}}$. It should be pointed out that the discrete KKT systems from both approaches are [illegible] necessarily the same.

No approach, however, has a clear advantage over the other [26]. In both cases, the derivation of the Lagrangian w.r.t. state (control, Lagrange multiplier) variables results in the adjoint (control, state, respectively) equations. OTD usually demands the weak formulation of the state PDEs and allows the use of different meshes and techniques (e.g., boundary conditions for infinite domains, or shock-wave capturing schemes for the case of flows) for the state and adjoint equations. DTO, on the other hand, guarantees a consistency between the gradients of the objective function and of the Lagrangian functional and is also suited for the use of automatic differentiation.

From now on, we work only with the DTO approach. For simplicity, let us omit the symbols "h" and "$\hat{\ }$" and use the notation $N \equiv n_s + n_u + m$ and $\mathbf{X} \equiv (\mathbf{x},\boldsymbol{\lambda}) \equiv (\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \in \mathbb{R}^N$. The KKT system becomes

$$\mathbf{F}(\mathbf{X}) \equiv \begin{pmatrix} \nabla_{\mathbf{x}}\mathcal{L} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L} \end{pmatrix} = \begin{pmatrix} \nabla\mathcal{F} + \boldsymbol{\nabla}\mathbf{C}^T\boldsymbol{\lambda} \\ \mathbf{C}(\mathbf{s},\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \nabla_{\mathbf{s}}\mathcal{F} + \boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^T\boldsymbol{\lambda} \\ \nabla_{\mathbf{u}}\mathcal{F} + \boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C}^T\boldsymbol{\lambda} \\ \mathbf{C}(\mathbf{s},\mathbf{u}) \end{pmatrix} = \mathbf{0}, \tag{2.3}$$

where $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$, $\nabla_{\mathbf{x}}\mathcal{L}$ denotes the gradient of $\mathcal{L}$ w.r.t. state and control variables, with similar meaning holding for $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$, $\nabla_{\mathbf{s}}\mathcal{F}$, and $\nabla_{\mathbf{u}}\mathcal{F}$, while $\boldsymbol{\nabla}\mathbf{C}$ denotes the Jacobian of $\mathbf{C}$ and $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ and $\boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C}$ denote the Jacobian of $\mathbf{C}$ w.r.t. state and control variables, respectively. In (2.3), we refer to $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ as the linearized forward operator.

System (2.3) can be solved with an inexact Newton method [21, 22]. Given an initial guess $\mathbf{X}^{(0)}$, at each iteration $k = 0, 1, 2, \ldots$, an approximate solution

$$\mathbf{p}^{(k)} \equiv \left(\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)}\right) \equiv \left(\mathbf{p}_{\mathbf{s}}^{(k)}, \mathbf{p}_{\mathbf{u}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)}\right)$$

of the linearized KKT system

$$\mathbf{K}^{(k)}\ \mathbf{p}^{(k)} \ = \ -\mathbf{F}^{(k)} \tag{2.4}$$

is computed, where $\mathbf{K}^{(k)} = \boldsymbol{\nabla}\mathbf{F}(\mathbf{X}^{(k)})$ and $\mathbf{F}^{(k)} = \mathbf{F}(\mathbf{X}^{(k)})$. The KKT matrix $\mathbf{K}^{(k)}$ is the transpose of the Hessian of the Lagrangian $\mathcal{L}$, is symmetric indefinite under sufficient smoothness assumptions, and can be computed by a finite difference approximation. System (2.4) can also be written as (all terms should be understood as evaluated at $\mathbf{X}^{(k)}$)

$$\begin{bmatrix} \boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)} & \boldsymbol{\nabla}\mathbf{C}^{(k)^T} \\ \boldsymbol{\nabla}\mathbf{C}^{(k)} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{p}_{\mathbf{x}}^{(k)} \\ \mathbf{p}_{\boldsymbol{\lambda}}^{(k)} \end{pmatrix} = - \begin{pmatrix} \nabla_{\mathbf{x}}\mathcal{L}^{(k)} \\ \mathbf{C}^{(k)} \end{pmatrix}.$$

Thus, if $\boldsymbol{\nabla}\mathbf{C}^{(k)}$ has full rank and $\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}$ is positive definite in the tangent space of the constraints (i.e., $\mathbf{d}^T[\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}]\mathbf{d} > 0$ for all $\mathbf{d} \neq \mathbf{0}$ such that $[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{d} = \mathbf{0}$), then we can interpret $(\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ as being the [illegible] solution and Lagrange multipliers of

$$\begin{cases} \min\limits_{\mathbf{p_x} \in \mathbf{S}\times\mathbf{U}} & \frac{1}{2}\mathbf{p}_{\mathbf{x}}^T[\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}]\mathbf{p}_{\mathbf{x}} + \nabla_{\mathbf{x}}\mathcal{L}^{(k)^T}\mathbf{p}_{\mathbf{x}} \\ \text{subject to} & \boldsymbol{\nabla}\mathbf{C}^{(k)}\mathbf{p}_{\mathbf{x}} + \mathbf{C}^{(k)} = \mathbf{0} \in \mathbf{Y}. \end{cases}$$

This interpretation justifies the use of the terminology [illegible] (SQP) for methods involving (2.4); see [36]. Alternatively, one can interpret $(\mathbf{p}_{\mathbf{x}}^{(k)}, \boldsymbol{\lambda}^{(k)} + \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ as the solution and Lagrange multipliers for the quadratic programming (QP) using, in its objective function, $\nabla\mathcal{F}^{(k)}$ instead of $\nabla_{\mathbf{x}}\mathcal{L}^{(k)}$.

After approximately solving (2.4), one may use a globalization method such as [illegible] or [illegible] [44]. In this study we focus on a line search approach, where the next iterate is $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}$ and the step length $\alpha^{(k)}$ is selected by backtracking until the sufficient decrease condition

$$\phi(\mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}) \leqslant \phi(\mathbf{X}^{(k)}) + \alpha^{(k)}c_1\nabla\phi(\mathbf{X}^{(k)})^T\mathbf{p}^{(k)} \tag{2.5}$$

is satisfied. Here ϕ is a merit function, and c_1 is a constant satisfying $0 < c_1 < 1/2$. For constrained optimization, merit functions such as l_1 or augmented Lagrangian [5, 33] are most commonly used. In contrast to the standard merit function $\|\mathbf{F}(\mathbf{X})\|_2^2/2$, which is commonly used for systems of nonlinear equations, these merit functions try to balance the sometimes conflicting goals of reducing the objective function and satisfying the constraints [36]. We use the augmented Lagrangian $\phi_{AL} : \mathbb{R}^N \to \mathbb{R}$ in our experiments in this paper. Given a penalty parameter $\rho > 0$, it is defined by

$$\phi_{AL}(\mathbf{X}; \rho) = \mathcal{L}(\mathbf{s}, \mathbf{u}, \boldsymbol{\lambda}) + \frac{\rho}{2}\ \|\mathbf{C}(\mathbf{s}, \mathbf{u})\|_2^2 \quad \forall\ \mathbf{X} = (\mathbf{s}, \mathbf{u}, \boldsymbol{\lambda}) \in \mathbf{S} \times \mathbf{U} \times \mathbf{Y}.$$

At iteration k, $\rho = \rho^{(k)}$ must be such that we obtain descent directions $\mathbf{p}^{(k)}$, i.e.,

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)}; \rho^{(k)}))^T\mathbf{p}^{(k)} = (\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)} + \rho^{(k)}\ \mathbf{C}^{(k)^\mathbf{T}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0. \tag{2.6}$$

Since $\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} = -\|\mathbf{C}^{(k)}\|_2^2$ for an exact step, it is reasonable to expect

$$\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0 \tag{2.7}$$

for approximate steps if $\mathbf{C}^{(k)} \neq \mathbf{0}$ and the tolerances for the Krylov subspace method are small enough and the preconditioner is good enough to guarantee that the Krylov subspace method does not stop by achieving the maximum allowed number of iterations. If (2.7) does not hold, we can continue the Krylov iterations, with eventual smaller tolerances, until it does. Once (2.7) holds, we then use a fairly common strategy where we demand that $\rho^{(k)}$ satisfy

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)};\rho^{(k)}))^T\mathbf{p}^{(k)} \leqslant \frac{\rho^{(k)}}{2}\ \mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)},$$

that is,

$$\rho^{(k)} \geqslant \overline{\rho}^{(k)} = -2\frac{(\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)}}{\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)}}.$$

We then choose

$$\rho^{(k)} = \max\{\overline{\rho}^{(k)}, \rho^{(k-1)}\},$$

where $\rho^{(-1)} > 0$ is a given positive value.

However, if $\mathbf{C}^{(k)} = \mathbf{0}$, then there is no way to guarantee a descent direction. This is a fundamental issue with line search methods. Some algorithms handle this by modifying the Hessian to make it positive definite on the null space of $\boldsymbol{\nabla}\mathbf{C}^{(k)}$, but for problems of the size we are considering there is no efficient way to check positive definiteness.

In all tests described in this paper, (2.7) held for every step generated, as long as we made the absolute Krylov tolerance small enough that at least one Krylov iteration was performed. In addition, it is worth noting that, in each run, the value of $\rho^{(k)}$ became fixed before 60% of the iterations had been made. Thus the heuristic merit parameter updating strategy described above appeared to work well for the examples of this paper.

3. Parallel full space SQP Lagrange–Newton–Krylov–Schwarz. In this section we briefly review some existing techniques and then present a new parallel algorithm based on a Newton method with line search and an overlapping Schwarz preconditioner for solving (2.1). The new algorithm will be referred to as the Lagrange–Newton–Krylov–Schwarz (LNKSz) algorithm. All these methods differentiate themselves by the way (2.4) is solved.

3.1. Brief review of existing methods. In this subsection we assume that the number of constraints is equal to the number of state variables, i.e.,

$$m = n_s, \tag{3.1}$$

in (2.1); that is, $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ is a square matrix of dimension m. Situation (3.1) usually happens after the discretization of PDE-constrained optimization problems that involve well-posed boundary value problems as constraints. To explain the existing techniques

[8, 9] we need the following block-LU factorization:

$$\mathbf{K}^{(k)} = \begin{bmatrix} \nabla^2_{\mathbf{ss}}\mathcal{L}\nabla_{\mathbf{s}}\mathbf{C}^{-1} & \mathbf{0} & \mathbf{I} \\ \nabla^2_{\mathbf{su}}\mathcal{L}\nabla_{\mathbf{s}}\mathbf{C}^{-1} & \mathbf{I} & \mathbf{L}^T \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \nabla_{\mathbf{s}}\mathbf{C} & \nabla_{\mathbf{u}}\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & \nabla_{\mathbf{s}}\mathbf{C}^T \end{bmatrix}, \tag{3.2}$$

where $\mathbf{H}_1 = \nabla^2_{\mathbf{uu}}\mathcal{L} - \mathbf{L}^T\mathbf{H}_2 - \nabla^2_{\mathbf{su}}\mathcal{L}\mathbf{L}$, $\mathbf{H}_2 = \nabla^2_{\mathbf{us}}\mathcal{L} - \nabla^2_{\mathbf{ss}}\mathcal{L}\mathbf{L}$, and $\mathbf{L} = \nabla_{\mathbf{s}}\mathbf{C}^{-1}\nabla_{\mathbf{u}}\mathbf{C}$. The matrix $\mathbf{H}_1$ is the Schur complement for $\mathbf{p}^{(k)}_{\mathbf{u}}$ and is called the reduced Hessian of the Lagrangian.

Full space SQP methods solve the entire system of linear equations (2.4) simultaneously, while reduced space SQP methods divide the problem into smaller ones, first solving for the n_u control variables only and then solving for state variables and Lagrange multipliers. Newton reduced space SQP (N-RSQP) methods basically correspond to the block-LU factorization (3.2). Quasi-Newton reduced space SQP (QN-RSQP) methods drop second order terms from the right-hand side of the equations in N-RSQP and also substitute $\mathbf{H}_1$ by quasi-Newton approximations, usually via BFGS [36].

Reduced methods have the advantage of requiring less memory. However, N-RSQP demands too many solutions of systems of m linear equations per outer iteration, while in QN-RSQP the number of outer iterations tends to increase too much as n_u grows. Full space methods try to overcome the lack of parallel scalability of reduced approaches, but they also present difficulties, not only because a descent direction is not guaranteed but also due to the linearized KKT system: it is more than twice the size of the forward problem, usually indefinite (a property known to slow down Krylov solvers), and very ill conditioned. One such method, Lagrange–Newton–Krylov–Schur (LNKSr), was presented in [8, 9], where four preconditioners based on the block factorization (3.2), as well as globalization techniques and heuristics (such as maintaining a BFGS approximation of $\mathbf{H}_1$ and using a QN-RSQP step in case the original LNKSr computed step fails), are proposed and tested. LNKSr attempts to transform the problem of finding a good preconditioner for the KKT matrix to the problem of finding a good preconditioner for the linearized forward operator.

3.2. Lagrange–Newton–Krylov–Schwarz. As mentioned earlier, the key element of a successful full space approach is the preconditioning of the Jacobian of the KKT system, which is indefinite and extremely ill conditioned. A good preconditioner has to be able to substantially reduce the condition number and, at the same time, provide good scalability so that the potential of massively parallel computers can be realized. The Schur complement preconditioner used in LNKSr is an [illegible] type of technique, in which a sequential block elimination step is needed to form the Schur complement w.r.t. the control variable. In contrast to operator-splitting, Schwarz-type preconditioners are [illegible] in the sense that all variables are treated equally and the partition is based completely on the physical domain Ω. Because there is no need to eliminate any variables from the system, there is one fewer sequential step in the preconditioning process. Another advantage of LNKSz method is that it does not demand assumption (3.1). With LNKSz we can, for instance, deal directly with full [30] boundary control problems, where an equation like (4.6) is explicitly added to the constraints.

Schwarz preconditioners can be used in one-level or multilevel approaches and, in each case, with a combination of additive and/or multiplicative algorithms [39, 42]. In this paper we deal only with one-level additive algorithms [20]. Let $\Omega \subset \mathbb{R}^2$ be a bounded open domain on which the control problem is defined. We consider only

simple box domains with uniform mesh of characteristic size h here. To obtain the overlapping partition, we first partition the domain into nonoverlapping subdomains Ω_j^0, $j = 1, \ldots, N_s$. Then we extend each subdomain Ω_j^0 to a larger subdomain Ω_j^δ, i.e., $\Omega_j^0 \subset \Omega_j^\delta$. Here $\delta > 0$ indicates the size of the overlap. Only simple box decomposition is considered in this paper; i.e., all the subdomains Ω_j^0 and Ω_j^δ are rectangular and made up of an integral number of fine mesh cells. For boundary subdomains, we simply cut off the part that is outside Ω. Let $H > 0$ denote the characteristic diameter of subdomains Ω_j. Let N, N_j^0, and N_j denote the number of degrees of freedom associated with Ω, Ω_j^0, and Ω_j^δ, respectively. Let $\mathbf{K}$ be an $N \times N$ matrix of a linear system that needs to be solved during the numerical solution process of the differential problem. Let d indicate the degree of freedom per mesh point. For simplicity let us assume that d is the same throughout the entire mesh. We define the $N_j \times N$ matrix $\mathbf{R}_j^\delta$ as follows: Its $d \times d$ block element $(\mathbf{R}_j^\delta)_{\ell_1,\ell_2}$ is either (a) an identity block if the integer indices $1 \leqslant \ell_1 \leqslant N_j/d$ and $1 \leqslant \ell_2 \leqslant N/d$ are related to the same mesh point and this mesh point belongs to $\Omega_j^{'}$ or (b) a zero block otherwise. The multiplication of $\mathbf{R}_j^\delta$ with an $N \times 1$ vector generates a smaller $N_j \times 1$ vector by discarding all components corresponding to mesh points outside $\Omega_j^{'}$. The $N_j \times N$ matrix $\mathbf{R}_j^0$ is similarly defined, with the difference that its application to an $N \times 1$ vector also zeros all those components corresponding to mesh points on $\Omega_j^{'} \setminus \Omega_j$. We denote by $\mathbf{K}_j$ the $N_j \times N_j$ subdomain matrix given by

$$\mathbf{K}_j = \mathbf{R}_j^\delta \ \mathbf{K} \ \left(\mathbf{R}_j^\delta\right)^T.$$

Let $\mathbf{B}_j^{-1}$ be either an inverse of $\mathbf{K}_j$ or a preconditioner for $\mathbf{K}_j$. The classical one-level additive Schwarz preconditioner $\mathbf{B}_{asm}^{-1}$ for $\mathbf{K}$ is defined as

$$\mathbf{B}_{asm}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^\delta\right)^T \mathbf{B}_j{}^{-1} \mathbf{R}_j^\delta.$$

In addition to this standard additive Schwarz method (ASM) described above, we also consider the newly introduced restricted version (RAS) of the method [11, 15]. For some applications, the restricted version requires less communication time since one of the restriction or extension operations does not involve any overlap. The RAS preconditioner is defined as

$$\mathbf{B}_{ras}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^\delta\right)^T \mathbf{B}_j{}^{-1} \mathbf{R}_j^0.$$

Some numerical comparisons of the ASM and RAS are presented later in the paper.

When the Schwarz preconditioner is applied to symmetric positive definite systems arising from the discretization of elliptical problems defined in $H_0^1(\Omega)$, the condition number κ of the preconditioned system satisfies $\kappa \leqslant C\,(1 + H/\delta)\,/H^2$, where C is independent of h, H, δ, and the shapes of Ω and Ω_j^δ [39]; that is, a Schwarz preconditioned Krylov subspace method is expected to have the following properties:

(3.3) The number of iterations grows approximately proportional to $1/H$;

(3.4) If δ is maintained proportional to H, the number of iterations is bounded independently of h and H/h (a parameter related to the number of degrees of freedom of each subproblem);

(3.5) The convergence gets better as δ is increased.

Theoretically, results (3.3)–(3.5) may not be applied immediately to Krylov subspace methods, e.g., GMRES [24, 38], for the solution of indefinite linearized KKT systems. Nonetheless, we carefully examine all the properties in our numerical experiments. In particular, let $\bar{l}$ be the average number of Schwarz preconditioned GMRES iterations per linearized KKT system. We look for the following scalability properties:

(3.6) For fixed h and δ, $\bar{l}$ increases as H decreases;

(3.7) For fixed H and δ, $\bar{l}$ is not very sensitive to the mesh refinement;

(3.8) For fixed h and H, $\bar{l}$ decreases as δ increases.

4. Boundary control of incompressible Navier–Stokes flows. In this section we discuss the boundary control of the two-dimensional steady-state incompressible Navier–Stokes equations in the velocity-vorticity formulation. The velocity is denoted by $\mathbf{v} = (v_1, v_2)$ and the vorticity by ω. Let Ω be an open and bounded polygonal domain in the plane, with $\Gamma = \partial\Omega$ its boundary and $\boldsymbol{\nu}$ the unit outward normal vector along Γ. Let $\mathbf{f}$ be a given external force defined in Ω. An example of a flow simulation problem consists of finding (v_1, v_2, ω) such that

$$
\left\{
\begin{array}{lcll}
-\Delta v_1 - \dfrac{\partial \omega}{\partial x_2} & = & 0 & \text{in } \Omega, \\
-\Delta v_2 + \dfrac{\partial \omega}{\partial x_1} & = & 0 & \text{in } \Omega, \\
-\Delta \omega + Re\ v_1 \dfrac{\partial \omega}{\partial x_1} + Re\ v_2 \dfrac{\partial \omega}{\partial x_2} - Re\ \text{curl}\ \mathbf{f} & = & 0 & \text{in } \Omega, \\
\mathbf{v} - \mathbf{v}_D & = & \mathbf{0} & \text{on } \Gamma, \\
\omega + \dfrac{\partial v_1}{\partial x_2} - \dfrac{\partial v_2}{\partial x_1} & = & 0 & \text{on } \Gamma,
\end{array}
\right. \tag{4.1}
$$

where Re is the Reynolds number, $\mathbf{v}_D$ is a prescribed boundary velocity satisfying $\int_\Gamma \mathbf{v}_D \cdot \boldsymbol{\nu}\, d\Gamma = 0$, and $\text{curl}\ \mathbf{f} = -\frac{\partial f_1}{\partial x_2} + \frac{\partial f_2}{\partial x_1}$. If $\mathbf{v} \in C^3(\Omega) \times C^3(\Omega)$, then [18]

$$
\omega + \frac{\partial v_1}{\partial x_2} - \frac{\partial v_2}{\partial x_1} = 0 \quad \text{in } \Omega \tag{4.2}
$$

and

$$
\nabla \cdot \mathbf{v} = 0 \quad \text{in } \Omega. \tag{4.3}
$$

These two equations show the consistency of the two-dimensional velocity-vorticity formulation of incompressible flows, since (4.2) is the formal definition of the vorticity and (4.3) is related to the physical law of mass conservation.

An example of a boundary control problem consists of finding $(v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$
\min_{(\mathbf{s},\mathbf{u}) \in \mathbf{S} \times \mathbf{U}} \mathcal{F}(\mathbf{s}, \mathbf{u}) = \frac{1}{2} \int_\Omega \omega^2\, d\Omega + \frac{c}{2} \int_{\Gamma_u} \|\mathbf{u}\|_2^2\, d\Gamma \tag{4.4}
$$

is achieved subject to the constraints

$$
(4.5)\quad \begin{cases}
-\Delta v_1 - \dfrac{\partial \omega}{\partial x_2} = 0 & \text{in } \Omega,\\
-\Delta v_2 + \dfrac{\partial \omega}{\partial x_1} = 0 & \text{in } \Omega,\\
-\Delta \omega + Re\ v_1 \dfrac{\partial \omega}{\partial x_1} + Re\ v_2 \dfrac{\partial \omega}{\partial x_2} - Re\ \text{curl}\ \mathbf{f} = 0 & \text{in } \Omega,\\
v_i - v_{D,i} = 0 & \text{on } \Gamma_{D,i}, \quad i = 1,2,\\
\dfrac{\partial v_i}{\partial \boldsymbol{\nu}} - v_{N,i} = 0 & \text{on } \Gamma_{N,i}, \quad i = 1,2,\\
\mathbf{v} - \mathbf{u} = \mathbf{0} & \text{on } \Gamma_u,\\
\omega + \dfrac{\partial v_1}{\partial x_2} - \dfrac{\partial v_2}{\partial x_1} = 0 & \text{on } \Gamma,\\
\displaystyle\int_\Gamma \mathbf{v}\cdot\boldsymbol{\nu}\, d\Gamma = 0, &
\end{cases}
$$

where, for $i = 1, 2$, $\Gamma = \Gamma_{D,i} \cup \Gamma_{N,i} \cup \Gamma_u$, $\Gamma_{D,i}$ ($\Gamma_{N,i}$) is the part of the boundary where the v_i velocity component is specified through a Dirichlet (Neumann) condition with a prescribed velocity $v_{D,i}$, and Γ_u is the part of the boundary where the velocity is specified through a Dirichlet condition with a control velocity $\mathbf{u}$. The positive constant parameter c is used to adjust the relative importance of the control norms in achieving the minimization, thus indirectly constraining the size of those norms. The physical objective behind problem (4.4)–(4.5) is the minimization of turbulence [1, 7, 30]. The last constraint, given by

$$
(4.6)\qquad \int_\Gamma \mathbf{v}\cdot\boldsymbol{\nu}\, d\Gamma = 0,
$$

is necessary for the consistency with the physical law of mass conservation. Therefore, the control $\mathbf{u} = (u_1, u_2)$ cannot be any control: it must belong to the space of functions satisfying (4.6). We denote problems like (4.4)–(4.5), where controls are allowed to assume nonzero normal values at the boundary, as [illegible] boundary control problems (BCPs) [30]. In these kinds of problems assumption (3.1) is not valid, due to the extra constraint (4.6). This fact also complicates the parallel finite differences approximation of Jacobian matrices, since one does not have only PDEs (i.e., equations with local behavior) anymore. The integral condition (3.1) couples nonneighboring mesh points. An alternative formulation, which compromises between the physical law of mass conservation and the complexity of parallel Jacobian approximations, eliminates the explicit constraint (4.6), thus making (3.1) valid, but adds to the objective function the term $\tilde{c}\left[\int_\Gamma \mathbf{v}\cdot\boldsymbol{\nu}\, d\Gamma\right]^2/2$, with $\tilde{c} \gg 1$ [9].

One can also deal with [illegible] BCPs, where the control is allowed to be just tangential to the boundary (i.e., $\mathbf{u}\cdot\boldsymbol{\nu} = 0$ on Γ_u) and the velocity $\mathbf{v}$ is assumed to satisfy $\int_{\Gamma\setminus\Gamma_u} \mathbf{v}\cdot\boldsymbol{\nu}\, d\Gamma = 0$. Thus, (3.1) is valid. Since tangential BCPs restrict even more the space where the control $\mathbf{u} = (u_1, u_2)$ can exist, one naturally expects better objective function values with full BCPs. In this paper we study only tangential boundary control problems.

5. Numerical experiments. In our numerical experiments we deal with two-dimensional rectangular domains $\Omega = (0, L_1) \times (0, L_2)$, $L_2 \leqslant L_1$. All notation related to the geometry of the computational domain is depicted in Figure 5.1. In particular,

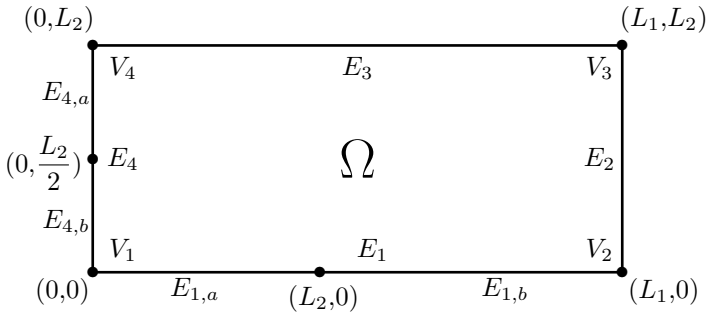


FIG. 5.1. *General rectangular domain* $\Omega = (0, L_1) \times (0, L_2)$*, with* $L_2 \leqslant L_1$*, involved in our numerical experiments.*

we define $E_{1,a} = \{(x_1, x_2) \in E_1 : 0 < x_1 \leqslant L_2\}$, $E_{1,b} = E_1 \setminus E_{1,a}$, $E_{4,a} = \{(x_1, x_2) \in E_4 : \dfrac{L_2}{2} \leqslant x_2 < L_2\}$, and $E_{4,b} = E_4 \setminus E_{4,a}$.

5.1. Flow problems. In order to simplify the implementation, we rewrite (4.4)–(4.5), defining the control $\mathbf{u}$ everywhere on Γ, but not necessarily allowing it to vary everywhere. We consider two flow problems: cavity flow and backward-facing step flow problems. In each case we describe a simulation problem and a tangential BCP. Also, we describe only the boundary conditions on the velocity and the constraints on the boundary controls, since all problems have in common the three differential equations in Ω and the boundary condition on ω that appear in (4.1). All control problems seek the minimization of the objective function (4.4) with $c = 10^{-2}$.

Lid-driven cavity flow. In this case, $L_1 = L_2 = 1$ and (see Figure 5.2(a))

$$\mathbf{f} = \left(-\sin^2(\pi x_1)\cos(\pi x_2)\sin^2(\pi x_2),\ \sin^2(\pi x_2)\cos(\pi x_1)\sin^2(\pi x_1)\right).$$

The simulation problem assumes slip rigid walls on Γ:

$$\begin{cases} \mathbf{v} \cdot \boldsymbol{\nu} &= 0 & \text{on } \Gamma, \\ \dfrac{\partial \mathbf{v}}{\partial \boldsymbol{\nu}} &= \mathbf{0} & \text{on } \Gamma. \end{cases} \tag{5.1}$$

The tangential BCP has $\Gamma_u = \Gamma$ and velocity and control constraints given by

$$\begin{cases} \mathbf{v} - \mathbf{u} &= \mathbf{0} & \text{on } \Gamma_u, \\ \mathbf{u} \cdot \boldsymbol{\nu} &= 0 & \text{on } \Gamma_u. \end{cases} \tag{5.2}$$

Backward-facing step flow. In this case, $L_1 = 6$, $L_2 = 1$, and $\mathbf{f} = \mathbf{0}$. We define $v_{\text{in}}(x_1, x_2) = 8(1 - x_2)(x_2 - \frac{1}{2})$ and $v_{\text{out}}(x_1, x_2) = x_2(1 - x_2)$. See Figures 5.3 and 5.6. The simulation problem has velocity boundary conditions given by

$$\begin{cases} v_1 &= 0 & \text{on } \{V_1\} \cup E_{1,b} \cup \overline{E}_3 \cup E_{4,b}, \\ v_1 &= v_{\text{in}} & \text{on } E_{4,a}, \\ v_1 &= v_{\text{out}} & \text{on } E_2, \\ \dfrac{\partial v_1}{\partial \boldsymbol{\nu}} &= 0 & \text{on } E_{1,a}, \\ v_2 &= 0 & \text{on } \Gamma \setminus E_{4,b}, \\ \dfrac{\partial v_2}{\partial \boldsymbol{\nu}} &= 0 & \text{on } E_{4,b}. \end{cases} \tag{5.3}$$

The tangential BCP has $\Gamma_u = E_{4,b} \cup \{V_1\} \cup E_{1,a}$ and velocity and control constraints given by

$$\text{(5.4)} \qquad \begin{cases} \mathbf{v} - \mathbf{u} &= \mathbf{0} & \text{on } \Gamma_u, \\ \mathbf{u} \cdot \boldsymbol{\nu} &= 0 & \text{on } \Gamma_u, \\ v_1 &= 0 & \text{on } E_{1,b} \cup \overline{E}_3, \\ v_1 &= v_{\text{in}} & \text{on } E_{4,a}, \\ v_1 &= v_{\text{out}} & \text{on } E_2, \\ v_2 &= 0 & \text{on } \Gamma \setminus \Gamma_u. \end{cases}$$

We run flow problems for $Re = 200$ and $Re = 300$. In this range of Reynolds numbers the flow is known to be steady and two-dimensional [30].

5.2. Details of numerical approaches. To discretize the flow problems, we use a five-point second order finite difference method on a uniform mesh.[1] All derivative terms of interior PDEs are discretized with a second order central difference scheme. The boundary condition $\omega + \partial v_1/\partial x_2 - \partial v_2/\partial x_1 = 0$ on Γ is also discretized with a second order approximation, using, however, only mesh points adjacent to the boundary, by applying (4.2) at these points also. This combination of second order approximation with only mesh points adjacent to the boundary is good for parallel performance reasons. We show how to achieve this combination for the boundary mesh points on edge E_1 and at corner V_1. Equations for the other boundary regions are similarly deduced. Let h denote the uniform mesh spacing. Given a mesh point, let subindices r, l, a, and b denote mesh points located, respectively, to the right of, to the left of, above, and below this given mesh point. The meanings for subindices ar, br, bl, al, rr, ll, aa, bb, etc. are straightforward. For mesh points located on edge E_1 we know that

$$w = -\frac{-3v_1 + 4v_{1,a} - v_{1,aa}}{2h} + \frac{v_{2,r} - v_{2,l}}{2h} + O(h^2),$$

$$w_a = -\frac{v_{1,aa} - v_1}{2h} + \frac{v_{2,ar} - v_{2,al}}{2h} + O(h^2).$$

Thus, taking the sum of the expressions, $v_{1,aa}$ is cancelled:

$$w + w_a + 2\frac{v_{1,a} - v_1}{h} - \frac{v_{2,ar} - v_{2,al}}{2h} - \frac{v_{2,r} - v_{2,l}}{2h} = O(h^2).$$

Using a similar technique, we obtain, for corner V_1,

$$w + w_a + w_r + w_{ar} + 2\frac{v_{1,a} - v_1}{h} + 2\frac{v_{1,ar} - v_{1,r}}{h} - 2\frac{v_{2,r} - v_2}{h} - 2\frac{v_{2,ar} - v_{2,a}}{h} = O(h^2).$$

To form an algebraic system of nonlinear equations from the finite difference equations, we need to order the unknowns and the corresponding functions. The unknowns are ordered mesh point by mesh point, in contrast to physical variable by physical variable as required by all SQP methods related to the matrix structure (3.2). At each mesh point the unknowns are ordered in the order of $v_1, v_2, \omega, u_1, u_2, \lambda_1, \lambda_2$, and λ_3. The mesh points are ordered subdomain by subdomain, for the purpose of

[1] Since our research focus in this paper is the iterative method LNKSz, we use a simple nonstaggered mesh. Another approach is to use staggered meshes in order to numerically better preserve the physical conditions (4.2) and (4.3) [18, 25, 37].

parallel processing. The ordering of the subdomains is not important since we use an additive method whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting during the sparse LU method (used in our experiments), the corresponding functions are ordered as $\nabla_{\lambda_1}\mathcal{L}$,$\nabla_{\lambda_2}\mathcal{L}$,$\nabla_{\lambda_3}\mathcal{L}$, $\nabla_{u_1}\mathcal{L}$,$\nabla_{u_2}\mathcal{L}$, $\nabla_{v_1}\mathcal{L}$,$\nabla_{v_2}\mathcal{L}$, and $\nabla_{\omega}\mathcal{L}$. Because the orderings for the unknowns and for the function components are different, the Jacobian matrix becomes nonsymmetric and so we use GMRES.

Some of the numerical results obtained using LNKSz for tangential boundary flow control problems are compared with the corresponding simulation results obtained using NKSz [12]. Although NKSz has been well studied for flow simulation problems, little is known about its applicability and performance for solving KKT systems arising from flow control problems. The NKSz algorithm can be briefly described as follows:

$$
\left\{
\begin{array}{ll}
1. & \text{Initialize the iteration } k = 0 \text{ and } \mathbf{X}^{(0)} = \mathbf{0};\\
2. & \text{Decide whether to stop based on } \|\mathbf{F}(\mathbf{X}^{(k)})\|_2;\\
3. & \text{Approximately solve } \left[\boldsymbol{\nabla}\mathbf{F}(\mathbf{X}^{(k)})\right]\mathbf{p}^{(k)} = -\mathbf{F}(\mathbf{X}^{(k)}),\\
 & \text{using GMRES with a left Schwarz preconditioner;}\\
4. & \text{Perform line search } \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}\\
 & \text{using the standard merit function } \|\mathbf{F}\|_2^2/2;\\
5. & \text{Set } k \leftarrow k+1; \text{ go to step 2.}
\end{array}
\right. \tag{5.5}
$$

LNKSz is very similar to NKSz, except that it uses an augmented Lagrangian merit function in step 4 of (5.5). Note that Reynolds continuation is not used in any of the algorithms.

In all experiments, the Jacobian matrix is constructed approximately using a multicolored central finite difference method with step size 10^{-5} [17]. In problem (4.4)–(4.5), since there is no variable with power greater than two, central finite difference approximations of the Jacobian are exact up to roundoff errors. To solve the Jacobian systems we use restarted GMRES with an absolute tolerance equal to 10^{-6}, a relative tolerance equal to 10^{-4}, a restart parameter equal to 90, and a maximum number of iterations equal to 5,000. The GMRES tolerances are checked over preconditioned residuals. Regarding the one-level additive Schwarz preconditioner, the number of subdomains is equal to the number of processors, and the extended subdomain problems have zero Dirichlet interior boundary conditions and are solved with sparse LU. The line search with the merit function defined in section 2 is performed with cubic backtracking, with $c_1 = 10^{-4}$ in (2.5) and a minimum allowed step length $\alpha^{(k)}$ equal to 10^{-6}. For augmented Lagrangian merit functions we follow the strategy explained in section 2 with $\rho^{(-1)}$=10. For Newton iterations we use an absolute stopping tolerance equal to 10^{-6} and a relative tolerance equal to 10^{-10} times the initial residual. The maximum allowed number of Newton iterations is 100.

5.3. Results of numerical experiments. All tests were performed on a cluster of Linux PCs, and our parallel object-oriented software was developed using the C++ programming language and the Portable, Extensible Toolkit for Scientific Computing (PETSc) library [2] from Argonne National Laboratory. Our main concern is the scalability of the algorithms in terms of the linear and nonlinear iteration numbers. Computing times are also reported, but they should not be taken as a reliable measure of the scalability of the algorithms because our network is relatively slow and is shared by other processes.

Results are grouped into tables according to a unique combination of problem type (simulation or control), Reynolds number Re, ASM type (standard or restricted), overlap δ, and merit function (standard or augmented Lagrangian). Each table presents

results for nine situations, related to three different sizes of meshes (parameter h) and three different numbers of processors (parameter H). For each case we report the following:

- the total number of Newton iterations: n;
- the average number of GMRES iterations per Newton iteration: $\bar{l}$;
- the total computing time in seconds spent on all Newton iterations: t_n;
- the average computing time, in seconds, per Newton iteration, spent on solving for the Newton steps: $\bar{t}_l$.

For each table we compare the behavior of $\bar{l}$ against (3.6)–(3.8). Predictions (3.6) and (3.7) can be checked by observing the values of $\bar{l}$ in a column (fixed h and δ) and in a row (fixed H and δ), respectively. Prediction (3.8) can be checked by observing the values of $\bar{l}$ at the same situation (fixed problem type, Re, ASM type, merit function, H, and h) in different tables (different δ). We also compare approximate values of

$$\|\omega\|_h^2 = \int_{\Omega_h} \omega^2 \, d\Omega_h. \tag{5.6}$$

We show some [illegible] results, defined as follows: the base computing time is obtained on 16 processors, and the parallel efficiencies E_n (related to t_n) and $\overline{E}_l$ (related to $\bar{t}_l$) are given by the expression

$$\frac{(\text{computing time } (t_n \text{ or } \bar{t}_l) \text{ using 16 processors}) \times 16}{(\text{computing time } (t_n \text{ or } \bar{t}_l) \text{ using } N_p \text{ processors}) \times N_p}. \tag{5.7}$$

5.3.1. Cavity flow problems. Table 5.1 presents results for the simulation problem (5.1) with $Re = 200$. The preconditioner is ASM with $\delta = 1/64$ and the standard merit function is used in the line search. The total number of Newton iterations does not change with the mesh size or the number of processors. The average number of Krylov iterations per Newton iteration changes as expected in predictions (3.6) and (3.7).

TABLE 5.1

Results for the cavity flow simulation problem with $Re = 200$, standard ASM with overlap $\delta = 1/64$ and standard merit function $\|\mathbf{F}\|_2^2/2$. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 198,147 and $\|\omega\|_h^2 \approx 55.4$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 5$	$t_n \approx 0.83$	$n = 5$	$t_n \approx 3.6$	$n = 5$	$t_n \approx 18.3$
	$\bar{l} \approx 46$	$\bar{t}_l \approx 0.13$	$\bar{l} \approx 47$	$\bar{t}_l \approx 0.62$	$\bar{l} \approx 47$	$\bar{t}_l \approx 3.29$
32	$n = 5$	$t_n \approx 0.71$	$n = 5$	$t_n \approx 2.7$	$n = 5$	$t_n \approx 12.1$
	$\bar{l} \approx 60$	$\bar{t}_l \approx 0.091$	$\bar{l} \approx 63$	$\bar{t}_l \approx 0.46$	$\bar{l} \approx 63$	$\bar{t}_l \approx 2.20$
64	$n = 5$	$t_n \approx 0.70$	$n = 5$	$t_n \approx 1.99$	$n = 5$	$t_n \approx 7.50$
	$\bar{l} \approx 69$	$\bar{t}_l \approx 0.077$	$\bar{l} \approx 80$	$\bar{t}_l \approx 0.32$	$\bar{l} \approx 79$	$\bar{t}_l \approx 1.35$

The next three tables present results for the tangential BCP (5.2) with $Re = 200$. An augmented Lagrangian merit function is used in the line search. Several different overlap values are used in the ASM preconditioner, and the results are summarized as follows: Table 5.2 for $\delta = 1/64$, Table 5.3 for $\delta = 1/32$, and Table 5.4 for $\delta = 1/16$. Changes in the total number of Newton iterations w.r.t. the mesh size and the number

TABLE 5.2

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/64$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.92$	$n = 8$	$t_n \approx 52.1$	$n = 6$	$t_n \approx 238$
	$\bar{l} \approx 92$	$\bar{t}_l \approx 1.21$	$\bar{l} \approx 85$	$\bar{t}_l \approx 5.78$	$\bar{l} \approx 100$	$\bar{t}_l \approx 36.4$
32	$n = 7$	$t_n \approx 10.3$	$n = 8$	$t_n \approx 53.3$	$n = 6$	$t_n \approx 264$
	$\bar{l} \approx 208$	$\bar{t}_l \approx 1.34$	$\bar{l} \approx 204$	$\bar{t}_l \approx 6.26$	$\bar{l} \approx 272$	$\bar{t}_l \approx 42.5$
64	$n = 7$	$t_n \approx 5.39$	$n = 8$	$t_n \approx 25.6$	$n = 6$	$t_n \approx 108$
	$\bar{l} \approx 187$	$\bar{t}_l \approx 0.67$	$\bar{l} \approx 182$	$\bar{t}_l \approx 2.96$	$\bar{l} \approx 216$	$\bar{t}_l \approx 17.1$

TABLE 5.3

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/32$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6). Case "(∗)" is discussed in subsection 5.3.1.

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 8.22$	$n = 8$	$t_n \approx 49.1$	$n = 7$	$t_n \approx 254$
	$\bar{l} \approx 58$	$\bar{t}_l \approx 0.96$	$\bar{l} \approx 59$	$\bar{t}_l \approx 5.38$	$\bar{l} \approx 62$	$\bar{t}_l \approx 33.1$
32	$n = 7$	$t_n \approx 8.32$	$n = 8$	$t_n \approx 46.6$	$n = 6$	$t_n \approx 199$
	$\bar{l} \approx 119$	$\bar{t}_l \approx 1.05$	$\bar{l} \approx 130$	$\bar{t}_l \approx 5.41$	$\bar{l} \approx 140$	$\bar{t}_l \approx 31.6$
64	$n = 7$ (*)	$t_n \approx 6.21$	$n = 8$	$t_n \approx 27.6$	$n = 6$	$t_n \approx 110$
	$\bar{l} \approx 155$	$\bar{t}_l \approx 0.78$	$\bar{l} \approx 132$	$\bar{t}_l \approx 3.18$	$\bar{l} \approx 143$	$\bar{t}_l \approx 17.4$

TABLE 5.4

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.65$	$n = 8$	$t_n \approx 59.2$	$n = 7$	$t_n \approx 744$
	$\bar{l} \approx 47$	$\bar{t}_l \approx 1.16$	$\bar{l} \approx 44$	$\bar{t}_l \approx 6.62$	$\bar{l} \approx 50$	$\bar{t}_l \approx 103$
32	$n = 7$	$t_n \approx 10.8$	$n = 8$	$t_n \approx 75.1$	$n = 7$	$t_n \approx 616$
	$\bar{l} \approx 100$	$\bar{t}_l \approx 1.39$	$\bar{l} \approx 108$	$\bar{t}_l \approx 8.96$	$\bar{l} \approx 149$	$\bar{t}_l \approx 86.3$
64	$n = 6$	$t_n \approx 6.36$	$n = 8$	$t_n \approx 56.5$	$n = 7$	$t_n \approx 331$
	$\bar{l} \approx 104$	$\bar{t}_l \approx 0.94$	$\bar{l} \approx 115$	$\bar{t}_l \approx 6.79$	$\bar{l} \approx 128$	$\bar{t}_l \approx 46.4$

of processors are not pronounced. We observe that $\bar{l}$ follows (3.8). With the same δ used on the simulation problem, we can see in Table 5.2 that the average number of GMRES iterations is now more sensitive to both h and, especially, H. Table 5.3 is

the one where $\bar{l}$ best follows both (3.6) and (3.7) and the computing times t_n and $\bar{t}_l$ for the finest mesh decrease with the increase in the number of processors. Table 5.4 shows that if δ gets too big, then the consequent decrease on $\bar{l}$ might not compensate for the increased time taken by sparse LU on the larger extended subdomains; that is, $\bar{t}_l$ increases. Comparing values of $\bar{t}_l$ in Tables 5.2 and 5.3 with the values in Table 5.1, we see that the average time spent on computing $\mathbf{p}^{(k)}$ can be more than 10 times longer in control problems than in simulation problems on the same mesh, instead of being around $8/3 \approx 3$ times longer, in accordance to the ratio between the number of variables per mesh point on control problems and on simulation problems.

As reported in subsection 5.2, we use a GMRES relative tolerance of 1.0×10^{-4}, a GMRES absolute tolerance of 1.0×10^{-6}, and a Newton absolute tolerance of 1.0×10^{-6}. Case "(*)" in Table 5.3, however, gives results for a Newton absolute tolerance of 1.2×10^{-6}. When a Newton absolute tolerance of 1.0×10^{-6} is used, although full steps are accepted in some iterations, the line search stalls once $||F||_2 \approx 1.14 \times 10^{-6}$. If we change the GMRES relative tolerance to 1.0×10^{-6} and the GMRES absolute tolerance to 1.0×10^{-13} (in order to obtain a more accurate Newton step), then we achieve $||F||_2 < 1.00 \times 10^{-6}$ with $n = 6$, $\bar{l} \approx 272$, $t_n \approx 9.38$, and $\bar{t}_l \approx 1.42$. Although we performed our tests with fixed GMRES tolerances, this experiment suggests that for more demanding Newton tolerances one might need to use decreasing GMRES tolerances as the outer loop proceeds, as expected by the theory for superlinear convergence of the inexact Newton method [19, 36].

In the next two tables, we change the preconditioner to RAS and everything else stays the same; i.e., these results are for the tangential BCP (5.2) with $Re = 200$ and we use an augmented Lagrangian merit function in the line search. We increase the overlap size in the RAS preconditioner as follows: Table 5.5 for $\delta = 1/32$ and Table 5.6 for $\delta = 1/16$. The average number of GMRES iterations continues to follow (3.8) but now it better follows (3.6) and (3.7). The computing times t_n and $\bar{t}_l$ for the finest mesh in Table 5.6 decrease with the increase in the number of processors. The average number of GMRES iterations is larger in Table 5.5 than in Table 5.3, but the saving in communications of RAS compensates for this increase so that $\bar{t}_l$ does not increase proportionally. By comparing finest mesh results on Tables 5.6 and 5.4 we see that RAS performs better than the standard ASM in terms of both $\bar{l}$ and $\bar{t}_l$, resulting in a smaller t_n.

Table 5.7 presents results for the tangential BCP (5.2) with $Re = 250$, restricted ASM with $\delta = 1/16$, and augmented Lagrangian merit function. We can see that, with a Reynolds number greater than that in the previous table, both nonlinear (n) and average linear ($\bar{l}$) complexities increase.

In order to determine if tighter fixed GMRES tolerances would improve the number of Newton iterations and thus, eventually, improve the total computing time, we rerun the tests of Table 5.7 with the same parameters, except for a GMRES relative tolerance of 1.0×10^{-6}, instead of 1.0×10^{-4}, and a GMRES absolute tolerance of 1.0×10^{-13}, instead of 1.0×10^{-6}. When compared to the results in Table 5.7, at least one Newton step is saved in all nine tests, but the increase in $\bar{l}$ (and consequently $\bar{t}_l$) makes t_n increase in all cases with 32 and 64 processors. In fact, t_n decreases just for the case of 256×256 mesh and 16 processors, from 1,090 seconds to 1,040 seconds. So, although there was some improvement in the number of Newton steps, it was usually not enough to compensate for the expected increase in linear complexity, resulting in a longer overall computing time t_n.

Next, in Table 5.8 we present some parallel efficiency results derived from data presented in previous tables.

TABLE 5.5

Results for the cavity tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/32$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 8.33$	$n = 9$	$t_n \approx 53.1$	$n = 7$	$t_n \approx 253$
	$\bar{l} \approx 59$	$\bar{t}_l \approx 0.98$	$\bar{l} \approx 57$	$\bar{t}_l \approx 5.15$	$\bar{l} \approx 62$	$\bar{t}_l \approx 32.9$
32	$n = 7$	$t_n \approx 8.47$	$n = 8$	$t_n \approx 46.9$	$n = 6$	$t_n \approx 211$
	$\bar{l} \approx 131$	$\bar{t}_l \approx 1.07$	$\bar{l} \approx 134$	$\bar{t}_l \approx 5.45$	$\bar{l} \approx 154$	$\bar{t}_l \approx 33.6$
64	$n = 7$	$t_n \approx 6.38$	$n = 8$	$t_n \approx 30.1$	$n = 6$	$t_n \approx 132$
	$\bar{l} \approx 175$	$\bar{t}_l \approx 0.81$	$\bar{l} \approx 162$	$\bar{t}_l \approx 3.50$	$\bar{l} \approx 184$	$\bar{t}_l \approx 21.1$

TABLE 5.6

Results for the cavity tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.66$	$n = 8$	$t_n \approx 60.7$	$n = 7$	$t_n \approx 743$
	$\bar{l} \approx 47$	$\bar{t}_l \approx 1.15$	$\bar{l} \approx 46$	$\bar{t}_l \approx 6.80$	$\bar{l} \approx 51$	$\bar{t}_l \approx 103$
32	$n = 7$	$t_n \approx 9.49$	$n = 8$	$t_n \approx 65.5$	$n = 6$	$t_n \approx 436$
	$\bar{l} \approx 90$	$\bar{t}_l \approx 1.21$	$\bar{l} \approx 86$	$\bar{t}_l \approx 7.7$	$\bar{l} \approx 98$	$\bar{t}_l \approx 71.0$
64	$n = 7$	$t_n \approx 7.18$	$n = 9$	$t_n \approx 56.9$	$n = 7$	$t_n \approx 309$
	$\bar{l} \approx 105$	$\bar{t}_l \approx 0.91$	$\bar{l} \approx 97$	$\bar{t}_l \approx 6.1$	$\bar{l} \approx 114$	$\bar{t}_l \approx 43.3$

TABLE 5.7

Results for the cavity tangential boundary flow control problem with $Re = 250$, restricted ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 50.2$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 11$	$t_n \approx 15.8$	$n = 13$	$t_n \approx 110$	$n = 10$	$t_n \approx 1090$
	$\bar{l} \approx 51$	$\bar{t}_l \approx 1.22$	$\bar{l} \approx 55$	$\bar{t}_l \approx 7.67$	$\bar{l} \approx 55$	$\bar{t}_l \approx 106$
32	$n = 11$	$t_n \approx 17.2$	$n = 13$	$t_n \approx 126$	$n = 9$	$t_n \approx 726$
	$\bar{l} \approx 107$	$\bar{t}_l \approx 1.42$	$\bar{l} \approx 112$	$\bar{t}_l \approx 9.23$	$\bar{l} \approx 123$	$\bar{t}_l \approx 79.0$
64	$n = 10$	$t_n \approx 12.6$	$n = 13$	$t_n \approx 102$	$n = 9$	$t_n \approx 504$
	$\bar{l} \approx 135$	$\bar{t}_l \approx 1.16$	$\bar{l} \approx 139$	$\bar{t}_l \approx 7.62$	$\bar{l} \approx 180$	$\bar{t}_l \approx 55.1$

Finally, we present a comparison of computed velocity fields for $Re = 200$. Figure 5.2(b) refers to the simulation problem (5.1), and Figures 5.2(c) and 5.2(d) refer to the tangential BCP (5.2). Recall that the objective of the tangential boundary control is to minimize the intensity of the velocity field in the vortex shown in Figure 5.2(b). Figure 5.2(d) shows that the tangential boundary control acts in the opposite direction

TABLE 5.8

Efficiency analysis of NKSz and LNKSz, w.r.t. the number of processors for cavity flow problems with a 256×256 *mesh and different values of Reynolds number Re and overlap* δ. E_n *is the efficiency related to* t_n, *the total time in seconds spent on all Newton iterations.* $\overline{E}_l$ *is the efficiency related to* $\bar{t}_l$, *the average time in seconds, per Newton iteration, spent on solving for Newton steps. See* (5.7). *Results are based on values of* t_n *and* $\bar{t}_l$ *from Tables* 5.1, 5.6, *and* 5.7.

# Procs.	Simulation		Boundary control			
	$Re = 200,\ \delta = 1/64$ Standard ASM		$Re = 200,\ \delta = 1/16$ Restricted ASM		$Re = 250,\ \delta = 1/16$ Restricted ASM	
	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$
16	1.00	1.00	1.00	1.00	1.00	1.00
32	0.76	0.75	0.85	0.73	0.75	0.67
64	0.61	0.61	0.60	0.59	0.54	0.48

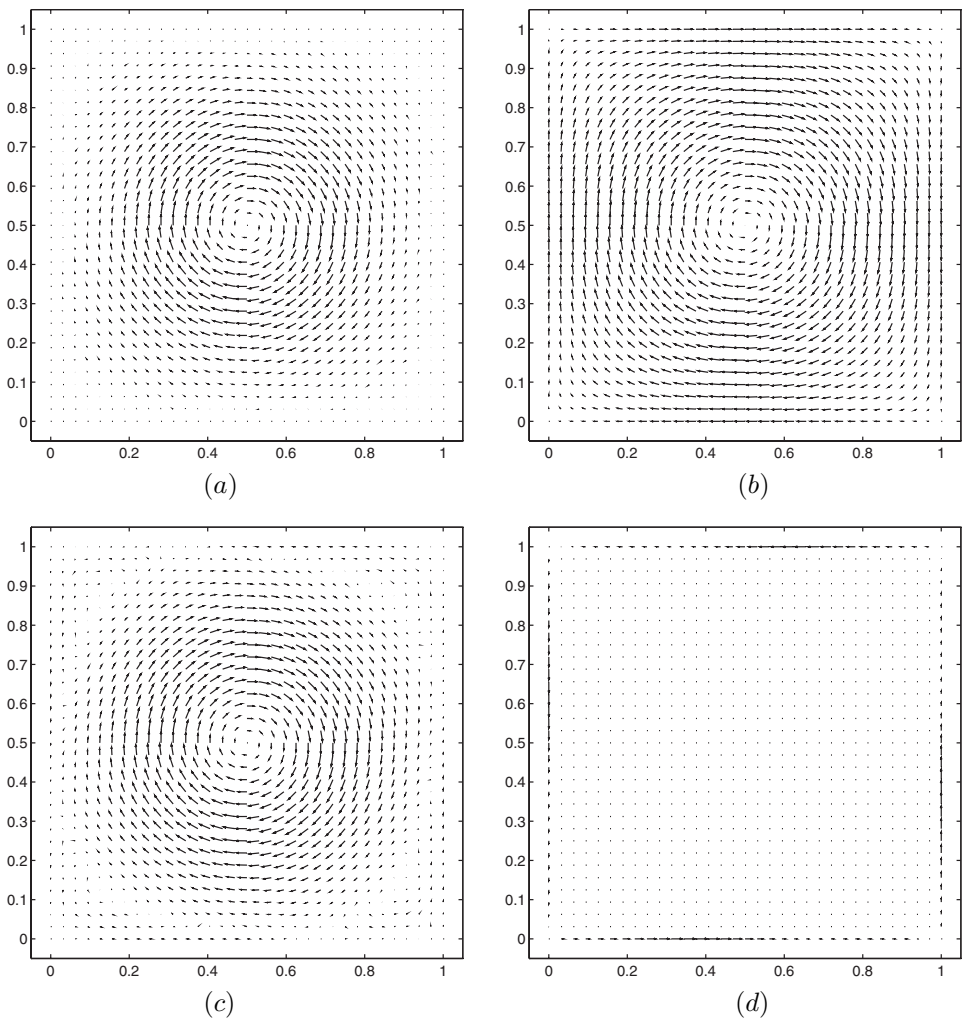


FIG. 5.2. *Information on cavity flow problems:* (a) *External force.* (b) *Computed velocity field for the simulation problem* (5.1) *for* $Re = 200$. (c) *Computed velocity field for the tangential BCP* (5.2) *for* $Re = 200$. (d) *Highlight of the boundary velocity in* (c).

of the interior flow. The achieved minimization can be observed in two ways: by comparing Figure 5.2(c) against Figure 5.2(b) and by comparing the value $\|\omega\|_h^2 \approx 32.5$ in Tables 5.2–5.6 to the value $\|\omega\|_h^2 \approx 55.4$ in Table 5.1.

5.3.2. Backward-facing step flow problems. In this subsection we present results for the backward-facing step problem. Table 5.9 contains results for the simulation problem (5.3) with $Re = 200$. The standard merit function is used in the line search, and an overlap $\delta = 1/32$ is used in the ASM preconditioner. The total number of Newton iterations does not change with the mesh size or the number of processors. The average number of GMRES iterations does not seem to follow (3.6) and (3.7) as closely as in the case of cavity flow simulation problems. But $\bar{l}$ behaves better when RAS is employed.

TABLE 5.9

Results for the backward-facing step flow simulation problem with $Re = 200$, standard ASM with overlap $\delta = 1/32$, and standard merit function $\|\mathbf{F}\|_2^2/2$. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 297,603 and $\|\omega\|_h^2 \approx 3.35$. See (5.6).

# Procs.	Mesh 192×32		Mesh 384×64		Mesh 768×128	
16	$n = 5$	$t_n \approx 1.35$	$n = 5$	$t_n \approx 6.49$	$n = 5$	$t_n \approx 37.2$
	$\bar{l} \approx 54$	$\bar{t}_l \approx 0.22$	$\bar{l} \approx 59$	$\bar{t}_l \approx 1.15$	$\bar{l} \approx 64$	$\bar{t}_l \approx 6.85$
32	$n = 5$	$t_n \approx 1.37$	$n = 5$	$t_n \approx 5.96$	$n = 5$	$t_n \approx 30.7$
	$\bar{l} \approx 92$	$\bar{t}_l \approx 0.22$	$\bar{l} \approx 109$	$\bar{t}_l \approx 1.09$	$\bar{l} \approx 121$	$\bar{t}_l \approx 5.82$
64	$n = 5$	$t_n \approx 0.96$	$n = 5$	$t_n \approx 3.35$	$n = 5$	$t_n \approx 13.6$
	$\bar{l} \approx 89$	$\bar{t}_l \approx 0.13$	$\bar{l} \approx 102$	$\bar{t}_l \approx 0.58$	$\bar{l} \approx 102$	$\bar{t}_l \approx 2.50$

Next, we consider the tangential BCP (5.4) with $Re = 200$ and an augmented Lagrangian merit function in the line search. In this case the standard ASM with $\delta = 1/32$ does not result in a good enough preconditioner for GMRES; i.e., the maximum allowed number (5,000) of Krylov iterations is eventually achieved without any tolerance being satisfied. Table 5.10 presents results for the standard ASM with $\delta = 1/16$. Interestingly, the total number of Newton iterations decreases as the mesh is refined, although the overall effort to find a solution increases. Comparing these results with Table 5.9, we can see that the average number of GMRES iterations is now much more sensitive to both h and H.

With $\delta > 1/16$ we expect a better convergence of LNKSz. But instead of applying the standard ASM with $\delta = 1/8$, we apply, in Table 5.11, RAS with $\delta = 1/8$, trying to balance more sparse LU time spent on bigger extended subdomains with less communication time spent on RAS. The number of Newton iterations continues to decrease as the mesh is refined. With an overlap δ four times as large as the one used on the simulation problem, the average number of GMRES iterations now follows (3.6) and (3.7) more closely. Comparing values of $\bar{t}_l$ in Tables 5.11 and 5.9, we see that the average time spent on computing $\mathbf{p}^{(k)}$ can be more than 20 times greater in control problems than in simulation problems with the same mesh, instead of being around $8/3 \approx 3$ times greater, in accordance to the ratio between the number of variables per mesh point on control problems and on simulation problems.

We now increase the Reynolds number to $Re = 300$, and the performance data for the tangential BCP (5.4) is provided in Table 5.12. We use the augmented Lagrangian merit function in the line search and $\delta = 1/8$ in RAS. With the increase in Re, both

TABLE 5.10

Results for the backward-facing step tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 and $\|\omega\|_h^2 \approx 3.28$. See (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 40$	$t_n \approx 89.1$	$n = 31$	$t_n \approx 392$	$n = 23$	$t_n \approx 1610$
	$\bar{l} \approx 81$	$\bar{t}_l \approx 1.92$	$\bar{l} \approx 85$	$\bar{t}_l \approx 11.5$	$\bar{l} \approx 90$	$\bar{t}_l \approx 64.8$
32	$n = 41$	$t_n \approx 322$	$n = 31$	$t_n \approx 2100$	$n = 23$	$t_n \approx 13400$
	$\bar{l} \approx 605$	$\bar{t}_l \approx 7.65$	$\bar{l} \approx 1110$	$\bar{t}_l \approx 67.0$	$\bar{l} \approx 1800$	$\bar{t}_l \approx 581$
64	$n = 41$	$t_n \approx 209$	$n = 31$	$t_n \approx 1460$	$n = 23$	$t_n \approx 8520$
	$\bar{l} \approx 701$	$\bar{t}_l \approx 4.97$	$\bar{l} \approx 1410$	$\bar{t}_l \approx 46.6$	$\bar{l} \approx 2200$	$\bar{t}_l \approx 369$

TABLE 5.11

Results for the backward-facing step tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/8$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 and $\|\omega\|_h^2 \approx 3.28$. See (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 40$	$t_n \approx 111$	$n = 31$	$t_n \approx 648$	$n = 23$	$t_n \approx 4990$
	$\bar{l} \approx 62$	$\bar{t}_l \approx 2.44$	$\bar{l} \approx 61$	$\bar{t}_l \approx 19.6$	$\bar{l} \approx 62$	$\bar{t}_l \approx 211$
32	$n = 41$	$t_n \approx 87.9$	$n = 32$	$t_n \approx 367$	$n = 23$	$t_n \approx 2250$
	$\bar{l} \approx 78$	$\bar{t}_l \approx 1.93$	$\bar{l} \approx 80$	$\bar{t}_l \approx 10.8$	$\bar{l} \approx 83$	$\bar{t}_l \approx 94.9$
64	$n = 40$	$t_n \approx 50.0$	$n = 31$	$t_n \approx 246$	$n = 23$	$t_n \approx 1560$
	$\bar{l} \approx 80$	$\bar{t}_l \approx 1.12$	$\bar{l} \approx 83$	$\bar{t}_l \approx 7.55$	$\bar{l} \approx 86$	$\bar{t}_l \approx 66.4$

TABLE 5.12

Results for the backward-facing step tangential boundary flow control problem with $Re = 300$, restricted ASM with overlap $\delta = 1/8$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 and $\|\omega\|_h^2 \approx 3.65$. See (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 53$	$t_n \approx 148$	$n = 41$	$t_n \approx 875$	$n = 30$	$t_n \approx 6600$
	$\bar{l} \approx 61$	$\bar{t}_l \approx 2.47$	$\bar{l} \approx 62$	$\bar{t}_l \approx 20.1$	$\bar{l} \approx 63$	$\bar{t}_l \approx 214$
32	$n = 55$	$t_n \approx 118$	$n = 41$	$t_n \approx 498$	$n = 31$	$t_n \approx 3090$
	$\bar{l} \approx 81$	$\bar{t}_l \approx 1.97$	$\bar{l} \approx 86$	$\bar{t}_l \approx 11.5$	$\bar{l} \approx 88$	$\bar{t}_l \approx 96.6$
64	$n = 53$	$t_n \approx 67.1$	$n = 41$	$t_n \approx 333$	$n = 30$	$t_n \approx 2050$
	$\bar{l} \approx 82$	$\bar{t}_l \approx 1.13$	$\bar{l} \approx 86$	$\bar{t}_l \approx 7.75$	$\bar{l} \approx 89$	$\bar{t}_l \approx 66.8$

nonlinear (n) and average linear ($\bar{l}$) complexities increase, although the increase in the total number of Newton iterations is more pronounced.

Again, as done in the case of cavity flow control problems, in order to determine if smaller fixed GMRES tolerances would improve the number of Newton iterations

TABLE 5.13

Efficiency analysis of NKSz and LNKSz, w.r.t. the number of processors, for backward-facing step flow problems with a 768×128 *mesh and different values of Reynolds number Re and overlap* δ. E_n *is the efficiency related to* t_n, *the total time in seconds spent on all Newton iterations.* $\overline{E}_l$ *is the efficiency related to* $\bar{t}_l$, *the average time in seconds, per Newton iteration, spent on solving for Newton steps. See* (5.7). *Results are based on values of* t_n *and* $\bar{t}_l$ *from Tables* 5.9, 5.11, *and* 5.12.

# Procs.	Simulation		Boundary control			
	$Re = 200,\ \delta = 1/16$ Standard ASM		$Re = 200,\ \delta = 1/8$ Restricted ASM		$Re = 300,\ \delta = 1/8$ Restricted ASM	
	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$
16	1.00	1.00	1.00	1.00	1.00	1.00
32	0.61	0.59	1.11	1.11	1.07	1.11
64	0.68	0.69	0.80	0.79	0.80	0.80

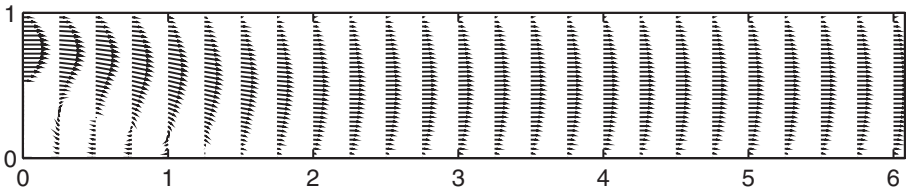


FIG. 5.3. *Computed velocity for the backward-facing step problem* (5.3) *with* $Re = 200$.

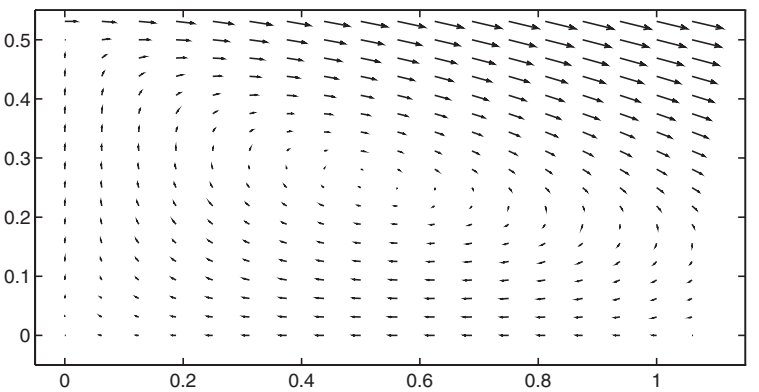


FIG. 5.4. *Highlight of the velocity field in the left bottom region of Figure* 5.3.

and thus, eventually, improve the total computing time, we rerun the tests of Table 5.12 with the same parameters, except for a GMRES relative tolerance of 1.0×10^{-6} instead of 1.0×10^{-4}, and a GMRES absolute tolerance of 1.0×10^{-13} instead of 1.0×10^{-6}. When compared to the results in Table 5.12, one or two Newton steps are saved in just two of the nine tests, while the increase in $\bar{l}$ (and $\bar{t}_l$, consequently) makes t_n increase in all nine cases.

In Table 5.13 some parallel efficiency results derived from data presented in previous tables are given.

Finally, we look at the computed velocity fields for $Re = 200$. Figures 5.3–5.5 are for the simulation problem (5.3), and Figures 5.6–5.8 are for the tangential BCP

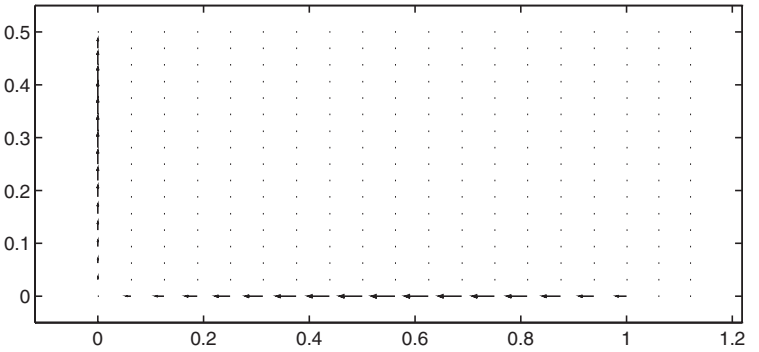


FIG. 5.5. *Highlight of the boundary velocity in the left bottom region of Figure* 5.3.

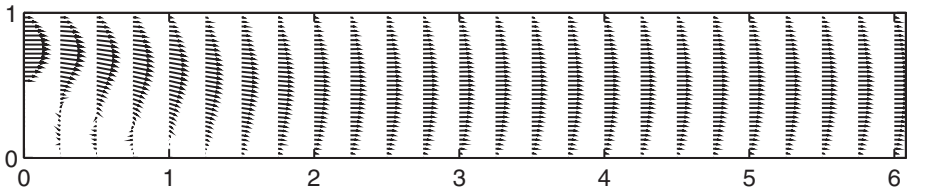


FIG. 5.6. *Computed velocity for the backward-facing step tangential BCP* (5.4) *with* $Re = 200$.

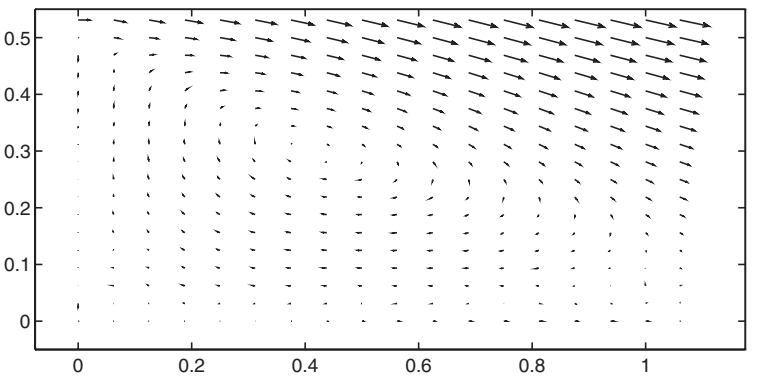


FIG. 5.7. *Highlight of the velocity field in the left bottom region of Figure* 5.6.

(5.4). The results are very similar to those of the cavity flow problem; more precisely speaking, Figures 5.4 and 5.8 show that the tangential boundary control acts in the opposite direction of the interior flow. The achieved minimization values can be measured either by comparing Figure 5.7 with Figure 5.4, or by comparing the value $\|\omega\|_h^2 \approx 3.28$ in Tables 5.10–5.11 with the value $\|\omega\|_h^2 \approx 3.35$ in Table 5.9.

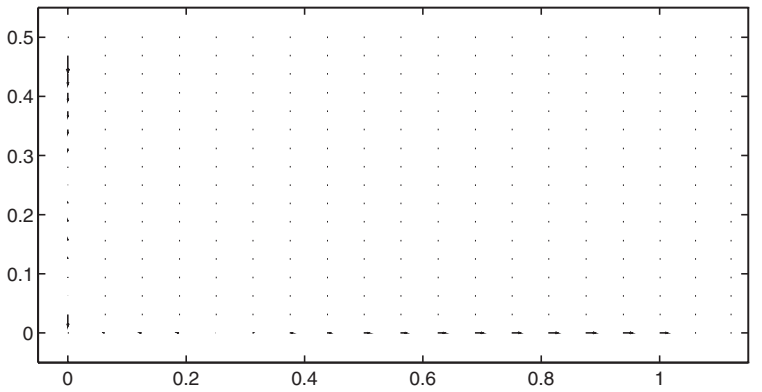


FIG. 5.8. *Highlight of the boundary velocity in the left bottom region of Figure* 5.6.

6. Conclusions. We have developed a general LNKSz algorithm for PDE constrained optimization problems and applied it to some tangential BCPs involving two-dimensional incompressible Navier–Stokes equations. In our numerical experiments the LNKSz algorithm, together with an augmented Lagrangian merit function, provides a fully parallel and robust solution method. The one-level additive Schwarz preconditioned GMRES, with a proper overlap, works well for the indefinite linearized KKT systems. A proper overlap for a control problem seems to be greater than a proper overlap for a simulation problem. More precisely, in our experiments the proper overlaps are two to four times greater in the control problems. For larger overlaps the restricted version of ASM seems to perform better than the standard ASM as a preconditioner for linearized KKT systems.

Our experiments also show that control problems are computationally more demanding, in terms of the total number of nonlinear iterations, the average number of linear iterations per Newton iteration, and the total computing time, than the corresponding simulation problems. The overall computational effort grows as Re increases, in terms of the number of both nonlinear and average linear iterations.

Acknowledgments. We would like to thank the PETSc team of Argonne National Laboratory for their help on using the PETSc library. We would also like to thank Professors G. Biros, O. Ghattas, S. Hou, and S. Ravindran, as well as two anonymous referees, for their useful comments.

REFERENCES

[1] F. ABERGEL AND R. TEMAM, *On some control problems in fluid mechanics*, Theoret. Comput. Fluid Dyn., 1 (1990), pp. 303–325.

[2] S. BALAY, K. BUSCHELMAN, W. D. GROPP, D. KAUSHIK, M. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *Portable, Extensible Toolkit for Scientific Computation (PETSc)*, http://www.mcs.anl.gov/petsc (2004).

[3] H. T. BANKS AND K. KUNISCH, *Estimation Techniques for Distributed Parameter Systems*, Systems Control Found. Appl. 1, Birkhäuser Boston, Boston, 1989.

[4] A. BATTERMANN AND M. HEINKENSCHLOSS, *Preconditioners for Karush-Kuhn-Tucker matrices arising in the optimal control of distributed systems*, in Control and Estimation of Distributed Parameter Systems, Internat. Ser. Numer. Math. 126, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser-Verlag, Basel, Switzerland, 1998, pp. 15–32.

[5] M. Bergounioux and K. Kunish, *Augmented Lagrangian algorithms for state constrained optimal control problems*, in Control and Estimation of Distributed Parameter Systems, Internat. Ser. Numer. Math. 126, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser-Verlag, Basel, Switzerland, 1998, pp. 33–48.

[6] M. Bergounioux and K. Kunish, *On the structure of the Lagrange multiplier for state-constrained optimal control problems*, Systems Control Lett., 48 (2002), pp. 16–176.

[7] T. R. Bewley, *Flow control: New challenges for a new renaissance*, Progr. Aerospace Sci., 37 (2001), pp. 21–58.

[8] G. Biros and O. Ghattas, *Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* I: *The Krylov–Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687–713.

[9] G. Biros and O. Ghattas, *Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* II: *The Lagrange–Newton solver and its application to optimal control of steady viscous flows*, SIAM J. Sci. Comput., 27 (2005), pp. 714–739.

[10] A. Borzi and K. Kunish, *A multigrid method for optimal control of time-dependent reaction diffusion processes*, in Fast Solution of Discretized Optimization Problems, Internat. Ser. Numer. Math. 138, K.-H. Hoffman, R. H. W. Hoppe, and V. Schulz, eds., Birkhäuser-Verlag, Basel, Switzerland, 2000, pp. 50–57.

[11] X.-C. Cai, M. Dryja, and M. Sarkis, *Restricted additive Schwarz preconditioners with harmonic overlap for symmetric positive definite linear systems*, SIAM J. Numer. Anal., 41 (2003), pp. 1209–1231.

[12] X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.

[13] X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.

[14] X.-C. Cai, D. E. Keyes, and L. Marcinkowski, *Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics*, Int. J. Numer. Meth. Fluids, 40 (2002), pp. 1463–1470.

[15] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[16] H. Choi, M. Hinze, and K. Kunish, *Instantaneous control of backward-facing step flows*, Appl. Numer. Math., 31 (1999), pp. 133–158.

[17] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[18] O. Daube, *Resolution of the* 2*D Navier-Stokes equations in velocity-vorticity form by means of an influence matrix technique*, J. Comput. Phys., 103 (1992), pp. 402–414.

[19] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics Appl. Math. 16, SIAM, Philadelphia, 1996.

[20] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.

[21] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[22] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.

[23] A. V. Fursikov, *Optimal Control of Distributed Systems. Theory and Applications*, Trans. Math. Monogr. 187, AMS, Providence, RI, 2000.

[24] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[25] G. Guj and F. Stella, *Numerical solutions of high-Re recirculating flows in vorticity-velocity form*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 405–416.

[26] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, Adv. Des. Control 5, SIAM, Philadelphia, 2002.

[27] M. D. Gunzburger and L. S. Hou, *Finite-dimensional approximation of a class of constrained nonlinear optimal control problems*, SIAM J. Control Optim., 34 (1996), pp. 1001–1043.

[28] M. Heinkenschloss, *Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet boundary control of Navier-Stokes flow*, in Optimal Control: Theory, Algorithms and Applications, Appl. Optim. 15, W. W. Hager and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998, pp. 178–203.

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

[30] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results*, SIAM J. Sci. Comput., 20 (1999), pp. 1753–1777.
[31] F.-N. Hwang and X.-C. Cai, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., 204 (2005), pp. 666–691.
[32] A. D. Ioffe and V. M. Tihomirov, *Theory of Extremal Problems*, Stud. Math. Appl. 6, North–Holland, Amsterdam, New York, 1979.
[33] K. Ito and K. Kunish, *The augmented Lagrangian method for equality and inequality constrained problems in Hilbert spaces*, Math. Programming, 46 (1990), pp. 341–360.
[34] K. Ito and K. Kunish, *The augmented Lagrangian method for parameter estimation in elliptic systems*, SIAM J. Control Optim., 28 (1990), pp. 113–136.
[35] B. Mohammadi and O. Pironneau, *Applied Shape Optimization for Fluids*, Numer. Math. Sci. Comput., The Clarendon Press, Oxford University Press, New York, 2001.
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[37] L. Quartapelle, *Numerical Solution of the Incompressible Navier-Stokes Equations*, Internat. Ser. Numer. Math. 113, Birkhäuser-Verlag, Basel, 1993.
[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[39] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[40] S. S. Sritharan, *Optimal Control of Viscous Flow*, SIAM, Philadelphia, 1998.
[41] V. M. Tihomirov, *Fundamental Principles of the Theory of Extremal Problems*, John Wiley & Sons, New York, 1986.
[42] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.
[43] M. Ulbrich, *Semismooth Newton methods for operator equations in function spaces*, SIAM J. Optim., 13 (2003), pp. 805–841.
[44] M. Ulbrich, S. Ulbrich, and M. Heinkenschloss, *Global convergence of trust-region interior-point algorithms for infinite-dimensional nonconvex minimization subject to pointwise bounds*, SIAM J. Control Optim., 37 (1999), pp. 731–764.
[45] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers Appl. Math., SIAM, Philadelphia, 2002.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1329–1346
© 2006 Society for Industrial and Applied Mathematics

ON OSCILLATIONS IN DISCONTINUOUS GALERKIN DISCRETIZATION SCHEMES FOR STEADY STATE PROBLEMS*

N. B. PETROVSKAYA†

Abstract. High order discontinuous Galerkin discretization schemes are considered for steady state problems. We discuss the issue of oscillations arising when Newton's method is employed to obtain a steady state solution. It will be demonstrated that flux approximation near flux extrema may produce spurious oscillations propagating over the domain of computation. The control over the numerical flux in the problem allows one to obtain nonoscillating convergent solutions.

Key words. discontinuous Galerkin, spurious oscillations, Newton's method

AMS subject classifications. 49M15, 65L20, 65L60

DOI. 10.1137/040603085

1. Introduction. In recent years a variety of discretization methods have been developed to solve complex problems of physics and engineering. One of them is a discontinuous Galerkin (DG) discretization scheme. Introduced in [11] and further developed by many authors (see [4] for the review of DG schemes), the DG method is a finite element scheme which uses piecewise polynomial approximation in space. The method also involves an approximate Riemann solver, since the approximate solution is discontinuous at grid interfaces.

The hyperbolic systems of conservation laws present a wide class of problems where the DG method can be successfully applied. The DG discretization scheme affords optimal orders of convergence for smooth problems by using high order approximating spaces. For the problems whose solutions have strong gradients and/or discontinuities, solution oscillations may occur when a high order DG scheme is used to discretize a conservation law. Since the nonphysical oscillations have a disastrous impact on the convergence of the approximate solution, a limiting procedure that eliminates the oscillations near discontinuities should be addressed. A number of authors have contributed to the issue of limiters for DG schemes in recent years (e.g., see [5], [6], [9]). It has been demonstrated many times that stabilization of the scheme by means of local limiters allows one to obtain accurate nonoscillating solutions to nonlinear hyperbolic problems.

The local limiters are not always helpful, however, when steady state solutions to conservation laws are considered. In practice, a time-dependent algorithm (e.g., a backward Euler integration) is used to approach a steady state solution. The time step is usually scaled as a function of the norm of the residual, so that the scheme with an infinitely large time step is equivalent to Newton's method. Thus it seems to be a reasonable strategy to solve time-dependent equations only at the early stages of computations. Once the basin of attraction has been approached, the Newton method may be exploited in order to provide a faster convergence rate. Meanwhile, our numerical experience shows that a transient solution may exhibit strong oscillations over

*Received by the editors January 12, 2004; accepted for publication (in revised form) June 14, 2005; published electronically January 6, 2006. This work was supported by The Boeing Company under contract 104AE.

http://www.siam.org/journals/sisc/27-4/60308.html

†School of Mathematics, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, United Kingdom (N.B.Petrovskaya@bham.ac.uk).

the entire domain of computation, if the Newton iteration method is used to solve a system of nonlinear equations obtained as a result of a high order DG discretization in space. Those oscillations may appear for a smooth solution as well as a discontinuous one, and their excitation does not depend on how close the initial guess for the Newton method is to the fixed point considered as a steady state solution for the problem. The spurious oscillations propagating over the domain cannot be eliminated by means of a standard limiting procedure [5], and their nature requires careful study.

In our work, we consider two nonlinear scalar equations in order to examine a high order DG discretization for steady state solutions. Simple enough, they nevertheless demonstrate the difficulties arising in the solution of steady state problems. In our first example the exact solution is smooth, while the solution to the second problem has a discontinuity. It will be shown that in both cases a standard high order DG discretization yields a divergent solution.

Based on our consideration, we conclude that a high order DG scheme is not able to recognize flux extrema that may result in a singular Jacobian when Newton's method is used to solve the problem. Moreover, a transient solution may generate nonphysical flux extrema which lead to a singular matrix as well. Thus, spurious solution oscillations occur in the problem due to incorrect flux approximation, so that a high order DG discretization requires flux control over each grid cell. We present a flux control procedure that allows one to obtain convergent solutions.

2. The problem statement. We consider an ordinary differential equation written for a function $u(x)$ in the conservative form

$$F_x(x,u)=0, \quad x\in\Omega=[0,1], \tag{1}$$

where $F(x,u(x))$ is a flux function. An appropriate boundary condition

$$\mathbf{B}u=0 \tag{2}$$

is provided for (1), where $\mathbf{B}$ denotes a boundary condition operator.

For numerical solution of the boundary-value problem (1), (2) we introduce the element partition G of the region, $G=\bigcup_{i=1}^{N} e_i$, $e_i=[x_i,x_{i+1}]$, $1\le i\le N$, where x_i is a nodal coordinate, and $h_i=x_{i+1}-x_i$ is a grid step size. We also use the notation x_i-0 and x_i+0 for the left and right limits at the point x_i.

Let $u(x)$ be the solution to the problem (1), (2). In order to find the approximate solution $u_h(x)$, a weak formulation of the problem is used. Multiplying (1) by test function $\phi_k(x)$, defined on the cell e_i for $k=0,1,\dots,K$ as

$$\phi_k(x)=\left(\frac{x-x_i}{h_i}\right)^k, \quad x\in e_i,$$

and integrating by parts over the cell e_i, we obtain

$$F(x_{i+1},u(x_{i+1}))\phi_k(x_{i+1})-F(x_i,u(x_i))\phi_k(x_i)-\int_{x_i}^{x_{i+1}}F(x,u)\frac{d\phi_k(x)}{dx}dx=0,$$
$$k=0,1,\dots,K. \tag{3}$$

We now replace the function $u(x)$ in (3) by the approximate solution $u_h(x)$. The DG discretization seeks for the approximation $u_h(x)$ to the solution $u(x)$ such that

$u_h(x)$ is a piecewise polynomial function over Ω. The approximate solution $u_h(x)$ is expanded on the cell e_i as

$$u_h(x) = \sum_{k=0}^{K} u_k \phi_k(x), \quad k = 0, 1, \ldots, K, \quad x \in e_i. \tag{4}$$

Since $u_h(x)$ is discontinuous at cell interfaces, (3) considered for the solution $u_h(x)$ requires us to define numerical flux $\tilde{F}(u_h)$. Suppose that the flux $\tilde{F}(u_h)$, which generally depends on the two values of the approximate solution at any grid point, is chosen for a given problem. Then the DG discretization scheme reads

$$\tilde{F}(u_h(x_{i+1}))\phi_k(x_{i+1}) - \tilde{F}(u_h(x_i))\phi_k(x_i) - \int_{x_i}^{x_{i+1}} F(x, u_h(x)) \frac{d\phi_k(x)}{dx} dx = 0, \tag{5}$$
$$k = 0, 1, \ldots, K.$$

For steady state problem (1), (2), the DG space discretization over the grid results in the following system of nonlinear equations:

$$\mathbf{R}(\mathbf{u}) = 0, \tag{6}$$

where the vector $\mathbf{R}(\mathbf{u})$ is the residual of the DG method given by (5) on each grid cell and $\mathbf{u}$ is the solution vector. We use Newton's iteration method to solve the nonlinear equations (6). Let $\mathbf{u}^n$ and $\mathbf{u}^{n+1}$ be the solution vector at the nth and $(n+1)$th solution iteration, respectively. Then the linearized system is

$$\mathbf{J}(\mathbf{u}^n)(\mathbf{u}^{n+1} - \mathbf{u}^n) = -\mathbf{R}(\mathbf{u}^n), \tag{7}$$

where the Jacobian matrix $\mathbf{J}(\mathbf{u}) = [\partial \mathbf{R}/\partial \mathbf{u}]$ and residual $\mathbf{R}(\mathbf{u})$ are taken from the nth iteration. The GMRES algorithm [12], [2] is used to solve numerically the algebraic system of linear equations obtained at each Newton iteration.

In the remainder of our paper we discuss oscillations appearing in solution (5), (7). One important observation about the DG scheme is that for the steady state problem (1), (2) the solution on the ith grid cell impacts on the solution on neighboring cells only by means of the numerical flux $\tilde{F}(u_h)$. Hence, the two possible ways of the excitation of oscillations at the nth Newton iteration are as follows.

1. The numerical flux required to define the DG discretization on the cell e_i is correct, but the approximate solution on the cell does not converge in a particular norm. That may happen, for instance, when a discontinuity presented in the cell is approximated by smooth function (4). The solution overshoots arising as a result of such approximation are local and do not affect the solution on other cells.
2. The numerical flux is not correct on the ith cell. The incorrect flux approximation produces solution oscillations that will propagate over the domain at the next Newton iterations and result in a divergent solution.

While local limiters can be successfully used to smooth the local solution overshoots, another approach is required to recognize and eliminate the spurious oscillations propagating over the grid. That approach will be discussed below.

3. The numerical flux in steady state problems. In this section, we address a numerical flux used in the formulation of the DG discretization. Usually, oscillations arising in the approximate solution are associated with solution discontinuities. Thus,

our aim is to verify the definition of the numerical flux and demonstrate that the oscillations may appear for a smooth solution as well as a discontinuous function.

We begin our consideration with a simple example of (1) that illustrates the problem. Let the flux $F(x,u)$ be

$$F(x,u) = p(x)f(u), \quad p(x) = \frac{1}{((x-x_0)(x-x_1))^2}, \quad f(u) = (u-A)^2. \tag{8}$$

The problem parameters x_0, x_1, and A and the boundary condition are chosen to provide a smooth solution to the problem

$$U(x) = A + C(x-x_0)(x-x_1), \quad x \in [0,1],$$

where C is a constant. Namely, we take $A = 1.0$, $C = -1.0$, $x_0 = -0.5$, and $x_1 = 1.5$, so that bifurcation points $x = x_0$ and $x = x_1$ lie outside the domain of computation. The boundary condition is

$$\int_0^1 u(x)dx = B, \tag{9}$$

where the value B is defined by integrating the exact solution with the parameters above.

The model problem (1), (8) is numerically solved by using the DG discretization approach. We choose the Engquist–Osher definition [10] to approximate the flux at grid interfaces. Let u_l and u_r be the left and right states at the interface x_i, respectively. The numerical flux reads

$$\tilde{F}^{EO}(u_l,u_r) = \int_0^{u_r} \min(F'(s),0)ds + \int_0^{u_l} \max(F'(s),0)ds + F(0). \tag{10}$$

For the problem (8), the flux has a single extremum point, $u = A$. Hence, the numerical flux (10) is as follows:

$$\tilde{f}(u_l,u_r) = \begin{cases} f(u_l), & u_l > A,\ u_r > A, \\ f(u_r), & u_l < A,\ u_r < A, \\ f(A), & u_l < A,\ u_r > A, \\ f(u_l) + f(u_r) - f(A), & u_l > A,\ u_r < A. \end{cases} \tag{11}$$

The numerical experience with the problem shows that the convergence of the Newton method depends strongly on the choice of initial guess. Consider a sine wave function

$$u_0(x,s_0) = \sin(2\pi x) + s_0,$$

where s_0 is a parameter. Let us consider $s_0^{\mathrm{I}} = 2.3$ and $s_0^{\mathrm{II}} = 1.8$. For the initial guess $u_0^{\mathrm{I}} = u_0(x, s_0^{\mathrm{I}})$, the flux $f(u)$ is a monotone function over the domain of definition $u_0 \in [u_{min}^{\mathrm{I}}, u_{max}^{\mathrm{I}}]$. For the function $u_0^{\mathrm{II}} = u_0(x, s_0^{\mathrm{II}})$, we have $u_{min}^{\mathrm{II}} < A$, $u_{max}^{\mathrm{II}} > A$, so that the flux approximation is required at the extremum point $u = A$ at the first Newton step.

Although the curves u_0^{I} and u_0^{II} are close to each other, i.e., $||u_0^{\mathrm{II}} - u_0^{\mathrm{I}}||_{L^\infty} = |s_0^{\mathrm{II}} - s_0^{\mathrm{I}}|$, $x \in [0,1]$, the convergence results are quite different for the two functions. Starting with the initial guess u_0^{I}, Newton's method rapidly converges to the approximate solution $u_h(x)$. The convergence results obtained on a sequence of uniform grids

confirm the consistency of the approximation (5), (11). In particular, the DG scheme with polynomial degree $K = 2$ provides a precise reconstruction of the quadratic function $U(x)$. Meanwhile, the choice of u_0^{II} as initial guess for the problem results in a divergent solution for any polynomial degree $K > 0$.

Let us look more closely at the numerical flux used in the problem. For a scalar flux function, the definition of the numerical flux is essentially based on the analysis of the flux derivative, the results of which depend on how the solution variation $u_r - u_l$ is determined. In the definition (10), the solution variation is assigned to the grid interface x_i, $u_l = u_h(x_i - 0)$, $u_r = u_h(x_i + 0)$. In other words, the definition (10) implies that the flux variation is due only to the solution variation at grid interfaces, i.e., $F(u) = \mathrm{const}$ within a grid cell. Evidently, this assumption will be correct if the approximate solution is constant over each cell. Meanwhile, for a high order DG discretization scheme the approximate solution varies in the domain $[x_i, x_{i+1}]$. The solution variation $\delta u_h = u_h(x_{i+1} - 0) - u_h(x_i + 0)$ may generate a flux extremum at the interior point of the cell e_i, while the flux remains a monotone function at both interfaces x_i and x_{i+1}. Below we compare flux approximation in a piecewise constant and a higher order DG discretization and demonstrate that for a high order scheme, considering a flux as a monotone function at the interfaces of the cell where the flux extremum is located may lead to incorrect DG approximation.

As an example of a high order DG scheme, let us consider a piecewise linear approximate solution

$$u_h(x) = u_0 + u_1\phi_1(x), \quad x \in e_i.$$

Since a DG discretization with polynomial degree K results in $K+1$ unknown solution values (degrees of freedom) per cell, two DG equations (5) per cell must be employed for a piecewise linear solution. Those equations are a discrete conservation law,

$$p(x_{i+1})\tilde{f}(u_l, u_r) - p(x_i)\tilde{f}(u_l, u_r) = 0, \tag{12}$$

and the momentum equation,

$$p(x_{i+1})\tilde{f}(u_l, u_r) - \frac{1}{h_i}\int_{x_i}^{x_{i+1}} p(x)f(u_h(x))dx = 0. \tag{13}$$

For the purpose of further discussion it is more convenient to define the DG discretization on a given cell in terms of the solution values rather than the solution derivatives in the expansion (4) (see [13] for more details). We now introduce the notation u_i^l and u_i^r for the left and right solution states at the interface x_i, respectively. Then the values u_i^r and u_{i+1}^l are defined by the piecewise linear function $u_h(x)$ on a given cell e_i. The evident transformation

$$u_i^r \equiv u_h(x_i + 0) = u_0, \quad u_{i+1}^l \equiv u_h(x_{i+1} - 0) = u_0 + u_1,$$

allows one to consider left and right solution states as unknown degrees of freedom for a piecewise linear solution.

The flux approximation in the DG equations at the first Newton step is illustrated in Figure 1 for a piecewise constant and a linear approximate solution. The flux function $f(u) = (u - A)^2$ is shown in the $(u, f(u))$-plane in Figure 1(a)–(b). The solution degrees of freedom are stationed at the u-axis, and the solution values required to approximate the flux at cell interfaces are displayed as black dots. The

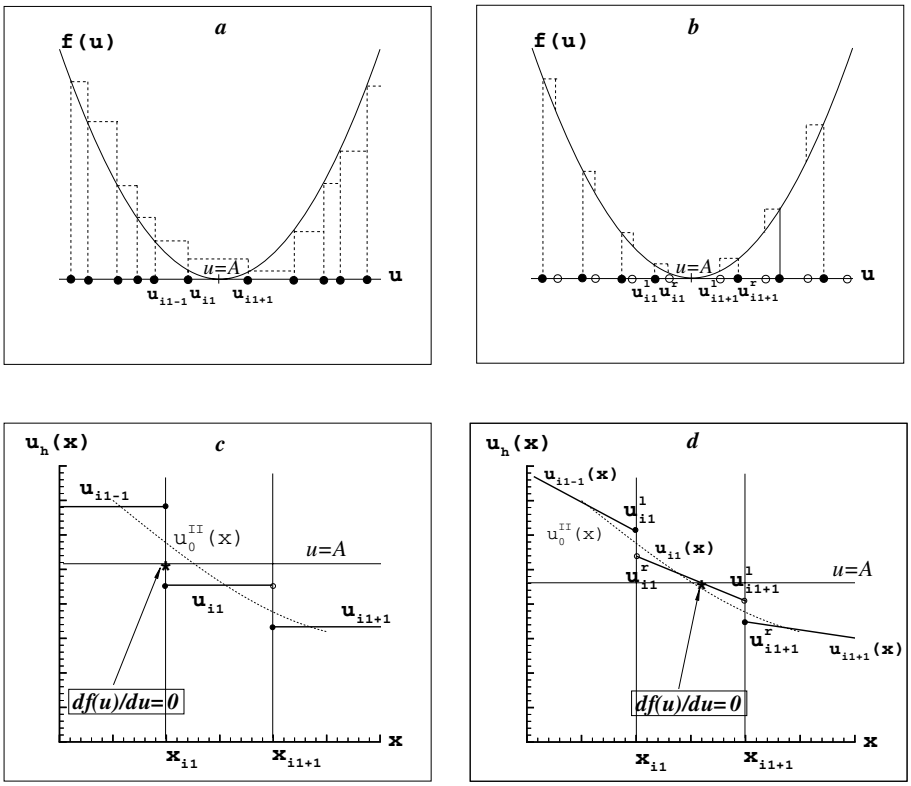


FIG. 1. (a)–(b) *The numerical flux near the extremum point for the problem* (1), (8). (a) *The numerical flux for the piecewise constant approximate solution. All degrees of freedom are involved in the flux approximation.* (b) *The flux approximation for a piecewise linear DG solution. Both degrees of freedom are not involved in the flux approximation on the cell where the flux extremum is located.* (c)–(d) *The DG discretization in the* $(x, u_h(x))$*-plane.* (c) *Piecewise constant solution approximation captures the flux extremum.* (d) *Higher order solution approximation misses the extremum point inside the grid cell.*

corresponding values of the numerical flux (10) at grid interfaces are shown at the curve $f(u)$. In the figure, the flux approximation is presented in the vicinity of the cell e_{i1} : $u^r_{i1} > A$, $u^l_{i1+1} < A$. In other words, the initial guess is a decreasing function near the extremum point, and the solution states at the grid interfaces corresponding to greater values of u are shown at the left from the extremum point $u = A$.

The flux approximation for the piecewise constant approximate solution (one degree of freedom per cell) is presented in Figure 1(a). It can be seen from the figure that, according to the definition (10), a discrete conservation law (12) involves one unknown solution value per cell. In other words, the number of DG equations to be solved over the grid is equal to the number of unknown solution values, as all degrees of freedom are involved in the flux approximation.

Now we look at the flux approximation in a piecewise linear DG scheme shown in Figure 1(b). Let flux be a monotone function over a given cell e_i. Again the DG equation (12) requires one degree of freedom per cell to balance the monotone fluxes at the left and right cell interfaces. That degree of freedom is chosen to provide the upwind discretization; i.e., we take u^r_i for $df(u)/du < 0$ or u^l_{i+1} for $df(u)/du > 0$ to approximate the flux. A second degree of freedom on the cell is not involved in the

discrete conservation equation (12), and thus another equation is required to define the second unknown solution value on the cell. This equation appears as the momentum equation (13) in the DG scheme. Hence, there are two DG equations for two unknown solution values on any cell where the flux is a monotone function through the cell.

The situation is different, however, on the cell where the flux extremum is located at the interior point. Now both degrees of freedom on the cell are not involved in the flux approximation, as the definition of the numerical flux concerns only the solution on the adjacent cells. Namely, we have $u^l_{i1} > A$, $u^r_{i1} > A$ at the left cell interface, so that the upwind flux is $f(u^l_{i1})$. Meanwhile, $u^l_{i1+1} < A$, $u^r_{i1+1} < A$, and the numerical flux is $f(u^r_{i1+1})$ at the right cell interface.

The DG equations on the cell e_{i1} are illustrated in the $(x, u_h(x))$-plane in Figure 1(c)–(d) for a piecewise constant and a linear discretization, respectively. Again, the solution values required to define the flux at the interfaces of the cell e_{i1} are shown as black dots. It can be seen from the figure that for the piecewise constant DG discretization, displayed in Figure 1(c), the extremum point at the interface is taken into account in the definition of the numerical flux. The discrete conservation law reads

$$p(x_{i1+1})f(u^r_{i1+1}) - p(x_{i1})(f(u^l_{i1}) + f(u^r_{i1}) - f(A)) = 0.$$

For the linear approximate solution, shown in Figure 1(d), the "phantom" solution on the cell e_{i1} is not involved in the flux definition. The discrete conservation law is

$$p(x_{i1+1})f(u^r_{i1+1}) - p(x_{i1})f(u^l_{i1}) = 0, \tag{14}$$

where $u^l_{i1} = u_h(x_{i1} - 0)$ and $u^r_{i1+1} = u_h(x_{i1+1} + 0)$ are the solution unknowns on the adjacent cells.

Thus, we still need two equations to define the unknowns (u^r_{i1}, u^l_{i1+1}), while the DG discretization provides us with only one more equation that involves the unknowns on the cell e_{i1}. Let us also note that, by the definition of the numerical flux (10), the unknowns (u^r_{i1}, u^l_{i1+1}) on the cell e_{i1} are not required for the discretization on neighboring cells (see Figure 1(b)), and thus the number of DG equations to be solved over the grid is not equal to the number of unknown solution values for a piecewise linear solution. This indicates that the solution should be reduced to piecewise constant approximation on the cell e_{i1} in order to avoid the underdetermined system of equations over the grid. In other words, the number of degrees of freedom is excessive on the cell e_{i1}, and only one of them should be used for the discretization.

The same conclusion can be made in the case of a DG discretization with any polynomial degree $K > 0$. For instance, a piecewise quadratic approximation $K = 2$ results in three unknown values per cell. Instead of coefficients (u_0, u_1, u_2) in the expansion (4), these unknowns can be defined as u^r_i, u^l_{i+1}, and u^m_i, where $u^m_i = u_h(x_m)$, $x_m = 0.5(x_i + x_{i+1})$. Again, the discrete conservation law (12) on the cell e_{i1} does not require any of these degrees of freedom, as only the solution on adjacent cells is concerned, and thus we have two equations per cell to find three unknown values.

Consequently, solving the underdetermined system of equations arising as a result of a high order DG discretization leads to a singular Jacobian in the Newton method. For a piecewise linear solution, let us define a local Jacobian $\mathbf{j}_{k_1k_2} = [\partial R_{k_1}/\partial u_{k_2}]$ by linearizing the two DG equations on the cell e_{i1}. Here the local index $k_1 = 0, 1$ is related to the DG residual on the cell, while the local index $k_2 = 0, 1$ refers to the corresponding degree of freedom from the set (u^r_{i1}, u^l_{i1+1}). Then the first row in the 2×2-matrix $\mathbf{j}_{k_1k_2}$ has only zero entries, as we have $j_{00} \equiv \partial R_0/\partial u^r_{i1} = 0$ and

$j_{01} \equiv \partial R_0/\partial u^l_{i1+1} = 0$ for the residual R_0 defined by (14). Hence, the rank of the local Jacobian is $\mathrm{rank}(\mathbf{j}_{k_1k_2}) = 1 < \dim(\mathbf{j}_{k_1k_2}) = 2$, and by reordering the rows and columns we can obtain a zero column in the matrix $\mathbf{j}_{k_1k_2}$.

We now consider the matrix $\mathbf{j}_{k_1k_2}$ as the block J_{lm} of the Jacobian $\mathbf{J}$ of the linearized system (7). The global indices for the block entries are $l = k_1 + M_1$ and $m = k_2 + M_1$, where $M_1 = 2\,i1 - 1$. Since the definition (10) provides the exact flux splitting for the problem, we have the Jacobian entries $J_{lM1} = 0$, $J_{lM1+1} = 0$, $\forall l$: $l < M_1$ or $l > M_1 + 1$. On the other hand, it follows from the above consideration that the rows and columns of the Jacobian matrix can be reordered to provide

$$J_{lM_1} = 0, \quad l = M_1, M_1 + 1,$$

while we still have $J_{lM1} = 0$, $\forall l$: $l < M_1$ or $l > M_1 + 1$ in the reordered matrix. Hence, a zero column appears in the Jacobian of the system (7). The singular Jacobian leads to an incorrect transient solution (whose appearance depends strongly on the robustness of the GMRES solver used in the problem). That solution, in turn, will impact on the flux at the next Newton iterations, so that the oscillations will rapidly propagate over the domain resulting in the divergence of the method. Let us also notice that a similar analysis can be carried out for a DG discretization with any polynomial degree $K > 0$ to demonstrate that a singular Jacobian will appear in the problem, if the number of unknowns on the cell exceeds the number of equations.

The above results bring us to the conclusion that the nature of oscillations arising in the steady state problem (8), (9) is different from that appearing in approximate solution to hyperbolic conservation laws. In the latter case the oscillations arise near a solution discontinuity, and the approximation implies a well-defined numerical flux over the computational domain. Now the numerical flux $\tilde{f}(u)$ is not a correct approximation to the flux function $f(u)$ at the extremum point, while the solution remains a smooth monotone function near the flux extremum.

We are now going back to the problem (8). The above analysis reveals that a high order approximate solution should be reduced to a piecewise constant approximation near a flux extremum in order to avoid nonphysical oscillations in the problem. If the flux extremum generates a "phantom" solution on a given grid cell e_i (i.e., the number of unknowns exceeds the number of equations on the cell) at the nth Newton iteration, then we compute $\bar{u} = \frac{1}{h_i}\int_{x_i}^{x_{i+1}} u_h(x)dx$ and define the approximate solution on the cell e_i as $u_h(x) = \bar{u}$. The equations

$$\begin{gathered} p(x_{i+1})\tilde{f}(u_h) - p(x_i)\tilde{f}(u_h) = 0, \quad k = 0, \\ u_k = 0, \quad k = 1, \ldots, K, \end{gathered}$$

are then considered on the cell e_i to obtain the solution at the $(n+1)$th Newton iteration. A modified DG discretization allows one to obtain the convergent solution with the polynomial degree $K > 0$ for the function $u_0^{\mathrm{II}}(x)$ considered as the initial guess.

Finally, let us mention that the problem of flux approximation in a high order DG discretization cannot be completely solved by considering another numerical flux in a problem. The discussion in this section demonstrates that the number of degrees of freedom should be equal to the number of equations on each grid cell in a high order DG scheme. Hence, many of the upwind fluxes which provide exact flux splitting at the flux extremum point are hardly appropriate for a high order discretization, as they do not meet the requirement above. For instance, the Godunov flux is defined

as

$$\tilde{F}^G(u_l,u_r)=\begin{cases}\min\limits_{u_l\le u\le u_r} F(u) & \text{if } u_l\le u_r,\\ \max\limits_{u_l\le u\le u_r} F(u) & \text{otherwise}\end{cases}$$

for the left state u_l and right state u_r at a given grid interface. It can be seen from the flux definition that the Godunov flux presents us with the same problem as the Engquist–Osher flux (10). Since the discrete conservation law does not involve the solution near the extremum for a high order DG discretization, the number of unknowns exceeds the number of equations on the cell, where the extremum point is located. On the other hand, the space-centered fluxes, such as the local Lax–Friedrichs flux,

$$\tilde{F}^{LLF}(u_l,u_r)=\frac{1}{2}[F(u_l)+F(u_r)-C(u_r-u_l)],$$
$$C=\max_{min(u_l,u_r)\le s\le max(u_l,u_r)}|F'(s)|,$$

use both left and right solution states to approximate the flux, which makes the number of degrees of freedom equal to the number of equations on each grid cell. However, the space-centered fluxes cannot be considered a priori as a preferable choice for a high order DG discretization, as the issues of accuracy also should be taken into account. For instance, it is well known that the Lax–Friedrichs flux is more dissipative than upwind fluxes, and thus it may not be acceptable for a problem under consideration (e.g., see [14]). This makes the choice of a numerical flux in steady state problems a complicated task which requires further study and discussion.

4. The numerical flux for time-dependent problems. Approximate Riemann solvers have been successfully used many times in numerical solution of the hyperbolic systems of conservation laws (e.g., see [7] and the references therein). Thus, it is instructive to compare the results obtained for the steady state problem above with the convergence of a nonlinear solver for a time-dependent problem. The inviscid Burgers equation

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=0 \tag{15}$$

is a well-known example of a nonlinear hyperbolic equation which provides us with a quadratic flux function $f(u)=\frac{u^2}{2}$ similar to that in (8). We solve (15) in the domain $x\in[0,1]$ due to a periodic boundary condition. The initial condition is taken from [3], where it has been chosen as a sine wave function,

$$u(x,0)=u_0(x)=\frac{1}{4}+\frac{1}{2}\sin(\pi(2x-1)).$$

The exact solution is smooth for any time $T<1/\pi$, while the shock appears at later times.

For numerical solution of the conservation law (15), a DG discretization in space is combined with a backward Euler time integration scheme which results in a system of nonlinear equations at each time step. This system is linearized in order to obtain the solution at the upper time level. Notice that the choice of the initial condition requires the flux approximation near the extremum point $u=0$. Nevertheless, the nonlinear solver provides a convergent solution at any time $T>0$. An approximate

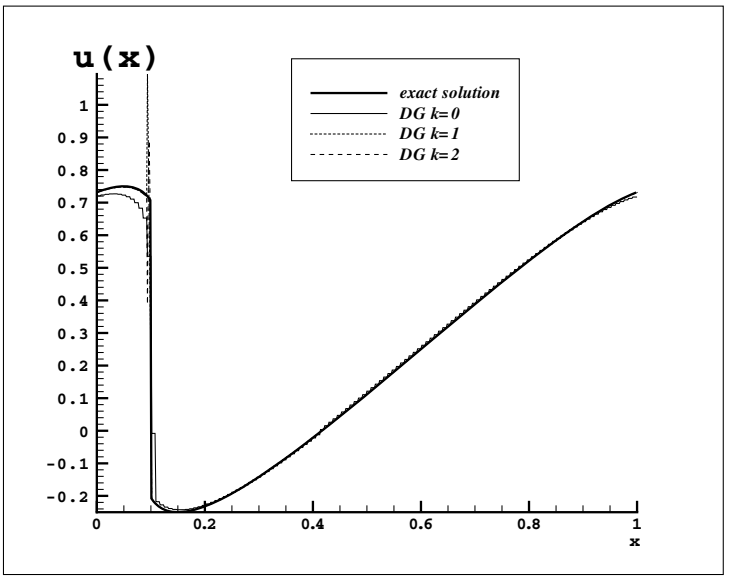


FIG. 2. *Exact and numerical solution to the Burgers equation. Smooth approximation to the solution generates local oscillations at the shock for the DG solution with polynomial degree $K > 0$. The solution overshoots can be eliminated by means of local limiters.*

solution at $T = 0.4$ is shown in Figure 2 for DG discretizations with polynomial degree $K \geq 0$ on a uniform grid of 128 cells. Let us notice that the DG $K = 1$ and $K = 2$ approximate solutions oscillate near the shock. However, those oscillations are local and can be eliminated by means of a limiting procedure [5].

The robustness of the nonlinear solver for the time-dependent problem (15) is readily explained based on the analysis of the Jacobian matrix. Consider the conservation law

$$\frac{\partial u}{\partial t} + F_x(x, u) = 0, \quad x \in \Omega. \tag{16}$$

The semidiscrete formulation of (16) on the cell e_i is

$$\int_{e_i} \frac{\partial u}{\partial t} \phi_k(x) dx + R_k^{DG}(\mathbf{u}) = 0, \quad k = 0, 1, \ldots, K,$$

where $R_k^{DG}(\mathbf{u})$ is the DG residual given by (5).

Let $\mathbf{u}^n$ and $\mathbf{u}^{n+1}$ be the solution vector over the grid at time t^n and $t^{n+1} = t^n + \Delta t$, respectively. After discretizing in time, the implicit scheme for the hyperbolic equation (16) reads

$$\mathbf{M}(\mathbf{u}^{n+1} - \mathbf{u}^n) = -\Delta t \mathbf{R}^{DG}(\mathbf{u}^{n+1}), \tag{17}$$

where positive diagonal matrix $\mathbf{M}$ is given by

$$M_{kl} = \int_{e_i} \phi_k(x) \phi_l(x) dx, \quad k, l = 0, 1, \ldots, K,$$

on each grid cell.

The linearization of the residual vector yields the system of equations

$$\mathbf{J}(\mathbf{u}^{n+1} - \mathbf{u}^n) = -\mathbf{R}(\mathbf{u}^n).$$

The Jacobian matrix is

$$\mathbf{J} = \frac{\mathbf{M}}{\Delta t} + \mathbf{J}^{DG},$$

where $\mathbf{J}^{DG}$ is the Jacobian of steady state problem (1), $\mathbf{J}^{DG} = [\partial \mathbf{R}^{DG}/\partial \mathbf{u}]$.

It can be seen from the above expression that the presence of mass matrix $\mathbf{M}$ in the discretization ensures diagonal entries in the Jacobian, even if matrix $\mathbf{J}^{DG}$ is singular. Hence, the time derivative can be considered as a stabilization term for high order DG discretizations.

5. The flux correction for steady state solutions. Lack of stabilization terms in steady state problems makes it difficult to use Newton's method for their numerical solution. On the one hand, it has been demonstrated in section 3 of the paper that an approximate Riemann solver may result in a singular matrix near the flux extremum. On the other hand, to obtain a convergent solution it is not sufficient to control the flux only near the extremum point. Below we demonstrate that a general case, unlike a simple model problem considered above, requires flux control on any grid cell.

Consider a scalar flux function $F(u)$. Any smooth function $F(u)$ that is not monotone in the domain of definition yields a multivalued solution $u(x)$ to the steady state equation (1). (From a geometric point of view, this means that the solution $F(x, u) = C$ to (1) intersects the curve $F(u)$ more than one time in the $(u, F(u))$-plane.) For the boundary-value problem (1), (2), the uniqueness of the solution is defined by a boundary condition.[1] However, a transient solution may experience jumps from one solution branch to another, until the basin of attraction is approached. Those local bifurcations may change the sign of the derivative dF/du and produce nonphysical flux extrema on the cell. As a result, a "phantom" solution appears on the cell, which, in turn, leads to incorrect flux approximation and a singular Jacobian in the problem.

The flux correction algorithm that we present in this section detects all flux extrema over the grid. It is important to notice here that one should distinguish between physical and nonphysical extrema in the problem. At the physical flux extremum, the solution should be reduced to piecewise constant approximation, as the number of degrees of freedom is not equal to the number of equations for a high order DG discretization. Nonphysical extrema appear in the regions where the flux should be a monotone function. In this case it is sufficient to render the flux monotone on a given cell, as the number of unknowns is equal to the number of equations on the cells where the flux is a monotone function.

The topic of the treatment of physical flux extrema has been covered in section 3. Below we discuss how to eliminate the nonphysical flux extrema in the problem. Let us denote the extremum points of the function $F(u)$ as $u_1, u_2, \ldots, u_{P-1}$. The domain of definition of the variable u can be partitioned as $\mathcal{D}_u = \bigcup_{p=0}^{P-1} [u_p, u_{p+1}]$, where u_0

[1] For a weak solution, additional constraints, such as the entropy condition (e.g., see [8]), are also required to provide the uniqueness of the solution.

and u_P are the boundary points of the domain. Consider the values $u(x_i - 0)$ and $u(x_{i+1} + 0)$, i.e., the approximate solution taken from the adjacent cells at the left and right interfaces of the cell e_i. Each of these values lies between two extremum (or boundary) points, $u(x_i - 0) \in [u_p, u_{p+1}]$, $u(x_{i+1} + 0) \in [u_q, u_{q+1}]$, where $0 \le p, q < P$. We now consider $u(x_i + 0)$ and $u(x_{i+1} - 0)$, which are the boundary values of the solution approximation in the cell e_i. Our goal is to detect nonphysical flux extrema within each grid cell. Instead of limiting the solution variation, we bound the flux variation in the cell in order to eliminate the flux oscillations.

Namely, we require that

$$u(x_i + 0) \in [u_p, u_{p+1}], \qquad u(x_{i+1} - 0) \in [u_q, u_{q+1}].$$

In other words, an approximate Riemann solver used in the problem must give the same choice of the numerical flux for the solution considered at $[u_l = u(x_i + 0), u_r = u(x_{i+1} - 0)]$ as for the interval $[u_l = u(x_i - 0), u_r = u(x_{i+1} + 0)]$.

From an algorithmic point of view, it is convenient to introduce the following formal description of our approach. Let us denote the left and right solution states at the interface x_i as u_{1i} and u_{2i}, respectively. Given numerical flux $\tilde{F}(u_h)$, we define $\mathbf{s}_i = (s_{1i}, s_{2i})^T$ at each grid interface x_i, $i = 1, \dots, N+1$, as follows:

$$\mathbf{s}_{li} = \begin{cases} 1 & \text{if } u_{li} \text{ is required to define } \tilde{F}(u_h), \quad l = 1, 2, \\ 0 & \text{otherwise.} \end{cases}$$

Once the state vector has been defined at each grid interface, the cell e_i can be described by the matrix $\mathbf{S}_i$,

$$\mathbf{S}_i = \begin{bmatrix} s_{1\,i} & s_{1\,i+1} \\ s_{2\,i} & s_{2\,i+1} \end{bmatrix},$$

where the columns of the matrix $\mathbf{S}_i$ are state vectors taken at the left and right cell interface, respectively.

The values $u(x_i - 0)$ and $u(x_{i+1} + 0)$ define the main diagonal of the matrix $\mathbf{S}_i$, while the $u(x_i + 0)$ and $u(x_{i+1} - 0)$ define the off-diagonal entries. Hence, the flux within the cell can be controlled by means of the matrix $\mathbf{S}_i$. In particular, it can be easily seen that zero off-diagonal entries of the matrix indicate the "phantom" solution, which yields incorrect flux approximation in the cell e_i.

Below we illustrate our approach with a nonlinear boundary-value problem known as the problem of mass flow in a convergent-divergent nozzle [1]. Let A(x) be the area of the nozzle, $A(x) = 1/2 + 2(x - 1/2)^2$, $0 \le x \le 1$, and let $u(x)$ be the velocity deviation.

The conservation law is

$$\frac{dF(x, u(x))}{dx} \equiv \frac{d(A(x)m(u))}{dx} = 0, \quad x \in [0, 1], \tag{18}$$

where the mass flux through the nozzle is given by

$$m(u) = \frac{1}{2}(1 - u^2). \tag{19}$$

The value $u_s = 0$ (sonic point) is a flux extremum point.

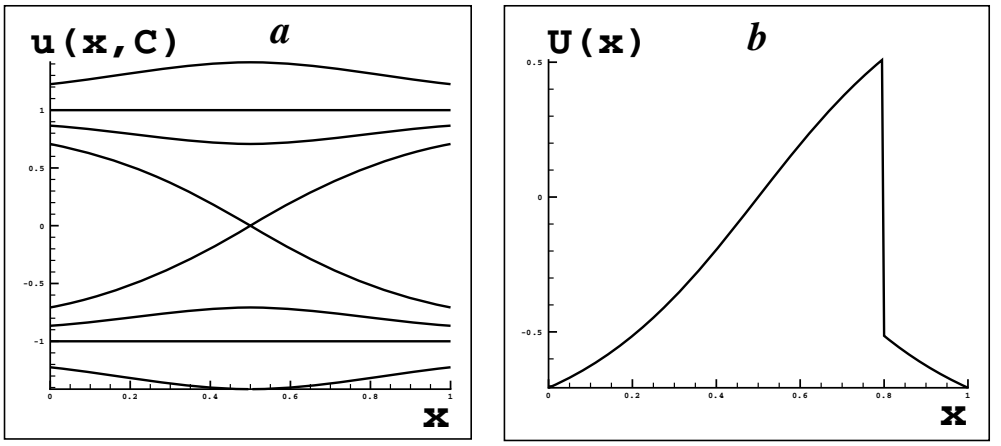


FIG. 3. *The nozzle problem.* (a) *The solution parametric field* $u(x, C)$. (b) *A discontinuous solution to the boundary-value problem* (18), (9).

A solution to the problem (18) is multivalued. The solutions are given by

$$u_{1,2}(x) = \pm\sqrt{1 - \frac{2C}{A(x)}},$$

where C is a constant. The solution parametric field $u(x, C)$ is shown in Figure 3(a). The value $x_s = 1/2$ is a solution extremum point for any $C \neq 0$ from the domain of the definition of C.

The value C is a controlling parameter for the problem. Let us choose $C = 1/4$, such that $u(x_s) = u_s$ and the point $P_s = (x_s, u_s)$ becomes the solution bifurcation point. Then the solution may be discontinuous at the point x_{sh}:

$$U(x) = \begin{cases} -\sqrt{(1 - 1/2A(x))}, & 0 \le x \le x_s \text{ or } x_{sh} + 0 \le x \le 1, \\ \sqrt{(1 - 1/2A(x))}, & x_s \le x \le x_{sh} - 0. \end{cases} \tag{20}$$

Equation (18) is solved due to the boundary condition (9) which determines the shock location x_{sh}. Integrating the solution over the domain $[0, 1]$ yields the following algebraic equation with respect to the variable x_{sh}:

$$I_1 + I_2(x_{sh}) + I_3(x_{sh}) = B,$$

where

$$I_1 = -\int_0^{x_s} \sqrt{1 - 1/2A(x)}dx, \quad I_2(x_{sh}) = \int_{x_s}^{x_{sh}} \sqrt{1 - 1/2A(x)}dx,$$

$$I_3(x_{sh}) = -\int_{x_{sh}}^{1} \sqrt{1 - 1/2A(x)}dx.$$

Solving this equation for a given value of B, the shock location can be defined. If we choose $B = -0.25$, then the shock will be located at $x_{sh} = 0.798074$. The discontinuous solution $U(x)$ is shown in Figure 3(b).

Consider the approximate solution $u_h(x)$ at the interface x_i. Given the left and right states at the interface, the Engquist–Osher numerical flux is similar to that in (11),

$$\tilde{m}(u_l,u_r)=\begin{cases} m(u_r), & u_l<0,\ u_r<0 \ \text{(subsonic case)},\\ m(u_l), & u_l>0,\ u_r>0 \ \text{(supersonic case)},\\ m(0), & u_l<0,\ u_r>0 \ \text{(sonic case)},\\ m(u_l)+m(u_r)-m(0), & u_l>0,\ u_r<0 \ \text{(shock case)}. \end{cases} \tag{21}$$

First, we use the standard DG approach to solve the boundary problem (18), (9). The problem is solved on a sequence of uniform grids. The initial guess on the first grid of eight nodes is chosen as $u_0(x)=\text{const}=-1.0$. The initial guess for the next finer grid is obtained by linear interpolation of the solution taken from a previous grid. The results with the standard DG discretization are that Newton's method fails to obtain a convergent solution for any polynomial degree $K>0$. Only a piecewise constant discretization reconstructs the discontinuous solution $U(x)$.

A DG discretization with polynomial degree $K>0$ yields a singular Jacobian on a shock cell. However, simple reduction to piecewise constant approximation near the shock is not successful in the problem and results in a divergent solution. A more thorough control of the numerical flux is required. For this purpose, we compute the matrix $\mathbf{S}_i$ on each grid cell e_i, $i=1,\dots,N$, at each Newton step. The definition (21) gives us the following formal classification of the matrix $\mathbf{S}_i$:

$$\mathbf{S}_i=\begin{bmatrix} 0 & s_{1\,i+1}\\ s_{2\,i} & 1 \end{bmatrix} \text{ (subsonic case)}, \quad \mathbf{S}_i=\begin{bmatrix} 1 & s_{1\,i+1}\\ s_{2\,i} & 0 \end{bmatrix} \text{ (supersonic case)},$$

$$\mathbf{S}_i=\begin{bmatrix} 0 & s_{1\,i+1}\\ s_{2\,i} & 0 \end{bmatrix} \text{ (sonic case)}, \qquad \mathbf{S}_i=\begin{bmatrix} 1 & s_{1\,i+1}\\ s_{2\,i} & 1 \end{bmatrix} \text{ (shock case)},$$

where $s_{1\,i+1}$ and $s_{2\,i}$ may take the value 0 or 1.

Based on the analysis of the state matrix $\mathbf{S}_i$, the correction algorithm, which eliminates nonphysical flux extrema for the problem (18), (9), is as follows.

1. [illegible] $u_h(x)$ [illegible] e_i, $i=1,\dots,N$ [illegible]
2. [illegible] $\mathbf{S}_i$ [illegible] e_i, $i=1,\dots,N$ [illegible] $\mathbf{S}_i$
3. [illegible] e_i [illegible]
   - 3.1 $s_{2i}\neq 1$, $s_{1\,i+1}\neq 0$ [illegible] $\mathbf{S}_i$
   - 3.2 $s_{2i}\neq 0$, $s_{1\,i+1}\neq 1$ [illegible] $\mathbf{S}_i$
   - 3.3 $s_{2i}=0$, $s_{1\,i+1}=0$ [illegible] $\mathbf{S}_i$ [illegible]
   - 3.4 $\mathbf{S}_i$ [illegible]
4. [illegible] e_i
$$u_h^{lin}(x)=u_{1i}+(u_{2\,i+1}-u_{1i})\phi_1(x).$$
5. [illegible] e_i [illegible] e_i
$$u_h^{c}(x)=\frac{1}{h_i}\int_{x_i}^{x_{i+1}} u_h^{lin}(x)dx.$$
6. [illegible] $u_h(x)$ [illegible]

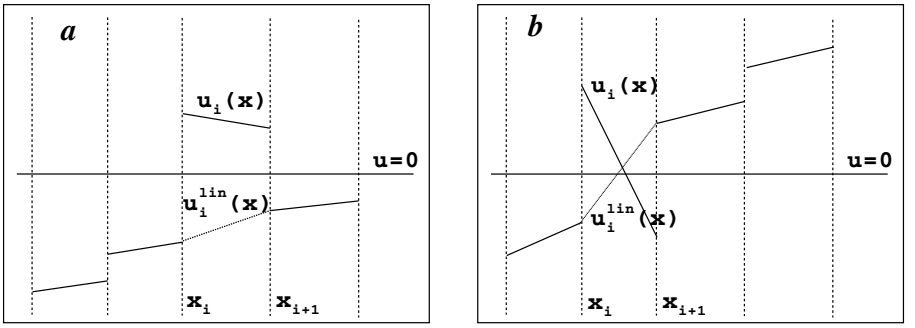


FIG. 4. *Examples of nonphysical flux extrema in the problem* (18), (9). (a) *The subsonic solution overshoot produces a sonic point and a shock at the cell interfaces.* (b) *The sonic solution overshoot produces a shock at the interior point of the cell and another sonic point at the cell interface.*

The above algorithm traces flux extrema over the grid for a transient solution at each Newton iteration. For the subsonic and supersonic cases, i.e., in the regions where the flux $m(u)$ is a monotone function, it is sufficient to control off-diagonal entries of $\mathbf{S}_i$ to ensure that there is no local bifurcation in the cell. We require that a transient solution generates the state matrix on the cell e_i as follows:

$$\mathbf{S}_i = \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \text{ (subsonic case)}, \quad \mathbf{S}_i = \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \text{ (supersonic case)}.$$

Let, for instance, the "subsonic" matrix be

$$\mathbf{S}_i = \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}.$$

This matrix is related to the solution shown in Figure 4(a). The matrix $\mathbf{S}_i$ indicates that a local solution overshoot appears on the cell e_i. That overshoot produces two nonphysical flux extrema (a sonic point and a shock) which must be eliminated. For this purpose, we linearly interpolate a transient solution on the cell e_i between the points $u(x_i - 0)$ and $u(x_{i+1} + 0)$ (see Figure 4(a)).

For the sonic case, we must eliminate the state matrix

$$\mathbf{S}_i = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix},$$

which indicates a "phantom" sonic solution. Again, the solution on the cell e_i will be linearly interpolated between the points $u(x_i - 0)$ and $u(x_{i+1} + 0)$ if a sonic cell yields the above matrix (see Figure 4(b)). The other "sonic" matrices are legal. The matrix $\mathbf{S}_i$, which has $s_{1\,i+1} = 1$ and $s_{2\,i} = 1$, corresponds to the sonic point inside the cell e_i, while the other two matrices indicate the sonic point at the interface.

As it has been earlier discussed, the linear solution interpolation between the points $u(x_i - 0)$ and $u(x_{i+1} + 0)$ is not sufficient for the shock. The interpolated solution eliminates nonphysical flux extrema on the cell, but it remains "phantom" in the presence of the shock. In other words, the number of degrees of freedom is not equal to the number of equations on the shock cell for the interpolated solution. Thus, in our algorithm we reduce the solution to piecewise constant approximation at the shock.

TABLE 1

The number N of Newton iterations required to converge on a given grid. N_c is the number of grid cells.

$N(K, N_c)$:	$N_c = 8$	$N_c = 16$	$N_c = 32$	$N_c = 48$	$N_c = 64$
K=0	9	5	4	3	4
K=1	9	5	5	4	4
K=2	10	5	5	4	4
K=3	10	5	4	4	4

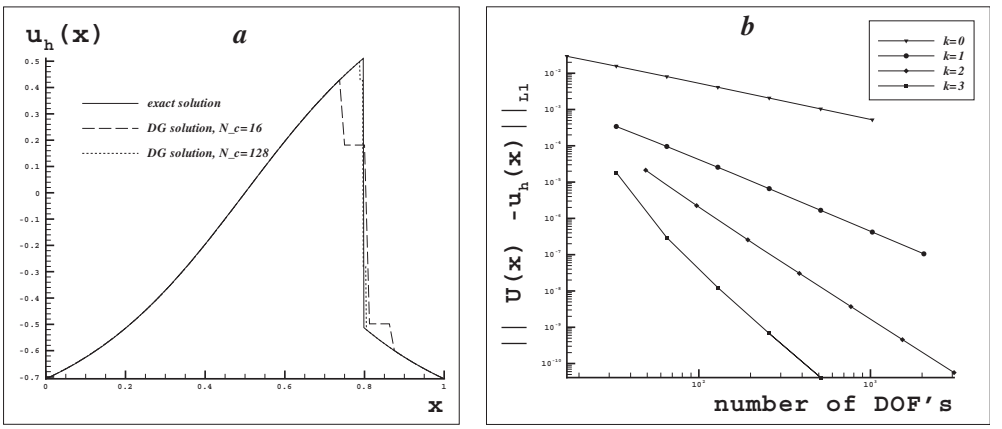


FIG. 5. *Numerical solution to the problem* (18), (9). (a) *An example of the DG solution (polynomial degree $K = 3$) on a coarse and a fine grid. N_c is the number of grid cells.* (b) *Convergence history for the DG solution with polynomial degree $K \geq 0$.*

The correction procedure allows one to obtain convergent solutions for DG $K > 0$ discretization schemes. Once the flux correction has been performed, the Newton method rapidly converges to the approximate solution. The number of Newton iterations required to converge on a given grid is displayed in Table 1 for the polynomial degree $K \geq 0$.

The approximate DG solution with polynomial degree $K = 3$ obtained by the flux correction is shown in Figure 5(a) on a coarse uniform grid of 16 cells and a fine grid of 128 cells. According to the correction algorithm, the shock is smeared over two adjacent grid cells, as an uncorrected solution has the shock at the grid interface at the final Newton step.

The convergence history on a sequence of uniform grids is plotted in Figure 5(b) for polynomial degree $K \geq 0$. The L_1-norm of the solution error,

$$||err||_{L_1} = \int_0^1 |U(x) - u_h(x)|dx,$$

is computed in regions where the solution is smooth (i.e., grid cells, which produce a shock state matrix, are not taken into account). The error norm is shown in the logarithmic scale. It can be seen from the convergence plots that the suggested algorithm keeps the order of approximation. The polynomial degree of the approximate solution is reduced only for a transient solution at the current Newton step. Once the solution is correct, the original polynomial degree will be restored on the cell at the next iterations. This approach allows one to obtain the optimal order of the convergence for high order DG discretizations.

6. Concluding remarks. We have considered high order DG schemes for steady state solutions. It has been shown that flux approximation near extremum points may generate spurious solution oscillations. Physical flux extrema require careful treatment to avoid a singular Jacobian in a steady state problem. Besides, false flux extrema may appear in a transient solution, when Newton's method is used to solve the problem. A high order DG discretization needs flux monitoring over each grid cell in order to eliminate nonphysical flux extrema.

The requirement of careful flux approximation makes Newton's method hardly appropriate for those steady state problems which do not have stabilization terms (e.g., diffusion and/or source terms) providing nonzero diagonal entries in the Jacobian. Although the flux control algorithm presented in the paper allows one to avoid a singular matrix in the one-dimensional case, it does not seem to always be efficient for multidimensional problems, where the construction of the state matrix on each grid cell becomes a complicated task.

The results of our paper confirm that a reasonable alternative to Newton's method is to use a time marching approach in order to obtain a steady state solution. It has been discussed in the paper that the time derivative can be considered as a stabilization term for high order DG schemes. However, an ill-conditioned Jacobian may appear at the end of the time stepping process when we approach "quasi-Newton" iterations. Thus, care should be taken of the flux approximation even in the case that a time stepping algorithm is used, and the issue of the numerical flux for high order DG discretizations requires further study when steady state problems are considered.

Acknowledgments. The author acknowledges the helpful discussions with F. T. Johnson, V. Venkatakrishnan, and other members of the GGNS team.

REFERENCES

[1] J. D. Anderson, Jr., *Fundamentals of Aerodynamics*, McGraw–Hill, New York, 1991.
[2] S. Balay, K. Buschelman, W. Gropp, D. Kaushik, M. Knepley, L. McInnes, B. Smith, and H. Zhang, *PETSc* Web page, http://www.mcs.anl.gov/petsc (2001).
[3] B. Cockburn, *Discontinuous Galerkin methods for convection-dominated problems*, in High-Order Discretization Methods in Computational Physics, Lect. Notes Comput. Sci. Eng. 9, T. Barth and H. Deconinck, eds., Springer-Verlag, Heidelberg, 1999, pp. 69–224.
[4] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, *The development of discontinuous Galerkin methods*, in Discontinuous Galerkin Methods. Theory, Computation and Applications, Lect. Notes Comput. Sci. Eng. 11, B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds., Springer-Verlag, New York, 2000, pp. 3–50.
[5] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws.* II. *General framework*, Math. Comp., 52 (1989), pp. 411–435.
[6] H. Hoteit et al., *New two-dimensional slope limiters for discontinuous Galerkin methods on arbitrary meshes*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 2566–2593.
[7] A. G. Kulikovskii, N. V. Pogorelov, and A. Yu. Semenov, *Mathematical Aspects of Numerical Solution of Hyperbolic Systems*, Chapman & Hall/CRC Monogr. Surv. Pure Appl. Math. 188, Chapman & Hall/CRC, Boca Raton, FL, 2001.
[8] R. J. LeVeque, *Numerical Methods for Conservation Laws*, Birkhäuser-Verlag, Basel, Switzerland, 1992.
[9] R. B. Lowrier, *Compact Higher-Order Numerical Methods for Hyperbolic Conservation Laws*, Ph.D. thesis, The University of Michigan, Ann Arbor, MI, 1996.
[10] B. Engquist and S. Osher, *One-sided difference equations for nonlinear conservation laws*, Math. Comp., 36 (1981), pp. 321–352.
[11] W. H. Reed and T. R. Hill, *Triangular Mesh Methods for the Neutron Transport Equation*, Technical Report LA-UR-73-479, Los Alamos National Laboratory, Los Alamos, NM, 1973.

[12] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Kent, UK, 1995.
[13] G. Strang and G. J. Fix, *An Analysis of the Finite Element Method*, Prentice–Hall, Englewood Cliffs, NJ, 1973.
[14] J. J. W. van der Vegt and H. van der Ven, *Space-time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows.* I. *General formulation*, J. Comput. Phys., 182 (2002), pp. 546–585.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1347–1370
© 2006 Society for Industrial and Applied Mathematics

WAVELET GALERKIN SCHEMES FOR BOUNDARY INTEGRAL EQUATIONS—IMPLEMENTATION AND QUADRATURE*

HELMUT HARBRECHT† AND REINHOLD SCHNEIDER†

Abstract. In the present paper we consider the fully discrete wavelet Galerkin scheme for the fast solution of boundary integral equations in three dimensions. It produces approximate solutions within discretization error accuracy offered by the underlying Galerkin method at a computational expense that stays proportional to the number of unknowns. We focus on algorithmical details of the scheme, in particular on numerical integration of relevant matrix coefficients. We illustrate the proposed algorithm by numerical results.

Key words. boundary integral equations, biorthogonal wavelet bases, matrix compression, numerical integration

AMS subject classifications. 47A20, 65F10, 65F50, 65N38, 65R20

DOI. 10.1137/S1064827503429387

Introduction. Various problems in science and engineering can be formulated as boundary integral equations. In general, boundary integral equations are solved numerically by the boundary element method (BEM). For example, BEM is a favorable approach for the treatment of exterior boundary value problems. Nevertheless, traditional discretizations of integral equations suffer from a major disadvantage. The associated system matrices are densely populated. Therefore, the complexity for solving such equations is at least $\mathcal{O}(N_J^2)$, where N_J denotes the number of equations. This fact seriously restricts the maximal size of the linear equations.

Modern methods for the fast solution of BEM reduce the complexity to a suboptimal rate $\mathcal{O}(N_J \log^\alpha N_J)$ or even an optimal rate $\mathcal{O}(N_J)$. Prominent examples for such methods are the [16], the [18], $\mathcal{H}$ [17], or the [1, 6, 10, 11, 31]. In fact, a Galerkin discretization with wavelet bases results in quasi-sparse matrices; i.e., most matrix entries are negligible and can be treated as zero. Discarding these nonrelevant matrix entries is called matrix compression. As shown in [6, 31] only $\mathcal{O}(N_J)$ significant matrix entries remain.

Concerning boundary integral equations, a strong effort has been expended on the construction of appropriate wavelet bases on surfaces [8, 12, 13, 14, 20, 26]. In order to achieve the optimal complexity of the wavelet Galerkin scheme, wavelet bases are required which provide a sufficiently large number of vanishing moments. Our realization is based on biorthogonal spline wavelets derived from the multiresolution developed in [3]. These wavelets are advantageous since the regularity of the duals is known [33]. Moreover, the duals are compactly supported, which preserves the linear complexity of the fast wavelet transform also for its inverse. This is an important task for the coupling of the finite element method and BEM; cf. [21, 22]. Additionally,

*Received by the editors June 11, 2003; accepted for publication (in revised form) June 8, 2005; published electronically January 6, 2006. This work is supported in part by the European Community's Human Potential Programme under contract HPRN-CT-202-00286 (BREAKING COMPLEXITY) and the SFB 393, "Numerical Simulation on Massive Parallel Computers," funded by the Deutsche Forschungsgemeinschaft.

http://www.siam.org/journals/sisc/27-4/42938.html

†Institute of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, Olshausenstr. 40, 24098 Kiel, Gemany (hh@numerik.uni-kiel.de, rs@numerik.uni-kiel.de).

in view of the discretization of operators of positive order, e.g., the hypersingular operator, globally continuous wavelets are available [2, 4, 12, 20]. We refer the reader to [23] for the details on the wavelet bases employed for our actual implementation of the wavelet Galerkin scheme.

In the present paper we describe algorithmical details of a fully discrete wavelet Galerkin scheme for the fast solution of boundary integral equations. Particularly, we realized the so-called second compression to achieve linear complexity [6, 31]. The efficient computation of the relevant matrix coefficients turned out to be the cornerstone for the successful application of the wavelet Galerkin method [20, 27, 31]. A fully discrete Galerkin scheme based on numerical quadrature is also required to treat nontrivial geometries. We expended much effort to realize an efficient quadrature strategy. Supposing that the given manifold is piecewise analytic, one applies an *hp*-quadrature scheme [20, 27, 31, 32] based on tensor product Gauss–Legendre quadrature formulas to arrive at an algorithm with asymptotically linear complexity without compromising the accuracy of the Galerkin scheme. However, a careful organization of the quadrature is indispensable to avoid wasteful and multiple calculations that affect the performance. We propose an element-based quadrature similar to the traditional single-scale Galerkin scheme. Especially we require only the quadrature routines of the traditional scheme to compute element-element interactions, except for the extension to different degrees of quadrature to deal with elements on different levels. The *h*-refinement is realized by replacing an element-element interaction by the element-element interactions related to the sons of the elements. Combined with a [illegible] to reuse previously computed element-element interactions (cf. sections 8 and 9) the work for quadrature decreases significantly. We emphasize that the proposed strategy greatly reduces the break-even point. By numerical experiments, considering the sphere and a nontrivial gearwheel, we confirm that we proceeded in developing a fast solver for the considered class of problems.

The outline of the present paper is as follows. First, in section 1, we introduce the class of problems under consideration. Then we provide the wavelet bases on manifolds in section 2 and recall the matrix compression in section 3. The next sections are concerned with implementational aspects: In section 4 a suitable data structure is introduced to handle a wide class of wavelets, in section 5 the numerical evaluation of distances is performed, in section 6 the compression pattern is set up, and in section 7 the computation of the compressed system matrix is reduced to the computation of certain element-element interactions which are investigated in section 8. Section 9 is concerned with the quadrature of element-element interactions which correspond mostly to nearly singular and singular integrals. We prove that, using our quadrature strategy, the compressed system matrix is computed within linear complexity. In section 10 we present numerical results employing both piecewise constant and bilinear wavelets.

We shall frequently write $a \lesssim b$ to express that a is bounded by a constant multiple of b, uniformly with respect to all parameters on which a and b may depend. Then $a \sim b$ means $a \lesssim b$ and $b \lesssim a$.

1. Problem formulation and preliminaries. We consider boundary integral equations on a closed boundary surface Γ of a three-dimensional domain $\Omega \subset \mathbb{R}^3$,

$$Au = f \quad \text{on } \Gamma, \tag{1.1}$$

where the boundary integral operator

$$Au(x) = \int_\Gamma k(x,y)u(y)d\Gamma_y$$

is assumed to be an operator of order $2q$, that is, $A : H^q(\Gamma) \to H^{-q}(\Gamma)$. The kernel functions under consideration are supposed to be piecewise smooth as functions in the variables x, y, apart from the diagonal $\{(x, y) \in \Gamma \times \Gamma : x = y\}$, and may have a singularity on the diagonal. Such kernels arise, for instance, by applying a boundary integral formulation to a second order elliptic problem. In general, they decay like a negative power of the distance of the arguments that depends on the operator order $2q$.

Throughout the remainder of this paper we shall assume that the boundary manifold Γ is given as a parametric surface consisting of smooth patches. More precisely, let $\square := [0,1]^2$ denote the unit square. The manifold $\Gamma \in \mathbb{R}^3$ is partitioned into a finite number of

$$\Gamma = \bigcup_{i=1}^{M} \Gamma_i, \qquad \Gamma_i = \gamma_i(\square), \qquad i = 1, 2, \ldots, M, \tag{1.2}$$

where each $\gamma_i : \square \to \Gamma_i$ defines a diffeomorphism of $\square$ onto Γ_i. The intersection $\Gamma_i \cap \Gamma_{i'}$, $i \neq i'$, of two different patches is supposed to be either empty or a common edge or vertex.

A mesh of level j on Γ is induced by dyadic subdivisions of depth j of the unit square into 4^j squares

$$\square_{j,k} := [2^{-j}k_1, 2^{-j}(k_1+1)] \times [2^{-j}k_2, 2^{-j}(k_2+1)] \subseteq \square,$$

where $k = (k_1, k_2)$ with $0 \le k_1, k_2 < 2^j$. This generates $4^j M$ (or elementary domains) $\Gamma_{i,j,k} := \gamma_i(\square_{j,k}) \subseteq \Gamma_i$, $i = 1, \ldots, M$.

In order to ensure that the collection of elements $\{\Gamma_{i,j,k}\}$ on the level j forms a regular mesh on Γ, the parametric representation is subjected to the following : A bijective, affine mapping $\Xi : \square \to \square$ exists such that for all $x = \gamma_i(s)$ on a common edge of Γ_i and $\Gamma_{i'}$ it holds that $\gamma_i(s) = (\gamma_{i'} \circ \Xi)(s)$. In other words, the diffeomorphisms γ_i and $\gamma_{i'}$ yield the same parametrization of the common edge except for orientation. This setting is frequently used and well understood in computer aided geometric design; cf. [24, 28], for example. A conforming mesh of a gearwheel is depicted in Figure 1.1.

Denoting the surface measure by $\kappa_i(s) := \|\partial_{s_1}\gamma_i(s) \times \partial_{s_2}\gamma_i(s)\|$, the canonical inner product in $L^2(\Gamma)$ is given by

$$\langle u, v \rangle := \int_\Gamma u(x)v(x)d\Gamma_x = \sum_{i=1}^{M} \int_\square u\big(\gamma_i(s)\big)v\big(\gamma_i(s)\big)\kappa_i(s)ds.$$

The associated Sobolev spaces are denoted by $H^t(\Gamma)$, endowed with the norms $\|\cdot\|_t$, where for $t < 0$ it is understood that $H^t(\Gamma) = (H^{-t}(\Gamma))'$. Of course, depending on the global smoothness of the surface, the range of permitted $t \in \mathbb{R}$ is limited to $t \in (-t_\Gamma, t_\Gamma)$.

In what follows it will be rather convenient to have access to the $\gamma_{i,j,k} : \square \to \Gamma_{i,j,k}$ given by

$$\gamma_{i,j,k}(s) := \gamma_i\big(2^{-j}\left[\begin{smallmatrix} k_1+s_1 \\ k_2+s_2 \end{smallmatrix}\right]\big), \qquad s = \left[\begin{smallmatrix} s_1 \\ s_2 \end{smallmatrix}\right] \in \square. \tag{1.3}$$

For $m \in \mathbb{N}$ we conclude that

$$\partial_{s_n}^m \gamma_{i,j,k}(s) = 2^{-jm}\,(\partial_{s_n}^m \gamma_i)\big(2^{-j}\left[\begin{smallmatrix} k_1+s_1 \\ k_2+s_2 \end{smallmatrix}\right]\big), \qquad n = 1, 2; \tag{1.4}$$

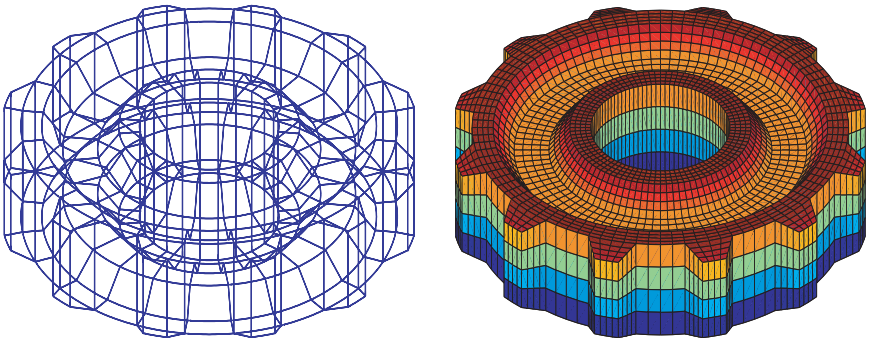


FIG. 1.1. *A gearwheel represented by* 336 *patches and the associated mesh on the level* 2.

in particular, the corresponding surface measure satisfies

$$\kappa_{i,j,k}(s) := \big\|\partial_{s_1}\gamma_{i,j,k}(s) \times \partial_{s_2}\gamma_{i,j,k}(s)\big\| = 2^{-2j}\kappa_i\big(2^{-j}\left[\begin{smallmatrix}k_1+s_1\\ k_2+s_2\end{smallmatrix}\right]\big). \tag{1.5}$$

We can now specify the kernel functions. To this end, we denote by $\alpha = (\alpha_1, \alpha_2)$ and $\beta = (\beta_1, \beta_2)$ multi-indices of dimension two and define $|\alpha| := \alpha_1 + \alpha_2$.

DEFINITION 1.1. [illegible] $k(x,y)$ [illegible] $2q$ [illegible]

$$k_{i,i'}(s,t) := k\big(\gamma_i(s), \gamma_{i'}(t)\big)\kappa_i(s)\kappa_{i'}(t), \qquad s,t \in \square, \qquad 1 \le i, i' \le M, \tag{1.6}$$

[illegible]

$$|\partial_s^\alpha \partial_t^\beta k_{i,i'}(s,t)| \le c_{\alpha,\beta}\|\gamma_i(s) - \gamma_{i'}(t)\|^{-(2+2q+|\alpha|+|\beta|)},$$

[illegible] $2 + 2q + |\alpha| + |\beta| > 0$

We emphasize that this definition requires patchwise smoothness but [illegible] global smoothness of the geometry. The surface itself needs to be only Lipschitz. Generally, under this assumption, the kernel of a boundary integral operator A of order $2q$ is a standard kernel of order $2q$. Hence, we may assume this property in what follows. We shall encounter further specifications in connection with numerical integration.

2. Biorthogonal wavelet bases. The nested trial spaces $V_j \subset V_{j+1}$ that we shall employ in the Galerkin scheme are the spaces of piecewise constant or bilinear functions on the given partition. These trial spaces have the [illegible] $d = 1$ and $d = 2$ in the case of the piecewise constants and bilinears, respectively; i.e., $\inf_{v_j \in V_j} \|v - v_j\|_0 \lesssim 2^{-jd}\|v\|_d$. They are spanned by so-called [illegible] $\Phi_j = \{\phi_{j,k} : k \in \Delta_j\}$ which can be specified as follows: On the level j, we find for each element $\Gamma_{i',j,k'}$ a piecewise constant scaling function $\phi_{j,k}$ with

$$\phi_{j,k}\big|_{\Gamma_{i',j,k'}} \equiv 2^j \tag{2.1}$$

and $\phi_{j,k}(x) = 0$ elsewhere. To describe the canonical piecewise bilinear scaling functions, we define four bilinear shape functions on the unit square:

$$\begin{aligned} p_1^\square(s) &:= (1-s_1)(1-s_2), & p_2^\square(s) &:= s_1(1-s_2),\\ p_3^\square(s) &:= s_1 s_2, & p_4^\square(s) &:= (1-s_1)s_2. \end{aligned} \tag{2.2}$$

Thus, $\phi_{j,k}$ is equal to 2^j in one node and equal to zero in the remaining nodes and, if its support contains an element $\Gamma_{i',j,k'}$, we find an $m \in \{1,2,3,4\}$ such that

$$\phi_{j,k}\big|_{\Gamma_{i',j,k'}}(x) = 2^j p_m^{\square}(s), \qquad x = \gamma_{i',j,k'}(s) \in \Gamma_{i',j,k'}. \tag{2.3}$$

Continuity is supposed on each patch, but along the interfaces of the patches we may consider double nodes or continuity. Notice that our definition yields the L^2-normalization $\|\phi_{j,k}\|_0 \sim 1$.

Associated with the [illegible] $\{V_j\}_{j\geq j_0}$ is always a [illegible] multiresolution sequence $\{\widetilde{V}_j\}_{j\geq j_0}$ which is generated by dual bases $\widetilde{\Phi}_j = \{\widetilde{\phi}_{j,k} : k \in \Delta_j\}$; i.e., one has $\langle \phi_{j,k}, \widetilde{\phi}_{j,k'}\rangle = \delta_{k,k'}$, $k,k' \in \Delta_j$. Here and below j_0 always stands for some fixed coarsest level of resolution that may depend on Γ. For the current type of boundary surfaces Γ the $\Phi_j, \widetilde{\Phi}_j$ are generated by first constructing dual pairs of single-scale bases on the interval $[0,1]$, using the dual components from [3] adapted to the interval [9]. Tensor products yield corresponding dual pairs on $\square$. Using the parametric liftings γ_i and gluing across patch boundaries leads to globally continuous single-scale bases $\Phi_j, \widetilde{\Phi}_j$ on Γ [2, 4, 13]. The dual spaces have approximation order $\widetilde{d} \geq d$ such that $d + \widetilde{d}$ is even.

Given the single-scale bases $\Phi_j, \widetilde{\Phi}_j$, one now can construct biorthogonal [illegible] $\Psi_j = \{\psi_{j,k} : k \in \nabla_j := \triangle_{j+1} \setminus \triangle_j\}$, $\widetilde{\Psi}_j = \{\widetilde{\psi}_{j,k} : k \in \nabla_j\}$, i.e., $\langle \psi_{j,k}, \widetilde{\psi}_{j',k'}\rangle = \delta_{(j,k),(j',k')}$, such that

$$\begin{aligned} W_j &:= \operatorname{span}\{\Psi_j\} \perp \widetilde{V}_j, \qquad & V_{j+1} &= V_j \oplus W_j, \\ \widetilde{W}_j &:= \operatorname{span}\{\widetilde{\Psi}_j\} \perp V_j, \qquad & \widetilde{V}_{j+1} &= \widetilde{V}_j \oplus \widetilde{W}_j, \end{aligned}$$

and $\operatorname{diam}\operatorname{supp}\psi_{j,k} \sim \operatorname{diam}\operatorname{supp}\widetilde{\psi}_{j,k} \sim 2^{-j}$ $(j \geq j_0)$; see, e.g., [2, 4, 12, 13] and [20, 23] for particularly useful local representations of important construction ingredients. We suppose these complement bases are normalized in $L^2(\Gamma)$.

A biorthogonal or [illegible] pair of wavelet bases is now obtained by taking the coarse single-scale basis and the union of the complement bases,

$$\Psi = \bigcup_{j\geq j_0-1} \Psi_j, \qquad \widetilde{\Psi} = \bigcup_{j\geq j_0-1} \widetilde{\Psi}_j,$$

where we have set for convenience $\Psi_{j_0-1} := \Phi_{j_0}$, $\widetilde{\Psi}_{j_0-1} := \widetilde{\Phi}_{j_0}$. Of course, in the infinite-dimensional case the notion of basis has to be made more specific. The key feature of the wavelet basis is now the fact that $\Psi, \widetilde{\Psi}$ are actually [illegible] in $L^2(\Gamma)$.

From biorthogonality and the fact that the dual spaces $\widetilde{V}_j$ have the approximation order $\widetilde{d}$ one infers [illegible] or the [illegible] of the primal wavelets

$$|\langle v, \psi_{j,k}\rangle| \lesssim 2^{-j(\widetilde{d}+1)} |v|_{W^{\widetilde{d},\infty}(\operatorname{supp}\psi_{j,k})}. \tag{2.4}$$

Here $|v|_{W^{\widetilde{d},\infty}(\Omega)} := \sup_{|\alpha|=\widetilde{d},\, x\in\Omega} |\partial^\alpha v(x)|$ denotes the seminorm in $W^{\widetilde{d},\infty}(\Omega)$. The fact that the concept of biorthogonality allows us to choose the order $\widetilde{d}$ of vanishing moments higher than the approximation order d is essential for deriving optimal compression strategies that could not be realized by orthonormal bases.

3. Matrix compression. We shall be concerned with the Galerkin method for the solution of the given boundary integral equation (1.1): Find $u_J \in V_J$ solving the variational problem

$$\langle Au_J, v_J \rangle = \langle f, v_J \rangle \quad \text{for all} \quad v_J \in V_J.$$

Traditionally this equation is discretized by the single-scale basis of V_J which yields a densely populated system matrix. Using instead wavelets with a sufficiently strong cancellation property (2.4), the system matrix becomes quasi-sparse and most matrix coefficients are negligible without compromising the order of convergence of the Galerkin scheme [6, 31].

But before we formulate this result, we introduce the following abbreviation:

$$\Omega_{j,k} := \operatorname{conv\,hull}(\operatorname{supp} \psi_{j,k}), \qquad \Omega'_{j,k} := \operatorname{sing\,supp} \psi_{j,k}. \tag{3.1}$$

Notice that the first expression denotes the convex hull of the support of a wavelet with respect to the Euclidean space $\mathbb{R}^3$. The second expression indicates the [illegible], i.e., that subset of Γ where the wavelet is not smooth.

THEOREM 3.1 (a priori compression). [illegible] $\Omega_{j,k}$ [illegible] $\Omega'_{j,k}$ [illegible] (3.1) [illegible] $\mathbf{A}_J$ [illegible] A [illegible]

$$[\mathbf{A}_J]_{(j,k),(j',k')} := \begin{cases} 0, & \operatorname{dist}(\Omega_{j,k}, \Omega_{j',k'}) > \mathcal{B}_{j,j'} \quad j, j' \geq j_0, \\ 0, & \operatorname{dist}(\Omega_{j,k}, \Omega_{j',k'}) \lesssim 2^{-\min\{j,j'\}} \\ & \operatorname{dist}(\Omega'_{j,k}, \Omega_{j',k'}) > \mathcal{B}'_{j,j'}, \quad j' > j \geq j_0 - 1, \\ & \operatorname{dist}(\Omega_{j,k}, \Omega'_{j',k'}) > \mathcal{B}'_{j,j'}, \quad j > j' \geq j_0 - 1, \\ \langle A\psi_{j',k'}, \psi_{j,k} \rangle & \text{[illegible]} \end{cases} \tag{3.2}$$

[illegible]

$$a, a' > 1, \qquad d < d' < \widetilde{d} + 2q, \tag{3.3}$$

[illegible] $\mathcal{B}_{j,j'}$ [illegible] $\mathcal{B}'_{j,j'}$ [illegible]

$$\begin{aligned} \mathcal{B}_{j,j'} &= a \ \max\left\{ 2^{-\min\{j,j'\}}, 2^{\frac{2J(d'-q)-(j+j')(d'+\tilde{d})}{2(\tilde{d}+q)}} \right\}, \\ \mathcal{B}'_{j,j'} &= a' \max\left\{ 2^{-\max\{j,j'\}}, 2^{\frac{2J(d'-q)-(j+j')d'-\max\{j,j'\}\tilde{d}}{\tilde{d}+2q}} \right\}. \end{aligned} \tag{3.4}$$

[illegible]

$$\|u - u_J\|_{2q-d} \lesssim 2^{-2J(d-q)} \|u\|_d \tag{3.5}$$

[illegible] u_J [illegible] u [illegible] Γ [illegible]

In [6, 20] we presented a general theorem which shows that the overall complexity of assembling the compressed system matrix can be kept of the order $\mathcal{O}(N_J)$, $N_J = \dim V_J$, even when a computational cost of logarithmic order is allowed for each entry. This theorem will be used later as the essential ingredient in proving that the quadrature strategy proposed in section 9 scales linearly.

THEOREM 3.2. $\mathbf{A}_J$ (3.2) $\mathcal{O}(N_J)$ $\alpha \geq 0$ $\mathcal{O}\big([2J-j-j']^4\big)$ $\langle A\psi_{j',k'}, \psi_{j,k}\rangle$

Numerical integration of relevant matrix coefficients has to be performed with sufficient accuracy. It turns out that the higher the level of resolution the lower the required accuracy. In accordance with [6, 20] we can formulate the following sufficient condition to retain the optimal order of convergence of the Galerkin scheme.

THEOREM 3.3. $\mathbf{A}_J$ 3.1 $\widetilde{\mathbf{A}}_J$

$$\big|[\mathbf{A}_J - \widetilde{\mathbf{A}}_J]_{(j,k),(j',k')}\big| \leq \varepsilon_{j,j'}$$

$\varepsilon_{j,j'}$

$$\varepsilon_{j,j'} \sim \min\Big\{2^{-|j-j'|}, 2^{-(2J-j-j')\frac{d'-q}{d+q}}\Big\} 2^{2Jq} 2^{-\delta(2J-j-j')} \tag{3.6}$$

$d' \in (d, \widetilde{d}+2q)$ (3.3) $\delta > d$ u_J (3.5) u Γ

If the order q of the boundary integral operator A is $\neq 0$, the compressed system matrix $\mathbf{A}_J$ becomes more and more ill conditioned when J increases. However, as a consequence of the norm equivalences of wavelet bases, the diagonally scaled system matrix has uniformly bounded spectral condition numbers, provided that the regularity $\widetilde{\gamma}$ of the dual wavelets satisfies $\widetilde{\gamma} > -q$ [5, 7, 10, 31]. Let us further remark that based on (3.6) an a posteriori compression is also available, which reduces again the number of nonzero coefficients; cf. [6, 20].

4. The data structure. In general the wavelet bases are not simply defined via tensor products of the univariate case. In particular, considering globally continuous wavelets, a single wavelet might be supported on several patches. This requires a suitable data structure to handle such wavelet bases.

4.1. The element tree. According to section 1 the elements $\Gamma_{i,j,k} = \gamma_i(\square_{j,k})$ are the images of the refined unit square under the parametric liftings γ_i. We introduce a hierarchical element tree with respect to "$\subseteq$" as follows. Starting with the M patches $\Gamma_i := \Gamma_{i,0,(0,0)}$ as the first generation, each element $\Gamma_{i,j,k}$ has the four sons $\Gamma_{i,j+1,k'} \subseteq \Gamma_{i,j,k}$, $k' \in \big\{(2k_1, 2k_2), (2k_1+1, 2k_2), (2k_1+1, 2k_2+1), (2k_1, 2k_2+1)\big\}$. Denoting by J the level of discretization, the tree consists of $J+1$ generations.

Notice that this element tree is given implicitly. The number of all elements is given by $N_\Gamma := \frac{M}{3}(4^{J+1}-1)$. Hence, we may utilize the bijective mapping

$$\lambda := \theta(i,j,k) = \frac{M}{3}(4^j - 1) + 4^j M i + 2^j k_2 + k_1 \tag{4.1}$$

to indicate the element $\Gamma_{i,j,k}$ uniquely by an integer $\lambda \in \{0, 1, \ldots, N_\Gamma - 1\}$. This numbering corresponds to indexing the element tree line-by-line. For the sake of brevity we frequently will write Γ_λ instead of $\Gamma_{i,j,k}$. Moreover, it is rather convenient to set $|\lambda| := j$.

4.2. The wavelet representation. The common way to perform the quadrature for, e.g., piecewise bilinear functions is element-based; that is, for all pairs of

elements one computes the element-element interactions of the associated shape functions and updates the system matrix correspondingly. Roughly speaking, a basis function is represented by in general four quadrilaterals (we call them the elements) and certain weights (namely equal to 2^j in the common vertex and zero elsewhere) of the shape functions associated with the vertices of these quadrilaterals. Since the wavelets of the level j are contained in V_{j+1}, we can represent the wavelet $\psi_{j,k}$ similarly by a set of elements $\Gamma_{i',j+1,k'} \subset \operatorname{supp} \psi_{j,k}$ and certain weight factors of the associated shape functions. This enables us to perform an element-based quadrature. It turns out that parts of the support of the wavelets are smooth even on the previous level, i.e., with respect to the elements $\Gamma_{i',j,k'}$; cf. [20, 23], for example. Therefore, we recommend a two level representation of the wavelets, which increases the performance of the wavelet Galerkin scheme enormously since fewer element-element interactions have to be evaluated.

More precisely, for each wavelet there exists a set $\mathcal{L}_{j,k}$ of elements $\Gamma_\lambda \in \operatorname{supp} \psi_{j,k}$, $|\lambda| \in \{j, j+1\}$, satisfying

$$\operatorname{supp} \psi_{j,k} = \bigcup_{\Gamma_\lambda \in \mathcal{L}_{j,k}} \Gamma_\lambda \subseteq \Omega_{j,k}, \qquad \Omega'_{j,k} \subseteq \bigcup_{\Gamma_\lambda \in \mathcal{L}_{j,k}} \partial\Gamma_\lambda \tag{4.2}$$

such that $\psi_{j,k} \circ \gamma_\lambda$ is either constant ($d = 1$),

$$\psi_{j,k}(x) \equiv 2^{|\lambda|} \omega_{j,k,\lambda}, \quad x = \gamma_\lambda(s), \quad s \in \square, \quad \Gamma_\lambda \in \mathcal{L}_{j,k}, \tag{4.3}$$

or piecewise bilinear ($d = 2$),

$$\psi_{j,k}(x) = 2^{|\lambda|} \sum_{m=1}^{4} \omega_{j,k,\lambda,m} p_m^{\square}(s), \quad x = \gamma_\lambda(s), \quad s \in \square, \quad \Gamma_\lambda \in \mathcal{L}_{j,k}, \tag{4.4}$$

with certain weights $\omega_{j,k,\lambda} \in \mathbb{R}$ and $\omega_{j,k,\lambda,m} \in \mathbb{R}$, respectively. This is illustrated in Figure 4.1 in the case of a piecewise bilinear wavelet.

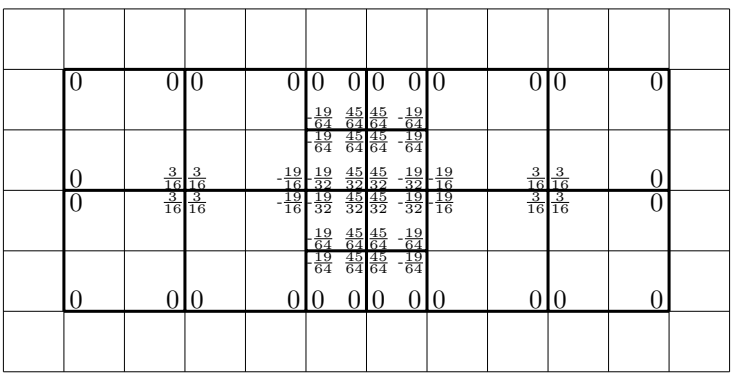


FIG. 4.1. *Element-based representation of a piecewise bilinear wavelet (four vanishing moments).*

Storing all weights is expensive, particularly for the piecewise bilinears. Since many wavelets correspond to identical masks, a suitable arrangement of the elements in the lists $\mathcal{L}_{j,k}$ induces identical lists of weights for such wavelets. Hence, we suggest storing only the different lists of weights. That way, the required memory is limited to $\mathcal{O}(1)$ since the number of different masks is independent of the level J.

Finally, in view of setting up the compression pattern, we introduce a wavelet tree defined with respect to the supports. Each wavelet has a certain number of sons with

$$\Omega_{j+1,\mathrm{son}} \subseteq \Omega_{j,\mathrm{father}} \qquad \text{if } \psi_{j+1,\mathrm{son}} \text{ is son of } \psi_{j,\mathrm{father}}. \tag{4.5}$$

We mention that the father-son relation might be described by fixed index operations independently of the geometry. However, the tree depends on the chosen wavelet construction.

5. Computing distances numerically. To determine the degrees of quadrature it is necessary to evaluate the distance between elements. Additionally, for the matrix compression, the distance between the supports of two wavelets, as well as the distance between the support of one wavelet and the singular support of another wavelet, has to be computed.

5.1. Computing distances between elements. For each element Γ_λ we determine a sphere $B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda}) := \{x \in \mathbb{R}^3 : \|x - m_{\Gamma_\lambda}\| \le r_{\Gamma_\lambda}\}$ which encloses the element. Then, the distance between elements is approximated by

$$\begin{aligned} \operatorname{dist}(\Gamma_\lambda, \Gamma_{\lambda'}) &\ge \operatorname{dist}\big(B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda}), B(m_{\Gamma_{\lambda'}}, r_{\Gamma_{\lambda'}})\big) \\ &= \max\{0, \|m_{\Gamma_\lambda} - m_{\Gamma_{\lambda'}}\| - r_{\Gamma_\lambda} - r_{\Gamma_{\lambda'}}\}. \end{aligned} \tag{5.1}$$

We start on the finest level $J = |\lambda|$ and compute the smallest sphere $B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda})$ containing the four vertices of a given element Γ_λ. Assuming that J is sufficiently large, this sphere satisfies $\Gamma_\lambda \subseteq B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda})$. Next, due to

$$\Gamma_\lambda = \bigcup_{\Gamma_{\lambda'} \text{ is son of } \Gamma_\lambda} \Gamma_{\lambda'},$$

the spheres to the elements on the lower levels are obtained recursively by determining the smallest spheres that contain the spheres of the associated sons. This ensures $\Gamma_\lambda \in B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda})$ for all $|\lambda| \le J$ even if the surface Γ is strongly curved.

In the next subsection, the computation of the distance of the singular support and the support of wavelets is reduced to the computation of the distance $\operatorname{dist}(\Gamma_{i,j,k}, \partial\Gamma_{i',j',k'})$ $(j > j')$. Clearly, we find

$$\operatorname{dist}(\Gamma_{i,j,k}, \partial\Gamma_{i',j',k'}) = \operatorname{dist}(\Gamma_{i,j,k}, \Gamma_{i',j',k'}), \qquad i \ne i', \tag{5.2}$$

which is evaluated according to (5.1). Thus, it remains to consider $i = i'$. Since γ_i defines a diffeomorphism, we deduce $\operatorname{dist}(\Gamma_{i,j,k}, \partial\Gamma_{i,j',k'}) \sim \operatorname{dist}(\square_{j,k}, \partial\square_{j',k'})$; i.e. we might reduce our problem to the unit square. Observing that the ℓ^2-norm and the ℓ^∞-norm are equivalent in $\mathbb{R}^2$, the square $\square_{j,k}$ might be interpreted as the sphere $B(m_{\square_{j,k}}, r_{\square_{j,k}})$ with the midpoint $m_{\square_{j,k}} := \gamma_i\big(2^{-(j+1)}\left[\begin{smallmatrix}2k_1+1\\2k_2+1\end{smallmatrix}\right]\big)$ and the radius $r_{\square_{j,k}} := 2^{-(j+1)}$. Therefore, we conclude

$$\operatorname{dist}(\Gamma_{i,j,k}, \partial\Gamma_{i,j',k'}) \sim \big|\|m_{\square_{j,k}} - m_{\square_{j',k'}}\|_\infty - r_{\square_{j',k'}}\big| - r_{\square_{j,k}} \tag{5.3}$$

since either $\square_{j,k} \subseteq \square_{j',k'}$ or $\square^\circ_{j,k} \cap \square^\circ_{j',k'} = \emptyset$.

5.2. Computing distances between wavelets. Computing for each wavelet the sphere $B(m_{\psi_{j,k}}, r_{\psi_{j,k}})$ with

$$B(m_{\psi_{j,k}}, r_{\psi_{j,k}}) := \inf_{B(m,r)\in\mathbb{R}^3} \left\{ B(m,r) \supseteq \bigcup_{\Gamma_\lambda\in\mathcal{L}_{j,k}} B(m_{\Gamma_\lambda}, r_{\Gamma_\lambda}) \right\}, \tag{5.4}$$

we conclude from (4.2) that $\Omega_{j,k} \subseteq B(m_{\psi_{j,k}}, r_{\psi_{j,k}})$. Thus, the distance between the wavelets $\psi_{j,k}$ and $\psi_{j',k'}$ is computed analogously to (5.1) by

$$\operatorname{dist}(\Omega_{j,k}, \Omega_{j',k'}) \geq \max\{0, \|m_{\psi_{j,k}} - m_{\psi_{j',k'}}\| - r_{\psi_{j,k}} - r_{\psi_{j',k'}}\}. \tag{5.5}$$

. . . . Of course, it suffices to compute only appropriate approximations to the infimum in (5.4). Moreover, one can use also axis-parallel boxes instead of spheres, which offers the advantage of resolving the high aspect ratios of the wavelets. However, one obtains different distances when rotating the geometry.

Next, the computation of $\operatorname{dist}(\Omega_{j,k}, \Omega'_{j',k'})$ is performed by invoking $\mathcal{L}_{j,k}$ and $\mathcal{L}_{j',k'}$. One readily infers

$$\operatorname{dist}(\Omega_{j,k}, \Omega'_{j',k'}) \gtrsim \min_{\Gamma_\lambda\in\mathcal{L}_{j,k}} \min_{\Gamma_{\lambda'}\in\mathcal{L}_{j',k'}} \big\{ \operatorname{dist}(\Gamma_\lambda, \partial\Gamma_{\lambda'}) \big\},$$

which is evaluated by (5.2) and (5.3), respectively.

6. Setting up the compression pattern. Checking the distance criteria (3.2) for each matrix coefficient in order to assemble the compressed matrix would require $\mathcal{O}(N_J^2)$ function calls. To realize linear complexity, we exploit the tree structure with respect to the supports of the wavelets to predict negligible matrix coefficients. Recall that each son $\psi_{j+1,\text{son}}$ of the wavelet $\psi_{j,\text{father}}$ satisfies $\Omega_{j+1,\text{son}} \subseteq \Omega_{j,\text{father}}$. The following observation is an immediate consequence of the relations $\mathcal{B}_{j,j'} \geq \mathcal{B}_{j+1,j'} \geq \mathcal{B}_{j+1,j+1'}$ and $\mathcal{B}'_{j,j'} \geq \mathcal{B}'_{j+1,j'}$ for $j > j'$.

LEMMA 6.1. . . $\Omega_{j+1,\text{son}} \subseteq \Omega_{j,\text{father}}$ . . $\Omega_{j'+1,\text{son}'} \subseteq \Omega_{j',\text{father}'}$ . . . . . .

1. $\operatorname{dist}(\Omega_{j,\text{father}}, \Omega_{j',\text{father}'}) > \mathcal{B}_{j,j'}$ . . . $\operatorname{dist}(\Omega_{j+1,\text{son}}, \Omega_{j',\text{father}'}) > \mathcal{B}_{j+1,j'}$ . . . $\operatorname{dist}(\Omega_{j+1,\text{son}}, \Omega_{j'+1,\text{son}'}) > \mathcal{B}_{j+1,j+1'}$
2. . . . . $j > j'$ . . $\operatorname{dist}(\Omega_{j,\text{father}}, \Omega'_{j',\text{father}'}) > \mathcal{B}'_{j,j'}$ . . . . . $\operatorname{dist}(\Omega_{j+1,\text{son}}, \Omega'_{j',\text{father}'}) > \mathcal{B}'_{j+1,j'}$

Thanks to this lemma we have to check the distance criteria only for coefficients that stem from subdivisions of calculated coefficients on a coarser level. Obviously, the resulting procedure of checking the distance criteria is then of linear complexity.

7. Assembling the system matrix. This section is concerned with assembling the relevant matrix coefficients $\langle A\psi_{j',k'}, \psi_{j,k}\rangle$. For applying product Gauss quadrature rules it is necessary to split the wavelets into the smooth parts of their supports.

First, for piecewise constant wavelets we introduce the . . . . by

$$\alpha_{\lambda,\lambda'} := 2^{|\lambda|+|\lambda'|} \int_\square \int_\square k\big(\gamma_\lambda(s), \gamma_{\lambda'}(t)\big)\kappa_\lambda(s)\kappa_{\lambda'}(t)dtds. \tag{7.1}$$

Then, from the wavelet representation (4.3) we conclude

$$\langle A\psi_{j',k'}, \psi_{j,k}\rangle = \sum_{\Gamma_\lambda\in\mathcal{L}_{j,k}} \sum_{\Gamma_{\lambda'}\in\mathcal{L}_{j',k'}} \omega_{j,k,\lambda}\omega_{j',k',\lambda'}\alpha_{\lambda,\lambda'}. \tag{7.2}$$

For piecewise bilinear wavelets, the element-element interactions are given by

$$\beta_{(m,\lambda),(m',\lambda')} := 2^{|\lambda|+|\lambda'|}\int_{\square}\int_{\square} k\big(\gamma_\lambda(s),\gamma_{\lambda'}(t)\big)p_m^{\square}(s)p_{m'}^{\square}(t)\kappa_\lambda(s)\kappa_{\lambda'}(t)dtds. \tag{7.3}$$

Thus, the representation (4.4) yields the equation

$$\langle A\psi_{j',k'},\psi_{j,k}\rangle = \sum_{\Gamma_\lambda\in\mathcal{L}_{j,k}}\sum_{\Gamma_{\lambda'}\in\mathcal{L}_{j',k'}}\sum_{m,m'=1}^{4}\omega_{j,k,\lambda,m}\omega_{j',k',\lambda',m'}\beta_{(m,\lambda),(m',\lambda')}. \tag{7.4}$$

Consequently, the computation of the stiffness matrix is reduced to the computation of element-element interactions. Moreover, since the support of a wavelet on the level j is subdivided into finitely many elements of the levels j and $j+1$, it suffices to compute the element-element interactions $\alpha_{\lambda,\lambda'}$ and $\beta_{(m,\lambda),(m',\lambda')}$ with the precision $\varepsilon_{|\lambda|,|\lambda'|}$ according to (3.3).

8. On element-element interactions. As we have seen in the last section, the assembling of the compressed system matrix reduces to the computation of element-element interactions. Utilizing the mapping θ from (4.1), these element-element interactions are identified uniquely with coefficients of a (sparse) matrix $\mathbf{Q}_J$, where, in the case of piecewise bilinear wavelets, a single entry consists of 16 double values. The pattern of $\mathbf{Q}_J$ is structured similarly to the corresponding compressed system matrix; namely, it is symmetric and [illegible]. In particular, the symmetry of the pattern suggests computing the element-element interaction of Γ_λ with $\Gamma_{\lambda'}$ and that of $\Gamma_{\lambda'}$ with Γ_λ simultaneously since the application of our quadrature algorithm yields many identical function calls.

Nearly all element-element interactions of $\mathbf{Q}_J$ represent singular or nearly singular integrals. Hence, we often have to subdivide the associated domain of integration for quadrature. We realize this subdivision efficiently by the following [illegible] yielding again element-element interactions which can be [illegible] when assembling the compressed system matrix. Considering a given element Γ_{father} with index father $= (i,j,k)$, the indices of its sons are given by

$$\begin{aligned}
&\text{son}_1 := (i,j+1,(2k_1,2k_2)), && \text{son}_2 := (i,j+1,(2k_1+1,2k_2)),\\
&\text{son}_3 := (i,j+1,(2k_1,2k_2+1)), && \text{son}_4 := (i,j+1,(2k_1+1,2k_2+1)).
\end{aligned}$$

In the case of piecewise constants, by (7.1) one readily verifies the recycling formulas

$$\begin{aligned}
\alpha_{\text{father},\lambda} &= (\alpha_{\text{son}_1,\lambda}+\alpha_{\text{son}_2,\lambda}+\alpha_{\text{son}_3,\lambda}+\alpha_{\text{son}_4,\lambda})/2,\\
\alpha_{\lambda,\text{father}} &= (\alpha_{\lambda,\text{son}_1}+\alpha_{\lambda,\text{son}_2}+\alpha_{\lambda,\text{son}_3}+\alpha_{\lambda,\text{son}_4})/2.
\end{aligned} \tag{8.1}$$

Similar formulas can be derived in the case of piecewise bilinears; for example, we have

$$\begin{aligned}
\beta_{(1,\text{father}),(m,\lambda)} &= \beta_{(1,\text{son}_1),(m,\lambda)}/2\\
&\quad+(\beta_{(2,\text{son}_1),(m,\lambda)}+\beta_{(4,\text{son}_1),(m,\lambda)}+\beta_{(1,\text{son}_2),(m,\lambda)}+\beta_{(1,\text{son}_3),(m,\lambda)})/4\\
&\quad+(\beta_{(3,\text{son}_1),(m,\lambda)}+\beta_{(4,\text{son}_2),(m,\lambda)}+\beta_{(2,\text{son}_3),(m,\lambda)}+\beta_{(1,\text{son}_4),(m,\lambda)})/8,\\
\beta_{(m,\lambda),(1,\text{father})} &= \beta_{(m,\lambda),(1,\text{son}_1)}/2\\
&\quad+(\beta_{(m,\lambda),(2,\text{son}_1)}+\beta_{(m,\lambda),(4,\text{son}_1)}+\beta_{(m,\lambda),(1,\text{son}_2)}+\beta_{(m,\lambda),(1,\text{son}_3)})/4\\
&\quad+(\beta_{(m,\lambda),(3,\text{son}_1)}+\beta_{(m,\lambda),(4,\text{son}_2)}+\beta_{(m,\lambda),(2,\text{son}_3)}+\beta_{(m,\lambda),(1,\text{son}_4)})/8.
\end{aligned} \tag{8.2}$$

Since a finite number of element-element interactions has to be calculated per relevant matrix coefficient, the number of nonzero coefficients of $\mathbf{Q}_J$ is $\mathcal{O}(N_J)$. However, we recommend not storing $\mathbf{Q}_J$ completely since this is so expensive that the computable number of unknowns is crucially restricted. Instead, we propose an element-based assembling of the system matrix. For a fixed element Γ_λ one computes and stores all relevant element-element interactions $\alpha_{\lambda,\lambda'}$ and $\alpha_{\lambda',\lambda}$ with $|\lambda| \geq |\lambda'|$ by the quadrature algorithm introduced in the next section, and likewise for the piecewise bilinears. Then, after updating the system matrix, these values can be deleted. According to $\varepsilon_{j,j'} \leq \varepsilon_{j+1,j'} \leq \varepsilon_{j+1,j'+1}$ for all $j_0 - 1 \leq j \leq j' < J$ (cf. (3.6)), the element-element interactions just have to be computed successively by starting on the coarse grid. Clearly, the proposed strategy requires a rearrangement of the loops within the wavelet Galerkin scheme. However, the number of multiple calculations is reduced enormously, while the requirement of memory is not too high.

9. Numerical integration. As we have seen in the previous sections it suffices to compute element-element interactions with respect to the domain of integration $\square \times \square$. Based on tensor product Gauss–Legendre rules we construct an adaptive quadrature algorithm which converges exponentially. In combination with Theorem 3.2 we realize linear complexity for the computation of the compressed system matrix.

9.1. Error estimates on the reference domain. For a given function $f \in C([0,1])$, we set $I^{[0,1]}f := \int_0^1 f(s)\,ds$. The g-point Gauss–Legendre formula on $[0,1]$ $Q_g^{[0,1]}f := \sum_{i=1}^g \omega_{g,i} f\big(\xi_{g,i}\big)$ applied to $f \in C^{2g}(0,1)$ produces the error

$$\big|R_g^{[0,1]}f\big| := \big|I^{[0,1]}f - Q_g^{[0,1]}f\big| = \frac{2^{4n+1}(n!)^4}{[(2n)!]^3(2n+1)}\big|f^{(2g)}(\xi)\big|, \qquad 0 < \xi < 1;$$

cf. [19], for example. Employing Stirling's formula $n! \sim \sqrt{2\pi n} n^n e^{-n}$, we arrive at the estimate

$$\big|R_g^{[0,1]}f\big| \lesssim \frac{2^{-4g}}{(2g)!} \max_{s\in[0,1]} \big|f^{(2g)}(s)\big|. \tag{9.1}$$

For $f \in C(\square)$ we define $I^\square f := \big(I^{[0,1]} \otimes I^{[0,1]}\big)f = \int_\square f(s)\,ds$ with $s = \left[\begin{smallmatrix} s_1 \\ s_2 \end{smallmatrix}\right]$. Approximating $I^\square f$ by the product Gauss–Legendre quadrature formula

$$Q_g^\square f := \big(Q_g^{[0,1]} \otimes Q_g^{[0,1]}\big)f = \sum_{i,i'=1}^g \omega_{g,i}\omega_{g,i'} f\Big(\left[\begin{smallmatrix} \xi_{g,i} \\ \xi_{g,i'} \end{smallmatrix}\right]\Big), \tag{9.2}$$

we find the following error estimate.

LEMMA 9.1. [illegible] $f \in C^{2g}(\square)$ [illegible] $R_g^\square f := I^\square f - Q_g^\square f$ [illegible] (9.2) [illegible]

$$\big|R_g^\square f\big| \lesssim \frac{2^{-4g}}{(2g)!}\Big[\max_{s\in\square}\big|\partial_{s_1}^{2g} f(s)\big| + \max_{s\in\square}\big|\partial_{s_2}^{2g} f(s)\big|\Big]. \tag{9.3}$$

[illegible]. Invoking (9.1), the classical tensor product argument

$$\begin{aligned} R_g^\square f &= \Big[\big(I^{[0,1]} \otimes I^{[0,1]}\big) - \big(Q_g^{[0,1]} \otimes Q_g^{[0,1]}\big)\Big]f \\ &= \Big[\big(I^{[0,1]} \otimes I^{[0,1]}\big) - \big(I^{[0,1]} \otimes Q_g^{[0,1]}\big) + \big(I^{[0,1]} \otimes Q_g^{[0,1]}\big) - \big(Q_g^{[0,1]} \otimes Q_g^{[0,1]}\big)\Big]f \\ &= \Big[I^{[0,1]} \otimes \big(I^{[0,1]} - Q_g^{[0,1]}\big)\Big]f + \Big[\big(I^{[0,1]} - Q_g^{[0,1]}\big) \otimes Q_g^{[0,1]}\Big]f \end{aligned}$$

leads us to the desired estimate (9.3). ∎

Next, to a function $f(s,t) \in C(\square \times \square)$ we apply the four-dimensional product Gauss–Legendre quadrature formula

$$Q_{g,g'}^{\square\times\square} f := \big(Q_g^{\square} \otimes Q_{g'}^{\square}\big) f = \sum_{i,i'=1}^{g} \sum_{j,j'=1}^{g'} \omega_{g,i}\omega_{g,i'}\omega_{g',j}\omega_{g',j'} f\Big(\begin{bmatrix} \xi_{g,i} \\ \xi_{g,i'} \end{bmatrix}, \begin{bmatrix} \xi_{g',j} \\ \xi_{g',j'} \end{bmatrix}\Big) \tag{9.4}$$

in order to approximate the integral $I^{\square\times\square} f := \big(I^{\square} \otimes I^{\square}\big) f = \int_{\square}\int_{\square} f(s,t)\, dt\, ds$.

LEMMA 9.2. [illegible] $f(s,t) \in C^{2g}(\square) \times C^{2g'}(\square)$ [illegible] $R_{g,g'}^{\square\times\square} f := I^{\square\times\square} f - Q_{g,g'}^{\square\times\square} f$ [illegible] (9.4) [illegible]

$$\begin{aligned} \big|R_{g,g'}^{\square\times\square} f\big| \lesssim & \frac{2^{-4g}}{(2g)!}\Big[\max_{s,t\in\square}\big|\partial_{s_1}^{2g} f(s,t)\big| + \max_{s,t\in\square}\big|\partial_{s_2}^{2g} f(s,t)\big|\Big] \\ & + \frac{2^{-4g'}}{(2g')!}\Big[\max_{s,t\in\square}\big|\partial_{t_1}^{2g'} f(s,t)\big| + \max_{s,t\in\square}\big|\partial_{t_2}^{2g'} f(s,t)\big|\Big]. \end{aligned}$$

[illegible]. This lemma is proven analogously to the previous lemma. □

9.2. Basic estimates. We now assume that the diffeomorphisms γ_i are analytical on $\square$ for all $i \in \{1,2,\ldots,M\}$. Then, in general, a given boundary integral operator $A : H^q(\Gamma) \to H^{-q}(\Gamma)$ of order $2q$ satisfies the following definition.

DEFINITION 9.3. [illegible] $k(x,y)$ [illegible] $2q$ [illegible] (1.6) [illegible]

$$|\partial_s^\alpha \partial_t^\beta k_{i,i'}(s,t)| \lesssim \frac{(|\alpha|+|\beta|)!}{r^{|\alpha|+|\beta|}} \big\|\gamma_i(s) - \gamma_{i'}(t)\big\|^{-(2+2q+|\alpha|+|\beta|)}$$

[illegible] $r > 0$ [illegible] $2 + 2q + |\alpha| + |\beta| > 0$

Defining the local transported kernels

$$k_{\lambda,\lambda'}(s,t) := k\big(\gamma_\lambda(s), \gamma_{\lambda'}(t)\big)\kappa_\lambda(s)\kappa_{\lambda'}(t) dt ds$$

for $s,t \in \square$, we find the relation

$$I^{\square\times\square} k_{\lambda,\lambda'} = 2^{-(|\lambda|+|\lambda'|)} \alpha_{\lambda,\lambda'}. \tag{9.5}$$

LEMMA 9.4. [illegible] $k(x,y)$ [illegible] $2q$ [illegible] $\operatorname{dist}(\Gamma_\lambda, \Gamma_{\lambda'}) > 0$ [illegible] $Q_{g,g'}^{\square\times\square}$ [illegible] (9.4) [illegible] (9.5) [illegible]

$$\begin{aligned} \big|R_{g,g'}^{\square\times\square} k_{\lambda,\lambda'}\big| \lesssim 2^{-2(|\lambda|+|\lambda|')} & \Bigg[\left(\frac{2^{-|\lambda|}}{4r}\right)^{2g} \operatorname{dist}(\Gamma_\lambda, \Gamma_{\lambda'})^{-(2+2q+2g)} \\ & + \left(\frac{2^{-|\lambda'|}}{4r}\right)^{2g'} \operatorname{dist}(\Gamma_\lambda, \Gamma_{\lambda'})^{-(2+2q+2g')}\Bigg]. \end{aligned} \tag{9.6}$$

[illegible]. Since the kernel is analytically standard of order $2q$, (1.4) and (1.5) imply the estimate

$$|\partial_s^\alpha \partial_t^\beta k_{\lambda,\lambda'}(s,t)| \lesssim \frac{(|\alpha|+|\beta|)!}{r^{|\alpha|+|\beta|}} \frac{2^{-|\lambda|(|\alpha|+2)} 2^{-|\lambda'|(|\beta|+2)}}{\big\|\gamma_\lambda(s) - \gamma_{\lambda'}(t)\big\|^{2+2q+|\alpha|+|\beta|}}. \tag{9.7}$$

Hence, we find

$$\max_{s,t\in\square}\left|\partial_{s_n}^{2g}k_{\lambda,\lambda'}(s,t)\right| \lesssim 2^{-2(|\lambda|+|\lambda'|)}\ 2^{-2g|\lambda|}\ \frac{(2g)!}{r^{2g}}\ \min_{s,t\in\square}\left\|\gamma_\lambda(s)-\gamma_{\lambda'}(t)\right\|^{-(2+2q+2g)}$$
$$\lesssim 2^{-2(|\lambda|+|\lambda'|)}\ 2^{-2g|\lambda|}\ \frac{(2g)!}{r^{2g}}\ \operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{-(2+2q+2g)}$$

for $n=1,2$. Analogously one infers

$$\max_{s,t\in\square}\left|\partial_{t_n}^{2g'}k_{\lambda,\lambda'}(s,t)\right| \lesssim 2^{-2(|\lambda|+|\lambda'|)}\ 2^{-2g'|\lambda'|}\ \frac{(2g')!}{r^{2g'}}\ \operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{-(2+2q+2g')}.$$

The integrand $k_{\lambda,\lambda'}$ is nonsingular if $\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})>0$. Therefore, we conclude the desired estimate by Lemma 9.2. ∎

In the case of piecewise bilinear wavelets, we have to compute in accordance with (7.3) the integrals

$$I^{\square\times\square}\left[k_{\lambda,\lambda'}(p_m^\square\otimes p_{m'}^\square)\right]=2^{-(|\lambda|+|\lambda'|)}\beta_{(m,\lambda),(m',\lambda')}. \tag{9.8}$$

Herein, $p_m^\square$, $p_{m'}^\square$ denote the bilinear polynomials on $\square$ defined by (2.2).

LEMMA 9.5. [illegible] $k(x,y)$ [illegible] $2q$ [illegible] $\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})\gtrsim 2^{-\min\{|\lambda|,|\lambda'|\}}$ [illegible] $Q_{g,g'}^{\square\times\square}$ (9.4) [illegible] (9.8) [illegible]

$$\left|R_{g,g'}^{\square\times\square}\left[k_{\lambda,\lambda'}(p_m^\square\otimes p_{m'}^\square)\right]\right| \lesssim \left(\frac{2^{-|\lambda|}}{4r}\right)^{2g}\frac{2^{-2|\lambda|'-|\lambda|}}{\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{1+2q+2g}}$$
$$+\left(\frac{2^{-|\lambda'|}}{4r}\right)^{2g'}\frac{2^{-2|\lambda|-|\lambda|'}}{\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{1+2q+2g'}}. \tag{9.9}$$

[illegible]. The $(2g)$th partial derivative of $k_{\lambda,\lambda'}(s,t)p_m^\square(s)p_{m'}^\square(t)$ with respect to s_n, $n=1,2$, is given by

$$\partial_{s_n}^{2g}\left[k_{\lambda,\lambda'}(s,t)p_m^\square(s)p_{m'}^\square(t)\right]=\partial_{s_n}^{2g}k_{\lambda,\lambda'}(s,t)p_m^\square(s)p_{m'}^\square(t)$$
$$+\partial_{s_n}^{2g-1}k_{\lambda,\lambda'}(s,t)\partial_{s_n}p_m^\square(s)p_{m'}^\square(t).$$

Obviously, there holds $\left|p_m^\square(s)p_{m'}^\square(t)\right|\le 1$ and $\left|\frac{\partial p_m^\square(s)}{\partial s_n}p_{m'}^\square(t)\right|\le 1$ for all $s,t\in\square$ and $1\le m,m'\le 4$; cf. (2.2). Therefore, according to (9.7) we find the bound

$$\max_{s,t\in\square}\left|\partial_{s_n}^{2g}\left[k_{\lambda,\lambda'}(s,t)p_m^\square(s)p_{m'}^\square(t)\right]\right|$$
$$\lesssim 2^{-2g|\lambda|}\frac{(2g)!}{r^{2g}}\frac{2^{-2|\lambda|'-|\lambda|}}{\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{1+2q+2g}}\cdot\underbrace{\left[\frac{2^{-|\lambda|}}{\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})}+\frac{r}{2g}\right]}_{\lesssim 1\text{ since }\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})\gtrsim 2^{-\min\{|\lambda|,|\lambda'|\}}}$$
$$\lesssim 2^{-2g|\lambda|}\frac{(2g)!}{r^{2g}}\frac{2^{-2|\lambda|'-|\lambda|}}{\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'})^{1+2q+2g}}.$$

Analogously one gets the corresponding result for the derivatives with respect to t_1 and t_2. Hence, (9.9) is valid according to Lemma 9.2. ∎

We now can formulate the following proposition, which is an immediate consequence of Lemmas 9.4 and 9.5.

PROPOSITION 9.6. $k(x,y)$ $2q$ ε Γ_λ $\Gamma_{\lambda'}$

$$\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}) \geq 2^{-\min\{|\lambda|,|\lambda'|\}s}, \qquad s > \frac{1}{4r}. \tag{9.10}$$

$Q_{g,g'}^{\square\times\square}$ (9.4) $\alpha_{\lambda,\lambda'}$ $\beta_{(m,\lambda),(m',\lambda')}$ $\lesssim \varepsilon$

$$\begin{aligned} g &= \left\lceil -\tfrac{1}{2}\cdot\frac{|\lambda|+|\lambda'|+\log_2(\varepsilon)+(2+2q)\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}{|\lambda|+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}\right\rceil, \\ g' &= \left\lceil -\tfrac{1}{2}\cdot\frac{|\lambda|+|\lambda'|+\log_2(\varepsilon)+(2+2q)\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}{|\lambda'|+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}\right\rceil \end{aligned} \tag{9.11}$$

$$\begin{aligned} g &= \left\lceil -\tfrac{1}{2}\cdot\frac{|\lambda'|+\log_2\varepsilon+(1+2q)\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}{|\lambda|+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}\right\rceil, \\ g' &= \left\lceil -\tfrac{1}{2}\cdot\frac{|\lambda|+\log_2\varepsilon+(1+2q)\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}{|\lambda'|+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}))}\right\rceil \end{aligned} \tag{9.12}$$

Notice that the error of quadrature does not tend to zero when increasing g and g' if the elements violate the distance criterion (9.10). For such integrals we propose an adaptive quadrature strategy in the next subsection.

A special situation occurs if $|\lambda| = |\lambda'|$ and $\Gamma_\lambda \cap \Gamma_{\lambda'} \neq \emptyset$, i.e., if both elements are identical or share a common edge or vertex. Then, the domain of integration contains the singularity. We utilize the to transform the singular integrands on nonsingular ones; cf. [15, 29, 30]. It suffices to consider an error of quadrature which behaves as in (9.6) and (9.9), respectively, with $\operatorname{dist}(\Gamma_\lambda,\Gamma_{\lambda'}) := 2^{-|\lambda|}$. Consequently, we have to subdivide the domain of integration $[0,1]^4$ into 16^M equal sized cubes. Herein, M denotes the nonnegative integer $M := \max\{0, \lceil \log_2 s\rceil\}$ with s from (9.10). After rescaling the domains of integration to $[0,1]^4$, the application of the quadrature formula $Q_{g,g'}^{\square\times\square}$ with

$$g = g' = \left\lceil \frac{q|\lambda| - (\log_2\varepsilon)/2}{2+M+\log_2 r}\right\rceil \tag{9.13}$$

yields a quadrature error $\lesssim \varepsilon$.

A refined analysis shows that the choice (9.13) can be weakened. Indeed, in certain coordinates the degree of quadrature can be chosen smaller; see [30] for the details.

9.3. An adaptive quadrature strategy. We are now in the position to formulate the recursive adaptive algorithm. We prove in Lemma 9.8 that the algorithm computes the element-element interaction of Γ_λ and $\Gamma_{\lambda'}$ with the precision $\varepsilon_{|\lambda|,|\lambda'|}$. Without loss of generality we assume $|\lambda'| \geq |\lambda|$. An example of the subdivision of the elements is depicted in Figure 9.1.

1. If the elements Γ_λ and $\Gamma_{\lambda'}$ violate the distance criterion (9.10), then goto item 2 if $|\lambda'| > |\lambda|$ or goto item 3 if $|\lambda| = |\lambda'|$. Otherwise apply the product Gauss–Legendre quadrature formula $Q_{g,g'}^{\square\times\square}$ choosing g and g' associated with the precision $\varepsilon_{|\lambda|,|\lambda'|}$ as in Proposition 9.6.

2. $|\lambda'| > |\lambda|$ Replace the larger element Γ_λ by its four sons and compute the associated element-element interactions with precision $\varepsilon_{|\lambda|,|\lambda'|}$ according to item 1. The desired element-element interaction is calculated via the recycling formulas.
3. $|\lambda| = |\lambda'|$ If the domain of integration contains the singularity, then apply the Duffy trick. Else, replace both elements Γ_λ and $\Gamma_{\lambda'}$ by their sons and compute the associated element-element interactions with precision $\varepsilon_{|\lambda|,|\lambda'|}$ according to item 1. The desired element-element interaction is calculated via the recycling formulas.

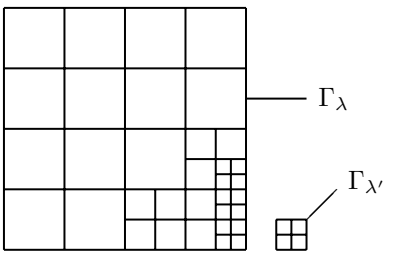


FIG. 9.1. *This subdivision of the elements* Γ_λ *and* $\Gamma_{\lambda'}$ *results if* $s = 3/2$.

Notice that the recycling is easily realized by inserting "return the previously computed element-element interaction if it exists" in item 1. The next lemma shows that the algorithm stops after at most $\mathcal{O}(||\lambda| - |\lambda'||)$ subdivision steps.

LEMMA 9.7. $\alpha_{\lambda,\lambda'}$ $\beta_{(m,\lambda),(m',\lambda')}$

1. $\mathcal{O}(||\lambda|-|\lambda'||)$ $\alpha_{\widehat{\lambda},\widehat{\lambda}'}$ $\beta_{(m,\widehat{\lambda}),(m',\widehat{\lambda}')}$ $|\widehat{\lambda}| \geq |\lambda|$ $|\widehat{\lambda}'| \geq |\lambda'|$
2. $|\lambda| \leq |\lambda'|$ $|\lambda| \leq |\widehat{\lambda}| \leq |\widehat{\lambda}'| \sim |\lambda'|$ $|\lambda'| \leq |\lambda|$
3. $|\widehat{\lambda}|$ $|\widehat{\lambda}'|$ $\alpha_{\widehat{\lambda},\widehat{\lambda}'}$ $\beta_{(m,\widehat{\lambda}),(m',\widehat{\lambda}')}$ $\mathcal{O}(1)$

. There is nothing to prove in the case of two elements Γ_λ and $\Gamma_{\lambda'}$ satisfying the distance criterion (9.10) or if the Duffy trick is applied. Hence, we consider first $|\lambda| = |\lambda'|$ and two disjoint elements Γ_λ and $\Gamma_{\lambda'}$ which violate the distance criterion (9.10). Since the mesh is quasi-uniform, a constant $c_\Gamma > 0$ exists such that $\operatorname{dist}(\Gamma_\lambda, \Gamma_{\lambda'}) \geq 2^{-|\lambda|} c_\Gamma$ for all pairs of elements with $\Gamma_\lambda \cap \Gamma_{\lambda'} = \emptyset$. Herein, c_Γ depends on only the manifold and its parametrization but not on $|\lambda|$ or J. The algorithm subdivides the desired element-element interaction into at most 16^M element-element interactions $\alpha_{\widehat{\lambda},\widehat{\lambda}'}$ or $\beta_{(m,\widehat{\lambda}),(m',\widehat{\lambda}')}$, where $|\lambda| \leq |\widehat{\lambda}| = |\widehat{\lambda}'| \leq |\lambda| + M$ and $M = \max\{0, \lceil \log_2(s/c_\Gamma) \rceil\}$.

Finally, we consider the case of two elements Γ_λ and $\Gamma_{\lambda'}$, $|\lambda| \neq |\lambda'|$, violating the distance criterion (9.10). Without loss of generality we assume $|\lambda| < |\lambda'|$. Moreover, we make use of the following observation. Due to $\operatorname{diam}(\Gamma_{\widehat{\lambda}}) \sim 2^{-|\widehat{\lambda}|}$, for a fixed level $|\widehat{\lambda}| \leq |\lambda'|$, only $\mathcal{O}(s^2)$ elements $\Gamma_{\widehat{\lambda}}$ exist with $\operatorname{dist}(\Gamma_{\widehat{\lambda}}, \Gamma_{\lambda'}) < s 2^{-|\widehat{\lambda}|}$. Consequently, our algorithm subdivides for all $|\lambda| < |\widehat{\lambda}| \leq |\lambda'|$ only $\mathcal{O}(s^2)$ element-element interactions while $\mathcal{O}(s^2)$ element-element interactions are evaluated since they satisfy the

distance criterion. The remaining $\mathcal{O}(s^2)$ element-element interactions associated with $|\widehat{\lambda}| = |\lambda|$ are subdivided into $\mathcal{O}(s^4)$ element-element interactions according to the aforementioned cases. □

We prove now that the proposed quadrature algorithm computes the desired element-element interaction with a precision that stays proportional to $\varepsilon_{|\lambda|,|\lambda'|}$.

LEMMA 9.8. [illegible] $\alpha_{\lambda,\lambda'}$ [illegible] $\beta_{(m,\lambda),(m',\lambda')}$ [illegible] $\varepsilon_{|\lambda|,|\lambda'|}$ [illegible]. Without loss of generality we assume $|\lambda| \le |\lambda'|$. According to Lemma 9.7, the computation of the desired element-element interaction is reduced to the computation of element-element interactions $\alpha_{\widehat{\lambda},\widehat{\lambda}'}$ or $\beta_{(m,\widehat{\lambda}),(m',\widehat{\lambda}')}$, where $|\lambda| \le |\widehat{\lambda}| \le |\widehat{\lambda}'| \sim |\lambda'|$. Hereby, for a fixed pair of levels $|\widehat{\lambda}|, |\widehat{\lambda}'|$, only $\mathcal{O}(1)$ element-element interactions are evaluated. Observing the damping factor $1/2$ in the recycling formulas (8.1) and (8.2), we might estimate the final quadrature error by $\sum_{|\widehat{\lambda}|=|\lambda|}^{|\lambda'|} \varepsilon_{|\lambda|,|\lambda'|} 2^{|\lambda|-|\widehat{\lambda}|} \lesssim \varepsilon_{|\lambda|,|\lambda'|}$. □

Consequently, by the above quadrature algorithm, the system matrix is computed consistent with Theorem 3.3. In combination with Theorem 3.2 we prove next that the proposed quadrature strategy in fact computes the compressed system matrix within linear complexity. This is achieved even without the recycling of previously computed element-element interactions. But we like to stress that the recycling accelerates the computation of the system matrix enormously.

THEOREM 9.9. [illegible] $\alpha_{\lambda,\lambda'}$ [illegible] $\beta_{(m,\lambda),(m',\lambda')}$ [illegible] $\varepsilon_{|\lambda|,|\lambda'|}$ [illegible] (3.6) [illegible] $\mathcal{O}\big([2J - |\lambda| - |\lambda'|]^4\big)$ [illegible]. (i) Throughout this proof we will abbreviate $j := |\lambda|$ and $j' := |\lambda'|$. We find $\frac{d'-q}{\widetilde{d}+q} < 1$ due to (3.3). Consequently, $\varepsilon_{j,j'}$ can be estimated by

$$\varepsilon_{j,j'} \sim \min\left\{2^{-|j-j'|}, 2^{-(2J-j-j')\frac{d'-q}{\widetilde{d}+q}}\right\} 2^{2Jq} 2^{-d'(2J-j-j')} \ge 2^{2Jq} 2^{-(d'+1)(2J-j-j')}.$$

Moreover, according to (9.4), the quadrature rule $Q_{g,g'}^{\square\times\square}$ applies $(gg')^2$ quadrature points. In view of Lemma 9.7 we may neglect the evaluation of the recycling formulas since this complexity scales linearly with the number $\mathcal{O}(|j-j'|)$ of computed element-element interactions and $|j-j'| \le 2J-j-j'$.

(ii) We estimate the degrees of quadrature to compute a single element-element interaction $\alpha_{\widehat{\lambda},\widehat{\lambda}'}$ or $\beta_{(m,\widehat{\lambda}),(m,\widehat{\lambda}')}$, which stems from our adaptive quadrature algorithm, with the precision $\varepsilon_{j,j'}$. Without loss of generality we assume $j' \ge j$, which implies $j' \sim \widehat{j}' := |\widehat{\lambda}'| \ge \widehat{j} := |\widehat{\lambda}| \ge j$ according to Lemma 9.7. Quadrature is applied if either the Duffy trick is employed or the distance criterion (9.10) is satisfied.

First, we treat the case that the Duffy trick is applied, which implies $\widehat{j} = \widehat{j}'$. We have to compute 16^M, $M = \max\{0, \lceil \log_2 s \rceil\}$, integrals with the degree of quadrature given by (9.13). The choice of M ensures $2 + M + \log_2 r = c > 0$, which leads to

$$g = g' \sim \tfrac{q\widehat{j} - (\log_2 \varepsilon_{j,j'})/2}{2+M+\log_2 r} \le \tfrac{1}{2c}[-2q(J-\widehat{j}) + (d'+1)(2J-j-j')].$$

Hence, we need $\mathcal{O}\big([2J-j-j']^4\big)$ quadrature points since

$$-2q(J-\widehat{j}) \lesssim \begin{cases} 1 & \text{if } q \ge 0, \\ -2q(2J-j-j') & \text{otherwise} \end{cases} \tag{9.14}$$

and $M \sim 1$ independently of J and $\widehat{j}$.

Next, let the distance criterion (9.10) be satisfied. Thus, there holds

$$\widehat{j} + 2 + \log_2 r + \log_2 \big(\operatorname{dist}(\Gamma_{\widehat{\lambda}}, \Gamma_{\widehat{\lambda}'})\big) \geq \widehat{j} + 2 + \log_2 r + \log_2(2^{-\widehat{j}} s) =: c > 0.$$

Inserting this inequality into (9.11) yields

$$\begin{aligned} g &\sim -\tfrac{1}{2} \cdot \tfrac{\widehat{j}+\widehat{j}'+\log_2(\varepsilon_{j,j'})+(2+2q)\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))}{\widehat{j}+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))} \\ &\leq -\tfrac{1}{2c}[\widehat{j} + \widehat{j}' + \log_2(\varepsilon_{j,j'}) + (2+2q)\log_2(2^{-\widehat{j}} s)] \\ &= \tfrac{1}{2c}[-2q(J-\widehat{j}) + \widehat{j} - \widehat{j}' + (d'+1)(2J-j-j') + (2+2q)\log_2(1/s)]. \end{aligned}$$

Inequality (9.14) and $\widehat{j} \leq \widehat{j}'$ imply $g \lesssim 2J - j - j'$. In complete analogy we estimate

$$\begin{aligned} g' &\sim -\tfrac{1}{2} \cdot \tfrac{\widehat{j}+\widehat{j}'+\log_2(\varepsilon_{j,j'})+(2+2q)\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))}{\widehat{j}'+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))} \\ &= -\tfrac{1}{2} \cdot \tfrac{\widehat{j}+\widehat{j}'+\log_2(\varepsilon_{j,j'})+(2+2q)\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))}{[\widehat{j}'-\widehat{j}]+[\widehat{j}+2+\log_2 r+\log_2(\operatorname{dist}(\Gamma_{\widehat{\lambda}},\Gamma_{\widehat{\lambda}'}))]} \lesssim \tfrac{1}{\widehat{j}'-\widehat{j}+c}(2J-j-j'). \end{aligned}$$

Consequently, $\mathcal{O}\big(\frac{1}{(\widehat{j}'-\widehat{j}+c_s)^2}[2J-j-j']^4\big)$ quadrature points are evaluated. One readily verifies that both estimates remain valid for the degrees of quadrature given by (9.12), i.e., in the case of piecewise bilinear functions.

(iii) According to Lemma 9.7 we have just to sum over $\widehat{j}$ with $j < \widehat{j} \leq j'$ to estimate the work of computing the desired element-element interaction. From

$$\sum_{\widehat{j}=j+1}^{j'} \frac{1}{(j'-\widehat{j}+c)^2} < \sum_{\widehat{j}=0}^{\infty} \frac{1}{(\widehat{j}+c)^2} < \frac{1}{c^2} + \sum_{\widehat{j}=1}^{\infty} \frac{1}{\widehat{j}^2} = \frac{1}{c^2} + \frac{\pi^2}{6} \lesssim 1$$

we derive indeed the desired complexity $\mathcal{O}\big([2J-j-j']^4\big)$. ∎

10. Numerical results. In order to confirm our analysis and to demonstrate the proposed method we shall present some numerical results in this section. We shall consider the Laplace equation $\Delta u = 0$ in Ω, where we employ first the unit sphere as domain Ω. It is well known that, denoting the spherical harmonics by Y_n^m, the functions of type $u(x) = \|x\|^2 Y_n^m(x/\|x\|)$ are solutions of the Laplace equation in the sphere. We will treat both the Dirichlet problem with data $f := u|_\Gamma = Y_2^0$ and the Neumann problem with data $g := \partial u/\partial n = 2Y_2^0$. Then, $u = \|x\|^2 Y_2^0(x/\|x\|)$ is the unique solution (modulo some constant in the case of the Neumann problem) of the interior Dirichlet and Neumann problem, respectively.

We solve both the interior Dirichlet and the interior Neumann problem by the indirect approach using the [illegible]

$$V : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma), \qquad (V\rho)(x) = \frac{1}{4\pi} \int_\Gamma \frac{1}{\|x-y\|} \rho(y) d\Gamma_y,$$

the [illegible] K and its adjoint $K^\star$,

$$K - 1/2 : L^2(\Gamma) \to L^2(\Gamma), \qquad (K\rho)(x) = \frac{1}{4\pi} \int_\Gamma \frac{\partial}{\partial n_y} \frac{1}{\|x-y\|} \rho(y) d\Gamma_y,$$

$$K^\star + 1/2 : L^2(\Gamma)/\mathbb{R} \to L^2(\Gamma)/\mathbb{R}, \qquad (K^\star\rho)(x) = \frac{1}{4\pi} \int_\Gamma \frac{\partial}{\partial n_x} \frac{1}{\|x-y\|} \rho(y) d\Gamma_y,$$

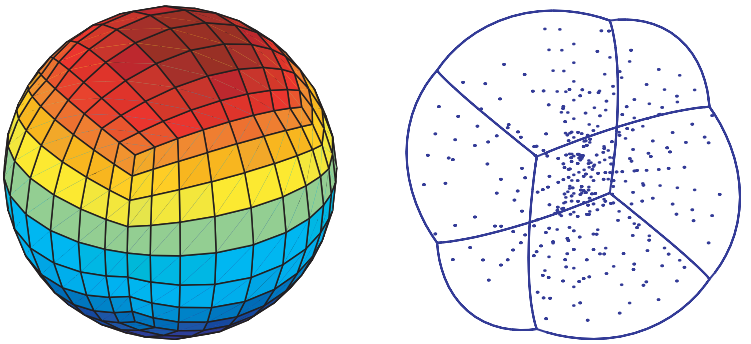


FIG. 10.1. *The mesh on the unit sphere ($j = 3$) and evaluation points of the solution.*

as well as the hypersingular operator

$$W : H^{1/2}(\Gamma)/\mathbb{R} \to H^{-1/2}(\Gamma)/\mathbb{R}, \qquad (W\rho)(x) = -\frac{1}{4\pi}\frac{\partial}{\partial n_x}\int_\Gamma \frac{\partial}{\partial n_y}\frac{1}{\|x-y\|}\rho(y)d\Gamma_y.$$

The indirect method yields a density ρ living on the boundary Γ which leads to the solution $u(x)$ by potential evaluation. More precisely, for the Dirichlet problem we derive the Fredholm integral equations of the first kind,

$$V\rho = f \text{ on } \Gamma, \qquad u(x) = (V\rho)(x) \text{ in } \Omega, \tag{10.1}$$

and of the second kind,

$$\left(K - \frac{1}{2}\right)\rho = f \text{ on } \Gamma, \qquad u(x) = (K\rho)(x) \text{ in } \Omega. \tag{10.2}$$

The Neumann problem is solved via the Fredholm integral equations of the first kind,

$$W\rho = g \text{ on } \Gamma, \qquad u(x) = (K\rho)(x) \text{ in } \Omega, \tag{10.3}$$

and of the second kind,

$$\left(K^\star + \frac{1}{2}\right)\rho = g \text{ on } \Gamma, \qquad u(x) = (V\rho)(x) \text{ in } \Omega. \tag{10.4}$$

The parametric representation of the sphere via six patches is derived by projecting the cube $[-1,1]^3$ onto the unit sphere; cf. Figure 10.1. We shall solve all these integral equations by piecewise constant and bilinear wavelets. Equations (10.1), (10.2), and (10.4) are discretized by piecewise constant wavelets with three vanishing moments as well as piecewise bilinear wavelets with double nodes along the interfaces and four vanishing moments. The discretization of (10.3) requires [illegible] piecewise bilinear wavelet bases since the energy space of the hypersingular operator is $H^{1/2}(\Gamma)/\mathbb{R}$. It suffices that the wavelets provide two vanishing moments. Since the kernel is hypersingular, we employ the identity $\langle W\rho, \mu\rangle = \langle V \operatorname{curl}_\Gamma \rho, \operatorname{curl}_\Gamma \mu\rangle$ (cf. [25]).

Since the spherical harmonics are eigenfunctions of all these boundary integral operators, the density ρ is a certain multiple of Y_2^0. Therefore, we are able to measure the $L^2(\Gamma)$-error of the density for which the error estimate $\|\rho - \rho_J\|_{L^2(\Gamma)} \lesssim 2^{-Jd}$

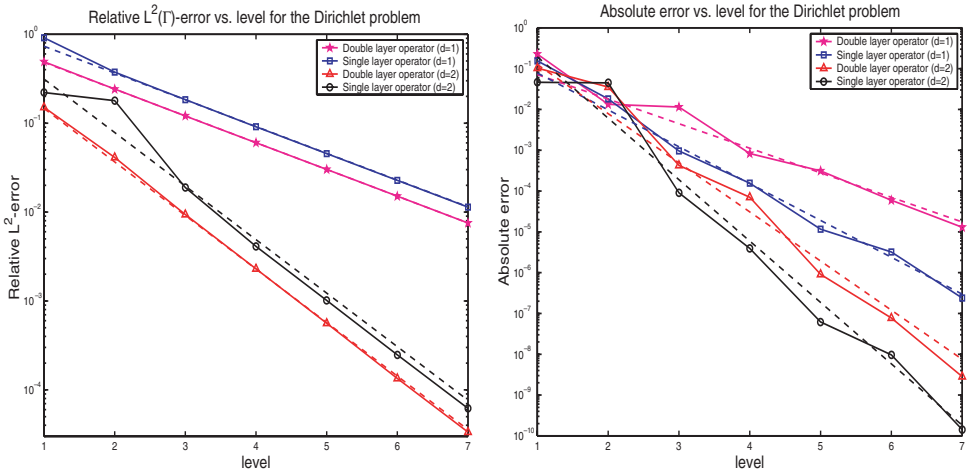


FIG. 10.2. *Dirichlet problem: errors of the density (left) and the potential evaluation (right).*

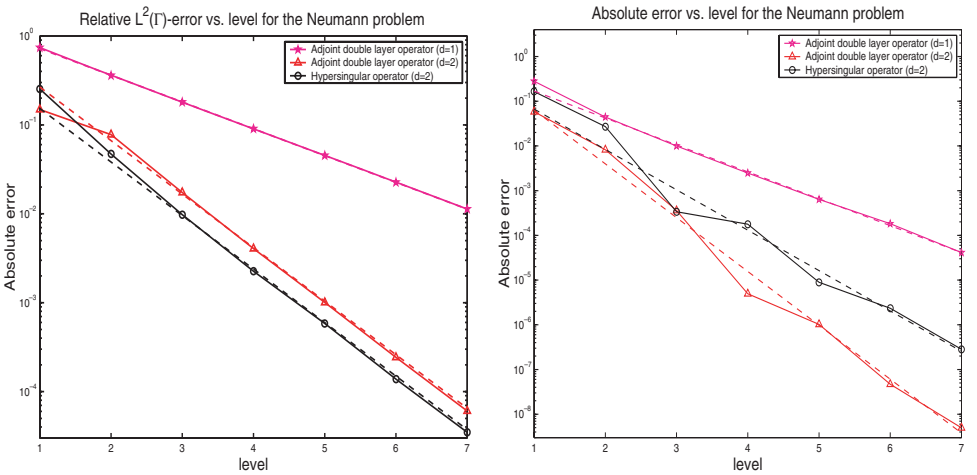


FIG. 10.3. *Neumann problem: errors of the density (left) and the potential evaluation (right).*

holds. In order to measure also the best possible order of convergence, we compute additionally the approximate solution u_J of the Laplace equation in a lot of points x_i distributed in the interior of the sphere as shown in the right plot of Figure 10.1. Depending on the approximation order d and the operator order $2q$, the optimal order of convergence (including the Aubin–Nitsche trick) of $u(x_i) - u_J(x_i)$ in a fixed evaluation point x_i is given by $|u(x_i) - u_J(x_i)| \lesssim 2^{-2J(d-q)}$; cf. [34]. However, the errors in a single point might oscillate extremely due to round-off errors. To be on safe ground we compute the ℓ^∞-norm of the measurements in all points. If this norm converges with the optimal order, we are sure that we achieve the optimal order of convergence. Instead of the ℓ^∞-norm we computed also the mean value and the ℓ^2-norm. But this has not changed the convergence behavior. The convincing results produced by our algorithm are presented in Figures 10.2 and 10.3 for the Dirichlet and the Neumann

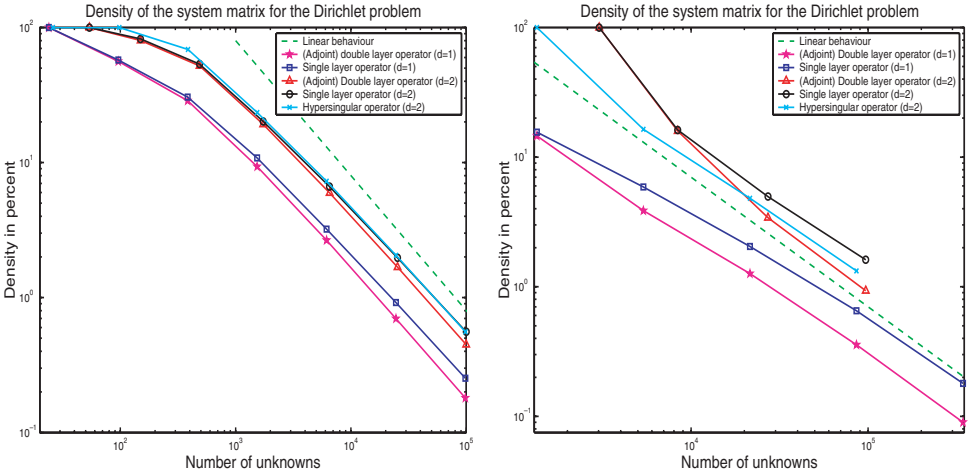


FIG. 10.4. *Compression rates for the sphere (left) and the gearwheel (right).*

problem, respectively. The left-hand plots are concerned with the relative $L^2(\Gamma)$-error of the density, while the right-hand plots show the absolute ℓ^∞-error of the potential evaluation. In all computations, the optimal orders of convergence, indicated by the dashed lines, are realized.

The density of the system matrix, measured by the ratio of relevant coefficients versus N_J^2, is visualized on the left-hand side of Figure 10.4. One figures out that the number of nonzero coefficients tends in fact to a linear behavior (dashed line).

In our second example we consider the gearwheel presented in Figure 1.1 as domain Ω and choose, as above, the Dirichlet or Neumann data of the harmonical function $u(x) = \langle x, [1,2,4]^T\rangle/\|x\|^3$, $x \notin \Omega$, as boundary conditions. Since in this case the density is unknown, we measure only the ℓ^∞-norm of the error of the solution with respect to about 8000 evaluation points x_i distributed inside the gearwheel. We emphasize that the density ρ exhibits strong singularities due to the presence of vertices and edges, even though the solution u is smooth. Hence, we cannot expect the same orders of convergence as on the sphere.

In Tables 10.1 and 10.2 we tabulated the results for the Dirichlet and the Neumann problem, respectively. The cpu-time refers to the computation of the approximate density, i.e., assembling and solving the linear system of equations by a CG or GMRES algorithm. The bracketed values correspond to the ratio of the error on the previous level to the actual error and reflect the order of convergence. The number of nonzero coefficients of the system matrix (density) is given in percent; see also Figure 10.4 for a visualization. The present example demonstrates that the wavelet Galerkin scheme remains applicable also in the case of complex geometries. It turns out that a piecewise bilinear discretization requires about twice the memory and five times the cpu-time of the corresponding piecewise constant discretization. However, the higher order yields a much higher accuracy. In Figure 10.5 we plot the absolute error versus the cpu-time. In the case of the Dirichlet problem (left-hand plot), one observes that the single layer operator discretized by piecewise bilinears yields the best performance. In the case of the Neumann problem (right-hand plot), the situation is not so clear. However, when extrapolating the graphs, the adjoint of the double layer operator discretized by piecewise bilinears seems to perform best.

TABLE 10.1
Numerical results with respect to the Dirichlet problem on the gearwheel.

Problem	J	N_J	$\|\mathbf{u}-\mathbf{u}_J\|_\infty$	Cpu-time (sec.)	Density (%)
Equation (10.1)	1	1344	6.60e-2	1	15
discretized by	2	5376	3.66e-2 (1.8)	8	5.9
piecewise constant	3	21504	4.17e-3 (8.8)	65	2.0
wavelets	4	86016	1.10e-3 (3.8)	502	0.65
	5	344064	2.68e-4 (4.1)	2693	0.18
Equation (10.2)	1	1344	3.34e-1	1	15
discretized by	2	5376	1.12e-1 (2.9)	8	3.9
piecewise constant	3	21504	2.22e-2 (5.2)	52	1.3
wavelets	4	86016	6.24e-3 (3.6)	389	0.37
	5	344064	6.34e-4 (10)	1717	0.09
Equation (10.1)	1	3024	2.84e-2	16	100
discretized by	2	8400	1.27e-2 (2.2)	32	17
piecewise bilinear	3	27216	4.91e-4 (26)	324	4.8
wavelets	4	97104	3.14e-5 (16)	2366	1.4
Equation (10.2)	1	3024	4.72e-1	10	100
discretized by	2	8400	1.02e-1 (4.6)	28	16
piecewise bilinear	3	27216	7.90e-3 (13)	309	3.4
wavelets	4	97104	1.10e-3 (7.2)	2334	0.92

TABLE 10.2
Numerical results with respect to the Neumann problem on the gearwheel.

Problem	J	N_J	$\|\mathbf{u}-\mathbf{u}_J\|_\infty$	Cpu-time (sec.)	Density (%)
Equation (10.4)	1	1344	9.51e-1	1	15
discretized by	2	5376	3.98e-1 (2.4)	8	3.9
piecewise constant	3	21504	4.31e-2 (9.2)	52	1.3
wavelets	4	86016	1.20e-2 (3.6)	389	0.37
	5	344064	2.52e-4 (4.8)	1717	0.09
Equation (10.3)	1	1344	8.59e-1	6	100
discretized by	2	5376	2.50e-1 (3.4)	88	16
piecewise bilinear	3	21504	1.44e-2 (17)	616	4.4
wavelets	4	86016	1.88e-3 (7.6)	3044	1.2
Equation (10.4)	1	3024	7.10e-1	10	100
discretized by	2	8400	2.38e-1 (3.0)	28	16
piecewise bilinear	3	27216	1.03e-2 (23)	309	3.4
wavelets	4	97104	2.34e-3 (4.4)	2343	0.92

All computations were performed by a standard workstation with 2 Gigabyte main memory. We never needed a degree of quadrature higher than 11. Moreover, we set $d' = \delta = 1.25$ if $d = 1$ and $d' = \delta = 2.25$ if $d = 2$, and $s = r = 1$. The parameter a was chosen equal to 1 for the sphere, which to our experience is a robust choice if all distances are scaled such that the largest patch has a diameter $\sqrt{2}$ that corresponds to the unit square. However, in our second example we increased this parameter to $a = 1.5$ to achieve an accuracy on the coarse grids that is comparable to that produced by the traditional Galerkin scheme. Finally, let us remark that we never subdivided elements living on the same levels, that is, $M = 0$. This choice does a good job if the constant c_Γ in the proof of Lemma 9.7 is sufficiently large, which is the case for both geometries. But we emphasize that M has to be increased if the geometry contains patches that meet in a very acute angle.

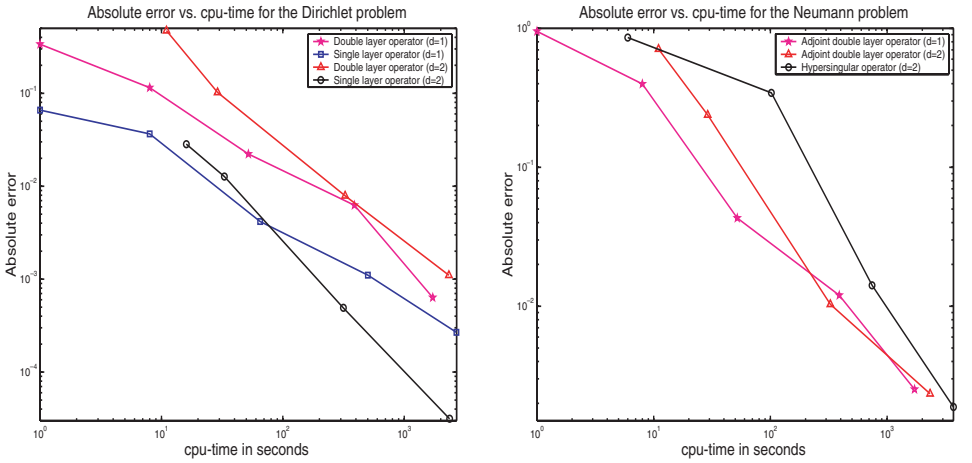


FIG. 10.5. *Performance with respect to the Dirichlet (left) and to the Neumann (right) problem.*

REFERENCES

[1] G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *The fast wavelet transform and numerical algorithms*, Comm. Pure Appl. Math., 44 (1991), pp. 141–183.

[2] C. CANUTO, A. TABACCO, AND K. URBAN, *The wavelet element method, part* I: *Construction and analysis*, Appl. Comput. Harmon. Anal., 6 (1999), pp. 1–52.

[3] A. COHEN, I. DAUBECHIES, AND J.-C. FEAUVEAU, *Biorthogonal bases of compactly supported wavelets*, Pure Appl. Math., 45 (1992), pp. 485–560.

[4] A. COHEN AND R. MASSON, *Wavelet adaptive method for second order elliptic problems: Boundary conditions and domain decomposition*, Numer. Math., 86 (2000), pp. 193–238.

[5] W. DAHMEN, *Wavelet and multiscale methods for operator equations*, Acta Numer., 6 (1997), pp. 55–228.

[6] W. DAHMEN, H. HARBRECHT, AND R. SCHNEIDER, *Compression techniques for boundary integral equations—asymptotically optimal complexity estimates*, SIAM J. Numer. Anal., 43 (2006), pp. 2251–2271.

[7] W. DAHMEN AND A. KUNOTH, *Multilevel preconditioning*, Numer. Math., 63 (1992), pp. 315–344.

[8] W. DAHMEN, B. KLEEMANN, S. PRÖSSDORF, AND R. SCHNEIDER, *A multiscale method for the double layer potential equation on a polyhedron*, in Advances in Computational Mathematics, Ser. Approx. Decompos. 4, H. P. Dikshit and C. A. Micchelli, eds., World Scientific, River Edge, NJ, 1994, pp. 15–57.

[9] W. DAHMEN, A. KUNOTH, AND K. URBAN, *Biorthogonal spline-wavelets on the interval—stability and moment conditions*, Appl. Comput. Harmon. Anal., 6 (1999), pp. 132–196.

[10] W. DAHMEN, S. PRÖSSDORF, AND R. SCHNEIDER, *Wavelet approximation methods for pseudodifferential equations*. II. *Matrix compression and fast solution*, Adv. Comput. Math., 1 (1993), pp. 259–335.

[11] W. DAHMEN, S. PRÖSSDORF, AND R. SCHNEIDER, *Multiscale methods for pseudodifferential equations on smooth closed manifolds*, in Wavelets: Theory, Algorithms, and Applications (Taormina, 1993), Wavelet Anal. Appl. 5, Academic Press, San Diego, 1994, pp. 385–424.

[12] W. DAHMEN AND R. SCHNEIDER, *Composite wavelet bases for operator equations*, Math. Comp., 68 (1999), pp. 1533–1567.

[13] W. DAHMEN AND R. SCHNEIDER, *Wavelets on manifolds* I: *Construction and domain decomposition*, SIAM J. Math. Anal., 31 (1999), pp. 184–230.

[14] W. DAHMEN AND R. STEVENSON, *Element-by-element construction of wavelets satisfying stability and moment conditions*, SIAM J. Numer. Anal., 37 (1999), pp. 319–352.

[15] M. G. DUFFY, *Quadrature over a pyramid or cube of integrands with a singularity at a vertex*. SIAM J. Numer. Anal., 19 (1982), pp. 1260–1262.

[16] L. GREENGARD AND V. ROKHLIN, *A fast algorithm for particle simulation*, J. Comput. Phys., 73 (1987), pp. 325–348.

[17] W. HACKBUSCH AND B. N. KHOROMSKIJ, *A sparse $\mathcal{H}$-matrix arithmetic*. II. *Application to*

multi-dimensional problems, Computing, 64 (2000), pp. 21–47.
[18] W. Hackbusch and Z. P. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.
[19] G. Hämmerlin and K.-H. Hoffmann, *Numerical Mathematics*, Springer-Verlag, New York, 1991.
[20] H. Harbrecht, *Wavelet Galerkin Schemes for the Boundary Element Method in Three Dimensions*, Ph.D. Thesis, Technische Universität Chemnitz, Chemnitz, Germany, 2001.
[21] H. Harbrecht, F. Paiva, C. Pérez, and R. Schneider, *Biorthogonal wavelet approximation for the coupling of FEM-BEM*, Numer. Math., 92 (2002), pp. 325–356.
[22] H. Harbrecht, F. Paiva, C. Pérez, and R. Schneider, *Wavelet preconditioning for the coupling of FEM-BEM*, Numer. Linear Algebra Appl., 10 (2003), pp. 197–222.
[23] H. Harbrecht and R. Schneider, *Biorthogonal wavelet bases for the boundary element method*, Math. Nachr., 269/270 (2004), pp. 167–188.
[24] K. Höllig and H. Mögerle, *G-splines*, Comput. Aided Geom. Design, 7 (1990), pp. 197–207.
[25] J.-C. Nedelec, *Integral equations with nonintegrable kernels*, Integral Equations Operator Theory, 5 (1982), pp. 562–572.
[26] T. von Petersdorff, C. Schwab, and R. Schneider, *Multiwavelets for second-kind integral equations*, SIAM J. Numer. Anal., 34 (1997), pp. 2212–2227.
[27] T. von Petersdorff and C. Schwab, *Fully discrete multiscale Galerkin BEM*, in Multiscale Wavelet Methods for PDEs, W. Dahmen, A. Kurdila, and P. Oswald, eds., Academic Press, San Diego, 1997, pp. 287–346.
[28] U. Reif, *Biquadratic G-spline surfaces*, Comput. Aided Geom. Design, 12 (1995), pp. 193–205.
[29] S. Sauter, *Über die effiziente Verwendung des Galerkin Verfahrens zur Lösung Fredholmscher Integralgleichungen*, Ph.D. Thesis, Christian–Albrecht–Universität Kiel, Kiel, Germany, 1992.
[30] S. Sauter and C. Schwab, *Quadrature for the hp-Galerkin BEM in* $\mathbb{R}^3$, Numer. Math., 78 (1997), pp. 211–258.
[31] R. Schneider, *Multiskalen- und Wavelet-Matrixkompression: Analysisbasierte Methoden zur Lösung großer vollbesetzter Gleichungssysteme*, B.G. Teubner, Stuttgart, Germany, 1998.
[32] C. Schwab, *Variable order composite quadrature of singular and nearly singular integrals*, Computing, 53 (1994), pp. 173–194.
[33] L. F. Villemoes, *Wavelet analysis of refinement equations*, SIAM J. Math. Anal., 25 (1994), pp. 1433–1460.
[34] W. L. Wendland, *On asymptotic error analysis and underlying mathematical principles for boundary element methods*, in Boundary Element Techniques in Computer Aided Engineering, NATO ASI Series E - 84, C. A. Brebbia, ed., Martinus Nijhoff, Dordrecht, Boston, Lancaster, 1984, pp. 417–436.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1371–1393
© 2006 Society for Industrial and Applied Mathematics

DISCONTINUOUS GALERKIN METHODS FOR A MODEL OF POPULATION DYNAMICS WITH UNBOUNDED MORTALITY*

MI-YOUNG KIM†

Abstract. Discontinuous Galerkin finite element methods in age-time domain are proposed to approximate a possibly nonsmooth solution to the model of population dynamics with unbounded mortality (coefficient) function. The strong stability is established and a priori L^2 error estimates are obtained. We show that the error estimates are optimal both in rate and in regularity away from the maximum age. Nonnegativity of the solution is also shown. The scheme is explicit and the solution is computed strip by strip marching in time. Several numerical examples are presented.

Key words. population dynamics, finite element method, discontinuous Galerkin method, error estimate, regularity of the solution

AMS subject classifications. 90A09, 65K10, 65M12, 65M60

DOI. 10.1137/050624182

1. Introduction.

We consider the following linear initial-boundary value problem due to Lotka and McKendrick [9]:

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial a} + \mu(a)u = 0, \quad 0 < a < a_\dagger, \quad t > 0, \tag{1.1}$$

$$u(0,t) = \int_0^{a_\dagger} \beta(a)u(a,t)da, \quad t > 0, \tag{1.2}$$

$$u(a,0) = u_0(a), \quad 0 \le a < a_\dagger. \tag{1.3}$$

Here, the function $u(a,t)$ denotes the age-specific density of individuals of age a at time t. The demographic parameters $\mu = \mu(a) \ge 0$ and $\beta = \beta(a) \ge 0$ are the age-specific mortality and natality rates, respectively. System (1.1)–(1.3) describes the evolution of the age density $u(a,t)$ of a population with a maximum age $a_\dagger < \infty$, whose growth is regulated by the vital rates $\beta(a)$ and $\mu(a)$, [8, 9, 21].

Throughout the paper we assume the following hypotheses [9]:

(H1) the nonnegative function u_0 belongs to $L^2[0, a_\dagger]$,
(H2) $\beta \in L^\infty[0, a_\dagger]$ and $0 \le \beta(a) \le \overline{\beta}$ for some positive constant $\overline{\beta}$,
(H3) $\mu(a) \ge 0$, $\mu \in L^1_{loc}[0, a_\dagger)$.

Under the assumptions (H1)–(H3), it is known [9] that problem (1.1)–(1.3) has a unique nonnegative solution, global in time.

Due to the finite maximum age $a_\dagger$, the mortality rate μ is unbounded near the maximum age [9]. Following [11], we assume the following general growth rates near $a_\dagger$:

(H4) There exist positive constants $a_* < a_\dagger$ and $\overline{\mu}$ such that $0 \le \mu(a) \le \overline{\mu}$ for $a \in [0, a_*]$,
(H5) $\mu(a) = \lambda/(a_\dagger - a)^\alpha$ on $(a_*, a_\dagger]$, for some $0 < \alpha < 1$, $\lambda > 0$.

*Received by the editors February 10, 2005; accepted for publication (in revised form) August 8, 2005; published electronically January 6, 2006. This research was supported by KOSEF(R04-2003-000-10024) and by KRF(2003-070-C00007).
http://www.siam.org/journals/sisc/27-4/62418.html
†Department of Mathematics, Inha University, Inchon 402-751, Korea (mikim@inha.ac.kr).

In traditional models of population dynamics, the maximum age $a_\dagger$ is assumed to be infinite and the mortality function μ is typically assumed to be Lipschitz continuous, for the sake of convenience. However, the growing complexity of physiologically structured population model makes it increasingly unrealistic to model the problems in ecology, epidemiology, population genetics, and cell growth with an infinite maximum age and a Lipschitz continuous mortality function. On the other hand, for the last several decades there has been much investigation for solving models with an infinite maximum age and a Lipschitz continuous mortality function. Most of them involve uniform time and age discretization with the age step chosen to be equal to the time step. We refer to [1, 7, 10, 12, 14, 16, 17, 18] and the references cited therein for more details. Recently, in [2] Ayati used a moving age discretization that allows nonuniform age and time discretizations.

In this paper, we consider the case that the maximum age is finite and the mortality function is unbounded. In such a case, regularity of the solution depends on the growth rate of the mortality function and so does the proper choice of the numerical method to approximate the solution; if the solution is not regular, finite difference methods such as the method of characteristics are not desirable or may not be applicable to the problem [11]. In this paper, we for the first time propose a method which is able to treat the problem with unbounded coefficient function and nonsmooth solution. More precisely, we consider the case that $0 < \alpha < 1$, $\lambda > 0$ and thus that the solution does not belong to $C^1([0, a_\dagger] \times [0, T])$ [15]. Discontinuous Galerkin finite element methods are proposed to approximate the model. There are difficulties in the problem treating nonlocal birth process (1.2) and the unbounded mortality function. To resolve them, we find some condition on the triangulation, say, (T1) in section 2, which guarantees the strong stability and accordingly the optimal L^2 error estimates. With locally defined test functions we could then show the (weak) stability with the L^2 norm. Using this result, we finally obtain the strong stability. Our method is easy to implement; the resulting discrete system decouples the nonlocal birth process from the system and it is blockwise solvable. We note that the discontinuous Galerkin method applied in age-time discretization allows varying age and time stepping. Thanks to the variable time steps one can adaptively choose the time steps to assure robustness and efficiency. Advantages of independent age and time discretizations are fewer computations and less memory use when the dependence on age is weak relative to the dependence on time. Our scheme is especially effective in a large system including space variables; if the considered model is especially spatially heterogeneous, then it becomes a more than four-dimensional problem (including two or three space dimensions) and it is very important to use an efficient mesh in age-time domain. In fact, there are many situations, such as Proteus mirabilis swarm colony development, in which there are very rapid changes in space requiring very small time steps, although the age dependence is smooth (see, for example, [2, 6, 20]). Our numerical method will help form a systematic basis for a simulation theory for physiologically structured such as age or body, individual-based population models, together with space variables. On the other hand, for the last several decades, discontinuous Galerkin methods have been successfully applied to many problems. For the discontinuous Galerkin methods applied to general problems, we refer to [5] and the references cited therein.

In [2, 3], Ayati considered moving age discretization along the characteristic (leaving the time variable to be continuous). There, the mortality function was assumed to be bounded. Stability of the approximate solution to the problem was not discussed. Nonnegativity of the solution was not treated, either. Our approximation can be viewed as a generalization of them. Furthermore, our method is simpler to implement

without any transformation of the problem as opposed to the case in [2, 3]. We also show that our discrete solution is strongly stable. Nonnegativity of the solution is very important for the analysis of the long-time behavior as well as from a biological point of view. We find some condition on the triangulation to preserve the nonnegativity of the solution to (1.1)–(1.3), which reflect the important fact that age and time advance together. Such a condition is a generalization of the CFL condition which is obtained by applying the finite difference method to the problem. Concerning the error estimate, we show that the discrete solution converges at the rate of $r + 1/2$ away from the maximum age and it converges at the rate of $l + 1/2 + \alpha/2 - 1/q$, if $u \in W^{l,2q}$ near the maximum age, where r is the degree of the polynomial of the approximation space, $1/2 \le l \le r + 1$, and $q \ge 1$.

It is noted that for $\alpha > 1$ or $\alpha = 1$ and $\lambda \ge 1$, the solution belongs to $C^1([0, a_\dagger] \times [0, T])$ provided that the compatibility condition at the origin is satisfied with smooth data [15]. In that case, a collocation method would be a proper choice for the effective computation. However, if the solution does not belong to $C^1([0, a_\dagger] \times [0, T])$, then the collocation method does not run well [15].

The organization of the remainder of this article is as follows. In the next section, we present a discontinuous Galerkin method in the age-time domain to approximate the solution to (1.1)–(1.3). In section 3, we show the local and global stabilities of the discrete solutions. In section 4, we discuss convergence results of the scheme, in detail. In section 5, we investigate the nonnegativity of the solution on the special type of triangulation. In section 6, we present some numerical examples. Finally, in section 7, we give some concluding remarks.

2. Numerical method. In this section we introduce a discontinuous Galerkin method in the age-time domain $\Omega = (0, a_\dagger) \times (0, T)$ so that the resulting problem is considered as an advection-reaction equation in two "space" dimensions. Now, let $0 = t^0 < t^1 < t^2 < \cdots < t^N = T$ be the time step, let $\mathcal{T}$ be a triangulation of Ω which is defined strip by strip in time, and let $\mathcal{T}^n \subset \mathcal{T}$ be the triangulation on the strip $S^n := (0, a_\dagger) \times (t^{n-1}, t^n)$. Then, the triangulation $\mathcal{T}$ of Ω may be defined as

$$\mathcal{T} = \left\{K_{ni}^-, K_{ni}^+ : n = 1, 2, \cdots, i = 1, 2, \dots\right\} \cup \left\{K_{n0}^+\right\},$$

where K_{ni}^- is a triangle having two vertices on $t = t^{n-1}$ and K_{ni}^+ has its two vertices on $t = t^n$ as in Figure 2.1. We assume that $\mathcal{T}$ satisfies the following:

(T1) On K_{ni}^-, the interior angle of the left vertex lying on $t = t^{n-1}$ is less than or equal to $\pi/4$ and the interior angle of the right vertex lying on $t = t^{n-1}$ is less than or equal to $\pi/2$.

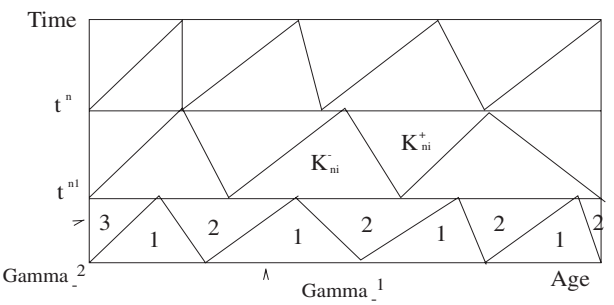


FIG. 2.1. *A sample triangulation.*

We here note that condition (T1) can be viewed as a generalization of the CFL condition which is obtained by applying the finite difference method to the linear hyperbolic problem.

Let $h = \max_{K\in\mathcal{T}} h_K$, where h_K denotes the diameter of $K \in \mathcal{T}$. We also assume that $\mathcal{T}$ is a regular family of triangulations in the sense that

$$\text{there exists a } \kappa > 0 \text{ such that } \frac{h}{h_{\min}} < \kappa, \quad h_{\min} = \min_{K\in\mathcal{T}} h_K. \tag{2.1}$$

Now, let Γ_- denote the inflow part of the boundary $\partial\Omega$: $\Gamma_- = \{x \in \partial\Omega : n(x)\cdot\gamma < 0\}$, where $\gamma = (1/\sqrt{2}, 1/\sqrt{2})$ is the unit vector of the convective direction and $n(x)$ is the outward unit normal to $\partial\Omega$ at x. Let $\Gamma_-^1 = \{(a,0) : 0 \le a \le a_\dagger\}$ and $\Gamma_-^2 = \{(0,t) : 0 \le t \le T\}$, and thus $\Gamma_- = \Gamma_-^1 \cup \Gamma_-^2$. Similarly, we denote $\partial_- K = \{x \in \partial K : n(x)\cdot\gamma < 0\}$ and $\partial_+ K = \{x \in \partial K : n(x)\cdot\gamma > 0\}$ for each $K \in \mathcal{T}$. Let $v^\pm(x) = \lim_{\epsilon\to 0} v(x \pm \epsilon\gamma)$ for $x \in (\cup_{K\in\mathcal{T}}\partial K)\backslash\partial\Omega$, and let $[v] = v^+ - v^-$.

A key mesh condition of our method is that the solution to (1.1)–(1.3) within each element K be uniquely determined by u on $\partial_- K$. The mesh will then be traversable in an explicit, element-by-element fashion, in the direction of increasing t, without violating any domain of dependence condition, as numbered in Figure 2.1. We here note that u on Γ_-^2 is not known in advance and u on K_{n0}^+ is determined in the end on each strip.

In what follows we use the space $L^p(\Omega)$ and the Sobolev spaces $W^{m,p}(\Omega)$ and $H^m(\Omega) = W^{m,2}(\Omega)$, $m \ge 1, 1 \le p \le \infty$ in their usual meaning for Ω a domain in R^2. $(\cdot,\cdot)$ denotes the inner product of $L^2(\Omega)$ and $(v,w)_{\Gamma_-} = \int_{\Gamma_-} vw|\gamma\cdot n|ds$, with analogous definition of $(\cdot,\cdot)_{\partial K}$. The norm in $L^p(\Omega)$ is denoted by $\|\cdot\|_{p,\Omega}$ if $p \ne 2$ and by $\|\cdot\|_\Omega$ if $p = 2$. Similarly, if Λ is a piecewise smooth curve or a union of such curves, $\|\cdot\|_{p,\Lambda}$ denotes the induced norm in $L^p(\Lambda)$, with the subindex p omitted if $p = 2$.

Let $P_r(K)$ denote the space of all polynomials of degree $\le r$ on K. Let W_h be the family of subspaces of $L^2(\Omega)$ given by

$$W_h = \{v \in L^2(\Omega) : v_{|K} \in P_r(K)\}. \tag{2.2}$$

Without loss of generality, we now write μ to represent $\frac{1}{\sqrt{2}}\mu$. Then, the discontinuous Galerkin method we shall analyze is given as follows. For each $K \in \mathcal{T}^n$, $1 \le n \le N$ and for given u_h^- on $\partial_- K$, find u_h on K such that $u_{h|K} \in P_r(K)$ and

$$\int_K (\gamma\cdot\nabla u_h + \mu u_h)v dx - \int_{\partial_- K} [u_h]v^+\gamma\cdot n ds = 0 \quad \forall\, v \in P_r(K) \tag{2.3}$$

with

$$u_h(a,0) = \widetilde{u_0}(a) \quad \text{if } (a,0) \in \Gamma_-^1, \tag{2.4}$$

$$u_h^-(0,t) = \int_0^{a_\dagger} \beta(a)u_h(a,t)da \quad \text{if } (0,t) \in \Gamma_-^2, \tag{2.5}$$

where $\widetilde{u_0}$ is the locally defined L^2-projection on Γ_-^1; see (3.18) for the definition of the locally defined L^2-projection. By summing over $K \in \mathcal{T}$ in (2.3) we obtain the following equivalent form of the discontinuous Galerkin method. Given r and $\mathcal{T}$, find

$u_h \in W_h$ such that for any $v \in W_h$,

$$\sum_{K\in\mathcal{T}} \int_K (\gamma\cdot\nabla u_h + \mu u_h) v dx - \sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} [u_h] v^+ \gamma\cdot n ds - \int_{\Gamma_-} u_h^+ v^+ \gamma\cdot n ds = -\int_{\Gamma_-^1} u_0 v^+ \gamma\cdot n ds - \int_{\Gamma_-^2} u_h^- v^+ \gamma\cdot n ds, \tag{2.6}$$

where $u_h^-(0,t)$ is given by (2.5).

System (2.3)–(2.5) is solved successively for u_h on each strip S^n, $1 \le n \le N$. We here note that u_h is always computed successively by starting from $K \in \mathcal{T}^n$ in our triangulation satisfying $\partial_- K \subset \Gamma_-^1$. As will be seen in Theorem 3.2, u_h is uniquely determined by (2.3)–(2.5). We here notice that the scheme is very easy to implement; the coefficient matrices of the discrete system on each strip are block diagonal matrices and each block is an $\{(r+1)(r+2)/2\} \times \{(r+1)(r+2)/2\}$ matrix.

Below, we denote by C a positive constant which may take different values on different occurrences. The constant may depend on the above parameter κ and r but not on other parameters, unless indicated explicitly.

3. Stability. For the stability analysis, we decompose $\mathcal{T}$ into two parts. Let $\mathcal{T}_1 = \{K \in \mathcal{T} :\ a < a_* \ \forall\, (a,t) \in \overline{K}\}$ and $\mathcal{T}_2 = \mathcal{T}\setminus\mathcal{T}_1$. We then see that $\mu(a) \le \overline{\mu}$ on $K\ \forall\ K \in \mathcal{T}_1$.

We shall frequently use the following inequalities, which are obtained by a scaling argument and by the equivalence of norms in a finite dimensional space. For any $w \in P_r(K)$ and $K \in \mathcal{T}$,

$$C^{-1}\{h_K^{-1/2}\|w\|_K + \|w\|_{\partial K}\} \le \{h_K\|\gamma\cdot\nabla w\|^2 + \|w\|_{\partial K}^2\}^{1/2} \le C h_K^{-1/2}\|w\|_K, \tag{3.1}$$

where C depends only on κ and K.

We also use the relations among various norms on the finite element space [4].

LEMMA 3.1. K R^n (2.1) $P_r(K)$ $W^{l,p}(K)\cap W^{m,q}(K)$ $1 \le p \le \infty,\ 1 \le q \le \infty,$ $0 \le m \le l$ $C = C(K,r,l,p,q,\kappa)$ $v \in P_r(K)$

$$\|v\|_{W^{l,p}(K)} \le C h^{m-l+n/p-n/q}\|v\|_{W^{m,q}(K)}. \tag{3.2}$$

One can always solve (2.3)–(2.5) successively, triangle by triangle, with u_h^- given on $\partial_- K$ either by the previously computed values or by the boundary conditions. Thus, u_h is determined uniquely if (2.3)–(2.5) is uniquely solved in each $K \in \mathcal{T}$ for given u_h^-. In fact, since the scheme (2.3)–(2.5) is linear, the unique solvability is a direct consequence of the following local stability.

THEOREM 3.2. h_0 κ K $h_0^{1-\alpha}\lambda < 1/16$ $h_0^\alpha < \min\{1/\sqrt{\overline{\mu}},\ 1/\overline{\mu}\}$ $h < h_0$

$$\|u_h\|_K + {h_K}^{1/2}\|u_h^+\|_{\partial_- K} + {h_K}^{1/2}\|u_h^-\|_{\partial_+ K} \le C {h_K}^{1/2}\|u_h^-\|_{\partial_- K},\ K \in \mathcal{T}, \tag{3.3}$$

$$\|\gamma\cdot\nabla u_h\|_K \le C\left\{\|u_h\|_K + h_K^{-1/2}\|[u_h]\|_{\partial_- K}\right\},\ K \in \mathcal{T}_1, \tag{3.4}$$

$$\|\gamma\cdot\nabla u_h\|_K \le C\left\{h_K^{-\alpha/2}\|\sqrt{\mu}u_h\|_K + h_K^{-1/2}\|[u_h]\|_{\partial_- K}\right\},\ K \in \mathcal{T}_2. \tag{3.5}$$

$\mathcal{T}_1 \cup \mathcal{T}_2$. From now on we denote $u_h = u_{h|K} \in P_r(K)$. Let

$$B_K(u_h, v) = \int_K (\gamma \cdot \nabla u_h + \mu u_h) v dx - \int_{\partial_- K} u_h^+ v^+ \gamma \cdot n ds. \tag{3.6}$$

To prove (3.3), we choose $v = u_h + \delta h_K \gamma \cdot \nabla u_h$, where $\delta < 1$ is determined below. We note, by integration by parts, that

$$\int_K u_h \gamma \cdot \nabla u_h dx = \frac{1}{2} \int_{\partial K} u_h^2 \gamma \cdot n ds. \tag{3.7}$$

Since $\delta < 1$, we easily see, by (3.1) and (3.7), that

$$\|v\|_K^2 \le (1 + C\delta + C\delta^2) \|u_h\|_K^2. \tag{3.8}$$

We have from (3.6) that

$$\begin{aligned} B_K(u_h, v) = & \int_K \mu u_h^2 dx + \delta h_K \int_K (\gamma \cdot \nabla u_h)^2 dx + \frac{1}{2} \int_{\partial K} u_h^2 |\gamma \cdot n| ds \\ & + \delta h_K \int_K \mu u_h \gamma \cdot \nabla u_h dx - \delta h_K \int_{\partial_- K} u_h^+ (\gamma \cdot \nabla u_h)^+ \gamma \cdot n ds \\ \ge & \, \delta h_K \|\gamma \cdot \nabla u_h\|_K^2 + \frac{1}{2} \|u_h\|_{\partial K}^2 \\ & + \delta h_K \int_K \mu u_h \gamma \cdot \nabla u_h dx - \delta h_K \int_{\partial_- K} u_h^+ (\gamma \cdot \nabla u_h)^+ \gamma \cdot n ds. \end{aligned} \tag{3.9}$$

If $K \in \mathcal{T}_1$ and $h^\alpha < 1/\sqrt{\overline{\mu}}$, then we see, from (3.9), that

$$B_K(u_h, v) \ge \frac{\delta}{4} \{1 - C_1 \delta - C_1 h_K \overline{\mu}\} \Big\{ h_K \|\gamma \cdot \nabla u_h\|_K^2 + \|u_h\|_{\partial K}^2 \Big\}. \tag{3.10}$$

Choose $\delta = \min\big\{1, 1/(2C_1)\big\}$. We then obtain that if $h_K \le 1/(4C_1 \overline{\mu})$,

$$B_K(u_h, v) \ge C^{-1} \Big\{ h_K^{-1} \|u_h\|_K^2 + \|u_h\|_{\partial K}^2 \Big\}. \tag{3.11}$$

On the other hand, by (2.3) and (3.8), we have

$$\begin{aligned} B_K(u_h, v) = \int_{\partial_- K} u_h^- v^+ |\gamma \cdot n| ds & \le C h_K^{-1/2} \|u_h^-\|_{\partial_- K} \|v\|_K \\ & \le C h_K^{-1/2} \|u_h^-\|_{\partial_- K} \|u_h\|_K. \end{aligned} \tag{3.12}$$

Combining (3.11) and (3.12), we obtain (3.3) for $K \in \mathcal{T}_1$.

If $K \in \mathcal{T}_2$, then we obtain, by (3.1) and Young's inequality, that

$$\begin{aligned} \int_K \mu u_h \gamma \cdot \nabla u_h dx & \le 4\lambda h_K^{-2\alpha} \|u_h\|_K^2 + \frac{1}{4} \|\gamma \cdot \nabla u_h\|_K^2 \\ & \le 4\lambda h_K^{1-2\alpha} \{h_K \|\gamma \cdot \nabla u_h\|_K^2 + \|u_h\|_{\partial K}^2\} + \frac{1}{4} \|\gamma \cdot \nabla u_h\|_K^2 \\ & \le 4\lambda h_K^{-\alpha} \{h_K \|\gamma \cdot \nabla u_h\|_K^2 + \|u_h\|_{\partial K}^2\} + \frac{1}{4} \|\gamma \cdot \nabla u_h\|_K^2 \end{aligned} \tag{3.13}$$

and that

$$\int_{\partial_- K} u_h^+(\gamma\cdot\nabla u_h)^+|\gamma\cdot n|ds \le \frac{1}{4}\|\gamma\cdot\nabla u_h\|_K^2 + 4h_K^{-1}\|u_h\|_{\partial K}^2. \tag{3.14}$$

Thus, combining (3.9) and (3.13)–(3.14), we see that

$$B_K(u_h,v)\ge \delta\left\{\frac{1}{2}-4\lambda h_K^{1-\alpha}\right\}h_K\|\gamma\cdot\nabla u_h\|_K^2 + \left\{\frac{1}{2}-4\delta\lambda h_K^{1-\alpha}-4\delta\right\}\|u_h\|_{\partial K}^2.$$

Now noting that $h_K^{1-\alpha}\lambda < \frac{1}{16}$, take $\delta \le \frac{1}{9}$. We then have, by (3.1), that

$$\begin{aligned} B_K(u_h,v) &\ge \frac{\delta}{4}\left\{h_K\|\gamma\cdot\nabla u_h\|_K^2 + \|u_h\|_{\partial K}^2\right\} \\ &\ge \frac{\delta}{4}C\left\{h_K^{-1}\|u_h\|_K^2 + \|u_h\|_{\partial K}^2\right\}. \end{aligned} \tag{3.15}$$

Combining (3.12) and (3.15), and by Young's inequality, we obtain (3.3) for $K \in \mathcal{T}_2$.

To prove (3.5), let $K \in \mathcal{T}_2$. By taking $v = \gamma\cdot\nabla u_h$, we see from (2.3) that

$$\int_K(\gamma\cdot\nabla u_h + \mu u_h)\gamma\cdot\nabla u_h dx = \int_{\partial_- K}[u_h](\gamma\cdot\nabla u_h)^+\gamma\cdot n ds$$

or

$$\|\gamma\cdot\nabla u_h\|_K^2 + \int_K \mu u_h\gamma\cdot\nabla u_h dx \le \|[u_h]\|_{\partial_- K}\|\gamma\cdot\nabla u_h\|_{\partial_- K}.$$

Thus, by Young's inequality, we have that, if $h^\alpha < 1/\overline{\mu}$,

$$\begin{aligned} \frac{1}{2}\|\gamma\cdot\nabla u_h\|_K^2 &\le C\left\{\|\mu u_h\|_K^2 + h_K^{-1/2}\|[u_h]\|_{\partial_- K}\|\gamma\cdot\nabla u_h\|_K\right\} \\ &\le C\left\{h_K^{-\alpha}\|\sqrt{\mu}u_h\|_K^2 + h_K^{-1/2}\|[u_h]\|_{\partial_- K}\|\gamma\cdot\nabla u_h\|_K\right\} \end{aligned} \tag{3.16}$$

or

$$\|\gamma\cdot\nabla u_h\|_K \le C\left\{h_K^{-\alpha/2}\|\sqrt{\mu}u_h\|_K + h_K^{-1/2}\|[u_h]\|_{\partial_- K}\right\}. \tag{3.17}$$

This completes the proof of (3.5). Similarly, one can prove (3.4) and we shall omit the proof . □

Now, let $\widetilde{\chi}$ denote the locally defined L^2-projection of χ into W_h for each $\chi \in L^2(\Omega)$. That is, $\widetilde{\chi}$ is defined by

$$(\chi - \widetilde{\chi}, v)_K = 0 \ \forall\ v \in P_r(K), \quad K \in \mathcal{T}. \tag{3.18}$$

Then we shall need the following approximation properties in what follows [19]. For $2 \le q \le \infty$,

$$\begin{aligned} &||\chi-\widetilde{\chi}||_K \le Ch_K^{r+1}|\chi|_{H^{r+1}(K)}, \\ &||\chi-\widetilde{\chi}||_{\partial K} \le Ch_K^{r+1/2}|\chi|_{H^{r+1}(K)}, \\ &||\chi-\widetilde{\chi}||_{L^q(K)} \le Ch_K^l||\chi||_{W^{l,q}(K)},\ \ 0 \le l \le r+1, \\ &\qquad\qquad \text{if } \chi \in L^2(\Omega)\cap W^{l,q}(K), \\ &||\chi-\widetilde{\chi}||_{L^q(\partial K)} \le Ch_K^{l-1/2}||\chi||_{W^{l,q}(K)},\ \ 1/2 \le l \le r+1, \\ &\qquad\qquad \text{if } \chi \in L^2(\Omega)\cap W^{l,q}(K). \end{aligned} \tag{3.19}$$

To establish the global stability, we define some notations. Let Γ_h be the set of all interior edges and Γ be the set of the boundary edges. That is,

$$\Gamma_h = \cup_K \partial_- K \backslash \Gamma_- = \cup_K \partial_+ K \backslash \Gamma_+, \qquad \Gamma = \Gamma_- \cup \Gamma_+,$$

where $\Gamma_+ = \{x \in \partial\Omega : n(x) \cdot \gamma > 0\}$.

We now define

$$\begin{aligned} |||v|||^2 &= \|v\|_\Omega^2 + \|\sqrt{\mu} v\|_\Omega^2 + \sum_{K \in \mathcal{T}} h_K \|\gamma \cdot \nabla v\|_K^2 \\ &\quad + \|[v]\|_{\Gamma_h}^2 + \|v^+\|_{\Gamma_-}^2 + \|v^-\|_{\Gamma_+}^2. \end{aligned} \tag{3.20}$$

For the (strong) global stability analysis, we first show the weak stability with the $L^2(\Omega)$-norm. In the following lemma, we need an angle condition on K_{n0}^+ to treat the nonlocal birth process (2.5).

LEMMA 3.3. [illegible] $(0, t^{n-1})$ [illegible] K_{n0}^+ [illegible] $\pi/4$ [illegible] $r \geq 0$ [illegible] C [illegible]

$$\|u_h\|_\Omega \leq C \|u_0\|_{\Gamma_-^1}, \tag{3.21}$$

$$\|u_h\|_{\Gamma_-^2} \leq C \|u_0\|_{\Gamma_-^1}. \tag{3.22}$$

The idea of the proof is similar to the one of the following theorem and we postpone the proof to the next theorem.

If $r \leq 1$, then the weak stability is established without the angle condition on K_{n0}^+. The proof is given by using the nonnegativity of u_h and it is given in the appendix.

LEMMA 3.4. [illegible] $r \leq 1$ [illegible] C [illegible]

$$\|u_h\|_\Omega \leq C \|u_0\|_{\Gamma_-^1},$$

$$\|u_h\|_{\Gamma_-^2} \leq C \|u_0\|_{\Gamma_-^1}.$$

Using the weak stability results, we now prove that u_h is globally stable with the norm $|||\cdot|||$.

THEOREM 3.5. [illegible] W_h [illegible] (2.2) [illegible] $r \geq 0$ [illegible] 3.3 [illegible] 3.4 [illegible] h_0 [illegible] κ [illegible] K [illegible] $h_0^{1-\alpha}\lambda < 1$ [illegible] $h < h_0$ [illegible] (2.6) [illegible]

$$|||u_h||| \leq C \|u_0\|_{\Gamma_-^1},$$

[illegible] $C(a_\dagger, T)$ [illegible] h [illegible] λ [illegible]. The discontinuous Galerkin method (2.6) is written as

$$B(u_h, v) = l(u_h; v), \tag{3.23}$$

where

$$\begin{aligned} B(u_h, v) &= \sum_{K \in \mathcal{T}} \int_K (\gamma \cdot \nabla u_h + \mu u_h) v dx - \sum_{K \in \mathcal{T}} \int_{\partial_- K \backslash \Gamma_-} [u_h] v^+ \gamma \cdot n ds \\ &\quad - \int_{\Gamma_-} u_h^+ v^+ \gamma \cdot n ds, \\ l(u_h; v) &= - \int_{\Gamma_-^1} \widetilde{u_0} v^+ \gamma \cdot n ds - \int_{\Gamma_-^2} u_h(0,t) v^+ \gamma \cdot n ds \end{aligned}$$

with $u_h(0,t)$ given by (2.5).

Following [13], we consider $\psi(a) = \exp(-\delta a)$, where $\delta = \ln 2/(a_\dagger + 1)$. Then, $\gamma \cdot \nabla\psi = -\delta\psi$, $\|\psi\|_{L^\infty(\Omega)} = 1, |\psi| > 1/2$. We now choose $v = \widetilde{\psi u_h}$. We see that

$$B(u_h, \widetilde{\psi u_h}) = B(u_h, \psi u_h) - B(u_h, \psi u_h - \widetilde{\psi u_h}).$$

On the other hand, we easily see, by integration by parts, that

$$\int_K (\gamma \cdot \nabla u_h)\psi u_h dx = \frac{1}{2}\left\{\int_{\partial K} \psi u_h^2 \gamma \cdot n ds + \delta \int_K \psi u_h^2 dx\right\}.$$

Thus, we have that

$$\begin{aligned} B(u_h, \psi u_h) =& \frac{1}{2}\delta \sum_{K\in\mathcal{T}} \int_K \psi u_h^2 dx + \sum_{K\in\mathcal{T}} \int_K \psi\mu u_h^2 dx + \frac{1}{2}\sum_{K\in\mathcal{T}} \int_{\partial K} \psi u_h^2 \gamma \cdot n ds \\ &- \sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} [u_h](\psi u_h)^+ \gamma \cdot n ds - \int_{\Gamma_-} u_h^+ (\psi u_h)^+ \gamma \cdot n ds \\ =& \frac{1}{2}\delta \sum_{K\in\mathcal{T}} \int_K \psi u_h^2 dx + \sum_{K\in\mathcal{T}} \int_K \psi\mu u_h^2 dx - \frac{1}{2}\sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} \psi [u_h]^2 \gamma \cdot n ds \\ &+ \frac{1}{2}\sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} \psi (u_h^-)^2 \gamma \cdot n ds \\ &+ \frac{1}{2}\sum_{K\in\mathcal{T}} \int_{\partial_+ K} \psi (u_h^-)^2 \gamma \cdot n ds - \frac{1}{2}\int_{\Gamma_-} \psi (u_h^+)^2 \gamma \cdot n ds. \end{aligned} \tag{3.24}$$

Since $\partial_- K\setminus\Gamma_-$ is a part of $\partial_+ K'$ for some $K' \in \mathcal{T}$ and the sign of the outer normal of $\partial_- K\setminus\Gamma_-$ is opposite of the one of $\partial_+ K'$, we have that

$$\sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} \psi (u_h^-)^2 \gamma \cdot n ds + \sum_{K\in\mathcal{T}} \int_{\partial_+ K} \psi (u_h^-)^2 \gamma \cdot n ds = \int_{\Gamma_+} \psi (u_h^-)^2 \gamma \cdot n ds.$$

Thus we have, from (3.24), that

$$\begin{aligned} B(u_h, \psi u_h) \geq& \frac{1}{4}\delta \|u_h\|_\Omega^2 + \frac{1}{2}\|\sqrt{\mu} u_h\|_\Omega^2 - \frac{1}{4}\sum_{K\in\mathcal{T}} \int_{\partial_- K\setminus\Gamma_-} [u_h]^2 \gamma \cdot n ds \\ &+ \frac{1}{4}\int_{\Gamma_+} \psi (u_h^-)^2 \gamma \cdot n ds - \frac{1}{4}\int_{\Gamma_-} (u_h^+)^2 \gamma \cdot n ds \\ \geq& \frac{1}{4}\delta \|u_h\|_\Omega^2 + \frac{1}{2}\|\sqrt{\mu} u_h\|_\Omega^2 + \frac{1}{4}\|[u_h]\|_{\Gamma_h}^2 + \frac{1}{4}\|u_h^+\|_{\Gamma_-}^2 + \frac{1}{4}\|u_h^-\|_{\Gamma_+}^2. \end{aligned} \tag{3.25}$$

Noting that $\|\psi\|_{W^{l,\infty}(K)} \leq C\delta^l$, $\|u_h\|_{H^l(K)} \leq Ch_K^{-l}\|u_h\|_K$, $|u_h|_{H^{r+1}(K)} = 0$, we have, by Leibniz's rule and (3.19), the following properties of the local projection. If $\delta h \leq 1$, then

$$\begin{aligned} &\|\psi u_h - \widetilde{\psi u_h}\|_K + h_K^{1/2}\|\psi u_h - \widetilde{\psi u_h}\|_{\partial K} \leq Ch_K^{r+1}|\psi u_h|_{H^{r+1}(K)} \\ &= Ch_K^{r+1}\sum_{l=0}^{r+1} \|\psi\|_{W^{l,\infty}(K)}|u_h|_{H^{r+1-l}(K)} \\ &= Ch_K^{r+1}\sum_{l=1}^{r+1} \|\psi\|_{W^{l,\infty}(K)}|u_h|_{H^{r+1-l}(K)} \end{aligned} \tag{3.26}$$

$$\leq Ch_K^{r+1}\sum_{l=1}^{r+1}\delta^l h_K^{l-(r+1)}\|u_h\|_K$$
$$\leq C\delta h\sum_{l=0}^{r}\delta^l h_K^l \delta h_K\|u_h\|_K$$
$$\leq C(r+1)\delta h_K\|u_h\|_K.$$

We also have, by (3.2) and (3.19), that

$$\begin{aligned}\|\psi u_h-\widetilde{\psi u_h}\|_{L^\infty(K)}+h_K^{1/2}\|\psi u_h-\widetilde{\psi u_h}\|_{L^\infty(\partial K)}&\leq Ch_K^{r+1}|\psi u_h|_{W^{r+1,\infty}(K)}\\ &\leq C(r+1)\delta h_K\|u_h\|_{L^\infty(K)}\\ &\leq C(r+1)\delta\|u_h\|_{L^2(K)}.\end{aligned}$$

Since $\int_K(\psi u_h-\widetilde{\psi u_h})w dx=0\ \forall\ w\in P_r(K)$, we see that

$$\int_K(\psi u_h-\widetilde{\psi u_h})\gamma\cdot\nabla u_h dx=0. \tag{3.27}$$

We thus have, by Hölder's inequality, that

$$\begin{aligned}B(u_h,\psi u_h-\widetilde{\psi u_h})=&\sum_{K\in\mathcal{T}}\int_K\mu u_h(\psi u_h-\widetilde{\psi u_h})dx\\ &-\sum_{K\in\mathcal{T}}\int_{\partial_- K\setminus\Gamma_-}[u_h](\psi u_h-\widetilde{\psi u_h})^+\gamma\cdot n ds\\ &-\int_{\Gamma_-}u_h^+(\psi u_h-\widetilde{\psi u_h})^+\gamma\cdot n ds\\ \leq&\frac{1}{4}\|\sqrt{\mu}u_h\|_\Omega^2+C\sum_{K\in\mathcal{T}_1}h_K^2\|\psi u_h-\widetilde{\psi u_h}\|_{L^\infty(K)}^2\\ &+C\lambda\sum_{K\in\mathcal{T}_2}h_K^{2-\alpha}\|\psi u_h-\widetilde{\psi u_h}\|_{L^\infty(K)}^2\\ &+\frac{1}{8}\|[u_h]\|_{\Gamma_h}^2+8\sum_{K\in\mathcal{T}}\|\psi u_h-\widetilde{\psi u_h}\|_{\partial_- K}^2+\frac{1}{8}\|u_h^+\|_{\Gamma_-}^2\\ \leq&\frac{1}{4}\|\sqrt{\mu}u_h\|_\Omega^2+C\delta\sum_{K\in\mathcal{T}_1}h_K^2\|u_h\|_K^2+C\lambda\delta\sum_{K\in\mathcal{T}_2}h_K^{2-\alpha}\|u_h\|_K^2\\ &+\frac{1}{8}\|[u_h]\|_{\Gamma_h}^2+8\delta\sum_{K\in\mathcal{T}}h_K\|u_h\|_K^2+\frac{1}{8}\|u_h^+\|_{\Gamma_-}^2.\end{aligned} \tag{3.28}$$

Combining (3.25) and (3.28), we then have, if $0<\lambda h<\frac{1}{16}$,

$$\begin{aligned}B(u_h,\widetilde{\psi u_h})\geq&\frac{1}{16}\delta\|u_h\|_\Omega^2+\frac{1}{4}\|\sqrt{\mu}u_h\|_\Omega^2\\ &+\frac{1}{8}\|[u_h]\|_{\Gamma_h}^2+\frac{1}{8}\|u_h^+\|_{\Gamma_-}^2+\frac{1}{8}\|u_h^-\|_{\Gamma_+}^2.\end{aligned} \tag{3.29}$$

We also see, by Lemma 3.3, that

$$
\begin{aligned}
B(u_h, \widetilde{\psi u_h}) &= -\int_{\Gamma_-^1} \widetilde{u_0}\widetilde{\psi u_h}^+ \gamma \cdot n ds - \int_{\Gamma_-^2} u_h(0,t)\widetilde{\psi u_h}^+ \gamma \cdot n ds \\
&\le \|\widetilde{u_0}\|_{\Gamma_-^1} \|\widetilde{\psi u_h}^+\|_{\Gamma_-^1} + \|u_h(0,t)\|_{\Gamma_-^2} \|\widetilde{\psi u_h}^+\|_{\Gamma_-^2} \\
&\le \left\{ \|\widetilde{u_0}\|_{\Gamma_-^1} + \|u_h(0,t)\|_{\Gamma_-^2} \right\} \left\{ \|u_h^+\|_{\Gamma_-} + \|u_h^-\|_{\Gamma_+} \right\} \\
&\le C\|\widetilde{u_0}\|_{\Gamma_-^1} \left\{ \|u_h^+\|_{\Gamma_-} + \|u_h^-\|_{\Gamma_+} \right\}.
\end{aligned} \tag{3.30}
$$

On the other hand, we note from (3.4)–(3.5) that

$$
\sum_{K\in\mathcal{T}} h_K \|\gamma\cdot\nabla u_h\|_K^2 \le C \left\{ \sum_{K\in\mathcal{T}_1} h_K \|u_h\|_K^2 + \sum_{K\in\mathcal{T}_2} \|\sqrt{\mu} u_h\|_K^2 + \sum_{K\in\mathcal{T}} \|[u_h]\|_{\partial_- K}^2 \right\}. \tag{3.31}
$$

Thus, from (3.20) and Lemma 3.3, and combining (3.29)–(3.31), we obtain that

$$
\begin{aligned}
|||u_h|||^2 &\le C \left\{ \sum_{K\in\mathcal{T}_1} \|u_h\|_K^2 + \sum_{K\in\mathcal{T}_2} \|\sqrt{\mu} u_h\|_K^2 + \|[u_h]\|_{\Gamma_h}^2 + \|u_h^+\|_{\Gamma_-}^2 \right\} \\
&\le C\|u_0\|_{\Gamma_-^1} \left\{ \|u_h^+\|_{\Gamma_-} + \|u_h^-\|_{\Gamma_+} \right\}
\end{aligned} \tag{3.32}
$$

or

$$
|||u_h||| \le C\|u_0\|_{\Gamma_-^1}.
$$

This completes the proof. □

3.3. Choose

$$
v(x) = \begin{cases} \widetilde{\psi u_h}(x) & \text{if } x \in K \text{ and } K \in \mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N, \\ 0 & \text{otherwise.} \end{cases} \tag{3.33}
$$

We then see that

$$
B(u_h, v) = B_0(u_h, \widetilde{\psi u_h}) - B_0(u_h, \psi u_h - \widetilde{\psi u_h}), \tag{3.34}
$$

where

$$
\begin{aligned}
B_0(u_h, w) = &\sum_{K\in\mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N} \int_K (\gamma\cdot\nabla u_h + \mu u_h) w dx \\
&- \sum_{K\in\mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N} \int_{\partial_- K\backslash\Gamma_-} [u_h] w^+ \gamma\cdot n ds - \int_{\Gamma_-^1} u_h^+ w^+ \gamma\cdot n ds.
\end{aligned}
$$

In a similar way to (3.24)–(3.25), we see, by integration by parts, that

$$
\begin{aligned}
B_0(u_h, \psi u_h) \ge &\frac{1}{4}\delta \sum_{K\in\mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N} \|u_h\|_K^2 + \frac{1}{2} \sum_{K\in\mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N} \|\sqrt{\mu} u_h\|_K^2 \\
&+ \frac{1}{4} \sum_{K\in\mathcal{T}\backslash\{K_{n0}^+\}_{n=1}^N} \|[u_h]\|_{\partial_- K\backslash\Gamma_-}^2 + \frac{1}{4}\|u_h^+\|_{\Gamma_-^1}^2 + \frac{1}{4}\|u_h^-\|_{\Gamma_+^1}^2.
\end{aligned} \tag{3.35}
$$

In a similar way to (3.28), we also see that

$$B_0(u_h, \psi u_h - \widetilde{\psi u_h}) \le \frac{1}{4} \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|\sqrt{\mu} u_h\|^2_K + \frac{1}{8}\delta \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|u_h\|^2_K$$
$$+ \frac{1}{8} \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|[u_h]\|^2_{\partial_- K\setminus\Gamma_-} + \frac{1}{8}\|u_h^+\|^2_{\Gamma^1_-}. \tag{3.36}$$

Combining (3.35)–(3.36), we have that

$$B_0(u_h, \widetilde{\psi u_h}) \ge \frac{1}{8}\delta \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|u_h\|^2_K$$
$$+ \frac{1}{8} \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|[u_h]\|^2_{\partial_- K} + \frac{1}{8}\|u_h^+\|^2_{\Gamma^1_-} + \frac{1}{8}\|u_h^-\|^2_{\Gamma_+}. \tag{3.37}$$

On the other hand, we have, by the choice of v and (3.34), that

$$\begin{aligned} B(u_h, v) &= l(u_h; v) \\ &= -\int_{\Gamma^1_-} \widetilde{u_0} v^+ \gamma\cdot n ds - \int_{\Gamma^2_-} u_h^- v^+ \gamma\cdot n ds \\ &= -\int_{\Gamma^1_-} \widetilde{u_0}\widetilde{\psi u_h}^+ \gamma\cdot n ds \\ &\le \|u_0\|_{\Gamma^1_-}\|u_h^+\|_{\Gamma^1_-}. \end{aligned} \tag{3.38}$$

Thus, we obtain by (3.37)–(3.38) that

$$\sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|u_h\|^2_K + \sum_{K\in\mathcal{T}\setminus\{K^+_{n0}\}_{n=1}^N} \|[u_h]\|^2_{\partial_- K} + \frac{1}{8}\|u_h^+\|^2_{\Gamma^1_-} + \frac{1}{8}\|u_h^-\|^2_{\Gamma_+}$$
$$\le C\|u_0\|^2_{\Gamma^1_-}. \tag{3.39}$$

We now have, by (2.5) and (3.3), and by the angle condition on K^+_{n0}, that

$$\begin{aligned} \|u_h\|^2_{K^+_{n0}} &\le h_{K^+_{n0}}\|u_h^-\|^2_{\Gamma^{2,n}_-} \\ &\le C h_{K^+_{n0}} \sum_{K\in\mathcal{T}^n} \|u_h\|^2_K. \end{aligned} \tag{3.40}$$

Summing over $n = 1, \dots, N$, we see from (3.40) that

$$\sum_{n=1}^N \|u_h\|^2_{K^+_{n0}} \le C \sum_{K\in\mathcal{T}} h_{K^+_{n0}} \|u_h\|^2_K. \tag{3.41}$$

Adding (3.39) and (3.41), we have, for sufficiently small $h_0 > 0$, that

$$\|u_h\|_\Omega \le C\|u_0\|_{\Gamma^1_-},$$

which proves (3.21). Thus, we have that

$$\|u_h\|_{\Gamma^{2,n}_-} \le C\|u_0\|_{\Gamma^1_-},$$

which shows (3.22). This completes the proof. □

4. L^2 error analysis. Convergence rates of the approximate solution obviously depend on the regularity of the solution to the problem. In this section, for the error estimate we let $\mathcal{T}_{22} = \{K \in \mathcal{T} : u \notin H^{r+1}(K)\}$.

We also consider the following extension of the projection $\widetilde{u}$ on Γ_-^2:

$$\widetilde{u}(0,t) = \int_0^{a_\dagger} \beta(a)\widetilde{u}(a,t)da. \tag{4.1}$$

To derive an error bound, we adopt the following notation: $e = u_h - u$, $\theta = u_h - \tilde{u}$, $-\rho = \tilde{u} - u$.

We now note from Hölder's inequality and (3.19) that if $K \in \mathcal{T}_{22}$, then

$$\begin{aligned} \|\rho\|_K &\le Ch_K^{1/p}\|\rho\|_{L^{2q}(K)} \\ &\le Ch_K^{1/p+l}\|u\|_{W^{l,2q}(K)} \\ &\le Ch_K^{l+1-1/q}\|u\|_{W^{l,2q}(K)} \end{aligned} \tag{4.2}$$

and

$$\|\rho\|_{\partial K} \le Ch_K^{l+1/2-1/q}\|u\|_{W^{l,2q}(K)}, \tag{4.3}$$

where $1/p + 1/q = 1$, $p, q \ge 1$, and $1/2 \le l \le r+1$. We also see from (1.2) and (2.5) that

$$\begin{aligned} \|\rho\|_{\Gamma_-^2}^2 &\le C\sum_{K\in\mathcal{T}} \|\rho\|_{L^2(K)}^2 \\ &\le C\Bigg\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+2}\|u\|_{H^{r+1}(K)}^2 + \sum_{K\in\mathcal{T}_{22}} h_K^{2l+2-2/q}\|u\|_{W^{l,2q}(K)}^2 \Bigg\}, \end{aligned} \tag{4.4}$$

where $q \ge 1$ and $0 \le l \le r+1$. Combining (3.19) and (4.2)–(4.4), we easily obtain the following lemma.

LEMMA 4.1. W_h (2.2 $r \ge 0$ u $H^{r+1}(K)$ $K \in \mathcal{T}\setminus\mathcal{T}_{22}$ $W^{l,2q}(K)$ $K \in \mathcal{T}_{22}$ $1/2 \le l \le r+1$ $q \ge 1$ $C > 0$ λ h

$$|||\rho||| \le C\Bigg\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{r+1/2}\|u\|_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}_{22}} h_K^{l+1/2-1/q}\|u\|_{W^{l,2q}(K)} \Bigg\}. \tag{4.5}$$

We are now ready to prove the main theorem of this section concerning the error estimates for the solution to (2.3)–(2.5).

THEOREM 4.2. W_h (2.2) $r \ge 0$ u $H^{r+1}(K)$ $K \in \mathcal{T}\setminus\mathcal{T}_{22}$ $W^{l,2q}(K)$ $K \in \mathcal{T}_{22}$ $1/2 \le l \le r+1$ $q \ge 1$ h_0 $(h_0\lambda < 1/16)$ κ r $h < h_0$

$$\begin{aligned} |||u - u_h|||^2 \le C\Bigg\{ &\sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+1}\|u\|_{H^{r+1}(K)}^2 + \sum_{K\in\mathcal{T}_{22}} h_K^{2l+1-2/q}\|u\|_{W^{l,2q}(K)}^2 \\ &+ \sum_{K\in\mathcal{T}} h_K^{2r+2}\|u_0\|_{H^{r+1}(\partial_- K\cap\Gamma_-^1)}^2 \Bigg\}, \end{aligned}$$

$C(a_\dagger, T)$ λ h

. We have the following error equation: $\forall\, v \in W_h$,

$$B(\theta, v) = B(\rho, v) - \int_{\Gamma_-^2} \theta^- v^+ \gamma \cdot n ds + \int_{\Gamma_-^2} \rho^- v^+ \gamma \cdot n ds + \int_{\Gamma_-^1} \rho_0 v^+ \gamma \cdot n ds, \tag{4.6}$$

where $\rho_0 = u_0(a) - \widetilde{u_0}(a)$. We now choose $v = \widetilde{\psi\theta}$. We note that (3.29) also holds for u_h replaced by θ. Thus, we have that

$$B(\theta, \widetilde{\psi\theta}) \geq C\Big\{ \|\theta\|_\Omega^2 + \|\sqrt{\mu}\theta\|_\Omega^2 + \|[\theta]\|_{\Gamma_h}^2 + \|\theta^+\|_{\Gamma_-}^2 + \|\theta^-\|_{\Gamma_+}^2 \Big\}. \tag{4.7}$$

To estimate the right-hand side of (4.6), we first rewrite $B(\rho, \widetilde{\psi\theta})$. Since $\int_K \rho\gamma \cdot \nabla(\widetilde{\psi\theta})dx \equiv 0$, we see, by integration by parts, that

$$\begin{aligned} B(\rho, \widetilde{\psi\theta}) &= \sum_{K\in\mathcal{T}} \int_K (\gamma \cdot \nabla\rho + \mu\rho)\widetilde{\psi\theta} dx - \sum_{K\in\mathcal{T}} \int_{\partial_- K \setminus \Gamma_-} [\rho] \widetilde{\psi\theta}^+ \gamma \cdot n ds - \int_{\Gamma_-} \rho^+ \widetilde{\psi\theta}^+ \gamma \cdot n ds \\ &= \sum_{K\in\mathcal{T}} \int_K \mu\rho\widetilde{\psi\theta} dx + \sum_{K\in\mathcal{T}} \int_{\partial_- K \setminus \Gamma_-} \rho^- [\widetilde{\psi\theta}] \gamma \cdot n ds + \int_{\Gamma_+} \rho^- \widetilde{\psi\theta}^- \gamma \cdot n ds. \end{aligned} \tag{4.8}$$

If $K \in \mathcal{T}_2$, then, by Hölder's inequality and (3.1), we see that

$$\|\sqrt{\mu}\rho\|_K^2 \leq C\|\mu\|_{L^1(K)}\|\rho\|_{L^\infty(K)}^2 \leq Ch_K^{2-\alpha}\|\rho\|_{L^\infty(K)}^2 \leq Ch_K^{-\alpha}\|\rho\|_K^2. \tag{4.9}$$

We thus obtain by (3.19), (4.2)–(4.4), (4.8), and (4.9) that

$$\begin{aligned} B(\theta, \widetilde{\psi\theta}) &= B(\rho, \widetilde{\psi\theta}) - \int_{\Gamma_-^2} \theta^- \widetilde{\psi\theta}^+ ds + \int_{\Gamma_-^2} \rho^- \widetilde{\psi\theta}^+ + \int_{\Gamma_-^1} \rho_0 \widetilde{\psi\theta}^+ \gamma \cdot n ds \\ &\leq C\Big\{ \sum_{K\in\mathcal{T}_1} \|\rho\|_K + \sum_{K\in\mathcal{T}_2} h_K^{-\alpha/2}\|\rho\|_K + \|\rho^-\|_{\Gamma_h} + \|\rho^-\|_{\Gamma_+} + \|\theta^-\|_{\Gamma_-^2} + \|\rho^-\|_{\Gamma_-^2} \\ &\qquad + \|\rho_0\|_{\Gamma_-^1} \Big\} \Big\{ \|\theta\|_\Omega + \sum_{K\in\mathcal{T}_2} \|\sqrt{\mu}\theta\|_K + \|[\theta]\|_{\Gamma_h} + \|\theta^+\|_{\Gamma_-} + \|\theta^-\|_{\Gamma_+} \Big\} \end{aligned} \tag{4.10}$$

$$\begin{aligned} &\leq C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{r+1/2}\|u\|_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}_{22}} h_K^{l+1/2-1/q}\|u\|_{W^{l,2q}(K)} \\ &\qquad + \sum_{K\in\mathcal{T}} h_K^{r+1}\|u_0\|_{H^{r+1}(\partial_- K\cap\Gamma_-^1)} + \|\theta^-\|_{\Gamma_-^2} \Big\} \\ &\qquad \times \Big\{ \|\theta\|_\Omega + \sum_{K\in\mathcal{T}_2} \|\sqrt{\mu}\theta\|_K + \|[\theta]\|_{\Gamma_h} + \|\theta^+\|_{\Gamma_-} + \|\theta^-\|_{\Gamma_+} \Big\}, \end{aligned}$$

where $q \geq 1$ and $0 < l \leq r+1$. Thus, combining (4.7) and (4.10), we have that

$$\begin{aligned} &\|\theta\|_\Omega + \|\sqrt{\mu}\theta\|_\Omega + \|[\theta]\|_{\Gamma_h} + \|\theta^+\|_{\Gamma_-} + \|\theta^-\|_{\Gamma_+} \\ &\leq C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{r+1/2}\|u\|_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}_{22}} h_K^{l+1/2-1/q}\|u\|_{W^{l,2q}(K)} \\ &\qquad + \sum_{K\in\mathcal{T}} h_K^{r+1}\|u_0\|_{\partial_- K\cap\Gamma_-^1} + \|\theta^-\|_{\Gamma_-^2} \Big\}, \end{aligned}$$

where $0 < l \le r+1$, $q \ge 1$. We here note that Lemma 3.3 holds with u_h replaced by θ. Thus we see that

$$\begin{aligned}\|\theta\|_\Omega + \|\sqrt{\mu}\theta\|_\Omega + \|[\theta]\|_{\Gamma_h} + \|\theta^+\|_{\Gamma_-} + \|\theta^-\|_{\Gamma_+} \\ \le C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{r+1/2}\|u\|_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}_{22}} h_K^{l+1/2-1/q}\|u\|_{W^{l,2q}(K)} \\ + \sum_{K\in\mathcal{T}} h_K^{r+1}\|u_0\|_{H^{r+1}(\partial_- K\cap\Gamma_-^1)}\Big\},\end{aligned} \tag{4.11}$$

where $0 < l \le r+1$, $q \ge 1$. Now, to estimate the term $\|\gamma\cdot\nabla\theta\|_K^2$, we first note the following local error equation:

$$\int_K \{\gamma\cdot\nabla e + \mu e\} v dx = -\int_{\partial_- K} [e] v^+ \gamma\cdot n ds,$$

or, by taking $v = \gamma\cdot\nabla\theta$,

$$\begin{aligned}\int_K \{\gamma\cdot\nabla\theta + \mu\theta\}\gamma\cdot\nabla\theta dx = \int_K \{\gamma\cdot\nabla\rho + \mu\rho\}\gamma\cdot\nabla\theta dx \\ + \int_{\partial_- K} [\rho](\gamma\cdot\nabla\theta)^+\gamma\cdot n ds - \int_{\partial_- K}[\theta](\gamma\cdot\nabla\theta)^+\gamma\cdot n ds.\end{aligned} \tag{4.12}$$

We now easily see that

$$\int_K \{\gamma\cdot\nabla\theta + \mu\theta\}\gamma\cdot\nabla\theta dx \ge \frac{7}{8}\|\gamma\cdot\nabla\theta\|_K^2 - 8h_K^{-1}\|\sqrt{\mu}\theta\|_K^2. \tag{4.13}$$

We also see by Young's inequality that

$$\begin{aligned}\int_K \{\gamma\cdot\nabla\rho + \mu\rho\}\gamma\cdot\nabla\theta dx + \int_{\partial_- K}[\rho](\gamma\cdot\nabla\theta)^+\gamma\cdot n ds - \int_{\partial_- K}[\theta](\gamma\cdot\nabla\theta)^+\gamma\cdot n ds \\ \le \|\gamma\cdot\nabla\rho\|_K\|\gamma\cdot\nabla\theta\|_K + Ch_K^{-1}\|\rho\|_K\|\gamma\cdot\nabla\theta\|_K \\ + h_K^{-1/2}\|[\rho]\|_{\partial_- K}\|\gamma\cdot\nabla\theta\|_K + h_K^{-1/2}\|[\theta]\|_{\partial_- K}\|\gamma\cdot\nabla\theta\|_K \\ \le \frac{1}{2}\|\gamma\cdot\nabla\theta\|_K^2 + 8Ch_K^{-1}\|[\theta]\|_{\partial_- K}^2 + 8\|\gamma\cdot\nabla\rho\|_K^2 \\ + 8Ch_K^{-2}\|\rho\|_K^2 + 8h_K^{-1}\|[\rho]\|_{\partial_- K}^2.\end{aligned} \tag{4.14}$$

Combining (4.12)–(4.14), we obtain that

$$\|\gamma\cdot\nabla\theta\|_K^2 \le C\Big\{h_K^{-1}\|\sqrt{\mu}\theta\|_K^2 + h_K^{-1}\|[\theta]\|_{\partial_- K}^2 + h_K^{-2}\|\rho\|_K^2 + \|\gamma\cdot\nabla\rho\|_K^2 + h_K^{-1}\|[\rho]\|_{\partial_- K}^2\Big\}$$

or

$$\begin{aligned}\sum_{K\in\mathcal{T}} h_K\|\gamma\cdot\nabla\theta\|_K^2 \le C\Big\{\|\sqrt{\mu}\theta\|_\Omega^2 + \sum_{K\in\mathcal{T}}\|[\theta]\|_{\partial_- K}^2 + \sum_{K\in\mathcal{T}} h_K^{-1}\|\rho\|_K^2 \\ + \sum_{K\in\mathcal{T}} h_K\|\gamma\cdot\nabla\rho\|_K^2 + \sum_{K\in\mathcal{T}}\|[\rho]\|_{\partial_- K}^2\Big\}.\end{aligned} \tag{4.15}$$

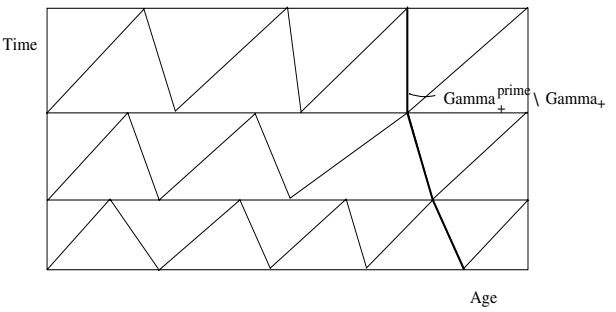


FIG. 4.1. *A sample triangulation.*

We thus have by Lemma 4.1, (3.20), (4.11), and (4.15) that

$$|||\theta|||^2 \le C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+1}\|u\|^2_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}_{22}} h_K^{2l+1-2/q}\|u\|^2_{W^{l,2q}(K)} + \sum_{K\in\mathcal{T}} h_K^{2r+2}\|u_0\|^2_{H^{r+1}(\partial_- K\cap\Gamma^1_-)}\Big\}. \tag{4.16}$$

This completes the proof. □

Let $\Omega' = \{x \in K : K \in \mathcal{T}\setminus\mathcal{T}_{22}\}$ be the regular region in Ω and let $\Gamma'_+ = \{x \in \partial\Omega' : n(x)\cdot\gamma > 0\}$ denote the outflow boundary of $\partial\Omega'$. Since u is not regular only in a and u_h is computed marching in time, we may assume that $\Gamma'_+\setminus\Gamma_+$ is geometrically conforming and $\overline{\partial\Omega'}\cap\Omega$ is a part of the outflow boundary $\partial\Omega'$ as in Figure 4.1, if necessary with a grid modification (calling it again $\mathcal{T}$). By following the proof of Theorem 4.2, we then obtain an optimal error estimate in order and in regularity away from the maximum age.

COROLLARY 4.3. [illegible] W_h [illegible] (2.2) [illegible] $r \ge 0$ [illegible] h_0 $(h_0\lambda < 1/16)$ [illegible] κ [illegible] r [illegible] $h < h_0$, [illegible]

$$\begin{aligned}
&\sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} \|u-u_h\|^2_K + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} \|\sqrt{\mu}(u-u_h)\|^2_K + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K\|\gamma\cdot\nabla(u-u_h)\|^2_K \\
&\quad + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} \|[u-u_h]\|^2_{\partial_- K\setminus\Gamma_-} + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} \|(u-u_h)^+\|^2_{\partial_- K\cap\Gamma_-} \\
&\quad + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} \|(u-u_h)^-\|^2_{\partial_+ K\cap\Gamma_+} \\
&\le C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+1}\|u\|^2_{H^{r+1}(K)} + \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+2}\|u_0\|^2_{H^{r+1}(\partial_- K\cap\Gamma^1_-)}\Big\},
\end{aligned}$$

[illegible] $C(a_\dagger, T)$ [illegible] λ [illegible] h [illegible] $q \ge 1$ $0 \le l \le r+1$.

[illegible]. Take the locally defined test function

$$v(x) = \begin{cases} \widetilde{\psi\theta}(x) & \text{if } x \in K \text{ and } K \in \mathcal{T}\setminus\mathcal{T}_{22}, \\ 0 & \text{otherwise,} \end{cases}$$

and follow the proof of Theorem 4.2 to obtain the desired results. □

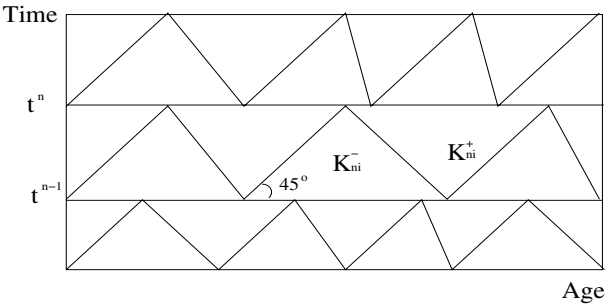


FIG. 5.1. *A sample triangulation.*

4.4. From Theorem 4.2 we easily see that

$$\sum_{K\in\mathcal{T}_{22}} \|u-u_h\|_K^2 \le C\Big\{ \sum_{K\in\mathcal{T}\setminus\mathcal{T}_{22}} h_K^{2r+1+\alpha}\|u\|_{H^{r+1}(K)}^2 + \sum_{K\in\mathcal{T}_{22}} h_K^{2l+1+\alpha-2/q}\|u\|_{W^{l,2q}(K)}^2 + \sum_{K\in\mathcal{T}} h_K^{2r+2}\|u_0\|_{H^{r+1}(\partial_- K\cap\Gamma_-^1)}^2 \Big\}.$$

5. Nonnegativity. In this section, we investigate the nonnegativity of the solution to (2.5)–(2.6). Nonnegativity of the solution is important; it is not only biologically meaningful but also it is essential to analyze long-time behavior of the solution. Throughout this section, we assume the followings on the triangulation $\mathcal{T}$:

(T2) On K_{ni}^-, the interior angle of the left vertex lying on $t = t^{n-1}$ is equal to $\pi/4$, as seen in Figure 5.1.

Under (T2), the age and time advance together along the characteristics and the discrete solution preserves the behavior of the solution to the model problem. Assumption (T2) can be viewed as an analogue of the CFL condition for the finite difference method applied to the hyperbolic equation (1.1)–(1.3). We here note that when the discontinuous Galerkin method is applied to a general linear hyperbolic problem, the same type of triangulation as (T2) is required to obtain a nonnegative solution. Furthermore, on actual computation, for the treatment of the nonlocal boundary condition of (2.3)–(2.5), one may do the following (Step 1)–(Step 3) while $1 \le n \le N$:

(Step 1) On $K \in \mathcal{T}^n\setminus\{K_{n0}^+\}$, compute u_h by (2.3)–(2.4).

(Step 2) On K_{n0}^+, compute

$$P_0 u_h(0,t) = \int_{t-t^{n-1}}^{a_\dagger} \beta(a)u_h(a,t)da$$

and let $Pu_h(a,t)$ be the solution to (2.3) with $u_h^-(0,t) = P_0u_h(0,t)$, and then compute

$$u_h^-(0,t) = \int_0^{t-t^{n-1}} \beta(a)Pu_h(a,t)da + \int_{t-t^{n-1}}^{a_\dagger} \beta(a)u_h(a,t)da.$$

(Step 3) On K_{n0}^+, compute u_h by (2.3) with $u_h^-(0,t)$ given by (Step 2).

We now notice that the triangle K satisfying (T2) is one of the cases of Figure 5.2.

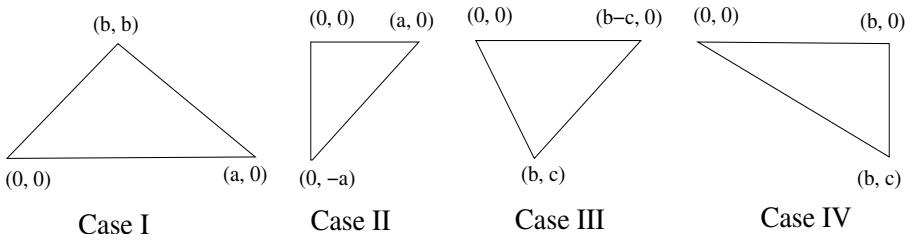


FIG. 5.2. *Sample triangles.*

LEMMA 5.1. [illegible] $r \le 1$ [illegible] u_h^- [illegible] $\partial_- K$ [illegible] K [illegible] 5.2 [illegible] u_h [illegible] K [illegible]. If we consider the mass lumping, then the term involving μ appears only through the diagonal. Without loss of generality we may thus assume that $\mu \equiv 0$. Let φ_i, $i = 1, 2, 3$, be the nodal basis on K and ψ_i, $i = 1, 2$, be the nodal basis on $\partial_- K$. Then $\psi_i = \varphi_{j_i}^+$ for some $j_i = 1, 2, 3$, $j_1 \neq j_2$, and $u_h^- = \alpha_1 \varphi_{j_1}^+ + \alpha_2 \varphi_{j_2}^+$, $\alpha_i \ge 0$, $i = 1, 2$. Let $u_h = U^1 \varphi_1 + U^2 \varphi_2 + U^3 \varphi_3$. Then by taking $v = \varphi_j$, $j = 1, 2, 3$, in (2.3) we see that

$$\int_K \gamma \cdot \nabla u_h \varphi_j dx - \int_{\partial_- K} u_h \varphi_j^+ \gamma \cdot n ds = - \int_{\partial_- K} u_h \varphi_j^+ \gamma \cdot n ds, \quad j = 1, 2, 3 \tag{5.1}$$

or, in matrix form, that $AU = B$, where $U = (U^1, U^2, U^3)^T$ and A and B are one of the following cases:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 2\alpha_1 + \alpha_2 \\ \alpha_1 + 2\alpha_2 \\ 0 \end{bmatrix} \quad \text{for Case I,}$$

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 0 & -1 & 1 \\ 1 & -1 & 3 \end{bmatrix}, \quad B = \begin{bmatrix} 2\alpha_1 + \alpha_2 \\ 0 \\ \alpha_1 + 2\alpha_2 \end{bmatrix} \quad \text{for Case II,}$$

$$A = \begin{bmatrix} 2b & -\sqrt{b^2 + c^2} & b + \sqrt{b^2 + c^2} \\ 0 & -1 & 1 \\ b & -\sqrt{b^2 + c^2} & 2b + \sqrt{b^2 + c^2} \end{bmatrix}, \quad B = \begin{bmatrix} b(2\alpha_1 + \alpha_2) \\ 0 \\ b(\alpha_1 + 2\alpha_2) \end{bmatrix} \quad \text{for Case III,}$$

$$A = \begin{bmatrix} 2b(b-c) - c\sqrt{b^2 + c^2} & (c-b)\sqrt{b^2 + c^2} & b(b-c) + b\sqrt{b^2 + c^2} \\ -c & c - b & b \\ b(b-c) - c\sqrt{b^2 + c^2} & (c-b)\sqrt{b^2 + c^2} & 2b(b-c) + b\sqrt{b^2 + c^2} \end{bmatrix},$$

$$B = \begin{bmatrix} b(b-c)(2\alpha_1 + \alpha_2) \\ 0 \\ b(b-c)(\alpha_1 + 2\alpha_2) \end{bmatrix} \quad \text{for Case IV.}$$

For Cases I–IV, we easily see that $(U^1, U^2, U^3) \ge 0$. □

We now prove that the solution to (2.3)–(2.5) is nonnegative.

TABLE 6.1
Convergence estimates for Example 6.1.

h	E_h^∞	r_h^∞	E_h^2	r_h^2
1/8	0.400262	1.938250	0.0905811	2.060904
1/16	0.102512	1.965158	0.0225618	2.005334
1/32	0.028436	1.933149	0.0057869	1.963026
1/64	0.00708297	1.922158	0.00149810	1.949663
1/128	0.00183668	1.947260	0.00038371	1.965067

TABLE 6.2
Convergence estimates for Example 6.2.

h	E_h^∞	r_h^∞	E_h^2	r_h^2
1/8	0.00714379	0.705638	0.00233010	1.219569
1/16	0.00398427	0.842377	0.000862253	1.434213
1/32	0.00217226	0.875121	0.000305325	1.497770
1/64	0.00114487	0.924016	0.000107350	1.508028
1/128	0.00059276	0.949652	0.000037815	1.505308

THEOREM 5.2. [illegible] $r \le 1$ [illegible] (2.3)–(2.5) u_h [illegible]

[illegible]. We easily see, by (2.3), that u_h on K_{ni}^-, $i \ge 1$ is nonnegative. We then see, by Lemma 5.1, that u_h on K_{ni}^+, $i \ge 1$, is nonnegative. We now have that $u_h^-(0,t)$ is nonnegative. Thus, in turn, we obtain that u_h is nonnegative on K_{n0}^+. □

6. Numerical results. In this section we present some numerical results. In the tests, we have computed twice the approximate solution and evaluated the errors twice, once with step size h and once with step size $h/2$. We estimated the errors in ℓ_2 and ℓ_∞ norms; the errors are denoted by, respectively, E_h^i and $E_{h/2}^i$, $i = 2$ or ∞. Then we computed the rate of convergence using the usual formula

$$r_h^i = \frac{\ln(E_h^i/E_{h/2}^i)}{\ln 2}, \quad i = 2 \text{ or } \infty.$$

In all the tests, we have used the piecewise linear polynomial space as an approximate space.

EXAMPLE 6.1. [illegible] (1.1)–(1.3) [illegible] $a_\dagger = 1$ $\beta(a) = 20a(1-a)$ $\mu(a) = 10exp(-100(1-a))$ $u_0(a)$ [illegible] $u(a,t) = \omega_0 exp(-\int_0^a \mu(\xi)\,d\xi - \alpha^* a)exp(\alpha^* t)$ [illegible] $\alpha^* \approx 2.78576939$ [illegible] $\omega_0 \approx 2.969447356$.

We took $u_0(a) = \omega_0 exp(\alpha^* t)$ as the initial data. We note that the compatibility condition at (0,0) is satisfied, which guarantees the continuity of the solution $u(a,t)$.

The results are presented in Table 6.1. From the table, we see that the effective rate of convergence is order two, which is a half order better than the expected one from the theoretical results (when the data are smooth and bounded).

In the next example we consider the case that the mortality function μ is unbounded.

EXAMPLE 6.2. [illegible] (1.1)–(1.3) [illegible] $a_\dagger = 1$ $\beta(a) = e$ $\mu(a) = \frac{1}{1-a}$, [illegible] $u_0(a) = (1-a)e^{-a}$

The results are shown to Table 6.2.

TABLE 6.3
Convergence estimates for Example 6.3.

h	E_h^∞	r_h^∞	E_h^2	r_h^2
1/8	0.0138519	0.73817	0.00358679	1.37705
1/16	0.0084722	0.70927	0.00132874	1.43263
1/32	0.0052516	0.68997	0.00049648	1.42024
1/64	0.0032805	0.678867	0.00019142	1.37499
1/128	0.0020576	0.672908	0.00007704	1.31307

When the mortality μ is unbounded and the other data are smooth, we obtain the convergent rate of the order of 3/2, which is in agreement with the theoretical results.

In the next example we consider the case that the solution $u \in H^1(\Omega)$ but $u \notin H^2(\Omega)$.

EXAMPLE 6.3. [illegible] (1.1)–(1.3) [illegible] $a_\dagger = 1$ $\beta(a) = 1/\int_0^1 (1-a)^{2/3} e^{-a} da$ $\mu(a) = \frac{2/3}{1-a}$, [illegible] $u_0(a) = (1-a)^{2/3} e^{-a}$

The exact solution is $u(a,t) = e^t(1-a)^{2/3}e^{-a}$. The results are shown to Table 6.3.

In Example 6.1, the mortality function is bounded and the solution is smooth and we obtain the convergence order of h^{r+1}. In Example 6.2, the mortality function is unbounded and the solution is smooth. In that case, we see from the theorem that the convergence rate is dominated by $h^{r+1/2}$ and our numerical example shows in agreement with the theorem. In Example 6.3, the mortality function is unbounded and the solution is not smooth. The results show that the solution of the discontinuous Galerkin method is stable and they indicate an $\mathcal{O}(h^{l+1/2})$ rate of convergence, which is the expected result from the theoretical estimate.

7. Conclusions. We have proposed a discontinuous Galerkin method to approximate the possibly nonsmooth solution to the model of population dynamics with unbounded mortality and nonlocal birth process. Due to the nonlocal birth process, some (reasonable) assumptions on the triangulation were made to obtain a stable approximate solution, which can be viewed as an analogue of the CFL condition for the finite difference method applied to (1.1)–(1.3). Regularity of the solution depends on the growth rate of the mortality function near the maximum age. In [15], it was shown that if $\alpha > 1$, or $\alpha = 1$ and $\lambda \geq 1$, then the solution is regular enough so that a collocation method can be used to approximate the solution to (1.1)–(1.3). In the present paper, we have considered the case that $0 < \alpha < 1$ and $\lambda > 0$. In such a case, the solution of the model is not regular. Discontinuous Galerkin finite element methods were proposed to approximate the solution. It was seen that the convergence rates were affected by the regularity of the solution rather than by the unboundedness itself of the coefficient of the differential equation. We conclude from the modeling point of view that it is important to choose an appropriate form of the mortality function (accordingly, an appropriate numerical method) for numerical simulations to be reliable and efficient, which are an important tool in indicating what possible dynamical properties solutions of equations have.

Finally, concerning the regularity of the solution to the problem, one may obtain by following [15] that, near the maximum age,

$$u \in H^1 \text{ but } u \notin H^2 \text{ if } 0 < \alpha < 1/2,$$
$$u \in H^1 \text{ but } u \notin H^2 \text{ if } \alpha = 1 \text{ and } 1/2 < \lambda < 1,$$

$u \notin H^1$ if $\alpha = 1$ and $0 < \lambda \le 1/2$,
$u \in W^{1,q}$ if $0 < \alpha < 1/q$ with $q \ge 1$.

8. Appendix.

3.4. We take $v = u_h$ in (2.3) and we see that

$$\int_K (\gamma \cdot \nabla u_h + \mu u_h) u_h dx - \int_{\partial_- K} (u_h^+)^2 \gamma \cdot n ds = - \int_{\partial_- K} u_h^- u_h^+ \gamma \cdot n ds. \tag{8.1}$$

By summing over all K in $\mathcal{T}$ and by integration by parts we see that

$$\begin{aligned} &\frac{1}{2}\Big\{ \sum_{K\in\mathcal{T}} \int_{\partial_+ K} (u_h^-)^2 \gamma \cdot n ds + \sum_{K\in\mathcal{T}} \int_{\partial_- K} (u_h^+)^2 \gamma \cdot n ds \Big\} + \sum_{K\in\mathcal{T}} \int_K \mu (u_h)^2 dx \\ &\quad - \sum_{K\in\mathcal{T}} \int_{\partial_- K} (u_h^+)^2 \gamma \cdot n ds = - \sum_{K\in\mathcal{T}} \int_{\partial_- K} u_h^- u_h^+ \gamma \cdot n ds. \end{aligned} \tag{8.2}$$

Since $\partial_- K \cap \Gamma_h$ is a part of $\partial_+ K'$ for some $K' \in \mathcal{T}$ and the sign of the outer normal of $\partial_- K \cap \Gamma_h$ is opposite of the one of $\partial_+ K'$, we have from (8.2) that

$$\begin{aligned} &-\frac{1}{2} \int_{\Gamma_h} [u_h]^2 \gamma \cdot n ds + \frac{1}{2\sqrt{2}} \int_0^{a_\dagger} u_h^2(a, t^n) da + \frac{1}{2\sqrt{2}} \int_0^{t^n} u_h^2(a_\dagger, t) dt \\ &\quad + \frac{1}{2\sqrt{2}} \int_0^{t^n} (u_h^+)^2(0,t) dt + \frac{1}{2\sqrt{2}} \int_0^{a_\dagger} (u_h^+)^2(a,0) da + \sum_{K\in\mathcal{T}} \int_K \mu u_h^2 dx \\ &\quad = \frac{1}{\sqrt{2}} \int_0^{t^n} u_h^-(0,t) u_h^+(0,t) dt + \frac{1}{\sqrt{2}} \int_0^{a_\dagger} \widetilde{u_0}(a) u_h^+(a,0) da. \end{aligned} \tag{8.3}$$

We see by Hölder's inequality that the right-hand side of (8.3) is bounded by

$$\begin{aligned} &2\sqrt{2} \int_0^{t^n} (u_h^-)^2(0,t) dt + \frac{1}{4\sqrt{2}} \int_0^{t^n} (u_h^+)^2(0,t) dt \\ &\quad + 2\sqrt{2} \int_0^{a_\dagger} (u_h^-)^2(a,0) da + \frac{1}{4\sqrt{2}} \int_0^{a_\dagger} (u_h^+)^2(a,0) da. \end{aligned} \tag{8.4}$$

Combining (8.3) and (8.4) we see that

$$\int_0^{a_\dagger} u_h^2(a, t^n) da \le 8 \|u_0\|^2 + 8 a_\dagger \overline{\beta} \int_0^{t^n} \int_0^{a_\dagger} u_h^2(a,t) da dt. \tag{8.5}$$

Let

$$w(t) = \int_0^{a_\dagger} u_h^2(a,t) da.$$

Then we see from (8.5) that

$$w(t^n) \le 8 \|u_0\|^2 + 8 a_\dagger \overline{\beta} \int_0^{t^n} w(t) dt. \tag{8.6}$$

Now let

$$\zeta^n = \begin{cases} \int_{t^{n-1}}^{t^n} w(t) dt, & n = 1, \dots, N, \\ 0, & n = 0. \end{cases} \tag{8.7}$$

Since $w(t)$ is quadratic on $[t^{n-1}, t^n]$, we then see that

$$\begin{aligned}\zeta^n &= \int_{t^{n-1}}^{t^n} w(t)dt \\ &= \frac{1}{3}(t^n - t^{n-1})\Big\{w(t^{n-1}) + 4w\big(\{t^{n-1}+t^n\}/2\big) + w(t^n)\Big\} \\ &\le h\{w(t^{n-1}) + w(t^n)\} \\ &\le h\Big\{16\|u_0\|^2 + 16a_\dagger\overline{\beta}\int_0^{t^{n-1}} w(t)dt + 8a_\dagger\overline{\beta}\int_{t^{n-1}}^{t^n} w(t)dt\Big\}.\end{aligned}$$

Now noting that

$$\sum_{m=1}^{n-1}\zeta^m = \int_0^{t^{n-1}} w(t)dt, \tag{8.8}$$

we have that

$$\zeta^n \le \frac{16h}{1-8a_\dagger\overline{\beta}h}\Big\{\|u_0\|^2 + a_\dagger\overline{\beta}\sum_{m=1}^{n-1}\zeta^m\Big\}. \tag{8.9}$$

Then, by Gronwall's inequality, we obtain that

$$\zeta^n \le C\exp^{CT} h\|u_0\|^2. \tag{8.10}$$

Therefore by (8.8) we see that

$$\int_0^T w(t)dt = \sum_{m=1}^{N}\zeta^m \le C\exp^{CT}\|u_0\|^2 \tag{8.11}$$

or that

$$\|u_h\|_\Omega \le C\exp^{CT}\|u_0\|, \tag{8.12}$$

where C is independent of h. This completes the proof. □

Acknowledgement. The author wishes to thank Ts. Selenge for her help on numerical experiments.

REFERENCES

[1] T. Arbogast and F. A. Milner, *A finite difference for a two sex model of population dynamics*, SIAM J. Numer. Anal., 26 (1989), pp. 1474–1486.
[2] B. P. Ayati, *A variable time step method for an age-dependent population model with nonlinear diffusion*, SIAM J. Numer. Anal., 37 (2000), pp. 1571–1589.
[3] B. P. Ayati and T. F. Dupont, *Galerkin methods in age and space for a population model with nonlinear diffusion*, SIAM J. Numer. Anal., 40 (2002), pp. 1064–1076
[4] S. C. Brenner and L. R. Scott, *The Mathematical Theory of Finite Element Methods*, Springer-Verlag, New York, 1994.
[5] B. Cockburn, G. E. Karniadakis, and C.-W. Shu, eds., *Discontinuous Galerkin Methods*, Springer-Verlag, New York, 2000.
[6] S. E. Esipov and J. A. Shapiro, *Kinetic model of Proteus mirabilis swarm colony development*, J. Math. Biol., 36 (1998), pp. 249–268.

[7] G. Fairweather and J. C. Lopez-Marcos, *A box method for a nonlinear equation of population dynamics*, IMA J. Numer. Anal., 11 (1991), pp. 525–538.

[8] M. Gurtin, and R. C. MacCamy, *Non-linear age-dependent population dynamics*, Arch. Ration. Mech. Anal., 54 (1974), pp. 281–300.

[9] M. Iannelli, Mathematical Theory of Age-Structured Population Dynamics, Applied Mathematics Monographs, Comitato Nazionale per le Scienze Matematiche, Giardini-Pisa, 1995.

[10] M. Iannelli, M.-Y. Kim, and E.-J. Park, *Splitting methods for the numerical approximation of some models of age-structured population dynamics and epidemiology*, Appl. Math. Comput., 87 (1997), pp. 69–93.

[11] M. Iannelli and F. Milner, *On the approximation of the Lotka-Mckendrick equation with finite life-span*, J. Comput. Appl. Math., 136 (2001), pp. 245–254.

[12] M. Iannelli, F. Milner, and A. Pugliese, *Analytical and numerical results for the age-structured S-I-S epidemic model with mixed inter-intracohort transmission*, SIAM J. Math. Anal., 23 (1992), pp. 662–688.

[13] C. Johnson and J. Pitkaranta, *An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation*, Math. Comp., 46 (1986), pp. 1–26.

[14] M.-Y. Kim, *Galerkin methods for a model of population dynamics with nonlinear diffusion*, Numer. Methods Partial Differential Equations, 12 (1996), pp. 59–73.

[15] M.-Y. Kim and Y. Kwon, *A collocation method for the Gurtin-MacCamy equation with finite life-span*, SIAM J. Numer. Anal., 39 (2002), pp. 1914–1937.

[16] M.-Y. Kim and E.-J. Park, *An upwind scheme for a nonlinear model in age-structured population dynamics*, Comput. Math. Appl., 30 (1995), pp. 5–17.

[17] M.-Y. Kim and E.-J. Park, *Mixed approximation of a population diffusion equation*, Comput. Math. Appl., 30 (1995), pp. 23–33.

[18] Y. Kwon and C.-K. Cho, *Second-order accurate difference methods for a one-sex model of population dynamics*, SIAM J. Numer. Anal., 30 (1993), pp. 1385–1399.

[19] E.-J. Park, *Mixed finite element methods for nonlinear second order elliptic problems*, SIAM J. Numer. Anal., 32 (1995), pp. 865–885.

[20] O. Rauprich, M. Matsushita, K. Weijer, F. Siegert, S. E. Esipov, and J. A. Shapiro, *Periodic phenomena in Proteus mirabilis swarm colony development*, J. Bacteriol., 178 (1996), pp. 6525–6538.

[21] G. F. Webb, Theory of Nonlinear Age-Dependent Population Dynamics, Marcel Dekker, New York, 1985.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1394–1423
© 2006 Society for Industrial and Applied Mathematics

DELAUNAY DECOUPLING METHOD FOR PARALLEL GUARANTEED QUALITY PLANAR MESH REFINEMENT*

LEONIDAS LINARDAKIS† AND NIKOS CHRISOCHOIDES†

Abstract. Creating in parallel guaranteed quality large unstructured meshes is a challenging problem. Parallel mesh generation procedures decompose the original mesh generation problem into smaller subproblems that can be solved in parallel. The subproblems can be treated as either completely or partially coupled, or they can be treated as completely decoupled. In this paper we present a parallel guaranteed quality Delaunay method for 2-dimensional domains which is based on the complete decoupling of the subproblems. As a result the method eliminates the communication and the synchronization during the meshing of the subproblems. Moreover, it achieves 100% code reuse of existing, fine-tuned, and well-tested sequential mesh generators. The approach we describe in this paper presents for the first time an effective way to create in parallel guaranteed quality meshes with billions of elements in a few hundreds of seconds, and at the same time demonstrates that these meshes can be generated in an efficient and scalable way. Our performance data indicate superlinear speedups.

Key words. mesh generation, domain decomposition, Delaunay triangulation, parallel algorithms, distributed memory computers

AMS subject classifications. 65M50, 65N50, 65L50

DOI. 10.1137/030602812

1. Parallel Delaunay mesh generation.

Parallel mesh generation methods decompose the original meshing problem into smaller subproblems that can be solved (i.e., meshed) in parallel. The requirements for the parallel and distributed solution of the subproblems are (1) —distributed meshes should retain the same level of quality of elements as the sequentially generated ones, (2) , and (3) , in order to leverage the ever-evolving basic sequential meshing techniques and software.

In [20, 25] parallel mesh generation methods for distributed memory computers or clusters of workstations (CoWs) are classified in terms of the and the the artificial boundary surfaces (interfaces) of the subproblems are meshed. Specifically, existing parallel methods are classified into three categories: (i) methods, which first mesh (either in parallel [34] or sequentially [41]) the interfaces of the subproblems and then mesh in parallel the individual subproblems; (ii) methods, which first solve the meshing problem in each of the subproblems in parallel and then mesh the interfaces [20] so that the global mesh is conforming; (iii) methods, which simultaneously mesh and improve the quality of the interfaces[1] as they mesh the individual subproblems [19, 13, 17, 36].

*Received by the editors December 31, 2003; accepted for publication (in revised form) November 18, 2004; published electronically January 27, 2006. This work was sponsored by National Science Foundation grants 0049086, 0085969, and 0203974. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

http://www.siam.org/journals/sisc/27-4/60281.html

†Department of Computer Science, College of William and Mary, P.O. Box 8795, Williamsburg, VA 23187-8795 (lxlina@wm.edu, nikos@cs.wm.edu).

[1]The improvement of the interfaces is measured in terms of the surface-to-volume ratio.

In this paper we present an a priori method that contributes to the state-of-the-art parallel mesh generation in the following three ways: (1) it guarantees the same level of quality of the mesh with the sequentially generated ones, (2) it eliminates communication and synchronization during the meshing of the subproblems and achieves superlinear speedups with respect to the best (to our knowledge) sequential guaranteed quality mesh generator, Triangle [43], and (3) achieves 100% code reuse, providing the ability to use the best sequential Delaunay mesh generators with no modifications. This is the first method (to the best of our knowledge) that eliminates communication and synchronization, and at the same time is based on a 100% code reuse of sequential codes. It is the only, so far, parallel guaranteed quality method that can achieve superlinear speedups, when compared to the best sequential mesh generation codes, and the first to create over 1 billion (B) elements. The decoupling method we propose can also be used for sequential mesh generation, in order to create larger meshes in less time on one processor.

In [24] Galtier and George present a parallel projective Delaunay meshing (P^2DM) method which guarantees the quality of the elements and eliminates communication and synchronization, but, depending on the geometry, it might suffer from setbacks which affect its efficiency. The setbacks are in the form of discarding completely the triangulation because the separators are not always Delaunay admissible as new points are inserted. The problem of computing Delaunay admissible separators in the context of parallel Delaunay mesh refinement is solved in this paper successfully for 2-dimensional domains.

A 2-dimensional divide-and-conquer Delaunay triangulation (DCDT) algorithm and its parallel implementation are presented in [6]. The DCDT is based on finding a Delaunay path, through a projection to a paraboloid, that separates the initial set into two equal-sized subsets. Although this is an elegant and efficient procedure for Delaunay triangulation, it cannot be used for parallel mesh generation and refinement, which require new point insertion in the mesh without significant extensions like the ones presented in [29], which introduce communication.

SMGP parallel guaranteed quality Delaunay mesh (PGQDM) generation methods appeared in [36] and in [18]. The PGQDM is communication intensive, and despite the fact that it tolerates (masks) up to 90% of communication, its speedup is about 6 for 16 processors [18]. The second SMGP method, the constrained Delaunay mesh (PCDM) generation [13], is based on constrained Delaunay triangulation [10]. It substantially reduces the communication and eliminates synchronization, but the speedup is still 5.75 for 8 processors [13]. The PCDM implementation, as the PGQDM, does not reuse existing sequential Delaunay mesh generators, due to additional care for cavities that are constrained by internal boundaries.

The method we present here requires high-quality domain decompositions that (1) satisfy certain geometric constraints [44] regarding the angles and (2) do not introduce significant constraints that will affect the efficiency of the mesh generator and the quality of the final mesh. In this paper we propose a novel domain decomposition method for 2-dimensional geometries based on the medial axis of the domain. This method satisfies (1) and (2) above, but it has the disadvantage of being difficult to extend to 3 dimensions.

In the rest of the paper, we present in section 3 the medial axis domain decomposition method. In section 4 we proceed to decouple the mesh generation process of the individual subdomains, by defining and preprocessing a zone around the internal boundaries of the subdomains. Contrary to past work [34, 24], we prove that the preprocessing of the zone completely decouples the subdomains. Finally, in section 7 we

present the complete parallel mesh generation procedure, and in section 9 we provide experimental results that demonstrate the efficiency of our method.

2. The domain decomposition problem. Guaranteed quality mesh generation algorithms [11, 12, 40] and software [44, 43] generate elements with good aspect ratio and good angles. These algorithms require that the initial boundary angles be within certain good bounds. For example, Ruppert's algorithm [40] requires boundary angles (the angles formed by the boundary edges) no less than 60° in order to guarantee the termination. Since the separators are treated as external boundary, the domain decomposition should create separators that meet the requirements of the mesh generation algorithm. Therefore, the constructed separator should form angles no less than a given bound Φ_o, which is determined by the sequential mesh generation procedure that will be used to mesh the individual subdomains.

The domain decomposition is used in parallel mesh generation to explore data-parallelism, as in many other areas of scientific computing. The three fundamental issues in data-parallel computations are [illegible]. The parallel mesh generation method we propose eliminates communication and synchronization, using a proper decoupling (see section 4) of the subdomains. However, we achieve the decoupling at the cost of some over-refinement, which is analogous to the size of the separators of the subdomains. Therefore, one of our objectives in the domain decomposition step is to minimize the size of the separators relative to the area of the subdomains. Then the over-refinement we introduce is insignificant (see section 11).

The third important issue that affects parallel program performance is the good balance of the workload among the processors. The equidistribution of processors' workload is achieved by over-decomposing [5] the domain, i.e., $N \gg P$, where N is the number of subdomains and P is the number of processors. The created subdomains are distributed on the processors using an a priori estimation of the workload, based on the area of the subdomains. This static, ab initio distribution approach gives good results for uniform cluster environments (see section 8). However, a dynamic load-balancing approach can be adopted using general purpose runtime systems, like the ones presented in [2, 35], to migrate at runtime subdomains from overloaded processors to ones that have completed their work. The area criterion for estimating the work load appears to be a good measure in the case of our method, for the following reason: the decomposition procedure, as we will see, creates "good" angles and small separators, and the created subdomains tend to have similar shapes after the over-decomposition of the domain; since the geometries are similar, the work of the mesher is approximately proportional to the area of the subdomains. The above intuition is confirmed by the results in section 8.

In summary, the domain decomposition criteria for parallel mesh generation are the following.

C1. Create good angles, i.e., angles no smaller than a given tolerance Φ_o. The value of Φ_o is determined by the sequential, guaranteed quality, mesh generation algorithm.

C2. The subdomains should have approximately equal size (areawise).

C3. The size of the separator should be relatively small, i.e., minimize the ratio $\max\{|\mathcal{H}|/|\Omega_i|\}$, where $|\mathcal{H}|$ is the length of the separator and $|\Omega_i|$ is the area of the subdomains.

The first condition is essential, since it is the one that guarantees the termination of the mesh generation procedure and at the same time prevents the creation of new

features that will lower the quality of the final mesh. Criteria 2 and 3 are not required, but are desired for the efficiency of the parallel computations.

The domain decomposition that we propose here is independent of the decoupling procedure described in section 4, and it can be used in other parallel mesh generation methods, like PCDM, that require good quality domain decompositions.

3. Medial axis domain decomposition method. The medial axis domain decomposition (MADD) method we propose is based on an approximation of the medial axis (MA) of the domain. The MA was introduced by Blum [7] as a way to depict the shape of an object and has been studied extensively during the last two decades [9, 8, 14, 33, 42, 48]. In the context of mesh generation the medial axis has been used in [1, 23, 26, 39, 45]. The existing domain decomposition methods aim mostly to solve the load-balancing problem and to minimize the communication [21, 30, 46]. For the first time in parallel mesh generation the medial axis was proposed as a domain decomposition technique in [15].

One of the contributions of this paper is that, in addition to the load-balancing goal, the MA is used to guarantee domain decompositions with separators which form good angles between them and the external boundary. Like existing methods our decomposition method also aims for separators whose size is small relatively to the areas of the subdomains.

In the rest of the paper we define as a domain Ω the closure of an open connected bounded set in $\mathbf{R}^2$, and the boundary $\partial\Omega$ is defined by a planar straight line graph (PSLG), which forms a set of (nonintersecting) line segments connecting pairs of points. A circle $C \subseteq \Omega$ is said to be maximal in Ω, if there is no other circle $C' \subseteq \Omega$ such that $C \subsetneq C'$. The closure of the locus of the circumcenters of all maximal circles in Ω is called the [illegible] Ω and will be denoted by $\mathrm{MA}(\Omega)$. The intersection of a boundary of Ω and a maximal circle C is not empty. The points $C \cap \partial\Omega$, where a maximal circle C intersects the boundary, are called [illegible] of c, where c is the center of C. Every point $c \in \mathrm{MA}(\Omega) \setminus \partial\Omega$ has at least two contact points.

The domain decomposition method we propose is based on the following simple geometric property.

LEMMA 3.1. [illegible] $c \in MA(\Omega)$ [illegible] cb [illegible] $\partial\Omega$ [illegible] b [illegible] $\pi/2$

[illegible]. We prove the lemma in the general case when Ω has a piecewise C^1 boundary. Suppose that the proposition is not true. Then there is a point $c \in MA(\Omega)$ of the medial axis and a contact point $b \in \partial\Omega$ of c, such that cb forms an angle $\phi < \pi/2$ with the boundary at b (see Figure 3.1). Take c to be the origin of the axes and cb to define the y axis. Without loss of generality, we assume that ϕ is formed by the tangent from the right. Let $(x(s), y(s))$ be locally the normal parametric representation of the curve, with $b = (x(0), y(0)) = (0, y(0))$ and $x(s) \geq 0$. We have $y(0) > 0$. Since $\phi < \pi/2$, we have $y'(0) < 0$. Let $R(s) = x^2(s) + y^2(s)$ be the square of the distance between c and the points of the curve. Because b is a contact point of c, it must be $R(s) \geq R(0) = |cb|^2$. We have $R'(0) = 2y(0)y'(0) < 0$. This means that locally $R(s) < R(0)$, which is a contradiction. □

The medial axis of Ω can be approximated by Voronoi points of a discretization of the domain [9, 8]. We make use of the property of Lemma 3.1 to construct separators that consist of linear segments which connect the Voronoi points to the boundary. The approximation of the $\mathrm{MA}(\Omega)$ is achieved in two steps: (1) discretization of the boundary, and (2) computation of a boundary conforming Delaunay triangulation using the points from step (1). The circumcenters of the Delaunay triangles are the

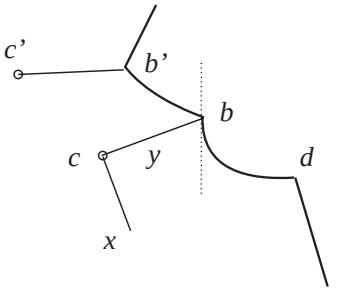


FIG. 3.1. *The angle cbd cannot be less than* 90°*, where c is a point of the medial axis and b its contact point. In some cases the angle is greater that* 90°*, as in the case* $c'b'$.

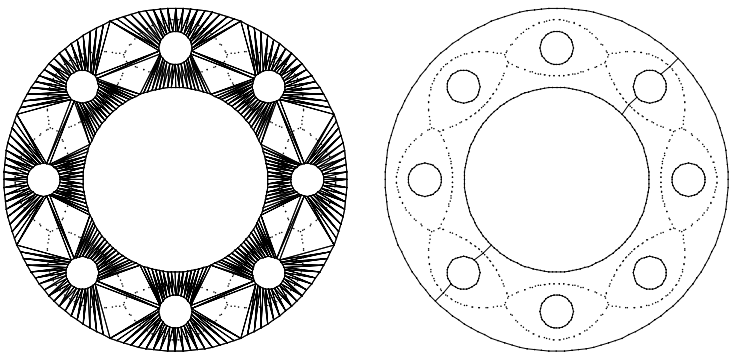


FIG. 3.2. *Left: The Delaunay triangulation of the pipe intersection. The circumcenters of the triangles approximate the medial axis. Right: The circumcenters are the Voronoi points. The separator is formed by selecting a subset of the Voronoi points and connecting them with the boundary.*

Voronoi points of the boundary vertices. The separators will be formed by connecting these circumcenters to the vertices of the Delaunay triangles. Figure 3.2 depicts the boundary conforming mesh of the cross section of a rocket (left), and the media axis approximation and a 2-way separator for the same geometry (right).

The level of the discretization of the boundary determines the quality of the approximation of the medial axis. However, our goal is not to approximate accurately the medial axis, but to obtain good angles from the separator. Therefore our criteria for the discretization of the domain will be determined from the quality of the angles formed between the separators and the boundary $\partial\Omega$. We achieve our goal by defining a new set of triangles (see Figure 3.3).

DEFINITION 3.2. $\mathcal{D}$ D $\partial\Omega$ $t \in \mathcal{D}$ junction triangle,

1. c
2. D
3. t $\geq \Phi_o$

The first criterion is set only for the simplicity of the MADD algorithm (see section 3.1.2) in order to avoid negative weights and guarantee that at least two angles between the segments are good. The second criterion prevents a decomposition that will create very small subdomains. The third criterion guarantees the quality of the angles. Let $a_1a_2a_3$ be the vertices of t. Then the third criterion demands the existence of at

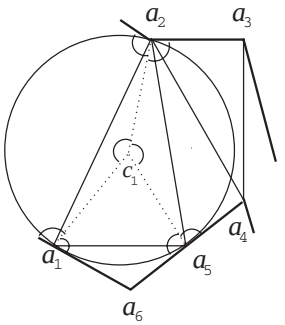


FIG. 3.3. *Triangle $a_1a_2a_5$ is a junction triangle, while $a_1a_5a_6$ has two edges on the boundary and $a_2a_4a_5$ does not include its cicrcumcenter, so they are not junction triangles.*

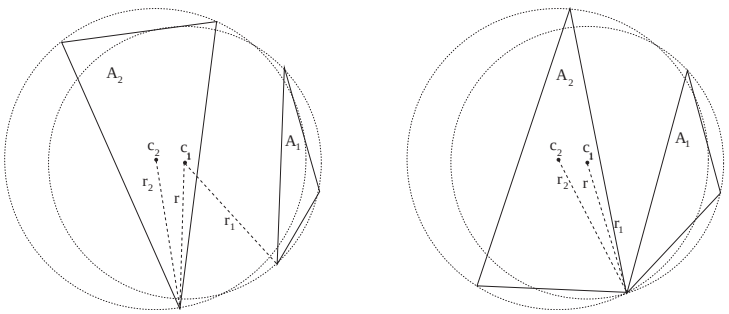


FIG. 3.4. *On the left, the two Delaunay triangles, A_1, A_2, do not have common vertices. On the right, the triangles share one common vertex; the case of two common vertices reduces to these.*

least one pair of segments a_ica_j, where c is the circumcenter of $a_1a_2a_3$, so that all the angles formed with these segments are greater than or equal to Φ_o. Such pairs a_ica_j are called , and they will be the candidates to form a complete separator. A complete separator decomposes a domain into two connected subdomains.

Let m be the number of holes of Ω. The level of refinement D we require for $\partial\Omega$ has to satisfy the following two conditions.

(i) In the Delaunay triangulation $\mathcal{D}$ of D there are at least $m+1$ junction triangles.

(ii) Every segment on the boundary D has an empty diametral circle.

The first condition in Definition 3.2 requires the existence of at least $m+1$ junction triangles. This ensures, as we will see in section 3.2, that there is at least one complete separator formed by partial separators. The second condition guarantees that all the segments of D will appear as edges in $\mathcal{D}$. It also guarantees that all the circumcenters of the triangles of $\mathcal{D}$ are contained in Ω [44]. This in turn guarantees the existence of at least one triangle that includes its circumcenter (see Lemma 3.4).

LEMMA 3.3. A_1, A_2 c_1 A_1 A_2 c_2 A_2 r_1 r_2 A_1 A_2 $r_1 < r_2$

. Let r be the smaller distance of c_1 from the vertices of A_2; see Figure 3.4. Then $r \geq r_1$. So we have $r_2 > r$, and consequently $r_2 > r_1$. □

LEMMA 3.4. D $\mathcal{D}$ D

. We know that, when the boundary segments have empty diametral circles, all the circumcenters of the triangles of $\mathcal{D}$ are in $\mathcal{D}$ [44]. We assume that the points are in general position; i.e., there are no cocircular points. We will prove the lemma by contradiction.

Suppose that the lemma is not true. Then for every triangle A_i there is another triangle $A_{i+1} \neq A_i$, such that the circumcenter c_i of A_i is included in A_{i+1}. Let r_i be the radius of the circumcircle of A_i. Since we assumed that no triangle includes its circumcenter, the sequence $\langle A_i \rangle$ is infinite. On the other hand, the set $\{t_i\}$ of all triangles in D is finite, so the sequence $\langle A_i \rangle$ includes an element t_k twice. Then $A_l = A_m = t_k$ for some $l < k$. From the previous lemma we have $r_l < r_{l+1} < \cdots < r_m$, which contradicts the fact that r_l and r_m are the radii of the same circle, and thus equal. So the lemma must hold. □

The discretization of the boundary is determined by the number of junction triangles we want to create, which in turn is determined by the geometry of the domain Ω. As we increase the refinement, the Voronoi points approximate the points of the medial axis and the formed angles with the boundary tend to be close to $\pi/2$. If we construct more junction triangles, and thus more partial separators, we have more choices to form a better separator, in terms of the quality of the angles, the size of the separator, and the balance of the areas of the subdomains. In our experiments a rather small refinement (less than 700 additional points for the 2-dimensional geometries we tried so far) gives satisfying results. Again this of course depends on the geometry, and a way to predefine the refinement level of the boundary of the domain is a subject of further research.

3.1. The MADD algorithm. The MADD algorithm uses as a starting point the approximation of the medial axis by the Delaunay triangulation $\mathcal{D}$, as described in the previous section. The complete separator is formed by partial separators, i.e., segments inserted in junction triangles of $\mathcal{D}$; these segments connect the circumcenter of the triangles to two of their vertices. Figure 3.2 (right) depicts a complete separator for a 2-way decomposition of the pipe.

The partial separators connect two points of the boundary, since $\mathcal{D}$ is a boundary conforming triangulation. The properties of junction triangles permit the construction of good angles between the partial separators and the external boundary of the geometry. The MADD algorithm will select to insert a set of partial separators that will guarantee the decomposition of the domain into two subdomains. The selection of the partial separators is based on the minimization of the ratio of the size of the separators to the areas of the subdomains.

The MADD algorithm maps the Delaunay triangulation $\mathcal{D}$ into a graph $G_{\mathcal{D}}$ which encapsulates the required information about the candidate partial separators. This information includes (1) the topology of $\mathcal{D}$, which is used to guarantee that the inserted partial separators form a complete separator and (2) the length of the partial separators and the area of the subdomains that will be created, which is used to optimize the ratio of the length of separators to the subdomains area. After $G_{\mathcal{D}}$ is constructed, the graph is contracted, so that only the junction triangles of $\mathcal{D}$ are represented. Then the contracted graph is partitioned; the graph partitioning can be obtained by using any of the well-known algorithms [32, 4, 27, 28, 30, 47] that decompose a connected graph into two connected subgraphs and minimize the ratio of the cut cost to the weights of the subgraphs. Finally the graph partition is translated into insertions of partial separators, which result into a 2-way decomposition. In summary the key steps of the algorithm are the following:

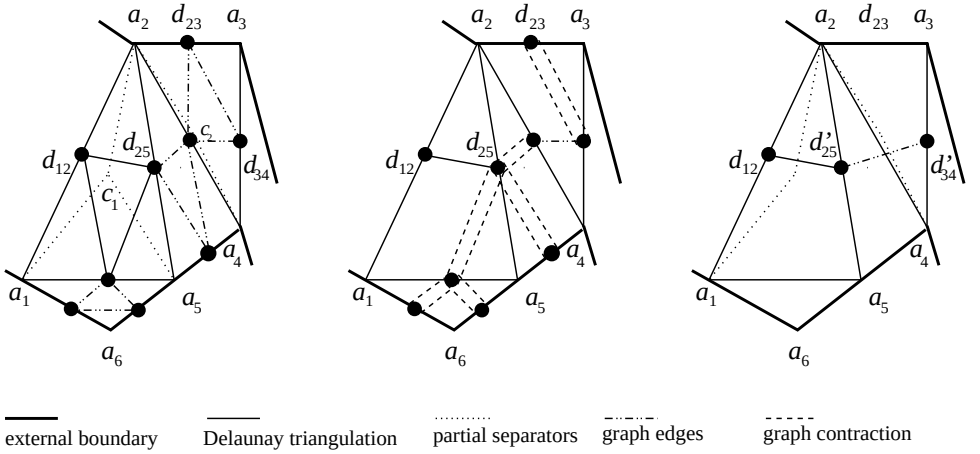


FIG. 3.5. *An example of creating the MADD graph. Left: a part of the Delaunay triangulation and the creation of the corresponding initial graph $G_{\mathcal{D}}$. Middle: the procedure of contracting the graph by combining the nodes of $G_{\mathcal{D}}$. The nodes connected by dashed lines are combined. Right: the final graph $G'_{\mathcal{D}}$ that corresponds to this part.*

1. Create a graph $G_{\mathcal{D}}$ from the Delaunay triangulation $\mathcal{D}$.
2. Contract $G_{\mathcal{D}}$ into the graph $G'_{\mathcal{D}}$, so that only the partial separators in the junction triangles are represented as edges of $G'_{\mathcal{D}}$.
3. Partition the graph $G'_{\mathcal{D}}$, optimizing the cut cost to subgraph weight ratio.
4. Translate the cuts of the previous partition into partial separators.

3.1.1. Construction of the graph $G_{\mathcal{D}}$. In this step the Delaunay triangulation $\mathcal{D}$ is represented as a weighted graph, the dual graph of the edges of the triangles. Two nodes of the graph are adjacent if their corresponding edges belong to the same triangle. The length of the radius of the circumcircle of this triangle will be the weight of the graph edge. The weights of the nodes are set to zero in this step, and they will be computed in the graph construction step (see section 3.1.2).

Figure 3.5 (left) depicts the step for constructing the graph $G_{\mathcal{D}}$. One graph node is created for each edge of the triangulation, and two nodes are connected if they belong to the same triangle. Let d_{ij} be the node corresponding to the edge a_ia_j. The weight of the edge connecting d_{ij}, d_{jk} is the length $|c_la_j|$, where c_l is the circumcenter of the triangle. For example, the edge that connects d_{12} and d_{25} has weight the length $|c_1a_2|$. The above procedure is described by the following algorithm.

ALGORITHM 1.

1. **for** all the edges a_ia_j in $\mathcal{D}$ **do**
2. Add node d_{ij} to the graph $G_{\mathcal{D}}$, with zero weight
3. **endfor**
4. **for** all triangles $t \in \mathcal{D}$ **do**
5. **for** the three pairs (a_ia_j, a_ja_k) of edges of t **do**
6. Create a graph edge between the corresponding nodes d_{ij}, d_{jk}, with weight the length of the circumradius of t
7. **endfor**
8. **endfor**

3.1.2. Graph contraction. In this step the graph $G_{\mathcal{D}}$ produced from the previous step is contracted into a new graph $G'_{\mathcal{D}}$, so that only the edges of junction triangles are represented as nodes in $G'_{\mathcal{D}}$. The nodes of $G_{\mathcal{D}}$ that correspond to edges of nonjunction triangles of $\mathcal{D}$ are contracted in $G'_{\mathcal{D}}$.

In order to contract the graph $G_{\mathcal{D}}$, first we iterate through all the triangles that are not junction triangles. The nodes of $G_{\mathcal{D}}$ that correspond to the three edges of a nonjunction triangle are combined into a single node and the new node replaces the initial nodes in the external graph edges, while the edges between the three initial nodes are deleted. The weight of the new node is the sum of the weights of the initial ones, plus the area of the triangle.

The remaining nodes correspond to the edges of junction triangles. Junction triangles contain candidate partial separators, whose number may vary from one to three. From the three possible partial separators we keep the one that forms the greater minimum angle. Since in junction triangles there is at least one partial separator that forms angles no less than Φ_o, the selected partial separator forms angles $\geq \Phi_o$. We establish this partial separator by combining the two of the three nodes that correspond to edges of the triangle. Let $a_1a_2a_3$ be a junction triangle and c its circumcenter. Let d_{ij} be the corresponding node to the edge a_ia_j, then the weight of the node d_{ij} is updated by adding the weight of the area included by the triangle ca_ia_j. Let a_jca_k be the partial separator that forms the greater minimum angle. Then the nodes d_{ji} and d_{ki} are contracted into a single node, where a_i is the remaining vertex. The procedure is illustrated by the following example.

Figure 3.5 (center) illustrates the procedure of contracting the graph. The bold lines indicate the external boundary. The triangles are part of the boundary conforming Delaunay triangulation of the domain. As above, we denote by d_{ij} the graph node that corresponds to the segment a_ia_j. We demonstrate four different cases.

I. The triangle $a_1a_5a_6$ has two edges on the boundary, so it is not a junction triangle, and the three corresponding nodes are combined in one. The edges connecting the new node d'_{15} are the external ones, i.e., the edges that connect d_{15} to d_{12} and d_{15} to d_{25}. The weight of d'_{15} is equal to the area of the triangle $a_1a_5a_6$.

II. The triangle $a_2a_4a_5$ does not include its circumcenter and so it is not a junction triangle. We follow the same procedure as in Case I. The nodes d_{25}, d_{24}, d_{45} are contracted into a new node d'_{25}. The new node has weight equal to the area of the triangle $a_2a_4a_5$ and is connected to the nodes $d_{12}, d'_{15}, d_{23}, d_{34}$.

III. The triangle $a_1a_2a_5$ is a junction triangle. The areas of the triangles formed by its circumcenter c_1 and its corners are added to the weight of the corresponding nodes. For example, the area $|a_2c_1a_1|$ is added to the node d_{12}, similarly the areas $|a_2a_5c_1|$ and $|a_1c_1a_5|$ are added to the nodes d'_{25} and d'_{15}, respectively. Suppose that the partial separator $a_1c_1a_2$ is the one that forms the greater minimum angle. Then the nodes d'_{15} and d'_{25} are contracted into a new node d'_{25} with its weight to be equal to the sum weights of the two previous nodes. The graph edge connecting the nodes d'_{15} and d'_{25} is deleted, while the two other graph edges are contracted into one edge connecting d'_{25} to d_{12}; the new edged weight is equal to the sum of the two previous edge weights, which is equal to the length of the partial separator $a_1c_1a_2$.

IV. The triangle $a_2a_3a_4$ is also a junction triangle. As for the previous triangle, first we add the areas of the triangles formed by the circumcenter c_2 and the vertices. The areas $|a_2c_2a_4|$, $|a_2c_2a_3|$, and $|a_3c_2a_4|$ are added to the weight of the nodes d'_{25}, d'_{23}, and d'_{34}, respectively. However, suppose in this case that the angle θ, formed by the segment c_2a_3 and the external boundary segment a_3b, is less than Φ_o.

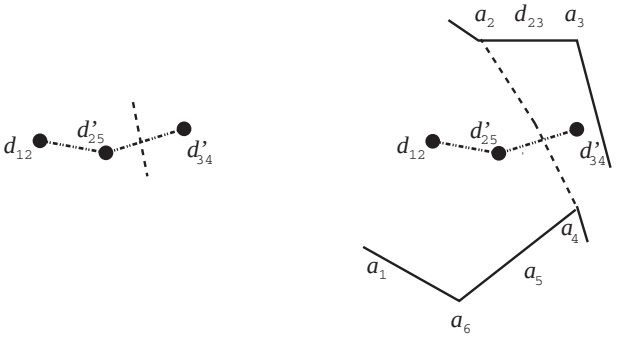


FIG. 3.6. *A partition of the graph and the corresponding separator, on the right, depicted with dashed lines.*

Then the two partial separators that include this segment are rejected and we keep the separator $a_2c_2a_4$, which is the one that forms the greater minimum angle. The nodes d_{23} and d_{34} are combined into the node d'_{34}. The new node is connected to d'_{25} by an edge with weight equal to the sum of the two previous edge weights, which is the length of the partial separator $a_2c_2a_4$. Figure 3.5 (right) shows the final graph.

The above procedure is described by the following algorithm.

ALGORITHM 2.

1. **for** all nonjunction triangles $t \in \mathcal{D}$ **do**
2. Combine the three nodes that correspond to the edges of t, generating a new node d'
3. Add the area of t to the weight of d'
4. **endfor**
5. **for** all junction triangles $t \in \mathcal{D}$ **do**
6. Let c be circumcenter of t
7. **for** all edges a_ia_j of t **do**
8. Add the area of the triangle a_ica_j to the weight of the corresponding node d_{ij}
9. **endfor**
10. Find the partial separator a_ica_j in t forming a max min angle
11. Combine the nodes d_{ik} and d_{jk}, where a_k is the remaining vertex
12. **endfor**

3.1.3. The construction of the separator. After contracting the graph, the constructed graph $G'_{\mathcal{D}}$ is partitioned. The number of the edges of the graph is less than or equal to the number of junction triangles; thus the size of the graph partitioning problem is significantly smaller than the elementwise dual graph of the boundary conforming Delaunay triangulation $\mathcal{D}$. Graph partitioning can be very expensive and has been an active area for several years [32, 4, 27, 28, 30, 47]. Any of the algorithms that give a partition of the graph into two connected subgraphs, with good cut cost to subgraph weight ratio, can be used as the graph partitioner for $G'_{\mathcal{D}}$. For algorithms that give nonconnected subgraphs, a check step must take place (see section 10).

After partitioning $G'_{\mathcal{D}}$, the final step of the MADD is to construct the separator of the geometry. From the previous step we have a partition of the graph $G'_{\mathcal{D}}$ into two connected subgraphs. This partition will give a corresponding separator for the geometry. Each edge of the graph corresponds to a partial separator of the form a_ica_j,

where c is a circumcenter of a junction triangle and a_i, a_j are two of its vertices. For every graph edge that is cut by the partition we will insert the related partial separator in the geometry. In our example above (see Figure 3.6) the partial separator $a_2c_2a_4$ is created in the case that the graph partitioner chooses to cut the edge e_2.

The construction of the separator is described in the following algorithm.

ALGORITHM 3.

1. **for** all triangles $t \in \mathcal{D}$ **do**
2. **if** one of the edges a_ia_j of t belongs to a different subgraph from the other two edges **then**
3. Insert the partial separator a_ica_j, where c is the circumcenter of t
4. **endif**
5. **endfor**

The algorithm traverses the list of all triangles and identifies those triangles whose edges correspond to disconnected nodes after the graph partition. In these triangles the partial separators are inserted, separating the edges that do not belong to the same subgraph. In Figure 3.6 the partial separator $a_2c_2a_4$ separates the edge a_2a_4 from the edges a_2a_3 and a_3a_4. The set of all these inserted partial separators establishes a (complete) separator for the domain, as we will see in section 3.2.

The ratio of the cost of the cut to the weight of the subgraphs is translated to the ratio of the total length of the separator to the area of the subdomains. Provided that the graph partitioner gives a good cut cost to subgraph weight ratio, the ratio of length of the separator to the area of the subdomains is also good. This way we obtain separators of relatively small size, and the areas of the subdomains are balanced. Moreover, since all the partial separators, by the construction of $G'_{\mathcal{D}}$, form good angles, the constructed separator forms good angles. In summary, the constructed separator meets the decomposition criteria C1–C3 in section 2.

3.2. Proof of correctness. In this subsection we prove that the MADD algorithm decomposes the domain Ω into two connected subdomains. We recall that the domain Ω is the closure of an open connected bounded set and the boundary $\partial\Omega$ is a PSLG that formed a set of linear segments which do not intersect. A separator $\mathcal{H} \subseteq \Omega$ is a finite set of simple paths (a continuous 1-1 map $h : [0,1] \to \Omega$) that do not intersect and define a decomposition A_1, A_2 of Ω in the following way: A_1 and A_2 are connected sets, with $A_1 \cup A_2 = \Omega$, and $U \cap \mathcal{H} \neq \emptyset$ for every path $U \subset \Omega$ which connects a point of A_1 to a point of A_2.

LEMMA 3.5. [illegible] m [illegible] Ω [illegible] n [illegible] $n > m$ [illegible] Ω [illegible]. We will prove the lemma by induction on m. If $m = 0$, then $n \geq 1$, and there is at least one partial separator. In this case, every partial separator is a separator for Ω, since every simple path $f : [a,b] \to \Omega$, with $f(a), f(b) \in \partial\Omega$ and $f(a,b) \subset \Omega^\circ$, is a separator for Ω.

Suppose the lemma is true for $m = q$, we will prove it is true for $m = q+1$. We have that $n > q+1$. If for a partial separator acb, where $a, b \in \partial\Omega$, we have that both a and b do not belong to the boundary of a hole; then acb forms a separator, as in the case $m = 0$. In the case that one of the points a, b belongs to the boundary of a hole O, then by inserting the partial separator acb we eliminate O. The new domain has q holes and $n - 1 > q$ junction triangles. Thus, by the inductive hypothesis, it can be decomposed by partial separators. Therefore there is a separator formed by partial separators, when the conditions of the lemma are satisfied. ∎

THEOREM 3.6. [illegible] Let $e_i, i = 1, \ldots, n$, be the edges of the contracted graph $G'_{\mathcal{D}}$ created by MADD. Each of these edges corresponds to a partial separator $h_i, i = 1, \ldots, n$. We will show that every decomposition of the graph $G'_{\mathcal{D}}$ corresponds to a decomposition of Ω formed by partial separators, and vice versa.

Let $E = \{e_i, i \in I\}$ be the set of edges that the graph partitioner cuts, creating two subgraphs G_1, G_2. Let $\mathcal{H} = \{h_i, i \in I\}$ be the set of partial separators that correspond to these edges. Finally, let $A_1, A_2 \subset \Omega$ be the two corresponding areas to the subgraphs G_1, G_2. Obviously $A_1 \cup A_2 = \Omega$. From the construction of the graph we have that the connected subgraphs correspond to path-connected areas of Ω. Assuming that the graph partitioner decomposes $G'_{\mathcal{D}}$ into two connected subgraphs, then G_1, G_2 are connected, and so A_1, A_2 are also connected. Every path $U \subset \Omega$ from a point of A_1 to a point of A_2 corresponds to a path U' in $G'_{\mathcal{D}}$ and forms a node of G_1 to a node of G_2. Since the edges E separate G_1 from G_2, we have $U' \cap E \neq \emptyset$. Let $e_j \in U' \cap E$. Then we have $U \cap h_j \neq \emptyset$, and the path U intersects $\mathcal{H}$. Thus $\mathcal{H}$ is a separator for Ω. Working backwards we see that a separator for Ω corresponds to a partition of the graph. The existence of such a separator is proved in Lemma 3.5, and this completes the proof. □

The algorithm always terminates, but the existence of a separator, according to the above theorem, depends on the number of junction triangles that we construct. If this number is greater than the genus of the domain, then a separator exists; there are cases though where fewer junction triangles are sufficient, depending on their position (for example, if none of their edges are on a hole). On the other hand, in order to achieve a better quality of the formed angles, as well as a better decomposition in terms of load-balancing, we would like to have as many junction triangles as possible, since this will increase the number of choices for partial separators. The number of junction triangles depends on the boundary refinement we will use, and two major parameters come into play: (a) the number of subdomains we wish to create, which largely depends on the size of the mesh we want to create; (b) the geometry of the domain. It is hard to define what a difficult geometry would be to decompose, since geometries that look complicated may form areas where "natural" cuts can be made, while geometries that look simple may lack these natural cuts. Small features of the geometry, which usually present a problem in the mesh generation, are not a problem for the decomposition, because they can be discarded (for example, they will form no junction triangles). In our experiments a small refinement (700 additional points) of the boundary produces enough junction triangles to form good separators (with an angle bound of $\Phi_o = 60°$); see Figure 10.1. Still an open problem, though, is a theory that would predefine the level of refinement (depending on the geometry) that would guarantee the creation of enough junction triangles and provide a good separator.

3.3. *N*-way decomposition. So far we have described the MADD procedure for a 2-way decomposition. In the following section we will describe a decoupling procedure which is applied on multiple subdomains and decouples the mesh generation procedure for all the given subdomains. In order to create more than two subdomains we can apply the MADD in a divide and conquer way (see Figure 3.7). When a 2-way separator is created, it is discretized and then every subdomain is decomposed independently. The resulting decomposition shows good adaptivity to the geometry. This approach requires one to recalculate the Delaunay triangulation of the subdomains. We can do that by just inserting the segments of the separator in the existing triangu-

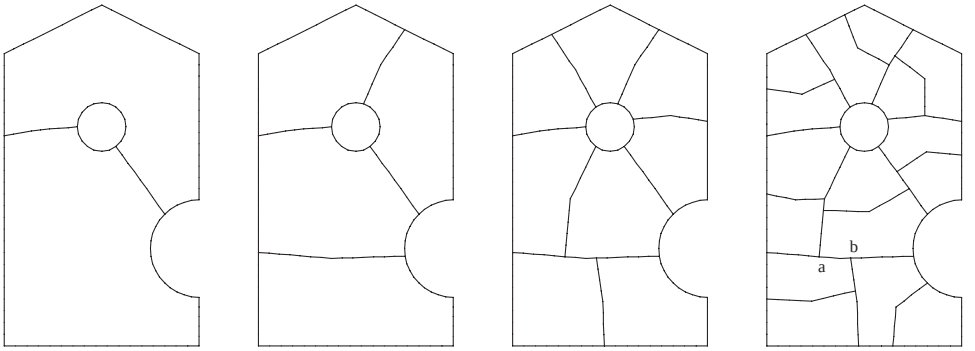


FIG. 3.7. *N-way partitions, where* $N = 2, 4, 8, 16$*, by the MADD divide and conquer method. METIS* [31] *was used as the graph partitioner and Triangle* [43] *produced the Delaunay triangulation.*

lation. These segments should be refined, and possibly the edges of the boundary also, so that the empty diametral circle property of the boundary, including the separators, is maintained.

Since every subdomain is decomposed independently, the discretization of the separators, which form the internal boundary, should be permanent. In practice, the size of the segments created by the discretization of the domain is much larger than the ones created by the mesh generation procedure. Here we should take into account that the level of decomposition is proportional to the size of the mesh we want to create. Thus, in the general case, the discretization does not create actual artificial constraints to the mesh. Figure 3.7 depicts that no new artifacts are introduced, given that segments like *ab* will be refined further.

In our method we refine even further the internal boundaries in order to decouple the subdomains, and our results show that the size and the quality of the mesh are not affected. For a more detailed experimental analysis see section 11.

An advantage of the divide and conquer approach is that it is easy to implement in parallel. In our implementation we have followed a parallel MADD divide and conquer strategy to create multiple subdomains. The method is described in detail in section 7.

4. The decoupling zone. The separators and the subdomains created by the MADD procedure have good quality in terms of the shape and size. Our goal though is to be able to create Delaunay meshes independently for each subdomain, and the previous procedure cannot guarantee this. In order to create the mesh independently in each subdomain we have to ensure that the final mesh will be Delaunay conforming. In the context of parallel Delaunay triangulation, a Delaunay conforming separator can be found using a projective method, presented in [6]. A study of conditions for a priori conformity for constrained Delaunay triangulations is presented in [37]. A method for independent mesh generation in each subdomain using a projective separator is presented in [24], but it does not always guarantee a priori Delaunay conformity.

In order to ensure the Delaunay conformity in the mesh generation context we will refine the separators using conditions derived from the mesh refining algorithm. A special "zone" around the segments of the separators (see Figure 4.1) will guarantee that the mesh generation procedure can be applied independently on each subdomain, giving a Delaunay conforming mesh for the whole domain, formed by the union of all the submeshes.

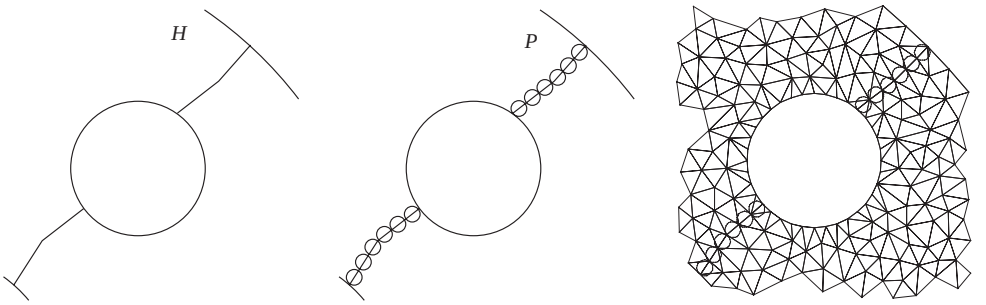


FIG. 4.1. *A fraction of the pipe intersection.* Left: *Part of the separators $\mathcal{H}$ inserted by MADD.* Middle: *Refining $\mathcal{H}$ gives a decoupling path $\mathcal{P}$; the decoupling zone $\mathcal{Z}_\mathcal{P}$ is depicted.* Right: *Ruppert's algorithm was applied on the subdomains with an element area restriction; $\mathcal{Z}_\mathcal{P}$ is empty and $\mathcal{P}$ is invariant. The final mesh is Delaunay conforming.*

Let $\mathcal{M}$ be a Delaunay mesh generation procedure. Let $D = \partial\Omega$ be a PSLG, where Ω is a domain as described in the previous section. Let $\mathcal{P}$ be a set of piecewise linear separators that decompose the domain Ω in n subdomains Ω_i and let $D_i = \partial\Omega_i$ be the boundaries of the subdomains.

DEFINITION 4.1.

$\mathcal{P}$ decoupling zone of $\mathcal{P}$ $\mathcal{Z}_\mathcal{P}$

DEFINITION 4.2. $\mathcal{P}$ decoupling path $\mathcal{M}$ $\mathcal{M}$ Ω_i $i = 1, \ldots, n$ $\mathcal{Z}_\mathcal{P}$

PROPOSITION 4.3. M_i $\mathcal{M}$ Ω_i $\mathcal{P}$ $\mathcal{M}$ $\cup M_i$

. Let M be the Delaunay triangulation of the vertices $V_M = \cup V_{M_i}$ of $\cup M_i$. We will prove that $M = \cup M_i$, by showing that the set of edges S of M are identical to the set of edges $\cup S_i$ of $\cup M_i$, thus the two triangulations are the same and $\cup M_i$ is a conforming Delaunay triangulation.

First we observe that $\mathcal{P}$ is a subset of both S and $\cup S_i$, because its decoupling zone is empty. For any edge $ab \in S$ there are two cases. (i) Both end points a, b belong to the same subdomain M_j, $a, b \in V_{M_j}$. (ii) $a \in M_i$ and $b \in M_j \setminus M_i$.

(i). Suppose $a, b \in V_{M_j}$. From the local Delaunay property, there is an empty circumcircle C of ab which does not include any points in V_M. Because $V_{M_j} \subseteq V_M$, C must be empty in the set V_{M_j}. Thus $ab \in S_j$ and $ab \in \cup S_i$.

(ii). We will show that this case cannot occur, there is no edge $ab \in S$ such that $a \in M_i$ and $b \in M_j \setminus M_i$. Suppose we have such an edge ab. Then $ab \subset D$ and since the subdomains M_i and M_j are separated by $\mathcal{P}$, a and b are separated by $\mathcal{P}$. So $ab \cap \mathcal{P} \neq \emptyset$. On the other hand, we have $\mathcal{P} \subseteq S$, which means that two edges of the triangulation M intersect. This contradicts the definition of a triangulation [23].

Since case (ii) cannot occur, we conclude from case (i) that $S \subseteq \cup S_i$. The two triangulations M and $\cup M_i$ must have the same number of edges, so we have $S = \cup S_i$, and thus $M = \cup M_i$. This proves the proposition. ∎

PROPOSITION 4.4. $\mathcal{M}$ $\mathcal{P}$ $\mathcal{M}$

. Suppose that during the procedure $\mathcal{M}$ an edge $s \in \mathcal{P}$ is destroyed. That means that the diametral circle C_s of s includes some point. Since $\mathcal{M}$ does not

remove points, C_s will not be empty after the termination of $\mathcal{M}$. This contradicts the definition of the decoupling path. □

Proposition 4.3 proves that, provided that we have constructed a decoupling path, the subdomains can be meshed independently and the final mesh will be Delaunay conforming. Our next step will be to construct a decoupling path from the separators created by MADD.

The decoupling path is defined with respect to a mesh generation procedure and, in many cases [11, 40], the stopping conditions of the mesh generation algorithm allow us to compute the length of the edges of the separators, so that these edges will form a decoupling path. Then we only have to refine the segments of the separators, acquiring this predefined length.

5. Ruppert's algorithm. For the mesh procedure we will consider Ruppert's algorithm [40]. This is a mesh refinement algorithm for 2 dimensions that guarantees the quality of the elements. It creates an initial triangulation and follows an incremental approach to refine the mesh. Triangles which have circumradius to shortest edge ratio greater than $\sqrt{2}$ are split, by inserting points in their circumcenters and constructing a new Delaunay triangulation. If a point to be inserted encroaches the diametral circle of a boundary edge, then, instead of inserting this point, the boundary edge is split in half. The algorithm maintains the Delaunay property after the insertion of each point. In order to guarantee the termination of this procedure the boundary angles should be at least 60°.

Let D be a PSLG, as defined above. An entity is either a vertex or a segment of the boundary; two entities are incident when they share a common point. The [illegible] of D is defined as the minimum distance between two nonincident entities [44];[2] it will be denoted by $\mathrm{lfs}_{\min}(D)$.

The following proposition holds [44].

PROPOSITION 5.1. [illegible] D [illegible] 60°. [illegible] D [illegible] $\sqrt{2}$ [illegible] $\mathrm{lfs}_{\min}(D)$

The only requirement for Ruppert's algorithm is that the boundary angles must be at least 60°. Provided that our initial boundary D satisfies this criterion, we can apply MADD to decompose Ω using an angle bound $\Phi_o = 60°$. So, both the constructed separators and the external boundaries form angles $\geq 60°$. Consequently the created subdomains are acceptable for this mesh generation algorithm.

6. The construction of the decoupling path. Let $D = \partial\Omega$ be the boundary of the domain Ω, and $\mathcal{H}$ the set of separators in Ω created by the MADD method using an angle bound of $\Phi_o = 60°$. Let $D_i = \partial\Omega_i$ be the boundaries of the created subdomains and $D_{\mathcal{H}} = D \cup \mathcal{H}$.

In order to construct a decoupling path $\mathcal{P}$ from the separators $\mathcal{H}$ we will refine $\mathcal{H}$ by inserting points along its edges, obtaining a desirable segment length. The calculation of this length is based on a parameter k. Let $L = \min\{|s|/\ s \text{ is a segment of } \mathcal{H}\}$.

[2]For a PSLG domain the minimum local feature size will be the minimum distance between two vertices, or a vertex and a segment [44]. In our implementation we use a simple brute force approach, checking all the distances. This approach is satisfactory for relatively simple domains with few initial vertices. For more complicated geometries better algorithms should be used, similar to the ones for finding the two closest pair of a set of points (see [38]), which have $\boldsymbol{O}(n \log n)$ complexity.

Let k be a real constant parameter, such that

$$0 < k \leq \min(\mathrm{lfs}_{\min}(D_{\mathcal{H}}), L/4). \tag{6.1}$$

The parameter k will be calculated from the conditions of the algorithm, so that it can be guaranteed that no edge will be created with length less than k.

The following lemma describes the refining procedure of $\mathcal{H}$.

LEMMA 6.1. [illegible] $\mathcal{H}$ [illegible] $\nu \in \mathbf{N}$ [illegible] $\nu - 1$ [illegible] b_i [illegible] $\frac{2}{\sqrt{3}}k \leq |b_i b_{i+1}| < 2k$ [illegible] b_i, b_{i+1}

[illegible]. Let l be the length of the segment s; there is $\nu \in \mathbf{N}$ such that $2(\nu-1)k \leq l < 2\nu k$. Then, by dividing the s into ν equal subsegments, we have for the length l' of the subsegments: $\frac{2(\nu-1)}{\nu}k \leq l' < 2k$. For $\nu \geq 3$, we have $\frac{2(\nu-1)}{\nu} > \frac{2}{\sqrt{3}}$, and this proves the lemma. □

Let $\mathcal{P}$ be the separators $\mathcal{H}$ after we have inserted the points b_i, as described in the previous lemma, and let $D_{\mathcal{P}} = D \cup \mathcal{P}$. The following lemmata hold.

LEMMA 6.2. [illegible] b_i, b_{i+1} [illegible] s [illegible] $\mathcal{H}$ [illegible] $b_i b_{i+1}$ [illegible]

[illegible]. The diametral circle C of $b_i b_{i+1}$ is contained in the diametral circle of s, which by the MADD construction does not include any of the points of $D_{\mathcal{H}}$.

The remaining points to be examined are the inserted points b_j. We have that all the angles are greater than 60° and, from Lemma 6.1, no created segment is less than half of any other created segment. Consequently, C cannot contain a point b_j created by the refining procedure. □

LEMMA 6.3. [illegible] $\mathrm{lfs}_{\min}(D_{\mathcal{P}}) \geq k$.

[illegible]. We have from relation (6.1) that $\mathrm{lfs}_{\min}(D_{\mathcal{H}}) \geq k$. We will examine the distances created by the inserted points.

Let b_i be a point inserted in a segment s of $\mathcal{H}$. For the distance d of b_i from a nonincident to s segment we have $d \geq \mathrm{lfs}_{\min}(D_{\mathcal{H}}) \geq k$. The same holds for the distance d' from points that are not incident to s, because we have $d' \geq d \geq k$.

For the distance d between b_i and an incident segment we have $d \geq \sin 60° \cdot \frac{2}{\sqrt{3}}k = k$. Finally, the distance between b_i and a point that belongs to an incident segment is greater than the distance d of the previous relation, and this completes the proof. □

The previous lemma demonstrates the property that will be used to prove that $\mathcal{P}$ is a decoupling path. Our next step will be to calculate the parameter k.

Ruppert's algorithm can be applied using either the quality criterion for the circumradius to shortest edge ratio, or by adding a criterion for the maximum area of the created elements. We will calculate k for these two cases separately. We will prove that $\mathcal{P}$ is a decoupling path for the two cases: (I) when Ruppert's algorithm is applied with only the quality criterion of the circumradius to shortest edge ratio; (II) when it is applied with an additional criterion for the maximum triangle area.

6.1. Case I: The ratio criterion. In this case we are interested only in the circumradius to shortest edge ratio. Since k gives a bound for the size of the created segments, we would like k to be as big as possible and at the same time satisfy relation (6.1). Proposition 5.1 and Lemma 6.3 indicate that we can define $k = \min\{\mathrm{lfs}_{\min}(D_{\mathcal{H}}), L/4\}$.

PROPOSITION 6.4. [illegible] $k = \min\{\mathrm{lfs}_{\min}(D_{\mathcal{H}}), L/4\}$ [illegible] $\mathcal{P}$ [illegible] 6.1. [illegible] $\mathcal{P}$ [illegible]

. According to Proposition 5.1, Ruppert's algorithm when applied to a subdomain D_i, will not create segments less than $\text{lfs}_{\min}(D_i)$. We will show ad absurdo that the decoupling zone $\mathcal{Z}_\mathcal{P}$ is empty after the termination of the algorithm.

Suppose that $\mathcal{Z}_\mathcal{P}$ is not empty after the mesh procedure and some points have been inserted in it. That means that some boundary segments of $\mathcal{P}$ have been encroached and thus have been split in half. From Lemma 6.1 the length of the segments of $\mathcal{P}$ is less than $2k$ and by splitting them the created segments will have length less than k. This contradicts Proposition 5.1 because, from Lemma 6.3, we have $\text{lfs}_{\min}(D_i) \geq \text{lfs}_{\min}(D_\mathcal{P}) \geq k$.

Thus the decoupling zone $\mathcal{Z}_\mathcal{P}$ is empty after applying Ruppert's algorithm, and $\mathcal{P}$ is a decoupling path with respect to this algorithm. □

COROLLARY 6.5. $\mathcal{P}$.

. Ruppert's algorithm does not remove points and maintains the Delaunay property after inserting a point. The corollary is a direct consequence of the previous proposition and of Proposition 4.4. □

Proposition 6.4 states that we can process the subdomains independently, using Ruppert's algorithm, and the final mesh will be Delaunay conforming and of guaranteed quality. Next we will examine the case where we have an additional condition for the area of the triangles.

6.2. Case II: The ratio and maximum area criteria. In this case, besides the circumradius to shortest edge ratio condition, we have an additional criterion for the maximum triangle area. In many cases we want to construct Delaunay meshes, not only with good quality of angles, but also of a desired maximum size. Let A be a bound to the maximum triangle area, then all the triangles of the final mesh will have an area at most A. To achieve this, the mesh generation algorithm will split the triangles in two cases: (a) because of the bad circumradius to shortest edge ratio; (b) because the area of the triangle is greater than A.

We will calculate k so that the previous results will remain valid.

LEMMA 6.6. l A $\sqrt{2}$ $l > \sqrt{\frac{A}{\sqrt{2}}}$.

. Let r be the circumradius of the triangle. Then $\frac{r}{l} \leq \sqrt{2}$ and $A < r \cdot l$. So, $A < r \cdot l \leq \frac{l^2}{\sqrt{2}} \Rightarrow l > \sqrt{\frac{A}{\sqrt{2}}}$. □

We want to define k in such a way that the mesh generation procedure will not create edges smaller than k. The previous lemma indicates that we should have $k \leq \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}$. We will take $k = \min\{\text{lfs}_{\min}(D_\mathcal{H}),\ L/4,\ \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}\}$. Then Lemma 6.3 holds, and we have the following theorem.

THEOREM 6.7. $k = \min\{\text{lfs}_{\min}(D_\mathcal{H}),\ L/4,\ \frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}}\}$ 6.1, $\mathcal{P}$ $\mathcal{P}$ $\sqrt{2}$ A

. There are two cases for splitting a triangle: (a) because of its circumradius to shortest edge ratio or (b) because of its area.

When Ruppert's algorithm splits a triangle because of its circumradius to shortest edge ratio it does not create edges smaller than $\text{lfs}_{\min}(D_\mathcal{P}) \geq k$. If a triangle is split because of its size, then from Lemma 6.6 we have that the smaller created edge will be no less than $\frac{1}{2}\sqrt{\frac{A}{\sqrt{2}}} \geq k$. In both cases no edge smaller than k will be created.

It is easy to see now that the decoupling zone $\mathcal{Z}_{\mathcal{P}}$ will be empty, after Ruppert's algorithm has been applied on the subdomains D_i with the additional condition of a maximum triangle area A. If this was not so, then some edge of $\mathcal{P}$ would be encroached and split. From Lemma 6.1 the new edges will be smaller than k, which is a contradiction. □

In summary, the procedure of preprocessing the separators created by MADD, as described in Lemma 6.1, creates a decoupling path with respect to Ruppert's algorithm, in both cases of the quality and the size criteria. In the first case, the construction is based on the minimum local feature size, while in the second the maximum area of the triangles is taken into account.

The size optimality (times a constant) of the mesh produced by Ruppert's algorithm, when only the angle criterion is used, is based on the local feature size [40, 44]. The size optimality, combined with the angle quality, provides the basis of the adaptivity to the geometry, that the Delaunay mesh displays. On the other hand, the insertion of separators by itself changes the geometry to be meshed, and the uniform refinement of the separators alternates the local feature size of the geometry. After applying the decoupling procedure, the size optimality of the mesh is not any more guaranteed. The use of the local feature size, instead of the global minimum, in creating the decoupling path, would improve the gradation and mesh size, especially when there are big differences in the local feature size. In cases, though, where the geometry is very simple but h-refinement is important [22], we would like to limit the area of the triangles, and in these cases the optimality of the mesh size is not based on the local feature size. The meshes produced using the area restriction are usually much larger, and thus more prompt for parallel processing. The experiments that we ran show that the over-refinement imposed by the decoupling procedure is insignificant (see section 11) when the area criterion is used.

The creation of the decoupling path allows us to generate Delaunay meshes, independently for each subdomain, with good angle quality and of the desired size. The final mesh, formed by the union of the submeshes, is Delaunay conforming. As a result, this procedure decouples the domain and enables us to parallelize the mesh generation procedure, while eliminating the communication between the processors.

7. The parallel Delaunay decoupling procedure. The procedure for the parallel mesh generation consists of two steps.

1. [illegible]: In this step the domain is decomposed using the divide and conquer MADD method.
2. [illegible]: This step is performed independently for each subdomain and includes two substeps.
   (a) The decoupling of the subdomains by refining the interfaces, as described in section 6.
   (b) The mesh generation on the subdomains. In this step the sequential mesh generator is used as a library and is applied independently on each subdomain.

During the PMADD phase the domain is over-decomposed (i.e., we create $N \gg P$ subdomains, where P is the number of processors), in order to achieve good load balancing (see section 8). The PMADD method is implemented using a master/worker model. Processor 0 is used as the master processor, while all the processors, including processor 0, are used as worker processors. The master processor maintains a sorted list of the areas assigned to each processor. In each iteration of the PMADD procedure a decomposition request is sent from the master processor to the processors assigned

with larger total areas. The processors that receive such requests decompose their larger subdomain into two subdomains using MADD. One of the two new subdomains is sent to a processor with small total assigned area. The procedure is repeated until all N subdomains are created. Observe that the MADD is sequential and it includes a Delaunay triangulation procedure. The resulting triangulations are discarded, since they cannot be used in the next phase. The decomposition gradually creates simple geometries (see Figure 10.1) so that the cost to triangulate them is small. Nevertheless, the cost of the initial Delaunay triangulation remains but is small related to the cost of creating big meshes.

The area of the subdomains is used to estimate the workload for the mesh procedure (see section 8). The goals of the PMADD is to minimize the larger area and to distribute the subdomains uniformly to the processors. Once the PMADD phase is finished, no data movement takes place. This is an approximate criterion for the load balance, other means [2] for dynamic load balancing can be used.

After the requested number of subdomains have been created, the master processor sends requests to all processors to mesh the subdomains assigned to them. Each processor iterates through its subdomains and performs two steps.

(a) It refines the interfaces, where the separators created by the MADD are refined by inserting vertices, as described in the decoupling procedure in section 6, according to the given mesh quality criteria. The parameter k, that determines the refinement of the separators, is computed before the mesh generation phase begins, and is used to refine the internal boundaries of all the subdomains, independently for each subdomain. Although each interface is refined independently for the two subdomains where it belongs, the result is conforming, because the same parameter k is used, the same orientation for the interfaces, and of course the same algorithm.

(b) The mesh generation procedure is applied on the subdomains independently. The sequential mesh generator is used as is, in the form of a library. The interfaces, since they are refined from the decoupling procedure, will not be further refined form the mesh generation procedure and they will remain unchanged, as proved in section 4. So, no communication is required, and the created meshes are Delaunay conforming.

The procedure terminates when all the meshes for subdomains have been created. The parallel procedure is described next.

ALGORITHM 4.

Master processor:

1. Read the definition of the domain Ω
2. Initialize and maintain a sorted list of the areas of the subdomains
3. **while** the current number of subdomains is less than N **do**
4. **send** decompose requests to processors that are assigned large area of subdomains
5. **receive** replies about decoupling and area information
6. **endwhile**
7. **send** requests to processors to mesh their subdomains
8. **receive** replies **until** all processors completed meshing
9. **send** requests for termination

Worker processors:

10. **while** not terminate **do**
11. **receive** request from Master and/or other workers
12. **if** request is to decompose **then**
13. Apply MADD on the largest subdomain

```
14.            send reply to Master
15.            send a new subdomain to other processor
16.          endif
17.          if request is to receive a subdomain then
18.            Add the new subdomain to this worker's mesh-queue
19.            send reply to Master
20.          endif
21.          if request is to start meshing then
22.            for each assigned subdomain do
23.                Refine the separators according to the decouple procedure
24.                Apply the sequential mesh generator on the subdomain
25.            endfor
26.            send completion message to master
27.          endif
28.      endwhile
```

During the PMADD phase, the first P subdomains are created in $\lg(P)$ iterations. The total number of iterations for the parallel MADD phase is $\frac{N-P}{P/2} + \lg(P) = 2(M-1) + \lg(P)$, where M is the average number of the final subdomains per processor. Typical values for M in our experiments vary between 12 and 20. The procedure is using in average $\frac{N-1}{2(M-1)+\lg(P)} = \frac{PM-1}{2(M-1)+\lg(P)}$ processors per iteration.

This divide and conquer approach is not optimal, but the cost is very small (see section 12) with respect to the cost for the mesh generation. On the other hand, it achieves a good load balance among the processors, which is a more significant factor for the total performance of the parallel mesh generation (see section 12.2). In the next section we present in detail the load balance attained using the parallel MADD.

8. Load balancing. Our experiments show that more than 99% of the total time is spent in the meshing phase (see section 12), which does not suffer from communication or synchronization cost. Thus, the workload balance among the processors is the main parameter that affects the performance of the method. The load-balancing problem for mesh refinement is a difficult problem, because of the unpredictable computational behavior of the meshing procedure. The problem becomes more approachable by the use of the PMADD for over-decomposing the domain. The resulting subdomains have similar geometric shapes, and their area is proved to be a good measure for estimating the workload for the mesh generator.

Our experimental data show, for the geometries we tested so far, that the parallel MADD procedure creates subdomains with similar "good" shape (see Figure 10.1), when the number N of subdomains is large. Figure 8.1 shows that, as we increase N and thus decrease the area of the subdomains, the meshing time converges, with very small differences between subdomains of similar size. This result demonstrates that the area of the subdomain can be used to estimate the workload of the mesher for this subdomain. Of course this depends on the geometry of the original domain, which is one of the parameters that determine the level of required decomposition. An adaptive to the geometry approach for the PMADD would optimize the results, and this is a subject of future work.

The load balance among the processors is achieved by balancing the total area of the subdomains assigned to each processor. The first effort to create subdomains with similar sizes takes place during the graph partition. This result, though, is not guaranteed, and the obtained subdomains can have differences in size. By over-decomposing we have the ability to distribute the subdomains, so that each processor

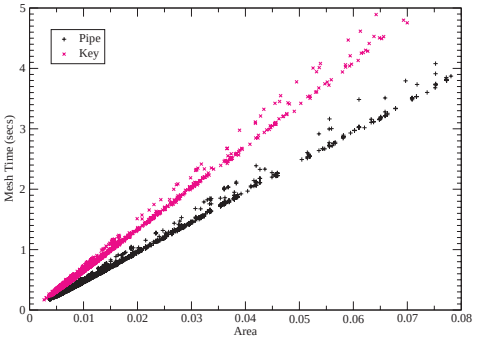


FIG. 8.1. *Mesh time for different sizes of subdomains of the key and the pipe geometry.*

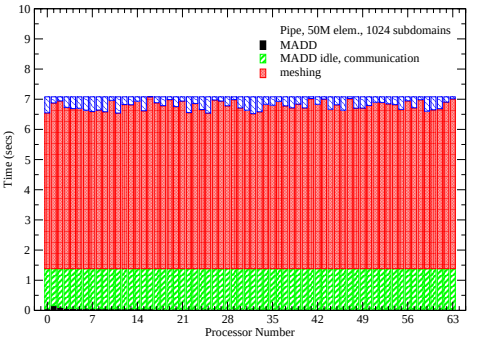


FIG. 8.2. *The work balance for* 64 *processors,* 50 *M elements.*

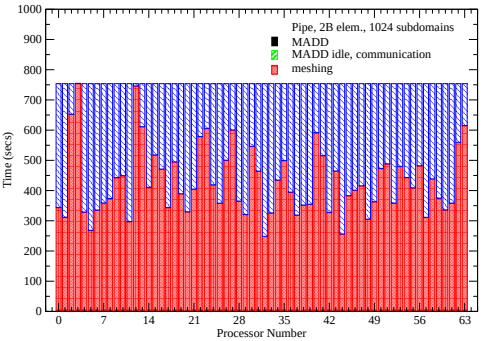


FIG. 8.3. *The work balance for* 64 *processors,* 2 *B elements,* 1024 *subdomains for the pipe.*

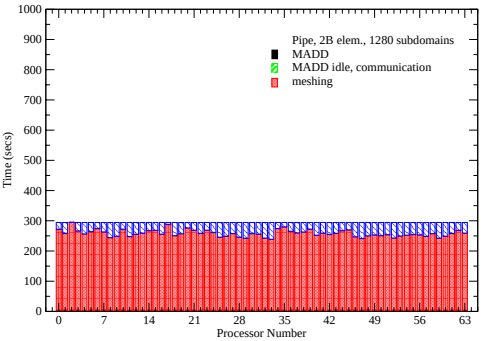


FIG. 8.4. *The work balance for* 64 *processors,* 2 *B elements,* 1280 *subdomains for the pipe.*

is assigned approximately the same total size. Moreover, the random distribution of the subdomains gives a more uniform assignment of subdomains that differ from the average in terms of size and geometry. The results of this simple approach are good. Figure 8.2 depicts the load balance among 64 processors for the pipe geometry, for 1024 subdomains and 50 M mesh size. This picture is typical in most cases. However, we have observed that the load balance depends not only on the geometry and the size of the subdomain but also on the size of the created mesh.

Figure 8.3 shows the load balance for the same decomposition of the pipe, as in Figure 8.2, this time for a mesh size of 2 billion elements. We see that the good load balance of Figure 8.2 is destroyed. The reason for this is that the time for creating larger meshes is much more sensitive to area and geometry differences. The answer to this problem is to increase N. In this way we improve two parameters: (i) the size of the mesh for each subdomain is decreased, and thus the time to create it is less sensitive to the differences, and (ii) a more uniform assignment of the subdomains can be accomplished. Figure 8.4 shows the balance for the same mesh size, 2 billion elements, by decomposing it into 1280 subdomains. This small increase of the number of subdomains gives an impressive improvement, the load balance is satisfactory and the total time is decreased to less than half; the reasons are described in sections 11

and 12.1.

The previous example shows that the load balance is sensitive to the size of the final mesh. The level of the required decomposition depends not only on the geometry and the number of processors, but mainly on the size of the final mesh. Let E be an estimation for the final size of the mesh in millions of elements. From our experiments we found that, for our setup, the number of subdomains should be at least $N = \frac{E}{1.6}$. This means that on average 1.6 M elements will be created for each subdomain. A higher decomposition has, of course, higher time cost, but this cost is insignificant against the gain, Figures 8.3 and 8.4, as well as the results in the next section demonstrate it.

9. Performance evaluation. We evaluate the parallel Delaunay decoupling (PDD) method with respect to three requirements listed in the introduction: (1) stability, (2) parallel efficiency, and (3) code reuse. Our experimental data indicate that the PDD method is stable; i.e., the elements of the distributed mesh retain the same good quality of angles as the elements generated by the Triangle (see Figures 11.3 and 12.3 (right)); at the same time it is very efficient as our fixed and scaled speedup data (see Figures 12.2 and 12.3 (left)) indicate. Finally it is based on 100% code reuse; i.e., existing sequential libraries like METIS and Triangle are used without any modifications for the parallel mesh generation.

10. Experimental setup. We have used two model domains (see Figure 10.1), with relatively simple geometries:[3] The , a cross section of a rocket from a NASA model problem where the peripheral pipes are used to cool the main cylinder in the center that contains combustion gases, and the , a domain provided with Triangle. We ran three sets of experiments: (1) to observe the behavior of the MADD and decoupling method in sequential execution for small meshes, 4–5 million (M) elements, (2) to calculate the fixed speedup for fixed size meshes of the order of 40–50 M elements, and (3) to compute the scaled speedup for meshes whose size range from 12 M to 2 billion (B) elements.

The programming language for our implementation was C++ and DMCS [3] was used as the communication substrate. The Triangle [43] library was used for the mesh generation procedure as well as for the creation of the Delaunay triangulation during the MADD procedure. The parameters passed to Triangle for the mesh generation were two: (a) for the quality of the elements (Ruppert's algorithm is used to achieve circumradius to shortest edge ration less then $\sqrt{2}$) and (b) for the maximum area of the generated elements. Also, METIS [31] was used for the graph partitioning step in the MADD procedure. The cases where METIS returned nonconnected subgraphs were recognized and discarded. All the libraries were used without modifications, minimizing the cost for the parallel implementation and achieving 100% code reuse.

The experiments ran on SciClone, a high-performance computing environment in the College of William and Mary. SciClone is a heterogeneous cluster of Sun workstations which use the Solaris 7 operating system. For our experiments we have used a subcluster of 32 dual-CPU Sun Ultra 60 workstations 360 MHz, with 512 MB memory and 18.2 GB local disk. Networking was provided by a 36-port 3Com Fast Ethernet switch (100Mb/s).

[3]The complexity of the geometry will challenge the PMADD and in particular the Delaunay triangulation procedure. Provided the efficiency of *Triangle*, this shouldn't be a problem. The mesh refinement procedure will be applied on the created subdomains, which have simple geometries. However, for three dimensional cases the complexity of the geometry is a much more serious issue.

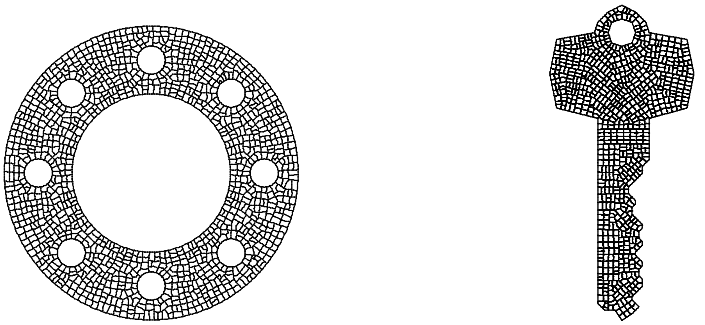


FIG. 10.1. *Left: The Pipe domain divided into* 1200 *subdomains. Right: The Key domain divided into* 768 *subdomains.*

11. Sequential experiments. We ran a set of sequential experiments in order to compare the sequential Delaunay decoupling method, where we over-decompose the domain, with Triangle, the best known publicly available sequential guaranteed quality Delaunay mesh generation code for 2-dimensional domains. In these experiments we examine the affects of the decoupling procedure with respect to the performance of the mesh procedure, the size of the final mesh, which indicates that the over-refinement we introduce is insignificant, and the quality of the elements in terms of the angle distribution. The size of the meshes we created is limited by the size (5.5 M) we were able to generate with Triangle due to memory limitations. However, using the

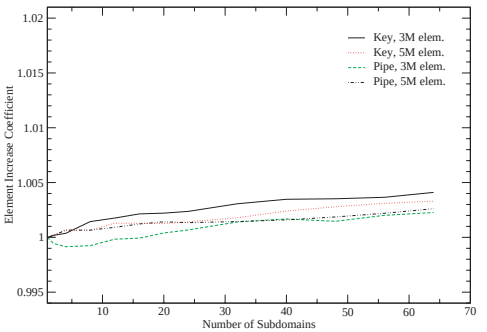


FIG. 11.1. *The increase of number of elements for decoupling into different number of subdomains.*

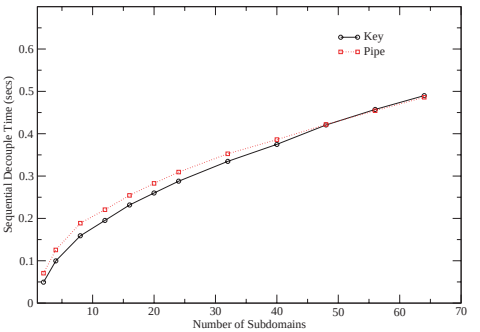


FIG. 11.2. *The time for the sequential MADD.*

Delaunay decoupling method we were able to generate more than 30 M on a single processor.

Figure 11.1 shows the ratio of the size of the decoupled meshes over the size of the nondecoupled mesh, which is a measure of the over-refinement we introduce when we decouple the domains.

Similarly, Table 11.1 presents the number of elements for different levels of decoupling. The over-refinement is insignificant; it is less than 0.4%, despite the intense over-decomposition (less than 90 K elements per subdomain).

The overhead of the sequential MADD method is approximately linear with respect to the number of subdomains; see Figure 11.2. This overhead is small compared to the mesh generation time. The total execution time using the sequential decoupling

TABLE 11.1
The number of elements and the total time (in seconds) for the same mesh generation parameters and for different levels of decoupling. The times do not include the mesh merging procedure.

Subdomains	1	8	16	32	48	64
Key elements	5,193,719	5,197,066	5,200,395	5,203,023	5,208,215	5,210,857
Total time	46.146	38.414	38.204	37.590	37.322	37.333
Pipe elements	5,598,983	5,602,668	5,605,819	5,607,055	5,609,404	5,613,624
Total time	59.263	41.342	41.046	40.370	40.352	40.147

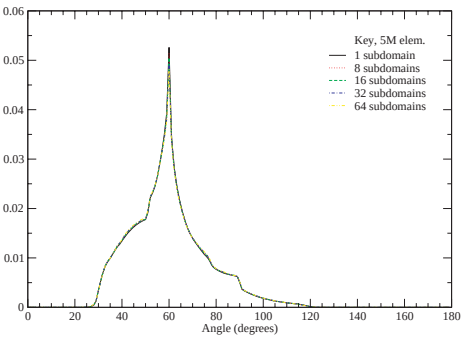


FIG. 11.3. *The angle distribution for different number of subdomains.*

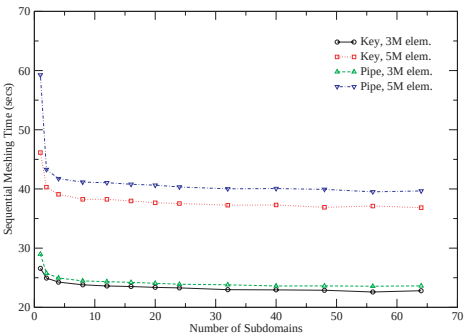


FIG. 11.4. *The time for sequential meshing after decoupling into subdomains. The times do not include the mesh merging procedure.*

procedure is decreased up to 68% of the time it takes for Triangle to generate a mesh with the same quality (Figure 11.4). As the size of the mesh increases, the performance of the decoupling procedure compared to Triangle is improving even further, because the size of the working set for each subdomain is smaller and the Delaunay mesh algorithm used in Triangle has a nonlinear time complexity [43].

The quality of the elements produced after the decoupling of the domain into subdomains is evaluated by comparing the distribution of angles. We compare the angles of the elements from both the nondecoupled mesh generated by Triangle and the decoupled ones generated by our method. Figure 11.3 shows that the distribution is the same. The above results hold as we scale the mesh size in our parallel experiments.

In summary, the decoupling method demonstrates merits even for sequential mesh generation. The gains in the performance from the better memory utilization cover the small overheads due to decoupling and over-refinement, while the element quality is independent of the decoupling, which shows that our method is stable regarding the quality of the mesh.

12. Parallel experiments. We performed two sets of experiments in order to calculate the fixed and scaled speedup using 8, 16, 32, and 64 processors. With 64 processors we were able to generate 2.1 billion (B) high quality elements for the Pipe in less than 3.5 minutes, while using Triangle [43] on a single workstation we were able to generate 5.5 million (M) elements in about one minute (see Tables 11.1 and 12.2).

In the rest of the section we present performance data for both the parallel medial axis domain decomposition (PMADD) method and the parallel mesh generation. The

PMADD procedure is evaluated in terms of its total parallel execution time which includes some communication and idle time and the maximum computation time spent on a single processor. The parallel mesh generation phase does not require communication and its performance is measured in terms of maximum and average computation time of processors. The ratio of these two numbers is used to measure the load imbalance of the parallel meshing phase.

Finally, we evaluate the scalability of the method in terms of two performance criteria: (1) the , , , , that it takes for one element to be created on a single processor, over all the processors and elements that are created, and (2) the , , cost (due to decomposition and parallelism) for each processor we use. Both criteria indicate that the parallel mesh generation method we present here is scalable and that we can generate billions of elements with insignificant overhead (see Table 12.2).

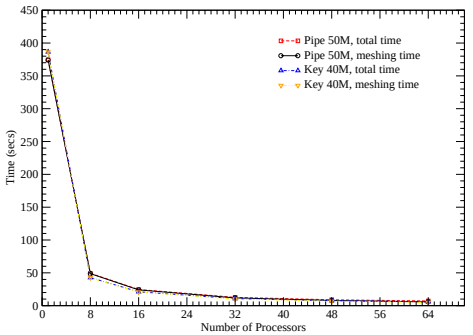


FIG. 12.1. *The performance for fixed size mesh.*

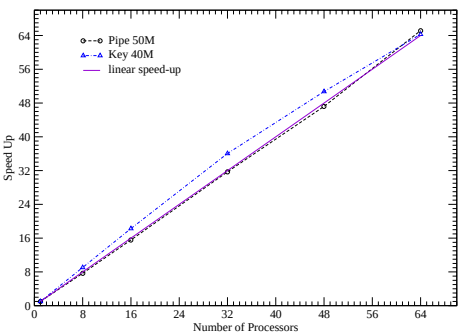


FIG. 12.2. *The speedup for fixed size mesh.*

12.1. Fixed size mesh experiments. In the fixed size set of parallel experiments we used a mesh of 40 M elements for the Key domain and 50 M for the cross section of the Pipe. For the Key domain we created 12 subdomains for each processor while for the Pipe we created 16 subdomains. The maximum triangle area is fixed throughout the experiments for each domain.

TABLE 12.1

Performance data for the Key and the Pipe geometry for a fixed maximum element area. All times are in seconds and mesh sizes are in millions (M).

No of processors	1	8	16	32	48	64
The domain						
No of subdomains	12	96	192	384	576	768
Mesh size (M)	43.32	43.34	43.37	43.41	43.43	43.45
PMADD time	0.20	0.37	0.44	0.60	0.83	1.05
Meshing time	386.32	42.35	20.72	10.12	6.79	4.96
Total time	386.52	42.72	21.16	10.72	7.62	6.01
The Pipe domain						
No of subdomains	16	128	256	512	768	1024
Mesh size (M)	50.93	50.97	51.00	51.05	51.08	51.11
PMADD time	0.27	0.51	0.60	0.89	1.07	1.47
Meshing time	374.15	48.80	24.03	11.80	7.93	5.74
Total time	374.42	49.29	24.63	12.69	9.00	7.21

The results are presented in Table 12.1. The data again indicate an unimportant increase in the number of elements for the different levels of over-decomposition, which

shows that the over-refinement we introduce is insignificant. The total execution time and the computation time for the actual mesh generation are depicted in Figure 12.1. These times are very close, because the PMADD overhead cost is very small. This cost is neutralized by the effect of over-decomposition, which, along with the good load balancing and zero communication during the parallel meshing, leads to superlinear speedup; see Figure 12.2. The speedup is calculated against the total time it takes to create the mesh on one processor, as is presented in Table 12.1.

12.2. Scaled size mesh experiments. A more practical way to evaluate the scalability and true performance of a parallel algorithm and software is to scale the size of the problem in proportion to the number of processors used. In the following experimental data we use the same level of decomposition for every configuration of processors, i.e., we keep the average number of subdomains per processor constant, and thus we eliminate the effect of over-decomposition in the resulting performance data. Theoretically we should be able to achieve the same creation time per element per processor for all the parallel configurations independently of the number of processors used. However, this is not feasible for the following two reasons: (1) the decomposition overhead, which increases very slowly but nevertheless there is an increase in the overhead as the number of processors increases and (2) load imbalances due to unpredictable and variable computation of the mesh generation kernel.

Table 12.2 shows some performance indicators for the two model problems we use, the Key and the Pipe geometry. In the experiments for the Key model we created 12 subdomains per processor and generated on average 1.6 M elements per subdomain, i.e., a total of 20 M per processor. For the Pipe model we created 20 subdomains per processor and generated on average 1.6 M elements per subdomain, i.e., a total of 32 M per processor. Small differences exist in the size of the mesh because our stopping criteria are based on the quality and size of elements, and thus the mesh size cannot be exactly predefined. It is clear from Table 12.2 that for larger processor configurations, like 64 processors, 99.5% of the total execution time is spent in the meshing phase by the Triangle. This suggests that for realistic problems the PMADD overhead is about 0.5% of the total execution time.

We observe that, while the maximum PMADD time on one processor remains almost constant, the time for PMADD phase increases as the number of processors increases. This is in agreement with the analysis in section 7. As the number of processors increases, the number of PMADD iterations increases, although the number of the subdomains per processor is constant. In each PMADD iteration all the processors finish the decomposition before the next iteration begins. This synchronization imposes an additional cost in the PMADD time. Moreover, the communication during this phase increases, as the number of processors increases. Fortunately, the communication and synchronization cost is less than 0.02 s per processor. In comparison with the total execution time this cost is very small.

The load imbalance is measured by the ratio of the maximum meshing time on one processor and the average meshing time for all the processors. In Table 12.2 we observe that the load balance for the Pipe is very good, 1.14 for 64 processors, while for the Key it is a satisfactory 1.25. The load balance is based on over-decomposing the domain and equi-distributing the areas, and although it depends on the size of the mesh as we saw in section 8, it also depends on the geometry and the number of the processors. Further improvement in the load balance can be achieved by using parallel runtime software systems that address load-balancing problems, as the one presented in [2].

TABLE 12.2

Performance data for the Key and the Pipe geometry. The meshing time includes the time of the decoupling procedure (MADD). The MADD phase includes the load balance estimation procedure and the distribution of the subdomains to the processors. The imbalance is measured as ratio of the max meshing processor time over the average. All times are in seconds except for the time/(elem./procs) which is in microsecs.

No of processors	1	8	16	32	48	64
The Key domain						
No of subdomains	12	96	192	384	576	768
Mesh size	20M	160M	320M	650M	860M	1.3B
Total time	152.43	177.31	192.41	213.91	166.10	205.26
Max meshing time	152.23	176.92	191.93	213.26	165.25	204.19
Aver. meshing time	152.23	165.75	168.04	170.31	137.70	163.14
Imbalance	1	1.067	1.142	1.252	1.200	1.252
MADD phase time	0.20	0.38	0.44	0.63	0.84	1.05
Max MADD time	0.20	0.14	0.13	0.13	0.12	0.13
Tot. time/(elem./procs)	7.33	8.73	9.47	10.54	9.20	10.11
Additional Cost /procs	0%	2.4%	1.8%	1.4%	0.5%	0.6%
The Pipe Domain						
No of subdomains	20	160	320	640	960	1280
Mesh size	32M	240M	500M	1B	1.4B	2.1B
Total time	236.00	247.10	245.32	279.59	246.59	294.39
Max meshing time	235.71	246.53	244.65	278.56	245.09	292.71
Aver. meshing time	235.71	226.78	231.15	253.59	218.56	255.87
Imbalance	1	1.087	1.058	1.098	1.121	1.144
MADD phase time	0.29	0.55	0.67	1.01	1.48	1.66
Max MADD time	0.29	0.19	0.17	0.17	0.16	0.18
Tot. time/(elem./procs)	7.30	8.23	7.94	8.51	8.45	8.96
Additional cost /procs	0%	1.6%	0.6%	0.5%	0.3%	0.4%

An important measure for evaluating the efficiency of a parallel meshing method is the (total) time spent for creating one element on one processor. Let $T^{(P)}$ be the total time running on P processors in order to create a mesh of size $S^{(P)}$. Then the time per element per processor is $T_e^{(P)} = \frac{T^{(P)} \cdot P}{S^{(P)}}$. This measure eliminates the differences in the mesh size, providing a more objective view of the scaled performance. We see in Table 12.2 that this time is almost constant, and thus the method is scalable. The slight increase of this time is mainly due to the imbalance increase, while the contribution of the overhead time cost is very small. This is evident in Figure 12.3, where the imbalance is depicted on the top and the scaled speedup on the bottom. The scaled speedup for P processors is measured as $U_P = \frac{T_e^s \cdot P}{T_e^{(P)}}$, where T_e^s is the time to create sequentially one element for a nondecomposed mesh of size 5 M. We again observe the superlinear speedup for the same reasons as in the fixed size experiments. In this figure the direct impact of the imbalance to the speedup is obvious.

Another measure for evaluating the scalability is the additional time cost for each processor that we use, relative to the total time when running on one processor. The additional cost C_P per processor, when using P processors, is computed as $C_P = \frac{T_e^{(P)} - T_e^{(1)}}{T_e^{(1)} \cdot P}$. Taking into account that the mesh size $S^{(P)}$ is approximately proportional to the number of processors P, we have $C_P \simeq \frac{T^{(P)} - T^{(1)}}{T^{(1)} \cdot P}$. We can consider the quantity $T^{(1)}$ as the ideal time for creating on P processors a mesh of size $S^{(P)} \simeq P \cdot S^{(1)}$, since the effect of over-decomposition is eliminated. In this way the additional cost C_P measures the distance from the ideal speedup, distributed to the number of processors used.

The time $T_e^{(P)}$ is increasing as P increases; the reasons were explained above.

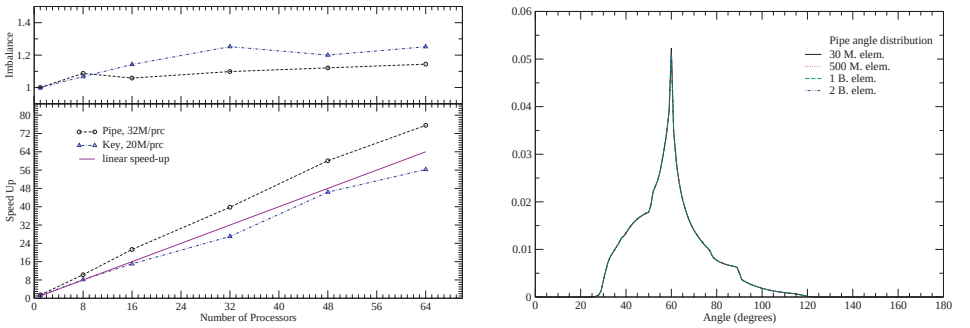


FIG. 12.3. *Left: At top is the imbalance and at bottom the speedup for the scaled experiments. The speedup is measured against the sequential creation of* 5 *M elements and is based on the overall time it takes for one element to be created. Observe the direct impact of the imbalance to the speedup. Right: The angle distribution for scaled mesh sizes of the Pipe.*

This increase, though, is small for the Key and even smaller for the Pipe domain. It is interesting to observe that the additional cost C_P tends to decrease as P increases. Although we have to pay a (small) cost in the performance for each additional processor we use, this cost tends to decrease, when measured in scale. This result underlines the scalability of the method.

Finally, we should compare the quality of the elements of scaled meshes that the decoupling procedure produces. In Figure 12.3 (right) is depicted the distribution of the angles of the elements, for meshes varying from 30 M triangles to 2.1 B. The quality is obviously the same.

13. Conclusions and future work. We presented a decoupling procedure for parallel Delaunay guaranteed quality mesh generation on distributed memory machines for 2-dimensional domains. The method eliminates the communication during the mesh generation and maintains the size and the quality of the final mesh. It also shows good speedup and scalability, making it suitable for creating very large meshes on distributed memory machines. A major advantage of our method is that a commercial, off-the-shelf and state-of-the-art, sequential mesher, like Triangle [43], can be used as a library, without any modification, achieving 100% code reuse. The method can be used at the same time for sequential mesh generation, in order to create larger meshes in less time using one processor. Because of the zero communication and the scalability for large meshes, this method seems to be suitable for Grid computing applications [16].

Future work for 2-dimensional geometries includes the extension of our approach for optimizing over-refinement of the interfaces, using the local feature size rather than $\text{lfs}_{\min}$. This will address the mesh gradation and improve the mesh-size optimality criterion.

Another problem is to determine the required level of the medial axis approximation that guarantees the creation of a minimum number of junction triangles, in order to produce separators that satisfy our geometric constraints.

It is also interesting to see how this approach can be applied in three dimensions for surface and volume parallel guaranteed quality mesh generation. The main issue in 3-dimensional domains is the creation of suitable domain decompositions, similar to the one we are able to create for the 2-dimensional cases.

Acknowledgment. The authors would like the thank the reviewers for their time in reviewing this paper and for their very useful suggestions.

REFERENCES

[1] C. Armstrong, D. Robinson, R. McKeag, T. Li, S. Bridgett, R. Donaghy, and C. McGleenan, *Medials for meshing and more*, in Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, NM, 1995, pp. 277–288.

[2] K. Barker, N. Chrisochoides, A. Chernikov, and K. Pingali, *A framework for load balancing of adaptive and asynchronous applications*, IEEE Trans. Parallel and Distributed Systems, 14 (2004), pp. 183–192

[3] K. Barker, N. Chrisochoides, J. Dobbelaere, D. Nave, and K. Pingali, *Data movement and control substrate for parallel adaptive applications*, Concurrency and Computation: Practice and Experience, 14 (2002), pp. 77–101.

[4] S. T. Barnard and H. D. Simon, *A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems*, Concurrency: Practice and Experience, 6 (1994), pp. 101–107.

[5] A. Beguelin, J. Dongarra, A. Geist, R. Manchek, K. Moore, and V. Sunderam, *PVM and HeNCE: Tools for heterogeneous network computing*, in Software for Parallel Computation, Vol. 106, J. S. Kowalik and L. Grandinetti, eds., Springer-Verlag, Berlin, 1993.

[6] G. E. Blelloch, G. L. Miller, J. C. Hardwick, and D. Talmor, *Design and implementation of a practical parallel Delaunay algorithm*, Algorithmica, 24 (1999), pp. 243–269.

[7] H. Blum, *A transformation for extracting new descriptors of shape*, in Models for the Perception of Speech and Visual Form, MIT Press, Cambridge, MA, 1967, pp. 362–380.

[8] J. W. Brandt, *Convergence and continuity criteria for discrete approximations of the continuous planar skeleton*, CVGIP: Image Understanding, 59 (1994), pp. 116–124.

[9] J. W. Brandt and V. R. Algazi, *Continuous skeleton computation by Voronoi diagram*, CVGIP: Image Understanding, 55 (1992), pp. 329–338.

[10] L. P. Chew, *Constrained Delaunay triangulations*, Algorithmica, 4 (1989), pp. 97–108.

[11] L. P. Chew, *Guaranteed Quality Triangular Meshes*, Technical report TR-89-983, Department of Computer Science, Cornell University, Ithaca, NY, 1989.

[12] L. P. Chew, *Guaranteed-quality mesh generation for curved surfaces*, in 9th Annual Symposium on Computational Geometry, ACM, San Diego, CA, 1993, pp. 274–280.

[13] L. P. Chew, N. Chrisochoides, and F. Sukup, *Parallel constrained Delaunay triangulation*, in ASME/ASCE/SES Special Symposium on Trends in Unstructured Mesh Generation, Evanston, IL, 1997, pp. 89–96.

[14] H. Choi, S. Choi, and H. Moon, *Mathematical theory of medial axis transform*, Pacific J. Math., 181 (1997), pp. 57–88.

[15] N. Chrisochoides, *An alternative to data mapping for parallel PDE solvers: Parallel grid generation*, in Scalable Parallel Libraries Conference, Mississippi State, MS, 1993.

[16] N. Chrisochoides, A. Fedorov, B. B. Lowekamp, M. Zangrilli, and C. Lee, *A case study of optimistic computing on the Grid: Parallel mesh generation*, in Next Generation Systems Program Workshop, IPDPS'03, Nice, 2003.

[17] N. Chrisochoides and D. Nave, *Simultaneous mesh generation and partitioning*, Math. Comput. Simulation, 54 (2000), pp. 321–339.

[18] N. Chrisochoides and D. Nave, *Parallel Delaunay mesh generation kernel*, Internat. J. Numer. Methods Engrg., 58 (2003), pp. 161–176.

[19] N. Chrisochoides and F. Sukup, *Task parallel implementation of the BOWYER-WATSON algorithm*, in Proceedings of Fifth International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields, Mississippi State, MS, 1996.

[20] H. de Cougny and M. Shephard, *Parallel volume meshing using face removals and hierarchical repartitioning*, Comput. Methods Appl. Mech. Engrg., 174 (1999), pp. 275–280.

[21] R. Diekmann, D. Meyer, and B. Monien, *Parallel decomposition of unstructured FEM-meshes*, Concurrency: Practice and Experience, 10 (1998), pp. 53–72.

[22] S. Dong and G. Karniadakis, *DNS of flow past a stationary and oscillating rigid cylinder at* RE = 10,000, in Flow Induced Vibrations, E. de Langre and F. Axisa, eds., Ecole Polytechnique, Paris, 2004.

[23] P. J. Frey and P.-L. George, *Mesh Generation*, Hermes Science Publishing, Oxford, UK, 2000.

[24] J. Galtier and P.-L. George, *Prepartitioning as a way to mesh subdomains in parallel*, in 5th International Meshing Roundtable, Pittsburgh, PA, 1996, pp. 107–122.

[25] P. L. George and H. Borouchaki, *Delaunay Triangulation and Meshing: Applications to Finite Element*, Hermes, Paris, 1998.

[26] H. N. Gursoy and N. M. Patrikalakis, *An automatic coarse and fine surface mesh generation scheme based on medial axis transform: Part* I. *Algorithms*, Engineering with Computers, 8 (1992), pp. 121–137.

[27] B. Hendrickson and R. W. Leland, *An improved spectral graph partitioning algorithm for mapping parallel computations*, SIAM J. Sci. Comput., 16 (1995), pp. 452–469.

[28] B. Hendrickson and R. W. Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings of ACM/IEEE Conference on Supercomputing, San Diego, CA, 1995.

[29] C. Kadow and N. Walkington, *Design of a projection-based parallel Delaunay mesh generation and refinement algorithm*, in 4th Symposium on Trends in Unstructured Mesh Generation, 2003.

[30] G. Karypis and V. Kumar, *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*, Technical report TR 95-035, Department of Computer Science, University of Minnesota, Minneapolis, MN, 1995.

[31] G. Karypis and V. Kumar, *METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Version* 2.0, 1995.

[32] B. W. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell Syst. Tech. J., 49 (1970), pp. 291–307.

[33] D. T. Lee, *Medial axis transformation of a planar shape*, IEEE Trans. Pattern Analysis and Machine Intelligence, PAMI 4 (1982), pp. 363–369.

[34] R. Lohner and J. Cebral, *Parallel advancing front grid generation*, in International Meshing Roundtable, Sandia National Labs, Albuquerque, NM, 1999.

[35] B. Maerten, D. Roose, A. Basermann, J. Fingberg, and G. Lonsdale, *DRAMA: A library for parallel dynamic load balancing of finite element applications*, in European Conference on Parallel Processing, Toulouse, 1999, pp. 313–316.

[36] D. Nave, N. Chrisochoides, and L. P. Chew, *Guaranteed-quality parallel Delaunay refinement for restricted polyhedral domains*, in Proceedings of the 18th ACM Symposium on Computational Geometry, ACM Press, New York, 2002, pp. 135–144.

[37] P. P. Pebay and J. F. Pascal, *A-priori Delaunay-conformity*, in 7th International Meshing Roundtable, 1998, Sandia National Labs, pp. 321–333.

[38] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[39] M. Price, C. Stops, and G. Butlin, *A medial object toolkit for meshing and other applications*, in Proceedings of 4th International Meshing Roundtable, 1995, Sandia National Labs, pp. 219–229.

[40] J. Ruppert, *A Delaunay refinement algorithm for quality* 2*-dimensional mesh generation*, J. Algorithms, 18 (1995), pp. 548–585.

[41] R. Said, N. Weatherill, K. Morgan, and N. Verhoeven, *Distributed parallel Delaunay mesh generation*, Comput. Methods Appl. Mech. Engrg., 177 (1999), pp. 109–125.

[42] E. C. Sherbrooke, N. M. Patrikalakis, and F.-E. Wolter, *Differential and topological properties of medial axis transforms*, Graphical Models and Image Processing, 55 (1996), pp. 574–592.

[43] J. R. Shewchuk, *Triangle: Engineering a* 2*D quality mesh generator and Delaunay triangulator*, in Applied Computational Geometry: Towards Geometric Engineering, Lecture Notes in Comput. Sci. 1148, M. C. Lin and D. Manocha, eds., Springer-Verlag, 1996, pp. 203–222.

[44] J. R. Shewchuk, *Delaunay Refinement Mesh Generation*, Ph.D. thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1997.

[45] T. K. H. Tam and C. Armstrong, 2*D finite element mesh generation by medial axis subdivision*, Adv. Engrg. Software, 56 (1991), pp. 313–324.

[46] C. Walshaw and M. Cross, *Mesh partitioning: A multilevel balancing and refinement algorithm*, SIAM J. Sci. Comput., 22 (2000), pp. 63–80.

[47] C. Walshaw, M. Cross, and M. G. Everett, *Parallel dynamic graph partitioning for adaptive unstructured meshes*, J. Parallel Distributed Comput., 47 (1997), pp. 102–108.

[48] F.-E. Wolter, *Cut Locus and Medial Axis in Global Shape Interrogation and Representation*, Technical report, Department of Ocean Engineering, Design Laboratory, MIT, Cambridge, MA, 1993.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1424–1437
© 2006 Society for Industrial and Applied Mathematics

ALTERNATIVE WEIGHTED DROPPING STRATEGIES FOR ILUTP*

JAN MAYER†

Abstract. This paper presents a weighted dropping strategy for the matrix L of the incomplete LU preconditioner with pivoting and dropping based on a dual threshold strategy (ILUTP). Preconditioners using this strategy often perform better than the standard ILUTP preconditioner for some difficult matrices. For matrices that are relatively easy to precondition, the results are similar. The implementation and setup times are somewhat longer than the ILUTP preconditioner employing the standard dropping strategy.

Key words. incomplete LU factorization, ILU, preconditioning, ILU with threshold, ILUT, ILUTP, iterative methods, sparse linear systems

AMS subject classifications. 65F10, 65F50

DOI. 10.1137/030602022

1. Introduction.

In this paper, we present a new method for dropping elements in the rows of the matrix L of the ILUTP preconditioner of a matrix A based on assigning weights to the elements of the row of L being calculated. The ILUT preconditioner, first introduced in [11], is based on an incomplete LU factorization of A with dropping being implemented using a dual threshold strategy in order to ensure sparsity. If the Gaussian elimination is performed with column pivoting, we obtain the ILUTP preconditioner found in [14], for which we present the weighted dropping strategy.

Different dropping strategies for ILU-type preconditioners have recently received more attention. An approach based on monitoring the growth of inverse factors (i.e., of L^{-1} and U^{-1}) can be found in [2]. A similar approach for an ILU-type preconditioner based on Crout's formulation called ILUC can be found in [9]. It is better for some matrices than the standard ILUT preconditioner but not as flexible as the ILUTP preconditioner because it does not permit pivoting.

Our approach is based instead on approximating the error made by dropping a single element in the LU factorization. For matrices for which the standard ILUTP preconditioner already yields satisfactory results (in other words, for matrices for which the standard ILUTP preconditioner can be chosen to be fairly sparse and the iterative method converges well), the performance of the new dropping method is at times slightly worse than the standard method. On the other hand, for the more difficult matrices considered here for which ILUTP preconditioners require more fill-in in order to work, the weighted dropping strategy often performs better, although this appears to depend somewhat on the application from which the matrix arises. Hence, the standard strategy should be preferred whenever it appears likely that ILUTP will work with little fill-in, whereas the weighted strategy is an option when ILUTP is expected to require fill-in with a factor of approximately 1.5 or more. As the setup times for the weighted strategy are somewhat longer, using it is only worthwhile if this is compensated by a reduction in the iterations (or the iteration time).

*Received by the editors November 28, 2003; accepted for publication (in revised form) January 10, 2005; published electronically January 27, 2006.
http://www.siam.org/journals/sisc/27-4/60202.html
†Institut für Angewandte Mathematik, Universität Karlsruhe, D-76128 Karlsruhe, Germany (jan.mayer@math.uni-karlsruhe.de).

Unfortunately, a similar dropping strategy for U is not available despite being possible theoretically. It is possible to implement the strategy for L because the ILUTP preconditioner is calculated by rows. Implementing a similar strategy for U would, however, require the preconditioner to be calculated by columns. In this sense, it is certainly possible to use such a strategy for U at the expense of not being able to use it for L. Implementing both, however, seems to be impossible.

For ILUC, on the other hand, L is calculated by columns and U by rows, so for this preconditioner it is theoretically possible to calculate the weights needed for both L and U. Using this approach, all the elements of a column of L being calculated (or a row of U) are assigned the same weight; in other words, the column of L (or row of U) is just scaled. Hence, using a dropping rule based on the relative sizes of elements yields the same results as a dropping rule without weights. However, a dropping rule using the absolute sizes of the elements yielded poor results. Thus, the weighted strategy does not appear to be suitable for ILUC.

Although the weighted strategies can be used with ILUT-based and ILUC-based multilevel preconditioners as implemented in pARMS (see [15]) or ILUPACK (see [3], [4], and [5]) without any modification, preliminary testing indicates that using the weighted strategy makes almost no difference in the quality of the resulting preconditioner. Hence, at this time, using weighted dropping in this situation does not appear to be advisable, but further investigation in this area may be worthwhile.

In the next section, we review the basic ideas concerning the ILUTP preconditioner. Then we present the theoretical justification for the weighted dropping strategy. Doing this rigorously appears to be possible only for dropping a single element, but this does provide a heuristic for dropping several elements at a time. We conclude this paper by presenting extensive numerical results that underline the observations alluded to before, i.e., that the weighted dropping strategy improves the ILUTP preconditioner for some difficult matrices.

2. The ILUTP preconditioner. The ILUTP preconditioner is calculated by an incomplete LU factorization using column pivoting. A simpler version without pivoting is called the ILUT preconditioner, which we describe first.

For a given matrix $A \in \mathbb{R}^{n\times n}$, the ILUT preconditioner is calculated by performing an LU factorization of A in such a manner that in the ith step ($i = 1, \ldots, n$) the ith row of both L and U are calculated completely and by dropping certain elements of these rows in order to ensure sparsity. The following algorithmic description is due to [14, p. 312]. For details, particularly on implementation issues, consult [4], [6], [11], and [12].

ALGORITHM (ILUT).

```
1.  for i = 1..n
2.      w := a_i                                  (a_i is the ith row of A)
3.      for k = 1..i − 1 and w_k ≠ 0
4.          w_k = w_k/a_kk
5.          Apply a dropping rule to w_k.
6.          if w_k ≠ 0 then
7.              w = w − w_k ∗ u_k                 (u_k is the kth row of U)
8.          end if
9.      end for k
10.     Apply a dropping rule to w.
11.     for j = 1..i − 1 : l_ij = w_j
12.     for j = i..n : u_ij = w_j
```

13. $w = 0$
14. end for i

Dropping in the ith step is usually done by a dual strategy based on two parameters $p \in \mathbb{N}$, which indicates the maximal number of elements in each row of L and U, and $\tau \in \mathbb{R}$, which gives a threshold for fill-in. Let w^L and w^U be the vectors consisting of the elements of w corresponding to L and U, respectively; i.e., w^L consists of the first $i-1$ elements of w and w^U consists of the remaining ones. For the standard dropping strategy, first all nonzero elements w_i^L of w^L satisfying $|w_i^L| < \tau$ are set to zero in line 5. Then all nonzero elements of w^U satisfying $|w_i^U| < \tau \cdot ||w^U||$ are set to zero in line 10. Finally, if there are still more than p elements in either w^L or w^U, then only the largest p elements by absolute value of w^L and w^U, respectively, are kept. For lack of a better name, we call this dropping strategy S0.

By choosing the largest element by absolute value as the pivot in each row and permuting the columns of A appropriately in the algorithm above, we obtain the ILUTP preconditioner. Often, elements are permuted only if their relative sizes exceed another parameter, the permutation tolerance. Based on the empirical results presented in [6], which indicate that pivoting yields somewhat better results, and the fact that our focus lies on fairly difficult matrices, we will always perform pivoting and not go into the details of implementing a permutation tolerance.

Although preprocessing is, strictly speaking, not part of the incomplete LU factorization, generally it improves the resulting preconditioner considerably and should be used; see [1].

3. New weighted dropping strategies. Let $L = (l_{ij})$, U and A be matrices such that $LU = A$, and $\hat{L}$ be the matrix obtained by dropping l_{ij} from L, i.e., by setting $l_{ij} = 0$. The error in the maximum norm made by using the factorization $\hat{L}U$ instead of LU is

$$||A - \hat{L}U||_\infty = ||LU - \hat{L}U||_\infty = ||(L - \hat{L})U||_\infty.$$

Because $L - \hat{L}$ is a matrix containing only zeros except for l_{ij}, the matrix $(L - \hat{L})U$ is a matrix having only zeros, except for the ith row, which contains $l_{ij} \cdot u_j$, u_j being the jth row of U. Hence, we have

$$||A - \hat{L}U||_\infty = |l_{ij}| \cdot ||u_j||_1.$$

Thus, if we wish to drop a single element from L such that the error obtained in the factorization above is minimal, we will need to drop the element l_{ij} such that $|l_{ij}| \cdot ||u_j||_1$ is minimal. Although these considerations are only accurate for dropping a single element, we do obtain heuristically that we should drop those elements l_{ij} for which $|l_{ij}| \cdot ||u_j||_1$ is smallest. In other words, the elements of the jth column of L should be weighted by the norm of the jth row of U. A similar consideration for dropping elements of U would indicate that the elements of the ith row of U should be weighted by the ith column of L. Because the ILUT preconditioner is calculated by rows, dropping elements in U using this approach is not possible in this case and we will not pursue it further. Finally, these considerations carry over to the ILUTP preconditioner as well without any modifications. For simplicity, we will just describe the implementation for the ILUT preconditioner.

As ILUT preconditioners are calculated by rows, in the ith step the first $i-1$ rows of L and U and the vectors w, w^L, and w^U are available. Let v^L be the vector of weighted elements, i.e., $v_j^L = w_j^L \cdot ||u_j||_1$. Using a double threshold strategy in

different ways, it is possible to construct several similar, but different, preconditioners. Simply replacing w^L of the standard dropping rule in line 5 of the algorithm by a weighted version may be the most straightforward approach, but the resulting preconditioner is generally poor. This is probably due to the fact that the elements dropped are important for the calculations in line 7, i.e., for calculating subsequent elements of L and U, even though the heuristic considerations above indicate that they are not important for the preconditioner itself. Hence, we would expect that using a weighted dropping rule in line 10 would be more successful and indeed this has been the case. We obtain a preconditioner with a weighted dropping strategy, which we call W0, if we use the standard strategy as described above and additionally drop the elements w_i^L with index i in line 10 of the algorithm for which $|v_i^L| < \tau \cdot ||v^L||$. If w^L still contains more than p elements, we keep only those p elements for which $|v_i^L|$ is largest. Note that the standard strategy performs no dropping in line 10 based on the threshold τ. However, it keeps only those p elements of w^L for which $|w_i^L|$ is largest. Here we need to point out that usually p is set to be very large, and thus only the threshold τ is responsible for dropping elements. These preconditioners tend to be better and the setup times are usually somewhat better, as the selection of the p largest elements (by absolute value) of a vector can be avoided. Consequently, for most practical purposes, the aspect of (weighted) dropping based on the parameter p is not particularly relevant. In any case, the weighted dropping rule described above leads to the idea that the standard dropping rule could be supplemented by additional dropping in L in line 10 based on τ. The corresponding dropping rule would be to drop those elements w_i^L of w^L for which $|w_i^L| < \tau \cdot ||w^L||$. Unlike W0, this strategy is not very effective for reasons which are not quite clear and we will not pursue it further.

Until now, we have used the same parameter τ for dropping in line 5 and line 10 of the algorithm. Dropping in L twice with the same parameter as for W0 may result in a preconditioner that is quite sparse and hence poor, so it may be advantageous to use different values for τ in lines 5 and 10. Using $0.1 \cdot \tau$ instead of τ in line 5 of W0, we obtain W1. Similarly, we obtain S1 if we use the standard strategy and drop with $0.1 \cdot \tau$ in line 5 and with τ in line 10. Both S1 and W1 use a denser w^L in the calculations of line 7, so their setup times are longer than for S0 and W0, but the resulting preconditioners are somewhat better. Nevertheless, these gains are usually not so significant as to justify longer setup times, in particular as W0 effects similar improvements. We summarize the use of τ in these dropping rules in Table 1. As mentioned before, the standard dropping rules S0 as well as S1 keep only the largest p elements by absolute value of w^L and w^U, whereas the weighted dropping rules W0 and W1 keep only those p elements whose weights are largest by absolute value.

TABLE 1

The use of τ in the dropping rules for L of ILUTP. The element w_i^L of w^L with index i is dropped if either of the conditions above is true.

Preconditioner	Dropping rule in line 5	Dropping rule in line 10
S0	$\|w_i^L\| < \tau$	none
S1	$\|w_i^L\| < 0.1 \cdot \tau$	$\|w_i^L\| < \tau \cdot \|\|w^L\|\|$
W0	$\|w_i^L\| < \tau$	$\|v_i^L\| < \tau \cdot \|\|v^L\|\|$
W1	$\|w_i^L\| < 0.1 \cdot \tau$	$\|v_i^L\| < \tau \cdot \|\|v^L\|\|$

4. Numerical results. We examined 163 matrices arising from different applications which we obtained from the University of Florida Sparse Matrix Collection [7] and the Matrix Market [10]. We tested these matrices using ILUPACK [4] implemention of SPARSKIT [12] using the standard routine for the ILUTP preconditioner as it is distributed, except that we used 1.0 as the permutation tolerance; i.e., we always pivoted. The preconditioners S1, W0, and W1 were constructed by minor modifications of the ILUTP preconditioner which this software provided. For testing these preconditioners, we used the routines implemented by ILUPACK as follows: If the matrix provided a right-hand side, then it was used. Otherwise, the right-hand side was constructed so that the exact solution of the resulting linear system is a vector consisting of all 1's. Next, the rows and columns were scaled and reordered using the PQ method (see [13]) as implemented by ILUPACK. Although other reordering methods also worked quite well, for example reverse Cuthill–McKee (see [8]), PQ seemed to yield the best overall results despite the fact that it was designed to be used with multilevel preconditioners. Hence, we report only on results for the PQ-reordering. Finally, GMRES(30) was used to attempt to solve the linear system. If the relative residual was decreased by a factor of $\sqrt{\text{eps}} \approx 3 \times 10^{-10}$ in at most 500 iterations, then the iteration was considered to be successful. The calculations were performed by an Athlon-XP 2.6 MHz CPU.

4.1. Summary of results for all matrices. All matrices were tested with $p = n$, n being the dimension of the matrix, and for various values of τ. Although we tested other combinations of p and τ, we restrict our attention to $p = n$ as the results are similar and $p = n$ is the default value of ILUPACK. Indeed, this usually yielded the best results. Hence, the preconditioners depended de facto only on the parameter τ. A notable disadvantage of this approach, particularly in large practical applications, is, however, that the memory requirement of the preconditioner is not known in advance, which can result in the calculation of the preconditioner terminating because of insufficient memory. This is remedied somewhat by ILUPACK requiring the user to provide an estimate of the required memory in advance. In any case, for each matrix and preconditioner we tested at least 20 different values of τ.

We attempted to choose the largest value of τ in order to obtain the sparsest preconditioner possible and the largest such that no more than 30 and 10 iterations, respectively, were needed to calculate the solution. Let γ_I be the least amount of fill-in that is needed to obtain the required reduction of the residual in no more than I iterations. Hence, for each preconditioner, we recorded $\gamma_{\min} = \gamma_{500}$, γ_{30}, and γ_{10}. Recall that the fill-in γ is defined to be the ratio of the sum of nonzero elements of L and U to those of A, the matrix tested. Although recording the various values for the fill-in seems to suggest that we were primarily interested in constructing preconditioners which are as sparse as possible, this is not the case. In other words, the new dropping strategies often enable sparser preconditioners, but usually they also require fewer iterations to calculate a solution than the standard preconditioner having the same fill-in. In any case, we chose to restrict our attention to this parameter because it appears to be most objective. Simply comparing the performance of preconditioners for a fixed τ is unsatisfactory, as for a given τ the weighted preconditioners, especially W0, are generally sparser but may require more iterations, and thus the comparison is often not very meaningful. On the other hand, comparing the number of required iterations for preconditioners for a fixed γ is difficult, as sensible values for γ depend highly on the matrix, making any choice for γ subjective. Additionally, if one preconditioner is better than another based on the fill-in required to obtain a

certain number of iterations, then this preconditioner is usually better based on another criterion: the number of iterations needed for a fixed γ. Hence, only recording the minimal fill-in needed to obtain no more than I iterations seems to be sufficient.

We report on the computation times only as part of the more extensive results for selected matrices. The additional time required to implement the weighted dropping is roughly proportional to the average number of elements in the row of L. Hence, if the original matrix is quite sparse and the fill-in required by the preconditioner is not too high, then weighted dropping results only in a small increase of setup times. On the other hand, if the preconditioner needs to be fairly dense, then implementing weighted dropping can require a considerable amount of extra time. For W0, this increase almost never exceeds a factor of about 1.5 and is often much lower. For S1 and W1, the setup times are usually increased by a factor of less than 2. Nevertheless, this time can generally only be amortized if the preconditioner is used to solve several systems or if the iteration times are reduced significantly by using one of the new strategies.

Before going into detail, we give an overview of the matrices tested. The first group, which we call hb58, consists of matrices of the Harwell-Boeing collection having size larger than 500. It has previously been used as a standard testing set; see [13]. As this set includes matrices from a wide range of applications, it generally gives a good overview of the quality of a preconditioner. In our particular case, the results for the various preconditioners are fairly similar, which is probably due to the fact that the differences between the preconditioners in question appear to be more significant for large matrices and the matrices in hb58 are fairly small. Next, we tested a set of 31 matrices arising from applications in chemical engineering, which we refer to as chem. The set pde consists of 19 matrices arising from partial differential equations, fidap consists of the 42 matrices from the fidap set having at least dimension 500, and sim consists of 13 matrices from circuit and device simulation problems. A list of the matrices tested can be found in Table 2.

We do not report on results comparing these preconditioners to multilevel preconditioners because extensive results comparing these preconditioners to ILUTP can be found on the homepage of ILUSTAB [3]. For that matter, these preconditioners go in a somewhat different direction. Although they are more robust than these ILUTP-based preconditioners and can often be constructed to be significantly sparser, which makes them very attractive alternatives to ILUTP, they often require significantly longer setup times, frequently much more than the additional work the strategies presented here require. Hence, if one of the ILUTP-type preconditioners presented here performs well for a given matrix (or is expected to perform well), then using a multilevel preconditioner, which may be considerably more expensive, is not necessary unless severe memory restrictions are an issue.

Before giving detailed results for selected matrices, we will summarize our results for all matrices and preconditioners tested in Tables 3, 4, and 5. In this summary, we also denoted "NC" for "no convergence" if γ of the preconditioners tested required a fill-in γ of more than 8 to solve the system in the prescribed number of iterations. In this case, the ILUTP-type preconditioners tested do not appear to be suitable for this matrix. If just one preconditioner succeeded with fill-in less than 8, we used one of the other classifications. We indicate that a preconditioner is "infinitely" better than another if calculating the solution in the prescribed number of iterations was possible for one preconditioner but not the other. In this case, we allowed γ to become very large to ensure that we record only significant failures of one strategy in this category. In other words, if for one preconditioner we achieved the required number of iterations for a particular amount of fill-in, we attempted to find a suitable γ for the others and

TABLE 2
The 163 *matrices tested.*

hb58		chem		pde	fidap		sim
bp200	nnc666	bayer01	lhr34	coater 2	fidap003	fidap037	bcircuit
bp400	nnc1374	bayer02	lhr34c	garon1	fidap004	fidapm02	circuit1
bp600	orani678	bayer03		garon2	fidap006	fidapm03	circuit2
bp800	orsirr1	bayer04		goodwin	fidap007	fidapm07	circuit3
bp1000	orsirr2	bayer05		rim	fidap008	fidapm08	circuit4
bp1200	orsreg1	bayer06		graham	fidap009	fidapm09	hcircuit
bp1400	pde2961	bayer07		raefsky1	fidap010	fidapm10	igbt3
bp1600	pores2	bayer08		raefsky2	fidap011	fidapm11	memplus
fs541 1	pores3	bayer09		raefsky3	fidap012	fidapm13	nmos3
fs541 2	psmigr1	bayer10		raefsky4	fidap013	fidapm15	onetone1
fs541 3	psmigr2	epb0		raefsky5	fidap014	fidapm29	onetone2
fs541 4	psmigr3	epb1		venkat01	fidap015	fidapm33	scircuit
fs680 1	saylr3	epb2		venkat25	fidap018	fidapm37	twotone
fs680 2	saylr4	epb3		venkat50	fidap019		
fs680 3	sherman1	hydr1		swang1	fidap020		
fs 760 1	sherman2	radfr1		swang2	fidap021		
fs 760 2	sherman3	rdist1		wang2	fidap022		
fs 760 3	sherman4	rdist2		wang 4	fidap023		
gemat11	sherman5	rdist3a			fidap024		
gemat12	shl0	lhr01			fidap025		
gre512	shl200	lhr02			fidap026		
gre1107	shl400	lhr04			fidap027		
jpwh991	steam2	lhr04c			fidap028		
lns511	watt1	lhr07			fidap029		
lns 3937	watt2	lhr07c			fidap031		
lnsp511	west0655	lhr17			fidap032		
lnsp3937	west0989	lhr17c			fidap033		
mahindas	west1505	lhr34			fidap035		
mcfe	west2021	lhr34c			fidap036		

TABLE 3
Summary of numerical results. Performance of $W0$ *in comparison to* $S0$.

		S0 better by				ND	W0 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	3	2	4	41	4	2	0	0	2
	γ_{30}	0	0	1	4	40	9	1	0	0	3
	γ_{10}	0	0	2	0	49	3	0	0	0	4
chem	$\gamma_{\min}$	0	2	1	2	17	5	1	0	0	3
	γ_{30}	0	1	0	3	22	1	0	1	0	3
	γ_{10}	0	0	0	0	26	1	0	0	0	4
pde	$\gamma_{\min}$	0	1	0	0	14	1	1	0	0	2
	γ_{30}	1	1	2	1	8	1	1	0	0	4
	γ_{10}	0	1	1	2	8	1	0	0	0	8
fidap	$\gamma_{\min}$	0	3	1	3	17	10	3	1	0	4
	γ_{30}	0	2	5	3	15	11	1	1	0	4
	γ_{10}	0	0	0	2	30	4	1	1	0	4
sim	$\gamma_{\min}$	0	1	0	1	7	0	2	1	0	1
	γ_{30}	0	0	0	0	9	0	3	0	0	1
	γ_{10}	0	0	0	0	6	2	4	0	0	1
Total	$\gamma_{\min}$	0	10	4	10	96	20	9	2	0	12
	γ_{30}	1	4	8	11	94	22	6	2	0	15
	γ_{10}	0	1	3	4	119	11	5	1	0	21

classified it. If the performance of two preconditioners was very similar (i.e., $\pm 10\%$) we indicate this by "ND," meaning "no difference."

TABLE 4
Summary of numerical results. Performance of $S1$ in comparison to $S0$.

		S0 better by				ND	S1 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	0	6	10	26	8	2	4	0	2
	γ_{30}	0	1	0	4	34	11	1	4	0	3
	γ_{10}	0	2	1	1	38	8	3	1	0	4
chem	$\gamma_{\min}$	0	4	3	3	13	3	1	1	0	3
	γ_{30}	0	2	1	3	19	1	2	0	0	3
	γ_{10}	0	0	2	1	23	1	0	0	0	4
pde	$\gamma_{\min}$	0	3	1	3	6	2	0	2	0	2
	γ_{30}	0	1	1	3	10	0	0	0	0	4
	γ_{10}	0	2	0	1	6	0	2	0	0	8
fidap	$\gamma_{\min}$	0	5	5	4	12	5	1	6	0	4
	γ_{30}	0	9	6	6	11	3	1	2	0	4
	γ_{10}	0	3	7	5	19	1	2	1	0	4
sim	$\gamma_{\min}$	0	3	0	2	4	1	2	0	0	1
	γ_{30}	0	2	0	0	6	2	2	0	0	1
	γ_{10}	0	0	0	0	8	2	1	1	0	1
Total	$\gamma_{\min}$	0	15	15	22	61	19	6	13	0	12
	γ_{30}	0	15	8	16	82	17	6	6	0	15
	γ_{10}	0	7	10	8	97	12	8	3	0	21

TABLE 5
Summary of numerical results. Performance of $W1$ in comparison to $S0$.

		S0 better by				ND	W1 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	1	4	14	25	7	3	2	0	2
	γ_{30}	0	0	2	6	35	8	3	1	0	3
	γ_{10}	0	2	0	2	42	6	2	0	0	4
chem	$\gamma_{\min}$	0	2	3	3	13	5	2	0	0	3
	γ_{30}	0	0	2	2	18	4	2	0	0	3
	γ_{10}	0	0	0	0	24	3	0	0	0	4
pde	$\gamma_{\min}$	0	2	1	3	5	2	1	3	0	2
	γ_{30}	0	1	1	2	8	3	0	0	0	4
	γ_{10}	0	2	1	0	5	2	1	0	0	8
fidap	$\gamma_{\min}$	0	1	0	1	15	13	2	6	0	4
	γ_{30}	0	1	4	4	18	7	1	3	0	4
	γ_{10}	0	0	0	2	29	5	1	1	0	4
sim	$\gamma_{\min}$	0	1	1	3	3	1	3	0	0	1
	γ_{30}	0	0	1	1	5	1	4	0	0	1
	γ_{10}	0	0	0	0	6	2	3	1	0	1
Total	$\gamma_{\min}$	0	7	9	24	61	28	11	11	0	12
	γ_{30}	0	2	10	15	84	23	10	4	0	15
	γ_{10}	0	4	1	4	106	18	7	2	0	21

A first quick glance at these results suggests that the weighted dropping strategy is no improvement over the standard one. A more careful inspection of the raw data, however, reveals that the standard strategy is often better than the weighted strategy whenever the preconditioner is particularly sparse, often even sparser than the original matrix. Not surprisingly, this suggests that using additional dropping for a preconditioner which is already quite sparse is not a good idea. In practice, this does not really pose a problem: for these matrices, preconditioning is easy and ILUTP with the standard dropping strategy does the job satisfactorily, so there is usually no need to find a better preconditioner. Furthermore, because these preconditioners are

TABLE 6
Summary of numerical results. Performance of $W0$ in comparison to $S0$ with $\gamma \geq 1.5$.

		S0 better by				ND	W0 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	0	0	0	6	3	2	0	0	2
	γ_{30}	0	0	0	0	8	8	1	0	0	3
	γ_{10}	0	0	0	0	26	3	0	0	0	4
chem	$\gamma_{\min}$	0	0	0	1	9	2	0	0	0	3
	γ_{30}	0	1	0	1	14	1	0	0	0	3
	γ_{10}	0	0	0	0	22	1	0	0	0	4
pde	$\gamma_{\min}$	0	0	0	0	1	1	1	0	0	2
	γ_{30}	1	0	2	0	3	1	0	0	0	3
	γ_{10}	0	1	1	1	4	0	0	0	0	8
fidap	$\gamma_{\min}$	0	0	0	0	3	7	3	1	0	4
	γ_{30}	0	1	0	0	9	7	1	1	0	4
	γ_{10}	0	0	0	1	23	4	1	1	0	4
sim	$\gamma_{\min}$	0	0	0	0	0	0	2	1	0	1
	γ_{30}	0	0	0	0	0	0	3	0	0	1
	γ_{10}	0	0	0	0	0	2	3	0	0	1
Total	$\gamma_{\min}$	0	0	0	1	19	13	8	2	0	12
	γ_{30}	1	2	2	1	34	17	5	1	0	15
	γ_{10}	0	1	1	2	75	10	4	1	0	21

TABLE 7
Summary of numerical results. Performance of $S1$ in comparison to $S0$ with $\gamma \geq 1.5$.

		S0 better by				ND	S1 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	0	0	0	3	5	0	3	0	2
	γ_{30}	0	1	0	0	7	5	1	3	0	3
	γ_{10}	0	1	0	0	18	8	2	0	0	4
chem	$\gamma_{\min}$	0	1	0	1	8	2	0	0	0	3
	γ_{30}	0	1	0	2	13	0	1	0	0	3
	γ_{10}	0	0	2	1	19	1	0	0	0	4
pde	$\gamma_{\min}$	0	0	0	0	1	2	0	0	0	2
	γ_{30}	0	0	1	1	5	0	0	0	0	4
	γ_{10}	0	1	0	1	4	0	1	0	0	8
fidap	$\gamma_{\min}$	0	3	3	1	4	1	0	2	0	4
	γ_{30}	0	5	3	3	3	3	0	2	0	4
	γ_{10}	0	2	6	4	15	0	2	1	0	4
sim	$\gamma_{\min}$	0	0	0	0	1	0	2	0	0	1
	γ_{30}	0	0	0	0	1	2	0	0	0	1
	γ_{10}	0	0	0	0	2	1	1	1	0	1
Total	$\gamma_{\min}$	0	4	3	2	17	10	2	5	0	12
	γ_{30}	0	7	4	6	29	10	2	5	0	15
	γ_{10}	0	4	8	6	58	10	6	2	0	21

quite sparse, the standard strategy is significantly better only in a relative sense. In an absolute sense, the difference between the respective γ's for both strategies is often quite small, usually no more than 0.2. Hence, using a new strategy even when the standard strategy works better does not result in significantly worse results. In any case, it is generally best simply not to use a strategy other than the standard strategy for preconditioners constructed to be quite sparse. As a rule of thumb, the standard strategy should be used if τ is larger than approximately 10^{-2}. For these reasons, the results presented in the second set of tables (Tables 6, 7, and 8) are more interesting because here the results are restricted to matrices for which we have $\gamma \geq 1.5$ for S0.

TABLE 8
Summary of numerical results. Performance of W1 in comparison to S0 with $\gamma \geq 1.5$.

		S0 better by				ND	W1 better by				NC
		∞	>30%	20–30%	10–20%		10–20%	20–30%	>30%	∞	
hb58	$\gamma_{\min}$	0	0	0	0	3	5	2	1	0	2
	γ_{30}	0	0	0	0	7	7	3	0	0	3
	γ_{10}	0	0	0	0	23	6	0	0	0	4
chem	$\gamma_{\min}$	0	0	1	0	8	3	0	0	0	3
	γ_{30}	0	0	2	1	10	3	1	0	0	3
	γ_{10}	0	0	0	0	21	2	0	0	0	4
pde	$\gamma_{\min}$	0	0	0	0	0	2	0	1	0	2
	γ_{30}	0	0	1	1	3	2	0	0	0	4
	γ_{10}	0	1	1	0	3	1	1	0	0	8
fidap	$\gamma_{\min}$	0	0	0	0	4	7	0	3	0	4
	γ_{30}	0	0	1	0	10	5	1	2	0	4
	γ_{10}	0	0	0	1	22	5	1	1	0	4
sim	$\gamma_{\min}$	0	0	0	0	0	0	3	0	0	1
	γ_{30}	0	0	0	0	0	0	3	0	0	1
	γ_{10}	0	0	0	0	0	2	2	1	0	1
Total	$\gamma_{\min}$	0	0	1	0	15	17	5	5	0	12
	γ_{30}	0	0	4	2	30	17	8	2	0	15
	γ_{10}	0	1	1	1	69	16	4	2	0	21

The results recorded in these tables seem to indicate the following:

- The strategies W0, S1, and W1 are often better than S0 for harder matrices. W0 already improves ILUTP, but W1 even more so. However, this comes at the cost of even higher setup times, and therefore it is probably better to use W0 than W1 as a general rule. Although S1 can improve ILUTP as much as W1, it is worse than S0 more often. Considering that the setup times for S1 and W1 are almost equal, W1 should be preferred. It appears that dropping fewer elements prior to calculation and some afterwards may be worthwhile if weighted dropping is performed (as for W1), but much less so if standard dropping is used (as for S1).
- As fewer elements are dropped, the differences between the various dropping strategies become less significant. This is not surprising as all dropping strategies applied to a given vector will yield more similar results if fewer elements are dropped.
- Weighted dropping appears to work particularly well for the matrices in the sets fidap and sim. The set hb58 contains a significant number of matrices which are fairly easy to precondition; hence for most matrices either strategy works well. For the harder ones, however, W0 is better, and S0 is better for the easier ones. For the set chem there are no significant differences, although for some matrices arising from chemical engineering problems, in particular the west matrices (which are recorded as part of hb58), W0 works quite well. Finally, for pde the standard strategy appears to be somewhat better. However, this set is characterized by a particularly large number of failures, suggesting that a different preconditioner should be used.

Next, we summarize the results of the specific matrices listed in Table 9 characterizing the observations mentioned in Table 10. Here, N refers to the dimension of the matrix, NNZ to the number of nonzero elements, and $\delta = \frac{\text{NNZ}}{\text{N}}$ is the average number of elements per row or column. These matrices were chosen to give an overview of the differences in performance. Hence, these results do not include matrices

TABLE 9
Matrices selected for specific results.

Matrix	N	NNZ	δ
bp1000	822	4 661	5.7
gemat11	4 929	33 185	6.7
lnsp3937	3 937	25 407	6.5
west1505	1 505	5 445	3.6
bayer10	13 436	71 594	5.3
epb2	25 228	175 027	6.9
lhr17c	17 576	381 975	21.7
goodwin	7 320	324 772	44.7
garon1	3 175	84 723	26.7
wang4	26 068	177 196	6.8
fidap004	1 601	32 287	20.2
fidap033	1 733	20 315	11.7
fidapm13	3 549	71 975	20.3
onetone2	36 057	222 596	6.2
scircuit	170 998	958 936	5.6

TABLE 10
Numerical results for selected matrices.

Matrix	$\gamma_{\min}$				γ_{30}				γ_{10}			
	S0	W0	S1	W1	S0	W0	S1	W1	S0	W0	S1	W1
bp1000	0.85	1.05	0.97	1.04	1.18	1.34	1.26	1.37	1.71	1.71	1.72	1.73
gemat11	2.12	1.85	1.96	1.87	2.49	2.22	2.16	2.13	4.05	3.71	4.02	3.74
lnsp3937	1.24	1.71	1.53	1.41	2.34	2.17	2.25	2.28	3.30	3.04	3.39	3.11
west1505	2.27	1.60	1.41	1.61	2.27	1.81	1.57	1.71	2.77	2.58	2.23	2.49
bayer10	3.01	2.65	2.59	2.29	3.01	2.77	2.77	2.77	3.49	3.29	3.30	3.28
ebp2	0.19	0.19	0.23	0.21	2.27	2.45	2.63	2.63	7.72	7.94	7.57	7.62
lhr17c	5.42	4.95	5.04	4.97	5.42	4.95	5.13	4.97	5.53	5.46	5.27	5.48
goodwin	3.87	2.93	3.34	2.67	4.12	3.64	4.49	3.44	5.93	5.73	6.50	5.30
garon1	0.72	0.72	0.44	0.39	2.16	2.18	2.03	2.02	7.65	9.66	8.76	9.37
wang4	0.33	0.32	0.59	0.44	2.28	2.77	2.86	2.85	failed	failed	failed	failed
fidap004	1.17	1.26	1.18	1.18	1.90	2.54	2.54	2.36	3.99	4.49	4.32	4.49
fidap033	0.21	0.21	0.20	0.20	3.92	1.45	1.07	1.24	4.56	2.81	1.94	2.47
fidapm13	2.60	2.04	1.40	1.43	2.98	2.41	2.41	2.11	3.82	3.30	3.87	3.06
onetone2	8.91	6.14	6.59	6.25	8.95	6.28	7.22	6.46	10.1	7.78	10.0	7.93
scircuit	4.10	3.04	4.05	3.10	4.10	3.04	4.05	3.13	5.26	3.85	4.05	3.96

for which the strategies produce similar results and underrepresent matrices for which S0 performs worse than the other strategies.

Finally, we present timing results in Table 11 for these matrices. For these results, we tested different values of τ in order to obtain similar values for γ for each preconditioner. We attempted to choose γ in order to obtain a fairly sparse preconditioner. All timing results are recorded in seconds. A result of "NM" (not measurable) indicates that the times measured were too short to be considered accurate, usually 0.01 seconds or less. As mentioned before, if a preconditioner is fairly sparse, then implementing the weighted strategy adds little to the setup time. Whenever the preconditioner is fairly dense (i.e., $\gamma \cdot \frac{\text{NNZ}}{\text{N}}$ fairly large), the weighted dropping strategy can increase the setup time significantly. In this case, using the weighted preconditioner is only worthwhile if the iteration times decrease sufficiently or if it is used for several systems.

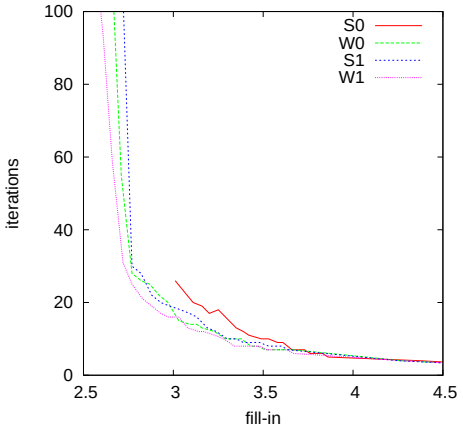
We conclude with a few graphical results in Figure 1 illustrating the improvements that can be obtained by using the weighted dropping strategies. These underline that

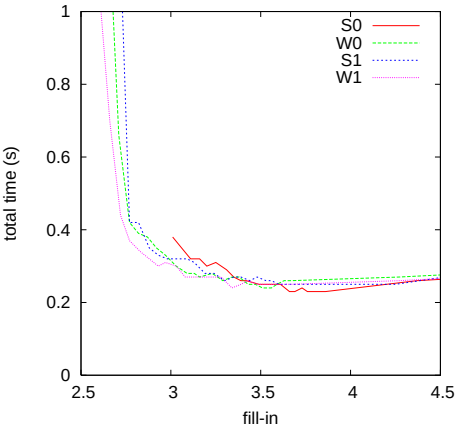
TABLE 11
Timing results for selected matrices.

Matrix	γ	Iterations				Setup time				Iteration time				Total time			
		S0	W0	S1	W1	S0	W0	S1	W1	S0	W0	S1	W1	S0	W0	S1	W1
bp1000	1.4	20	20	19	22	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
gemat11	2.5	30	23	22	21	0.03	0.05	0.07	0.06	0.10	0.07	0.06	0.08	0.13	0.12	0.13	0.13
lnsp3937	2.4	22	17	26	19	0.02	0.02	0.03	0.03	0.05	0.03	0.05	0.04	0.07	0.05	0.08	0.07
west1505	2.3	15	13	10	12	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM	NM
bayer10	3.1	20	14	16	12	0.14	0.16	0.17	0.16	0.18	0.12	0.14	0.11	0.32	0.28	0.31	0.27
ebp2	3.0	21	22	23	23	0.20	0.22	0.30	0.30	0.41	0.45	0.45	0.45	0.61	0.67	0.75	0.75
lhr17c	5.5	10	10	14	10	2.07	2.41	2.68	2.59	0.36	0.37	0.50	0.36	2.43	2.78	3.18	2.95
goodwin	3.9	53	23	53	18	1.04	1.51	2.23	2.10	1.24	0.54	1.23	0.42	2.28	2.05	3.46	2.52
garon1	1.9	31	35	29	30	0.09	0.12	0.28	0.28	0.15	0.18	0.14	0.15	0.24	0.30	0.42	0.43
wang4	1.6	57	79	60	82	0.14	0.15	0.18	0.18	0.88	1.23	0.95	1.28	1.02	1.38	1.13	1.46
fidap004	2.0	24	53	55	51	0.03	0.04	0.06	0.06	0.04	0.09	0.10	0.09	0.07	0.13	0.16	0.15
fidap033	2.0	79	18	9	15	0.02	0.03	0.03	0.03	0.09	0.02	0.01	0.02	0.11	0.05	0.04	0.05
fidapm13	2.8	61	17	20	12	0.08	0.11	0.13	0.14	0.31	0.09	0.10	0.06	0.39	0.20	0.23	0.20
onetone2	5.6	11	8	11	8	0.90	1.35	1.70	1.56	0.33	0.24	0.34	0.24	1.23	1.59	2.04	1.80
scircuit	4.1	17	7	8	7	3.63	5.65	7.98	8.05	3.06	1.00	1.14	0.99	6.69	6.65	9.12	9.04

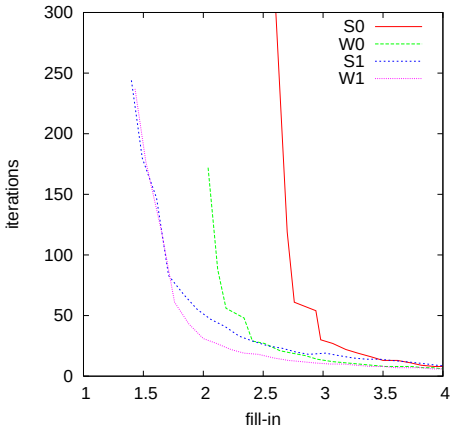
the preconditioners using weighted dropping strategies can often be constructed to be sparser than their standard counterparts and that, if a particularly sparse preconditioner is required, the iterations needed will be lower for a weighted preconditioner, but the total time to solve a system usually cannot be reduced if fill-in is chosen optimally with regard to reducing the total time. On the other hand, as the number of iterations is often reduced more effectively by a weighted strategy, the total times will usually be less if several systems are solved.

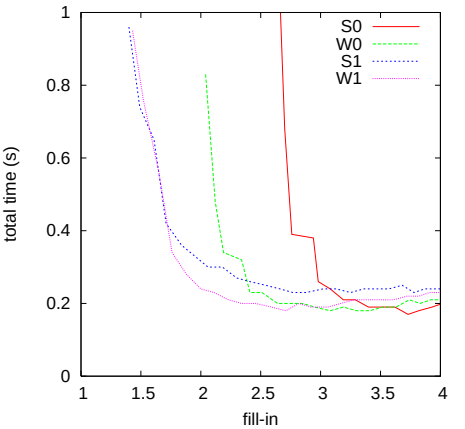
5. Conclusion. The weighted dropping strategy W0 presented in this paper is an alternative to the standard dropping method S0 usually employed in calculating the ILUTP preconditioner. W0 often yields a better preconditioner, especially for more difficult matrices. For matrices which are easy to precondition, results are often worse, so care must be taken when choosing to implement the weighted strategy. The setup times for the preconditioner are generally somewhat longer, but this additional time is often compensated for by shorter iteration times, making this strategy particularly worthwhile if the same linear system is solved for several different right-hand sides. On the other hand, the setup times for the other strategies S1 and W1 are significantly longer, making them attractive alternatives to S0 and W0 only in exceptional cases.



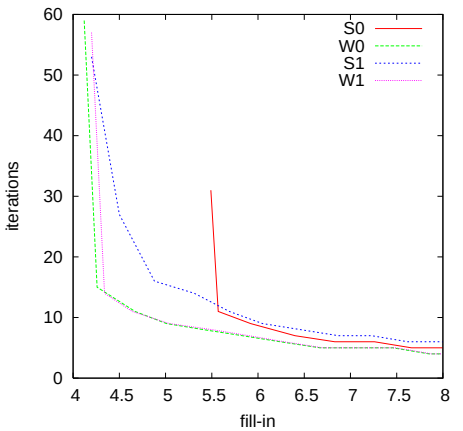


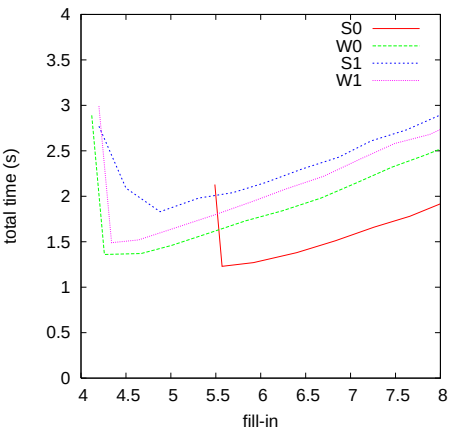
bayer10





fidapm13





onetone2

FIG. 1. *Comparison of dropping strategies for selected matrices.*

REFERENCES

[1] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[2] M. Bollhöfer, *A robust ILU with pivoting based on monitoring the growth of inverse factors*, Linear Algebra Appl., 338 (2001), pp. 201–218.

[3] M. Bollhöfer, *A robust and efficient ILU that incorporates the growth of the inverse triangular factors*, SIAM J. Sci. Comput., 25 (2003), pp. 86–103.

[4] M. Bollhöfer and Y. Saad, *ILUPACK—Preconditioning Software Package*, release V1.0, http://www.math.tu-berlin.de/ilupack/ (2004).

[5] M. Bollhöfer and Y. Saad, *Multilevel Preconditioners Constructed from Inverse-Based ILUs*, manuscript, 2004.

[6] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.

[7] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices, ftp://ftp.cise.ufl.edu/pub/faculty/davis/matrices.

[8] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices*, Monographs on Numerical Analysis, Clarendon Press, Oxford, UK, 1986.

[9] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Sci. Comput., 25 (2003), pp. 716–728.

[10] National Institute of Standards, *Matrix Market*, http://math.nist.gov/MatrixMarket/.

[11] Y. Saad, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[12] Y. Saad, *SPARSKIT*, version 2, http://www-users.cs.umn.edu/~saad/software/SPARSKIT/sparskit.html (1999).

[13] Y. Saad, *Complete Pivoting ILU: A Multilevel Approach*, ftp://ftp.cs.umn.edu/dept/users/saad/reports/index.html (2003).

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[15] Y. Saad, *pARMS: Parallel Algebraic Recursive Multilevel Solvers*, version 1.1, http://www-users.cs.umn.edu/~saad/software/pARMS/ (2004).

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1438–1457
© 2006 Society for Industrial and Applied Mathematics

PRECONDITIONING LANCZOS APPROXIMATIONS TO THE MATRIX EXPONENTIAL*

JASPER VAN DEN ESHOF† AND MARLIS HOCHBRUCK†

Abstract. The Lanczos method is an iterative procedure to compute an orthogonal basis for the Krylov subspace generated by a symmetric matrix A and a starting vector v. An interesting application of this method is the computation of the matrix exponential $\exp(-\tau A)v$. This vector plays an important role in the solution of parabolic equations where A results from some form of discretization of an elliptic operator. In the present paper we will argue that for these applications the convergence behavior of this method can be unsatisfactory. We will propose a modified method that resolves this by a simple preconditioned transformation at the cost of an inner-outer iteration. A priori error bounds are presented that are independent of the norm of A. This shows that the worst case convergence speed is independent of the mesh width in the spatial discretization of the elliptic operator. We discuss, furthermore, a posteriori error estimation and the tuning of the coupling between the inner and outer iteration. We conclude with several numerical experiments with the proposed method.

Key words. matrix exponential, iterative methods, restricted rational approximation, matrix-free time integration methods, preconditioning

AMS subject classifications. 65F10, 65L05, 65N22

DOI. 10.1137/040605461

1. Introduction.

Using the matrix exponential operator in the numerical solution of large time-dependent problems has been an extensive area of research in the last few years. A key aspect of such exponential integrators is that they require the evaluation or approximation of the product of the exponential of the Jacobian with a vector. In this paper we consider the approximation of

$$y(\tau) = \exp(-\tau A)v, \qquad \|v\| = 1, \tag{1.1}$$

where A results from a finite difference or finite element discretization of an elliptic partial differential operator.

In 1978, Moler and van Loan [24] published their famous paper discussing nineteen dubious ways to compute the exponential of a matrix. Since then, Krylov subspace methods have been an important development towards tackling the problem (1.1) when the matrix A is very large and sparse. Hence the updated paper [23] mentions Krylov subspace methods as the "twentieth" method. For matrices A that stem from a self-adjoint elliptic partial differential equation, it has been shown [7, 8, 13] that Krylov approximations to $y(\tau)$ always yield superlinear error reduction with the superlinear decay starting after the number of steps exceeds $\|A\|^{1/2}$. Unfortunately, numerical experiments typically show that these error bounds are fairly sharp for these applications. This means that the computational complexity grows like $n^{1+1/d}$ for a uniform discretization of an elliptic operator on a d-dimensional cube with n spatial grid points. The aim of this paper is to present an algorithm which allows us to compute the approximation with $\mathcal{O}(n)$ operations. Speeding up the Lanczos process is

*Received by the editors March 22, 2004; accepted for publication (in revised form) January 11, 2005; published electronically January 27, 2006. This work was supported by the Deutsche Forschungsgemeinschaft.

http://www.siam.org/journals/sisc/27-4/60546.html

†Mathematisches Institut, Heinrich-Heine Universität Düsseldorf, Universitätsstr. 1, D–40225 Düsseldorf, Germany (eshof@am.uni-duesseldorf.de, marlis@am.uni-duesseldorf.de).

achieved by a preconditioned operation. The emphasis in this paper is on the computation of the vector in (1.1), and we will not discuss details of the spatial discretization resulting in the matrix A. For ease of presentation, throughout this paper we assume that A is symmetric and positive semidefinite, although most of the techniques apply to discretizations of sectorial operators as well. The demand that the matrix is positive definite is not an essential restriction since indefinite matrices can easily be handled by shifting the matrix and multiplying the result with a suitable factor.

The paper is organized as follows. In section 2 we discuss the motivation for this work. The central idea in this paper is to transform the spectrum in such a way that convergence is much faster. This leads to a new method, which is proposed in section 3, consisting of an inner and outer iteration. The convergence behavior of the new method is treated in section 3.1, and termination strategies are the subject of section 4. In section 5 we discuss the tuning of the coupling between the inner and outer iteration in such a way that the overall procedure is as efficient as possible. We conclude by giving some examples of typical numerical experiments in section 6.

We have learned recently that the method proposed in this paper has been studied earlier by Moret and Novati in [25]. In that work the authors derive the method from a different point of view and consider the more general problem of approximating the matrix exponential of sectorial operators. Unfortunately, this makes it much more challenging to give insight into the behavior and properties of the method. In that sense the results in this paper supplement the work in [25] since we consider Hermitian matrices, which allows us to give a more thorough discussion. Moreover, the central question in the present paper is on how to include a preconditioner efficiently into Lanczos approximation methods for the matrix exponential, whereas in [25] the linear systems are solved with a direct method.

2. Motivation. When approximating the vector $y(\tau)$ in (1.1) for high dimensional matrices A, it becomes essential to exploit the sparsity of the matrix and the fact that only the action on a given vector is required. A standard idea is to use a polynomial approximation to the exponential function. In most cases this approach can be divided into two parts. First, there is the construction of a basis for the m-dimensional Krylov subspace. This part can often be summarized by the following matrix formulation:

$$AV_m = V_m T_m + \beta_m v_{m+1} e_m^*, \quad \text{where} \quad V_m e_1 = v. \tag{2.1}$$

The matrix T_m is upper Hessenberg, and the vector e_j denotes the jth column of the identity matrix. The second ingredient is the approximation to the product of the matrix exponential, and the vector v and is usually given by

$$y(\tau) \approx y_m(\tau) = V_m \exp(-\tau T_m) e_1. \tag{2.2}$$

Different choices for T_m lead to different polynomial approximations. The simplest choice is when T_m is defined by

$$T_m e_j = e_{j+1} \quad \text{for } j < m \quad \text{and} \quad T_m e_m = 0.$$

In this case the polynomial approximation (2.2) coincides with the standard Taylor series expansion of the matrix exponential [23, section 3]. Other attempts have been based on Chebyshev polynomials, e.g., [38], or, in the non-Hermitian case, on Faber polynomials in the complex plane, e.g., [29].

In this paper we focus on methods where (2.1) is constructed by means of the Lanczos method in which case the columns of the matrix V_m form an orthogonal basis for the Krylov subspace. The resulting method has been discussed by several authors and analyzed in various papers; see, for instance, [6, 7, 8, 10, 13, 33, 42] and [23] for a more extensive bibliography. We will refer to the vector $y_m(\tau)$ in this case as the [illegible] approximation. The advantage here is that these Lanczos approximations have a potential for exploiting the discrete nature of the spectrum of A: By interpolating the exponential in eigenvalues of A, the required degree of the polynomial approximation can be much smaller than for Chebyshev approximations where, for example, a uniform approximation is constructed on an interval containing the spectrum of A. Moreover, no a priori knowledge is required about the spectral radius of the method.

Unfortunately, when A is a discrete representation of the elliptic operator, Lanczos approximations can have two distinct drawbacks. First of all, the Lanczos method often is unsuccessful in exploiting the discrete character of the spectrum, as mentioned above. One example is when A is the usual finite difference/element discretization of a one-dimensional Poisson operator with Dirichlet boundary conditions. In this case, the eigenvalues of A coincide with the roots of a shifted and scaled Chebyshev polynomial and the Lanczos method cannot identify any eigenvalue of A quickly, e.g., [21, section 4.1]. This explains why we often observe for parabolic problems that Lanczos approximations do not require significantly fewer multiplications with the matrix than use of a polynomial approximation based on Chebyshev polynomials, whereas they do have some overhead cost compared to this method. A related issue is that convergence analysis shows that the required number of steps is proportional to $\|A\|^{1/2}$. This means that the number of operations grows faster than linearly in the number n of spatial grid points. This should be compared with the solution of an elliptic problem which can be accomplished in $\mathcal{O}(n)$ flops using multigrid techniques.

An extension of the Lanczos approximation is to somehow precondition the iterative procedure as is done when solving linear systems. This is the aim of this paper. Other attempts in this direction have been made in [4]. The main idea there is to choose a matrix $M \approx A$ (the preconditioner) for which $\exp(-\tau M)v$ is cheap to compute and to combine this with a generalized Runge–Kutta method for the solution of the differential equation for the defect. It should also be mentioned that for the solution of parabolic partial differential equations, the idea of preconditioning the time-differencing is not new. Two important developments in this area are parabolic multigrid methods [12] and methods based on waveform relaxation, e.g., [15].

3. Preconditioning Lanczos approximations. The exponential function is a quickly decaying function. This implies that the vector $\exp(-\tau A)v$ is mostly determined by the smallest eigenvalues and their corresponding invariant subspaces. Recall again that the Lanczos method does not necessarily utilize this since the first eigenvalues (and their eigenvectors) are difficult to find. This suggests that we should transform the spectrum in such a way that the Lanczos method can quickly find these eigenpairs. The simplest idea is to apply the Lanczos method to the matrix $(I+\gamma A)^{-1}$ (with $\gamma > 0$) which emphasizes the eigenvalues of importance. The Lanczos relation for the spectrally transformed matrix reads

$$(3.1) \quad (I+\gamma A)^{-1} V_m = V_m T_m + \beta_m v_{m+1} e_m^*, \quad \text{where} \quad V_m e_1 = v \text{ and } V_m^* V_m = I.$$

We define the function

$$f_\gamma^\tau(t) = \exp((1-t^{-1})\tau/\gamma) \quad \text{for } t \in (0,1], \quad f_\gamma^\tau(0) = 0,$$

and note that $f_\gamma^\tau((I+\gamma A)^{-1}) = \exp(-\tau A)$. As an approximation to $y(\tau)$ we use

$$y_m(\tau) = V_m f_\gamma^\tau(T_m)e_1 = V_m \exp(-\tau \widetilde{T}_m)e_1, \quad \widetilde{T}_m = \frac{1}{\gamma}(T_m^{-1} - I).$$

It is well known that the Lanczos method usually quickly finds approximations to the eigenvalues that are in some sense nicely separated from the other eigenvalues in the spectrum. A quantitative measure that expresses this separation is the ratio between the eigenvalue spread and gap. For the invariant subspace that corresponds to the smallest eigenvalue, this quantity is given by

$$\frac{\lambda_n - \lambda_1}{\lambda_2 - \lambda_1}, \tag{3.2}$$

where λ_1, λ_2, and λ_n denote, respectively, the smallest, second smallest, and largest eigenvalue of A. This quantity appears in upper bounds on the convergence speed of the Lanczos method for finding eigenpairs; see, e.g., [17]. On the other hand, for the transformed system the ratio of the eigenvalue spread and gap of the invariant subspace corresponding to the smallest (positive) eigenvalue of the transformed system is bounded by

$$\frac{(1+\gamma\lambda_1)^{-1} - (1+\gamma\lambda_n)^{-1}}{(1+\gamma\lambda_1)^{-1} - (1+\gamma\lambda_2)^{-1}} \leq \frac{(1+\gamma\lambda_1)^{-1}}{(1+\gamma\lambda_1)^{-1} - (1+\gamma\lambda_2)^{-1}} = \frac{1+\gamma\lambda_2}{\gamma(\lambda_2 - \lambda_1)}.$$

The first expression follows directly from (3.2) by plugging in the transformed eigenvalues. When A stems from an appropriate discretization of an elliptic operator, then the eigenvalues λ_1 and λ_2 approach the eigenvalues of the elliptic operator when the accuracy of the spatial discretization increases. As a consequence this ratio does not become arbitrarily large when the size of the matrix A increases as a result of decreasing the mesh width in the spatial discretization. For example, for the already mentioned one-dimensional Poisson operator with Dirichlet boundary conditions the spread/gap ratio for the smallest invariant subspace of the continuous operator is unbounded since the spread of the spectrum is infinite. On the other hand, the spread/gap ratio for the transformed spectrum can be bounded by $(1+\gamma 4\pi^2)/(3\gamma\pi^2)$.

The practical importance of these observations is that a good approximation to the leading invariant subspaces can in turn speed up the convergence of the method in a way that is analogous to what is often witnessed for the conjugate gradient method [41]. Our hope is that, due to the spectral transformation, the Lanczos method is more effectively able to exploit the discrete nature of an important part of the spectrum.

Finally, we mention that in our [illegible] Lanczos method we have to invert a fairly standard elliptic operator in every step. This can be done efficiently by very effective preconditioned methods that can be found in the literature. This is exploited in our numerical experiments in section 6. The idea of this paper is to incorporate preconditioning in the Lanczos method by using it to accomplish a spectral transformation that results in more favorable convergence speed at the cost of an inner-outer iteration. We will further investigate this in the coming sections.

3.1. A priori error estimation. The Lanczos approximations are given by

$$V_m f_\gamma^\tau(T_m)e_1 = p((I+\gamma A)^{-1})v = (I+\gamma A)^{-(m-1)}q(A)v, \quad p, q \in \Pi_{m-1}.$$

The space Π_{m-1} is the space of all polynomials of degree $m-1$ or less. Our method can be alternatively characterized as constructing iterates from the class of [illegible]

rational approximations defined by

$$\mathcal{R}_i^j = \left\{ p(t)(1+\gamma t)^{-i} \,|\, p \in \Pi_j \right\}.$$

This shows that the Lanczos approximations now are matrix rational approximations to $\exp(-\tau t)$ with all poles fixed at $-1/\gamma$. In the present section we will exploit this viewpoint to get more insight into the impact of our spectral transformation on the convergence speed, and we will neglect the discrete nature of the spectrum. We define

$$E_i^j(\gamma) := \inf_{r \in \mathcal{R}_i^j} \sup_{t \geq 0} |r(t) - \exp(-t)|. \tag{3.3}$$

We can now easily derive the following result.

LEMMA 3.1. [illegible] $A - \mu I$ [illegible]

$$\|V_m f_\gamma^\tau(T_m) e_1 - \exp(-\tau A) v\| \leq 2 \exp(-\tau\mu) E_{m-1}^{m-1}(\widetilde{\gamma}) \quad [illegible] \quad \widetilde{\gamma} = \frac{\gamma}{\tau(1+\gamma\mu)}.$$

[illegible]. The eigenvalues of A are all larger than μ, and as a direct consequence of [33, Lemma 4.1] we have that

$$\begin{aligned}
\|V_m f_\gamma^\tau(T_m) e_1 - \exp(-\tau A) v\| &\leq 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0, (1+\gamma\mu)^{-1}]} |f_\gamma^\tau(t) - p(t)| \\
&= 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,1]} \left| f_\gamma^\tau\left(\frac{t}{1+\gamma\mu}\right) - p(t) \right| \\
&= 2 \inf_{p \in \Pi_{m-1}} \sup_{t \in (0,1]} \left| \exp(-\mu\tau) \exp\left(\left(1 - \frac{1}{t}\right) \frac{\tau(1+\gamma\mu)}{\gamma} \right) - p(t) \right| \\
&= 2 \exp(-\mu\tau) E_{m-1}^{m-1}(\widetilde{\gamma}). \qquad \square
\end{aligned}$$

It is important to stress that this error estimate is independent of the norm of A and only the first (smallest) eigenvalue of A plays a modest role in the form of μ. We also see that the restriction of A to positive semidefinite matrices is too stringent since reasonable upper bounds are obtained if the spectrum is bounded from below by a modest constant.

A priori error bounds for the proposed method can be obtained by estimating $E_{m-1}^{m-1}(\widetilde{\gamma})$. A good choice for $\widetilde{\gamma}$ automatically leads to a good choice for γ (if τ and a reasonable estimate for μ are known). Therefore we want to derive upper bounds to (3.3) where we, without loss of generality, assume that $\tau = 1$ and $\mu = 0$, which leads to $\widetilde{\gamma} = \gamma$. Our attempts for bounding this quantity have resulted in upper bounds which are too pessimistic. However, some insight can be obtained by considering the asymptotic situation for $m \to \infty$. The use of the class $\mathcal{R}_j^{j-1}$ to approximate $\exp(-t)$ for $t \in [0, \infty)$ is discussed by Saff, Schönhage, and Varga [34]. The authors' analysis implies that the best shift is given by $\gamma = 1/j$, which was shown to lead to geometric convergence with an asymptotic decay rate between $1/5.828$ and $1/2$. This result was later sharpened by Andersson [1], who showed the following result.

THEOREM 3.2 (Andersson [1]). [illegible] γ [illegible] $\sqrt{2}/j$ [illegible]

$$\lim_{j \to \infty} \left(E_j^j(\sqrt{2}/j) \right)^{1/j} = \frac{1}{\sqrt{2}+1}.$$

Hence, the value of γ should ideally be chosen to be inversely proportional to the number of Lanczos steps. Our shift-and-invert strategy requires a fixed a priori chosen value of γ. In this case we cannot expect linear convergence. Based on work in [34] we can show the following result.

THEOREM 3.3. [illegible] $0 < \gamma < 1$ [illegible]

$$\left(E_j^{j-1}(\gamma)\right)^{1/\sqrt{j}} \leq \kappa_j, \quad \text{[illegible]} \quad \lim_{j\to\infty} \kappa_j = \exp\left(-\sqrt{\frac{2(1-\gamma)}{\gamma}}\right).$$

[illegible]. Equations (3.4) and (3.14) in [34] are

$$E_j^{j-1}(\gamma) \leq \sqrt{2}\rho_{j-1}(1/\gamma), \tag{3.4}$$

where

$$\rho_{j-1}(1/\gamma)^2 < \frac{1}{2}\exp(1/\gamma)\int_0^\infty \exp(-s/\gamma)\left(\frac{s}{s+1}\right)^{2j-2}\omega_j(s)ds$$

and

$$\omega_j(t) = \frac{(j+1)(2j+1)t^2}{(1+2t)(1+t)} + \frac{(4j+5)t^4}{(1+2t)^2(1+t)} + \frac{4t^6}{(1+2t)^3(1+t)}.$$

We have

$$\omega_j(t) \leq \frac{t^2}{(1+t)^2}\overline{\omega_j}(t), \qquad \overline{\omega_j}(t) := (j+1)(2j+1) + \frac{4j+5}{4}t + \frac{1}{2}t^2.$$

Using the identity $\exp(-s/\gamma) = \exp(-s/\gamma + s)\exp(-s)$ and the mean value theorem we obtain

$$\begin{aligned}\rho_{j-1}(1/\gamma)^2 &< \frac{1}{2}\max_{t\geq 0}\left\{\exp\left(\frac{1-t}{\gamma}+t\right)\left(\frac{t}{t+1}\right)^{2j}\right\}\int_0^\infty \exp(-s)\overline{\omega_j}(s)ds \\ &\leq \max_{t\geq 0}\exp(\psi_j(t))\left(j^2+2j+\frac{13}{8}\right) \\ &\leq 5j^2\max_{t\geq 0}\exp(\psi_j(t))\end{aligned}$$

with

$$\psi_j(t) = \frac{1}{\gamma}(1-t) + t + 2j(\log(t) - \log(t+1)).$$

We find that

$$\max_{t\geq 0}\psi_j(t) = \frac{1}{2\gamma}\left(3 - D_j - \gamma + 4j\gamma\log\left(-\frac{\gamma - 1 + D_j}{\gamma - 1 - D_j}\right)\right) =: \overline{\psi_j},$$

with $D_j = \sqrt{(1-\gamma)(8j\gamma - \gamma + 1)}$ (the maximum is attained at $t = \frac{1}{2}(-1 + \frac{D_j}{1-\gamma})$).

Taking the limit of $\overline{\psi_j}/\sqrt{j}$ for $j \to \infty$ and using (3.4) provides the desired result. □

In our context we are interested in approximations where the degree is modest. In the absence of insightful analytical estimates, we have decided to give numerical

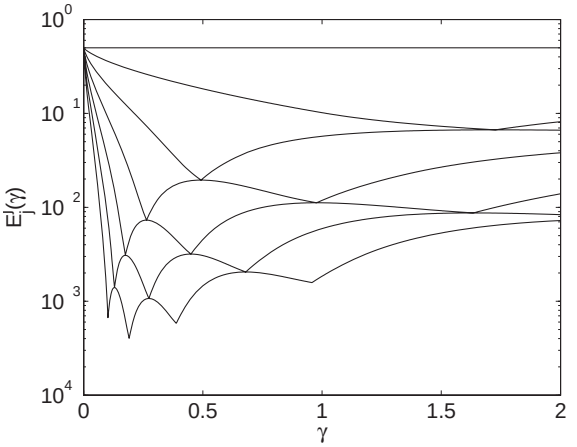


FIG. 3.1. $E_j^j(\gamma)$ *as a function of* γ *for* $j = 0, \ldots, 6$.

TABLE 3.1
Numerical approximations to the optimal value of γ, γ_{opt}, *and the corresponding value* $E_j^j(\gamma_{opt})$.

j	$E_j^j(\gamma_{\text{opt}})$	γ_{opt}	j	$E_j^j(\gamma_{\text{opt}})$	γ_{opt}
1	$6.7 \cdot 10^{-2}$	$1.73 \cdot 10^{0}$	11	$4.0 \cdot 10^{-6}$	$9.90 \cdot 10^{-2}$
2	$2.0 \cdot 10^{-2}$	$4.93 \cdot 10^{-1}$	12	$1.6 \cdot 10^{-6}$	$1.19 \cdot 10^{-1}$
3	$7.3 \cdot 10^{-3}$	$2.64 \cdot 10^{-1}$	13	$6.1 \cdot 10^{-7}$	$1.00 \cdot 10^{-1}$
4	$3.1 \cdot 10^{-3}$	$1.75 \cdot 10^{-1}$	14	$2.5 \cdot 10^{-7}$	$8.64 \cdot 10^{-2}$
5	$1.4 \cdot 10^{-3}$	$1.30 \cdot 10^{-1}$	15	$1.0 \cdot 10^{-7}$	$7.54 \cdot 10^{-2}$
6	$4.0 \cdot 10^{-4}$	$1.91 \cdot 10^{-1}$	16	$4.0 \cdot 10^{-8}$	$8.67 \cdot 10^{-2}$
7	$1.6 \cdot 10^{-4}$	$1.44 \cdot 10^{-1}$	17	$1.6 \cdot 10^{-8}$	$7.63 \cdot 10^{-2}$
8	$6.5 \cdot 10^{-5}$	$1.90 \cdot 10^{-1}$	18	$6.6 \cdot 10^{-9}$	$6.78 \cdot 10^{-2}$
9	$2.4 \cdot 10^{-5}$	$1.47 \cdot 10^{-1}$	19	$2.7 \cdot 10^{-9}$	$7.62 \cdot 10^{-2}$
10	$9.7 \cdot 10^{-6}$	$1.19 \cdot 10^{-1}$	20	$1.1 \cdot 10^{-9}$	$6.82 \cdot 10^{-2}$

approximations to the values of $E_{m-1}^{m-1}(\gamma)$ and, more interestingly, the location of the optimal value of γ, that is, the value that minimizes $E_{m-1}^{m-1}(\gamma)$. These results can be obtained using the Remez method, e.g., [31, section 1.3], by constructing the optimal polynomial approximation to $f_\gamma^1(t)$ on the interval $[0, 1]$. See the proof of Lemma 3.1 for more details.

In Figure 3.1 we have plotted $E_{m-1}^{m-1}(\gamma)$ as a function of γ for seven different values of m. This shows a very interesting pattern that is summarized by the following observation.

CONJECTURE 3.1. [illegible] $E_{j-1}^{j-1}(\gamma)$ [illegible] $j = \{2, 3, \ldots\}$ [illegible] $j - 1$ [illegible] [illegible] $E_j^j(\gamma)$ [illegible] $E_{j-1}^{j-1}(\gamma)$ [illegible] $E_j^j(\gamma)$

If the curve $E_j^j(\gamma)$ has a local minimum, this appears to coincide with the fact that the error curve for the polynomial approximation at this point seems to have one additional alternation point. In Table 3.1 the optimal value of γ and the corresponding error have been reported. We used a straightforward search algorithm for this in combination with Remez approximations.

As mentioned previously we must choose our shift γ before we start our iterative procedure. Changing it during the process is too expensive. Although this appears

to be a drawback from a theoretical point of view, in practice this is no problem since the required precision is known beforehand. For instance, if one is interested in a precision of about 10^{-5} one can consider Table 3.1 and decide to choose $\gamma = 0.119\tau$, since this γ minimizes our upper bound from Lemma 3.1.

3.2. Discussion. One might wonder if there is anything to gain by using different real shifts in the Lanczos method, that is, if it pays to construct an approximation from the class

$$\widetilde{\mathcal{R}}_i^j = \left\{ p(t) \prod_{i=1}^{i} (1 + \gamma_i t)^{-1} \,|\, p \in \Pi_j \right\}, \quad \text{where } \gamma_i \in \mathbb{R}.$$

Using different shifts in a Lanczos method can be facilitated by rational Krylov methods [32]. Interestingly enough, it was conjectured in two independent papers [19, 22] that the optimal approximation from $\widetilde{\mathcal{R}}_i^j$ is also contained in $\mathcal{R}_i^j$. This conjecture was later shown to be true [2, Theorem 1]. If one is willing to work with quadratic factors, or linear factors with complex shift, one can consider using Chebyshev rational approximations. A comparison for solving linear parabolic equations between the Crank–Nicolson method and using a Chebyshev rational approximation can be found in [5].

Using optimal restricted rational approximations on the negative real axis to approximate the exponential function for solving semidiscretized parabolic differential equations is discussed in [34, p. 319]. The authors emphasize the computational advantages when only shifted systems with constant real shifts have to be solved. Many other rational approximations, for example, Chebyshev rational approximations and the diagonal elements of the Padé table, cannot be factored into linear factors with real coefficients. Restricted rational approximations to the matrix exponential of Padé type have been used by Nørsett and Trickett [26, 27, 28]; see also the paper by Van Iseghem [43]. These rational approximations play an important role in the development of so-called semi-implicit diagonal Runge–Kutta methods, which are attractive since in one step of the method only systems have to be inverted with a constant real shift. In [16] explicit expressions for the restricted rational approximations are derived that interpolate the exponential function in an equispaced mesh. It is shown by a numerical example that rational interpolations are better suited for approximating the matrix exponential than restricted Padé approximations when A originates from a Poisson operator. The main difference between all these techniques and the method considered in this paper is that the use of the Lanczos method allows us to exploit the favorable eigenvalue spectrum of the transformed matrix as discussed previously. This will become clear when we discuss numerical examples in section 6.

The proposed method in this paper is also related to the work on Krylov subspaces in [9], where the authors are interested in Krylov subspaces for A extended by a Krylov subspace generated by A^{-1}. They show that these subspaces have a superior approximation quality (although they consider a different function class). One essential difference is that, in the present paper, we work with shifted inverses and, considering the results in the previous section, we mention that this is pivotal. Also, the a priori error bounds presented in [9] depend on the condition number of the matrix A. Finally, a difference is that we are interested in preconditioned solvers for the solution of linear systems. We will discuss this further in section 5.

4. A posteriori error estimation. In this section we consider strategies for terminating the shift-and-invert Lanczos process as soon as $y_m(\tau)$ is within a predefined

distance to the sought-after vector. To this purpose we first derive an explicit expression for the error. In practice this relation cannot be evaluated exactly. Subsequently we will discuss a few ideas to come to practical error estimators.

By rewriting (3.1) we find that

$$V_m T_m^{-1} = (I + \gamma A)V_m + \beta_m \widetilde{v}_{m+1} z_m^*, \quad \widetilde{v}_{m+1} = (I + \gamma A)v_{m+1}, \quad z_m = T_m^{-1} e_m.$$

Hence, we obtain the following relation:

$$AV_m = V_m \widetilde{T}_m - \frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^*, \quad \widetilde{T}_m = \frac{1}{\gamma}(T_m^{-1} - I). \tag{4.1}$$

Using this relation we see that our approximation satisfies

$$\begin{aligned} y_m'(t) &= -V_m \widetilde{T}_m \exp(-t\widetilde{T}_m) e_1 = -A y_m(t) + g_m(t), \\ g_m(t) &= -\frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^* \exp(-t\widetilde{T}_m) e_1, \end{aligned}$$

with $y_m(0) = v$, whereas the exact solution fulfills

$$y'(t) = -Ay(t), \qquad y(0) = v.$$

We define $e_m(t) = y_m(t) - y(t)$ and get

$$e_m'(t) = -Ae_m(t) + g_m(t), \quad e_m(0) = 0. \tag{4.2}$$

Finally, an explicit expression for the error follows by writing down the solution of this ODE using variation of constants, which yields

$$\begin{aligned} e_m(\tau) &= \int_0^\tau \exp(-(\tau - t)A) g_m(t)\, \mathrm{d}t \\ &= -\frac{\beta_m}{\gamma} \int_0^\tau \exp(-(\tau - t)A) \widetilde{v}_{m+1} z_m^* \exp(-t\widetilde{T}_m) e_1\, \mathrm{d}t \end{aligned} \tag{4.3}$$

or, equivalently,

$$e_m(\tau) = -\frac{\beta_m}{\gamma} X_m v_{m+1}, \tag{4.4}$$

where

$$X_m := \int_0^\tau e_m^* (I + \gamma \widetilde{T}_m) \exp(-t\widetilde{T}_m) e_1 \exp(-(\tau - t)A)(I + \gamma A)\, \mathrm{d}t. \tag{4.5}$$

Hence, the error is determined by the norm of the operator X_m. Unfortunately, the precise size of the error is hard to assess through (4.3) since this quantity is difficult to evaluate. In the remainder of this section we focus on practical error estimators.

We substitute

$$\exp(-(\tau - t)A) = I - (\tau - t)A + \frac{1}{2}(\tau - t)^2 A^2 + \cdots$$

in (4.3) and obtain

$$e_m(\tau) = -\frac{\beta_m}{\gamma} \int_0^\tau \left(I - (\tau - t)A + \frac{1}{2}(\tau - t)^2 A^2 + \cdots \right) \widetilde{v}_{m+1} z_m^* \exp(-t\widetilde{T}_m) e_1\, \mathrm{d}t.$$

By defining the functions

$$\phi_j(-\tau A) = \frac{1}{(j-1)!}\int_0^\tau (\tau - t)^{j-1}\exp(-tA)\,\mathrm{d}t, \tag{4.6}$$

we get the following result.

LEMMA 4.1. [illegible] $e_m(\tau)$ [illegible]

$$e_m(\tau) = -\frac{\beta_m}{\gamma}\sum_{j=0}^{\infty} z_m^* \phi_{j+1}(-\tau\widetilde{T}_m)e_1(-A)^j(I+\gamma A)v_{m+1}. \tag{4.7}$$

By partial integration it can be shown that the ϕ-functions can be computed by means of a simple recursion. This series is similar to the one given in [33] for standard Lanczos approximations. A practical error estimator follows by taking only the first summand of the series:

$$\|e_m(\tau)\| \approx \frac{\beta_m}{\gamma}|z_m^*\phi_1(-\tau\widetilde{T}_m)e_1|\,\|(I+\gamma A)v_{m+1}\|. \tag{4.8}$$

More advanced strategies can be taken as in the software package [35] for the standard Lanczos approximation which is based on the analogous expansion in [33]. We will not follow this direction.

Our derivation of (4.7) immediately suggests alternative approximations to the error by using more advanced approximations than the Taylor series expansion, e.g., using Chebyshev series, or again by Lanczos approximations. We can substitute, for instance, into (4.3) the following approximation:

$$(I+\gamma A)\exp(-(\tau-t)A)v_{m+1} \approx V_{m+j}(I+\gamma\widetilde{T}_{m+j})\exp(-(\tau-t)\widetilde{T}_{m+j})e_{m+1}.$$

The obvious advantage of this approximation is that the error estimator only requires information that is already computed in the Lanczos process, so there is no additional work necessary. Not surprisingly, it is very easy to see that with this substitution the norm of (4.3) is equal to $\|y_{m+j}(\tau) - y_m(\tau)\|$.

Estimating the error by the difference of two (consecutive) approximations is straightforward and simple; however, it has some less appealing properties. For example, stagnation of the process leads to excessively optimistic error estimates. In our practical experience with the presented method, no problems were encountered if γ is sensibly chosen and therefore convergence is fast enough. A different problem occurs in the first few steps where we see that $\|y_m(\tau)\| \ll \|y(\tau)\|$, which can be explained by the fact that the eigenvalues of $\widetilde{T}_m$ are far from the smallest eigenvalue of A. Numerical experience shows that an effective upper bound for the [illegible] error in step m is given by

$$\delta_m = \frac{\|y_m(\tau) - y_{m-j}(\tau)\|}{\|y_m(\tau)\|}.$$

With this assumption we have that $\|y(\tau)\| \lesssim \delta_m\|y(\tau)\| + \|y_m(\tau)\|$ and we correct the error estimator as follows:

$$\|e_m(\tau)\| \lesssim \frac{\delta_m}{1-\delta_m}\|y_m(\tau)\|. \tag{4.9}$$

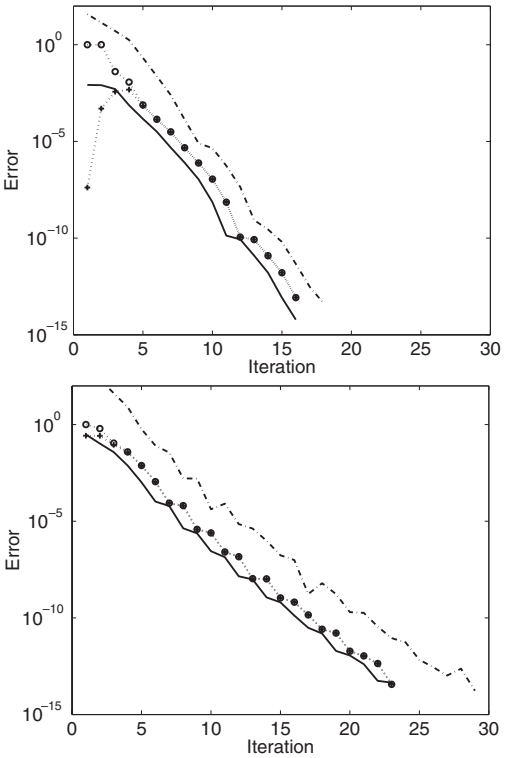


FIG. 4.1. *Error of the shift-and-invert Lanczos approximation (solid) and the error estimators* (4.8) *(dash-dot),* (4.9) *(o), and the difference of two iterates (+). In both pictures* $\gamma = \tau/10$. *Top picture:* $\tau = 1/5$*; bottom picture:* $\tau = 1/100$.

A trivial upper bound is $\|e_m(\tau)\| \leq 1 + \|y_m(\tau)\|$. In our experiments, we take the minimum of this bound and (4.9) as an error estimator to circumvent very large error estimates in the first steps. Furthermore, in all our experiments we have taken j to be equal to one.

The corrected estimator (4.9) turns out to be very effective in experiments and we will now demonstrate this with a simple example. The matrix A is a central finite difference discretization with 30 grid points in both directions of the Poisson operator on the unit square with homogeneous Dirichlet boundary conditions. Figure 4.1 illustrates that for $\tau = 1/5$ estimating the error by the difference of two iterates can lead to overly optimistic estimates in the start. This behavior is corrected by (4.9). Typically we see that the estimator starts in the neighborhood of one and after a few iterations it quickly decreases as more information on the norm of the error becomes available. We remark that these two figures demonstrate some interesting properties of the convergence of the proposed method; we will discuss this in detail in section 6.

5. Tuning the inner-outer iteration. We want to compute an approximation $y_m(\tau)$ which is at an absolute distance of about ϵ from the true vector. In step j of the shift-and-invert Lanczos process we have to solve a linear system involving an elliptic operator. If this is accomplished using an iterative solver, we have to prescribe

a constant η_j that determines the precision for this inner solve, that is,

$$\|r_j\| \le \eta_j \quad \text{with} \quad r_j = v_j - (I + \gamma A)c_j. \tag{5.1}$$

The vector c_j is the approximate solution of the linear system. In this section we look in more detail at the influence of these inaccuracies on the Lanczos approximation. Numerical experiments show that the sensitivity towards these errors in the Lanczos process is similar to that witnessed in other applications of the Lanczos method where matrix-vector products are perturbed. A precise error analysis, however, turns out to be very challenging, and we restrict ourselves to more heuristical arguments here.

The errors in the solution of the linear systems in (5.1) result in a perturbed Lanczos relation (3.1) which after some manipulation can be recast as a perturbed form of (4.1):

$$AV_m = V_m \widetilde{T}_m - \frac{\beta_m}{\gamma} \widetilde{v}_{m+1} z_m^* - \frac{1}{\gamma} F_m T_m^{-1}. \tag{5.2}$$

Here, the jth column of the matrix F_m equals the residual of the linear system that is solved in the jth step of the Lanczos process; in other words, we have $F_m e_j = r_j$. We warn the reader that the tridiagonal matrix T_m and the matrix V_m in this relation are different from the tridiagonal matrix generated by the Lanczos method in the error-free case. For instance, there is no guarantee that the matrix V_m is orthogonal. Another source of errors is rounding errors resulting from the use of floating point arithmetic. Since these errors are typically many orders of magnitude smaller than the approximation errors introduced in the solution of the shifted systems, we will neglect rounding errors. Using the relation (5.2) and reasoning along the same lines as in the previous section, we find the following bound for the error:

$$\|e_m(\tau)\| \le \frac{\beta_m}{\gamma} \|X_m v_{m+1}\| + \frac{1}{\gamma} \left\| \int_0^\tau \exp(-(\tau - t)A) F_m T_m^{-1} \exp(-t\widetilde{T}_m) e_1 \, \mathrm{d}t \right\|.$$

The matrix X_m is again as defined in (4.5). If we assume that $\eta_j = \epsilon$, and therefore $\|r_j\| \le \epsilon$, then a crude estimate for the second quantity is given by

$$\frac{1}{\gamma} \left\| \int_0^\tau \exp(-(\tau - t)A) F_m T_m^{-1} \exp(-t\widetilde{T}_m) e_1 \, \mathrm{d}t \right\| \le \|T_m^{-1} e_1\| \frac{\sqrt{m}\tau}{\gamma} \epsilon.$$

Here we have used the Cauchy–Schwarz inequality. This shows that

$$\|e_m(\tau)\| \le \frac{\beta_m}{\gamma} \|X_m v_{m+1}\| + \|(I + \gamma \widetilde{T}_m) e_1\| \frac{\sqrt{m}\tau}{\gamma} \epsilon. \tag{5.3}$$

We see that in the first few iterations the first term is the dominating quantity and, in fact, determines the convergence behavior of the perturbed method. Standard practice would be to view the first term in (5.3) as the error computed by an exact Lanczos process on a nearby matrix. This is a common approach in the analysis of perturbed Lanczos methods; see, for instance, [6]. Unfortunately, this idea does not easily carry over to the method of the present paper since the matrix X_m defined by (4.5) depends not only on the tridiagonal matrix T_m but also on the original matrix A.

As an alternative we focus on the expansion in Lemma 4.1 and in particular the quantities $z_m^* \phi_{j+1}(-\tau \widetilde{T}_m) e_1$ as a function of the iteration number m. We assume that for all m, the eigenvalues of T_m are contained in an interval $(0, \alpha]$. In practice the

positive definiteness of the tridiagonal matrix T_m can be guaranteed by using a version of the Lanczos method based on coupled recurrences in combination with solving the linear systems by the conjugate gradient method. Bounds on the eigenvalues of the tridiagonal matrix T_m generated by a perturbed Lanczos process can be found in [30]. For every polynomial $p \in \Pi_{m-2}$, we have

$$\begin{aligned} |z_m^* \phi_{j+1}(-\tau \widetilde{T}_m) e_1| &= |e_m^* (T_m^{-1} \phi_{j+1}(-\tau \widetilde{T}_m) e_1 - p(T_m)) e_1| \\ &\leq \inf_{p \in \Pi_{m-2}} \sup_{t \in (0,\alpha]} |\tfrac{1}{t} \phi_{j+1}(\tfrac{\tau}{\gamma}(1 - \tfrac{1}{t})) - p(t)|. \end{aligned}$$

This suggests that the coefficients in the expansion (4.8) decrease as a function of m with our assumption on the eigenvalues of the tridiagonal matrices. This argument confirms the observation that the vector $\|X_m v_{m+1}\|$ gets many orders of magnitude smaller than ϵ before it stagnates. As a consequence, this means that in the end the attainable precision of the method is essentially bounded by the second term in (5.3), and we have argued that the method can achieve a final precision of $\mathcal{O}(\epsilon)$ when all linear systems are solved with a fixed residual precision of ϵ.

There are some recent research efforts in the analysis of the effect of approximate matrix-vector products on Krylov subspace methods, e.g., [3, 11, 36, 39]. These works show that when solving linear systems or computing eigenvectors by means of a Krylov subspace method, accurate approximations to the matrix-vector product are necessary in the first iterations, but this precision can be relaxed as the method converges. It is interesting to see if such strategies can be extended to the current context where inexact matrix-vector products result from the errors in the solution of the shifted systems.

A key issue of the analysis in [11, 36, 39] is that the computed vector in the approximation subspace has a decreasing pattern. Suppose that after m steps of the error-free method we have computed an approximation that is sufficiently accurate, i.e., $\|y_m(\tau) - y(\tau)\| \leq \epsilon$. We have the following for the component of the solution in the direction of the vector $v_j = V_m e_j$:

$$\begin{aligned} |v_j^* y_m(\tau)| &\leq |v_j^*(y_m(\tau) - y(\tau))| + |v_j^* y(\tau)| \\ &\leq \epsilon + \|(I - V_{j-1} V_{j-1}^*) y(\tau)\| \leq \epsilon + \|e_{j-1}(\tau)\|. \end{aligned}$$

In the last step we have assumed that the iterate from step $j-1$ is approximately equal to the optimal approximation from the $(j-1)$th Krylov subspace. Intuitively, this shows that the errors in the Lanczos process in the first steps have a relatively large impact since the solution lies mainly in the direction of the first Lanczos vectors. For this reason, we propose the following strategy for controlling the tolerances of the linear systems:

$$\eta_j = \frac{\epsilon}{\|e_{j-1}(\tau)\| + \epsilon}. \tag{5.4}$$

Practical strategies result from replacing $\|e_{j-1}(\tau)\|$ in (5.4) with a suitable upper bound as discussed in the previous section, for instance, (4.9). Following nomenclature from [3] we will refer to (5.4) as a [illegible] strategy for the remainder of this paper. In the current context such a strategy turns out to be much harder to analyze than for linear systems due to the nonlinear character. Nevertheless, this strategy worked very satisfactorily in all our numerical experiments. We will give an example of this in section 6.1.

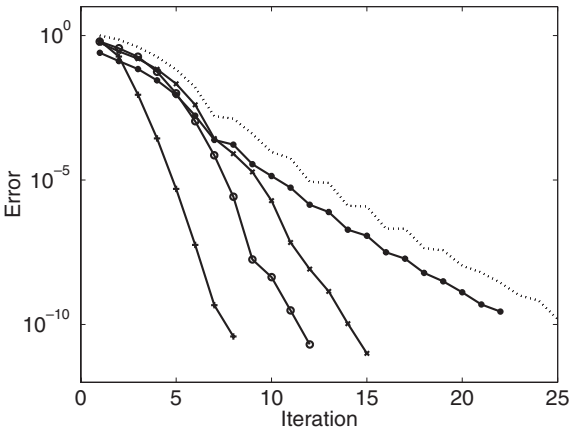


FIG. 6.1. *Error as a function of the number of iterations for* $\tau = 1/2$ *(+),* $\tau = 1/20$ *(o),* $\tau = 1/50$ *(×),* $\tau = 1/2000$ *(•), and the upper bound from Lemma* 3.1 *(dotted).*

6. Numerical experiments. In this section we discuss some numerical experiments that are typical of our experiences with the presented method. All experiments are conducted in MATLAB. The purpose of our first experiment is to illustrate properties of the convergence behavior of the new method. The matrix A results from a simple central finite difference discretization of the one-dimensional Poisson operator with periodic boundary conditions. Although this matrix is of little practical importance, it leads to representative results and, more importantly, it allows us to compute the true solution with a fast Fourier transform for comparison purposes. Moreover, the smallest eigenvalue of this matrix is zero, which means that the norms of the true solutions are not very sensitive to the choice of τ. The dimension of the matrix is 10^5 and in all experiments in this section we take $\gamma = \tau/10$. This latter choice is not necessarily optimal. However, using this fixed strategy in all experiments shows that the method is not very sensitive towards a very precise choice for γ. Notice that the traditional Lanczos approximation is not an option here since upper bounds suggest, and experiments often show, that for one-dimensional elliptic operators convergence starts after the number of iteration steps is almost equal to the number of spatial grid points. The results of our numerical experiments are reported in Figure 6.1, where we have included as a reference the upper bound from Lemma 3.1.

Interestingly, the number of Lanczos iterations increases when τ becomes smaller. An explanation lies in the fact that for larger values of τ a relatively small portion of the eigenvalues essentially determines the final vector. Due to the nice separation of these eigenvalues, as discussed in section 3, Krylov subspaces quickly come to contain approximations to the corresponding invariant subspaces and therefore only a few iteration steps are required. When τ decreases, loosely speaking, more and more eigenvalues determine the final result and a more uniform approximation is more effective then. This shows that the Lanczos approximation indeed takes advantage of the nice eigenvalue distribution of the transformed system, and it shows that convergence might be much faster than is expected from treating the problem as a uniform approximation problem on an unbounded interval as we did in our upper bounds in section 3.1.

6.1. The impact of errors in the solution of the linear systems. In the remainder of this section we take as a model equation the parabolic partial differential equation in two spatial dimensions:

$$\frac{\partial}{\partial t}u(t,x,y) = \mathcal{L}u(t,x,y), \quad (x,y) \in \Omega = (0,1)^2, \quad 0 \le t \le \tau, \tag{6.1}$$

subject to the initial condition

$$u(0,x,y) = u_0(x,y).$$

The operator $\mathcal{L}$ is a self-adjoint linear second-order elliptic operator,

$$\mathcal{L} = \frac{\partial}{\partial x}a(x,y)\frac{\partial}{\partial x} + \frac{\partial}{\partial y}b(x,y)\frac{\partial}{\partial y} + c(x,y), \tag{6.2}$$

with time-independent boundary conditions. The matrix A originates from discretizing the operator $\mathcal{L}$ using central finite differences on a square grid.

We investigate the impact of errors in the solution of the shifted systems (the inner iteration) on the final accuracy of the overall method. Therefore we have experimented with two choices of η_j in (5.1): taking $\eta_j = \epsilon$ (fixed) and the relaxation strategy given by (5.4) in section 5. For the latter strategy we have replaced the norm of the error by the approximation given by (4.9). Moreover, the Lanczos process was terminated as soon as this quantity dropped below ϵ. The matrix A stems from discretizing the operator $\mathcal{L}$ in (6.2) with

$$a(x,y) = 1 + y - x, \quad b(x,y) = 1 + x + x^2, \quad c(x,y) = 0,$$

and homogenous Dirichlet boundary conditions on the western and eastern boundaries and a homogenous Neumann boundary condition on the northern and southern boundaries of the domain. For the discretization of the operator we have used 40 grid points in both directions, which still allows us to compute the true solution with a dense method for comparison purposes. We are interested in a precision of $\epsilon = 10^{-6}$.

The results of our numerical experiments are summarized in Table 6.1. The shifted linear systems are solved with the conjugate gradient method (CG) preconditioned by a standard incomplete LU preconditioner. In the next section we use more advanced and effective preconditioners. As a measure for the total amount of work we have reported the total number of applications of the ILU preconditioner in the inner iterations and the number of outer iterations. Moreover, we have included the error of the computed approximation at termination as well as the error estimated at this point by (4.9).

TABLE 6.1
Numerical results for $\gamma = \tau/10$.

	Fixed			Relaxed		
τ	In./out.	Error	Estimate	In./out.	Error	Estimate
1	228/5	$3.8 \cdot 10^{-8}$	$3.9 \cdot 10^{-7}$	152/5	$3.7 \cdot 10^{-8}$	$4.0 \cdot 10^{-7}$
1/5	348/10	$6.5 \cdot 10^{-8}$	$3.1 \cdot 10^{-7}$	230/10	$1.9 \cdot 10^{-7}$	$3.1 \cdot 10^{-7}$
1/10	326/12	$2.3 \cdot 10^{-7}$	$8.9 \cdot 10^{-8}$	202/12	$5.7 \cdot 10^{-7}$	$9.6 \cdot 10^{-8}$
1/50	182/13	$8.2 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$	114/13	$8.9 \cdot 10^{-7}$	$7.0 \cdot 10^{-7}$
1/100	140/14	$1.2 \cdot 10^{-6}$	$2.3 \cdot 10^{-7}$	82/14	$1.3 \cdot 10^{-6}$	$2.1 \cdot 10^{-7}$

The table shows that in all cases we achieve the required precision of about 10^{-6}. The use of a relaxation strategy as opposed to using a fixed precision for the inner

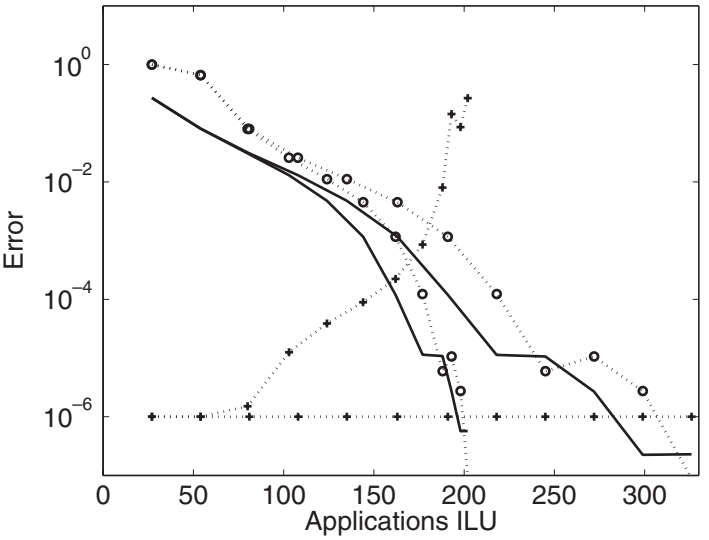


FIG. 6.2. *Error (solid), tolerance η_j (+), and error estimator (o) as a function of the total number of applications of the ILU preconditioner for $\eta_j = \epsilon$ and η_j as in* (5.4) *for $\tau = 1/10$. See also Table* 6.1.

iterations has no influence on the total number of Lanczos (outer) iterations. The amount of work in the inversions of the linear systems, measured as the total number of applications of the ILU preconditioner, is reduced by about 30 to 40 percent. This is comparable with reductions that are seen in applications of inexact Krylov methods in other areas; for a more detailed discussion see [40, section 3]. For the purpose of illustration we have included in Figures 6.2 and 6.3 a visual representation of the results for $\tau = 1/10$ and $\tau = 1/100$ for both strategies.

Again, we have that the number of Lanczos iterations increases when τ becomes smaller, as in the previous example. This is a combination of the fast convergence of the method for modest values of τ and, furthermore, is partially explained by the small norm of the true solution. On the other hand, we see that the number of CG steps per outer iteration decreases for smaller values of τ. This is a result of our choice $\gamma = \tau/10$, which makes the matrix more and more diagonally dominant when τ becomes smaller. For comparison purposes we mention that solving a linear system with this matrix and right-hand side to a precision of 10^{-6} requires 49 steps of the GMRES method preconditioned by an incomplete LU decomposition, a factor 4 less than for computing the exponential with $\tau = 1/10$.

From Figures 6.2 and 6.3 it seems that the convergence of the method stagnates now and then for one iteration. A possible explanation is that the Lanczos approximation mimics a property of the optimal uniform approximation. This is related to the observation in Conjecture 3.1 and can also be seen in Figure 3.1. Since for smaller values of τ the Lanczos method constructs a more and more uniform approximation on an interval, this might explain why these plateaus occur more often for small values of τ; see Figures 6.2, 6.3, and 4.1. We notice that in this experiment the error estimator (4.9) is a little too optimistic in the neighborhood of 10^{-5} for $\tau = 1/10$, which is caused by the temporary stagnation of the method at that point. Although such underestimation is not large and, in our experience, seldomly occurs, it can be resolved by switching to $j = 2$ in (4.9).

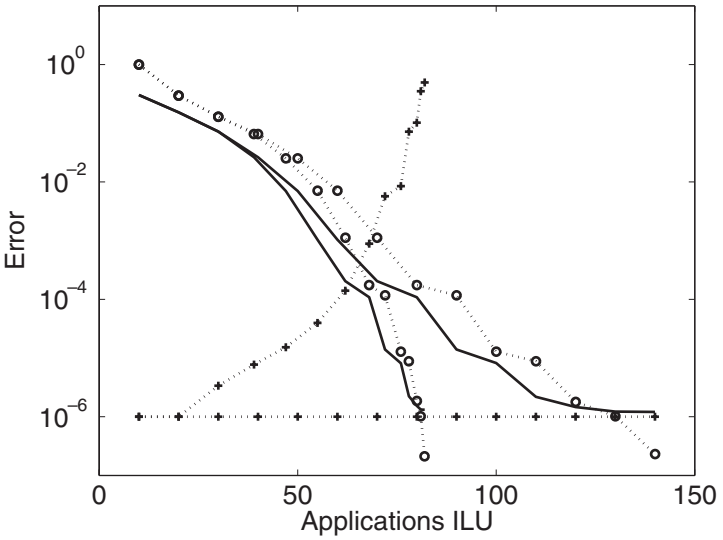


FIG. 6.3. *Error (solid), tolerance η_j (+), and error estimator (o) as a function of the total number of applications of the ILU preconditioner for $\eta_j = \epsilon$ and η_j as in* (5.4) *for $\tau = 1/100$. See also Table* 6.1.

6.2. The influence of mesh refinement. In our next experiment we explore the impact of the number of spatial grid points on the method. In order to be able to handle large matrices we have used the SAMG algebraic multigrid package [37] for solving the shifted systems. This package is written in Fortran and we have interfaced it with MATLAB. For ease of illustration we have taken the standard Poisson operator on a square grid with Dirichlet boundary conditions, i.e., $a(x, y) = b(x, y) = 1$ and $c(x, y) = 0$ in (6.2). We aim at a precision of $\epsilon = 10^{-8}$. In Table 6.2 we have reported the total number of cycles done in the inner iterations and the number of Lanczos iterations necessary to reduce the error estimator below 10^{-8} for different combinations of n (number of spatial grid points) and τ. Again we have used two strategies for choosing η_j: $\eta_j = \epsilon$ and η_j as in (5.4) with the norm of the error replaced by the estimator (4.9). The matrices were unfortunately too large to check if the obtained solution fulfills our accuracy requirement by comparing it against the outcome of a dense method. Instead, we have verified the results via another sparse method set to a smaller tolerance. In all cases the computed vectors have an error that is at most of the order of the required precision.

The table shows that increasing the accuracy in the spatial discretization has no dramatic influence on the total number of Lanczos iterations. For example, for $\tau = 1$, in all tests we needed four steps. This shows that even in situations where convergence is heavily based on exploiting the eigenvalue spectrum, the convergence is grid independent. This is expected since the upper estimate of the spread/gap ratio does not depend on the norm of A. When τ is decreased we see again that more steps of the Lanczos method are required. The computational advantage of a relaxation strategy is modest for a large value of τ. The reason is that in this situation the upper bound is around one in the first iterations and then very quickly drops in the final two iterations. For smaller values of τ the gain is comparable to that for the example in the previous section. We warn the reader that the SAMG package detects diagonal

TABLE 6.2
Total number of cycles/number of Lanczos iterations as a function of τ and the size of the matrix.

n	32^2	64^2	128^2	256^2	512^2	1024^2
			Fixed			
$\tau = 1$	20/4	24/4	24/4	27/4	28/4	28/4
$\tau = 1/10$	65/13	70/13	78/13	78/13	90/13	91/13
$\tau = 1/100$	104/15	48/16	89/18	90/18	108/18	108/18
$\tau = 1/1000$	23/8	52/11	123/14	64/16	76/16	80/16
			Relaxed			
$\tau = 1$	18/4	21/4	21/4	23/4	24/4	24/4
$\tau = 1/10$	43/13	48/13	52/13	53/13	58/13	60/13
$\tau = 1/100$	62/15	32/16	48/18	55/18	61/18	64/18
$\tau = 1/1000$	15/8	30/11	66/14	39/16	42/16	48/16

dominance of a matrix and automatically switches to a different solution strategy if this is more appropriate. This means that one cannot straightforwardly compare the total number of cycles for different combinations of τ and n.

7. Summary and outlook. The aim of this paper is to generalize the concept of preconditioning for linear systems to the computation of the product of the matrix exponential and a vector. The idea explored is to apply the Lanczos process to a shifted and inverted matrix that better emphasizes the important eigenvalues. This results in an inner-outer iteration scheme where the solution of the shifted systems can be accomplished by preconditioned solvers. The worst case convergence behavior of the method can be bounded in terms of a rational approximation problem on the positive real axis. However, convergence speeds observed in practice are often much higher than these bounds suggest. This is due to the fact that the spectral transformation facilitates the exploitation of the discrete nature of the spectrum of the matrix. We have argued and demonstrated that the method shows convergence speeds independent of the norm of the matrix and, if an appropriate method is used for the solution of the linear systems, we can compute the sought-after vector in a time proportional to the number of spatial grid points. Furthermore, we have proposed an empirical strategy for choosing the tolerances for the errors in the solution of the shifted linear systems. This strategy can reduce the amount of work in the solution of the shifted systems by up to 40 percent compared with using a fixed tolerance on the error.

For the sake of clarity, in this paper we have restricted our attention to the computation of the exponential function. However, the same approach seems to carry over successfully to the computation of the matrix functions related to (4.6) that play, besides the exponential function, an important role in exponentially based integrators; e.g., see [14, 18, 20]. Also, we have assumed throughout this paper that the matrix is symmetric positive semidefinite. It is clear how our work can be extended to deal with so-called sectorial operators as well. Much of the theory and heuristics here are expected to hold in this case too. For example, it has been shown that there is a sequence of approximations from the classes $\mathcal{R}_m^{m-1}$ that converges geometrically to $\exp(-t)$ in a sector symmetrically around the real axes; see [34].

In our future work we plan to investigate other ideas for incorporating preconditioning in the computation of the matrix exponential. This is particularly important in applications where the matrix is skew-symmetric and, therefore, has purely imaginary eigenvalues. In this case it is not possible to exploit the rapid decay of the exponential functions, and a different approach is required.

Acknowledgments. We would like to thank Klaus Stüben and Tanja Clees of the GMD for providing us with the SAMG package and for their assistance in using the package. The authors appreciate the useful suggestions and comments of the referees that improved the presentation of this paper. We are thankful to Paolo Novati and Igor Moret for providing us with copy of a preliminary version of [25]. Our gratitude goes to our colleague Jörg Niehoff for helpful discussions.

REFERENCES

[1] J.-E. Andersson, *Approximation of e^{-x} by rational functions with concentrated negative poles*, J. Approx. Theory, 32 (1981), pp. 85–95.
[2] P. B. Borwein, *Rational approximations with real poles to e^{-x} and x^n*, J. Approx. Theory, 38 (1983), pp. 279–283.
[3] A. Bouras and V. Frayssé, *A Relaxation Strategy for Inexact Matrix-Vector Products for Krylov Methods*, Technical report TR/PA/00/15, CERFACS, France, 2000.
[4] P. Castillo and Y. Saad, *Preconditioning the Matrix Exponential Operator with Applications*, Technical Report UMSI 97/142, Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1997.
[5] J. C. Cavendish, W. E. Culham, and R. S. Varga, *A comparison of Crank–Nicolson and Chebyshev rational methods for numerically solving linear parabolic equations*, J. Comput. Phys., 10 (1972), pp. 354–368.
[6] V. Druskin, A. Greenbaum, and L. Knizhnerman, *Using nonorthogonal Lanczos vectors in the computation of matrix functions*, SIAM J. Sci. Comput., 19 (1998), pp. 38–54.
[7] V. L. Druskin and L. A. Knizhnerman, *Two polynomial methods of calculating functions of symmetric matrices*, Comput. Math. Math. Phys., 29 (1989), pp. 112–121.
[8] V. L. Druskin and L. A. Knizhnerman, *Krylov subspace approximations of eigenpairs and matrix functions in exact and computer arithmetic*, Numer. Linear Algebra Appl., 2 (1995), pp. 205–217.
[9] V. L. Druskin and L. A. Knizhnerman, *Extended Krylov subspaces: Approximation of the matrix square root and related functions*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 755–771.
[10] E. Gallopoulos and Y. Saad, *Efficient solution of parabolic equations by Krylov approximation methods*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 1236–1264.
[11] G. H. Golub, Z. Zhang, and H. Zha, *Large sparse symmetric eigenvalue problems with homogeneous linear constraints: The Lanczos process with inner-outer iterations*, Linear Algebra Appl., 309 (2000), pp. 289–306.
[12] W. Hackbusch, *Parabolic multigrid methods*, in Computing Methods in Applied Sciences and Engineering VI (Versailles, 1983), North-Holland, Amsterdam, 1984, pp. 189–197.
[13] M. Hochbruck and Ch. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925.
[14] M. Hochbruck, Ch. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574.
[15] G. Horton and S. Vandewalle, *A space-time multigrid method for parabolic partial differential equations*, SIAM J. Sci. Comput., 16 (1995), pp. 848–864.
[16] A. Iserles, *Rational approximations to* $\exp(-x)$ *with applications to certain stiff systems*, SIAM J. Numer. Anal., 18 (1981), pp. 1–12.
[17] S. Kaniel, *Estimates for some computational techniques in linear algebra*, Math. Comp., 20 (1966), pp. 369–378.
[18] A.-K. Kassam and L. N. Trefethen, *Fourth-order time stepping for stiff PDEs*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.
[19] E. H. Kaufman, Jr. and G. D. Taylor, *Best rational approximations with negative poles to e^{-x} on $[0, \infty)$*, in Padé and Rational Approximation (Proc. Internat. Sympos., Univ. South Florida, Tampa, FL, 1976), Academic Press, New York, 1977, pp. 413–425.
[20] S. Krogstad, *Generalized integrating factor methods for stiff PDEs*, J. Comput. Phys., to appear.
[21] A. B. J. Kuijlaars, *Which eigenvalues are found by the Lanczos method?,* SIAM J. Matrix Anal. Appl., 22 (2000), pp. 306–321.
[22] T. C.-Y. Lau, *Rational exponential approximation with real poles*, BIT, 17 (1977), pp. 191–199.
[23] C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev., 45 (2003), pp. 3–49.

[24] C. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix*, SIAM Rev., 20 (1978), pp. 801–836.
[25] I. Moret and P. Novati, *RD rational approximations of the matrix exponential*, BIT, to appear.
[26] S. P. Nørsett, *Restricted Padé approximations to the exponential function*, SIAM J. Numer. Anal., 15 (1978), pp. 1008–1029.
[27] S. P. Nørsett and St. R. Trickett, *Exponential fitting of restricted rational approximations to the exponential function*, in Rational Approximation and Interpolation (Tampa, FL, 1983), Lecture Notes in Math. 1105, Springer-Verlag, Berlin, 1984, pp. 466–476.
[28] S. P. Nørsett and St. R. Trickett, *Order-constrained uniform approximations to the exponential based on restricted rationals*, Constr. Approx., 2 (1986), pp. 189–195.
[29] P. Novati, *Solving linear initial value problems by Faber polynomials*, Numer. Linear Algebra Appl., 10 (2003), pp. 247–270.
[30] C. C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.
[31] P. P. Petrushev and V. A. Popov, *Rational Approximation of Real Functions*, Cambridge University Press, Cambridge, UK, 1987.
[32] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.
[33] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 29 (1992), pp. 209–228.
[34] E. B. Saff, A. Schönhage, and R. S. Varga, *Geometric convergence to e^{-z} by rational functions with real poles*, Numer. Math., 25 (1975/76), pp. 307–322.
[35] R. B. Sidje, *Expokit: A software package for computing matrix exponentials*, ACM Trans. Math. Software, 24 (1998), pp. 130–156.
[36] V. Simoncini and D. B. Szyld, *Theory of inexact Krylov subspace methods and applications to scientific computing*, SIAM J. Sci. Comput., 25 (2003), pp. 454–477.
[37] K. Stüben and T. Clees, *User's Manual SAMG*, http://www.scai.fraunhofer.de/samg.htm (August 2003).
[38] H. Tal-Ezer, *Spectral methods in time for parabolic problems*, SIAM J. Numer. Anal., 26 (1989), pp. 1–11.
[39] J. van den Eshof and G. L. G. Sleijpen, *Inexact Krylov subspace methods for linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 125–153.
[40] J. van den Eshof, G. L. G. Sleijpen, and M. B. van Gijzen, *Relaxation strategies for nested Krylov methods*, preprint 1268, Dept. Math., University Utrecht, Utrecht, The Netherlands, 2003.
[41] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.
[42] H. A. van der Vorst, *An iterative solution method for solving $f(A)x = b$, using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.
[43] J. Van Iseghem, *Padé-type approximants of $\exp(-z)$ whose denominators are $(1 + z/n)^n$*, Numer. Math., 43 (1984), pp. 283–292.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1458–1470
© 2006 Society for Industrial and Applied Mathematics

PARTITIONS OF A SIMPLEX LEADING TO ACCURATE SPECTRAL (FINITE) VOLUME RECONSTRUCTION*

QIAN-YONG CHEN†

Abstract. In this paper, we present several systematic techniques, based on the Voronoi diagram and its variants, to partition a one- and two-dimensional simplex. The Fekete points are used as input to generate the Voronoi diagram, as they concentrate near the edges and are almost optimal for polynomial interpolation in a simplex.

Spectral (finite) volume reconstructions on the resultant partitions have small Lebesgue constants. When using the Dubiner basis, the reconstruction matrix is well conditioned. Moreover, the total number of edges of the partitions (the total work when being used in spectral volume methods) is shown to be at most twice the minimum number of edges of all partitions for reconstructions of the same order accuracy. These suggest that the obtained partitions are well suited for spectral volume methods and other numerical methods which rely on reconstructions from cell averages.

Key words. spectral volume method, Voronoi diagram, Lebesgue constant, finite volume reconstruction

AMS subject classifications. 65M06, 65M70, 35L65

DOI. 10.1137/030601387

1. Introduction.

In [4], Wang proposed a new finite volume method, called the spectral volume method, for hyperbolic conservation laws. The spectral volume method has several good properties: it is high-order accurate, conservative, geometrically flexible, and computationally efficient. (A comparison with the discontinuous Galerkin methods is given in [5].) In the spectral volume method, a volume or a cell (named spectral volume) is partitioned into nonoverlapping subcells called control volumes. Then cell-averaged solutions on the control volumes are used to perform high-order reconstructions, i.e., spectral volume reconstructions. The spectral volume reconstruction is different from the reconstruction procedure of previous finite volume methods, which employs cell-averaged solutions on neighbor cells to perform the reconstruction.

The spectral volume reconstruction is basically an approximation problem. Given a smooth function and an approximation space, the accuracy of the spectral volume reconstructions only depends on the partition of the spectral volume. Not all partitions produce good results. For example, uniform partitions [6] yield bad results for high-order reconstructions because of the Runge phenomenon. As far as we know, no systematic technique has been developed to partition an m-dimensional simplex, $S^m \subset R^m$, except that in [6] Wang gave a few partitions for up to the fourth-order reconstruction on a standard equilateral triangle. It is difficult to directly build good high-order partitions because there are too many parameters, such as the position of points, the number of edges for each subcell, and the topology of the subcells. Of course, one can compute the so-called L^2 optimal partitions as in [7]. But even for the one-dimensional case, the L^2 optimal partitions are not as satisfactory as the correspondents for the interpolation [8].

*Received by the editors October 24, 2003; accepted for publication (in revised form) January 25, 2005; published electronically January 27, 2006.

http://www.siam.org/journals/sisc/27-4/60138.html

†Institute for Mathematics and Its Applications, University of Minnesota, Minneapolis, MN 55455 (qchen@ima.umn.edu).

However, another approximation problem on a simplex, the interpolation based on node values, has been extensively studied in the past (see [9] and the references therein). Some almost optimal nodal sets for polynomial interpolation on a two-dimensional simplex are given in [7, 9, 10]. In this paper, we develop several systematic techniques, based on the well-known Voronoi diagram and its variants with those optimal nodal sets as the input, to generate partitions of a one- and two-dimensional simplex. Using these techniques, we obtained partitions for up to the 14th order polynomial reconstruction on an equilateral triangle.

The remainder of the paper is organized as follows. In section 2, we restate the spectral volume reconstruction problem on a two-dimensional simplex. The Lebesgue constant is introduced as one measurement of the quality of spectral volume reconstructions. Section 3 describes the systematic techniques to partition a one- and two-dimensional simplex. Finally, we summarize the paper and make some concluding remarks in section 4.

2. Spectral volume reconstruction. In this section, we define the spectral volume reconstruction problem on an equilateral triangle, S^2. Some issues related to the quality of spectral volume reconstructions are also addressed.

The spectral volume reconstruction is a key element of the recently proposed spectral volume method [4, 6], in which a target cell is divided into nonoverlapping subcells. The cell-averaged solutions on the subcells are then used to reconstruct an approximate solution on the target cell. The number of subcells is the same as the dimension of the approximation space. In general, the approximation space can consist of any functions. Here, we focus on the space of polynomials of degree up to n, denoted as P_2^n. The dimension of this approximation space is

$$N_n^2 = \dim P_2^n = \binom{2+n}{2}. \tag{1}$$

$N = N_n^2$ and $P^n = P_2^n$ will be used to simplify the notations if there is no confusion.

The spectral volume reconstruction problem can be formally stated as follows.

Spectral Volume Reconstruction Problem on S^2. [illegible] $u(x,y)$ [illegible] S^2 [illegible] $u \in C^0(S^2)$ [illegible]

1. [illegible] Π_n [illegible] S^2

$$S^2 = C_1 \cup \cdots \cup C_N,$$

[illegible] $\{C_1, \ldots, C_N\}$ [illegible] N [illegible]

2. [illegible] $I_{\Pi_n} u :\in P^n$ [illegible]

$$\int_{C_i} (I_{\Pi_n} u)\, dV = \int_{C_i} u(x,y) dV, \qquad i = 1, \ldots, N; \tag{2}$$

[illegible] u [illegible] $I_{\Pi_n} u$ [illegible]

The projection, $I_{\Pi_n} u$, can be computed once the partition Π_n is known. Express $I_{\Pi_n} u$ as a series sum of a complete basis of P^n, $\{p_1(x,y), \ldots, p_N(x,y)\}$:

$$I_{\Pi_n} u = \sum_{i=1}^{N} a_i\, p_i(x,y). \tag{3}$$

Denote $\bar{u}_i$ as the average of $u(x,y)$ over the subcell C_i, i.e.,

$$\bar{u}_i = \frac{1}{V_i}\int_{C_i} u(x,y)\, dV, \qquad i = 1,\ldots,N, \tag{4}$$

where V_i is the area of C_i. Plug (3) and (4) into (2), and rewrite the new equation in a matrix form

$$A\, a \;=\; \bar{u}, \tag{5}$$

where $\bar{u} = (\bar{u}_1,\ldots,\bar{u}_N)^T$, $a = (a_1,\ldots,a_N)^T$, and the reconstruction matrix A takes the form

$$A = \begin{pmatrix} \frac{1}{V_1}\int_{C_1} p_1(x,y)dV & \cdots & \frac{1}{V_1}\int_{C_1} p_N(x,y)dV \\ \cdots & \cdots & \cdots \\ \frac{1}{V_N}\int_{C_N} p_1(x,y)dV & \cdots & \frac{1}{V_N}\int_{C_N} p_N(x,y)dV \end{pmatrix}. \tag{6}$$

When the partition is nonsingular, i.e., the matrix A is nonsingular, we solve (5) and substitute the solution a back into (3) to obtain

$$I_{\Pi_n} u = \sum_{i=1}^{N} \bar{u}_i\, L_i(x,y), \tag{7}$$

where the cardinal basis functions $L = (L_1(x,y),\ldots,L_N(x,y))$ are given as

$$L = (p_1(x,y),\ldots,p_N(x,y))\; A^{-1}. \tag{8}$$

Then we equip the space P^n and $C^0(S^2)$ with an L^∞ norm (supremum-norm, denoted as $\|\cdot\|$) and the induced functional norm

$$\|I_{\Pi_n}\| \;=\; \sup_{\|u\|\neq 0} \frac{\|I_{\Pi_n} u\|}{\|u\|}.$$

Since $|\bar{u_i}| \le \|u\|$ for $i = 1,\ldots,N$, one can show that

$$\|I_{\Pi_n}\| = \max_{(x,y)\in S^2} \sum_{i=1}^{N} |L_i(x,y)|. \tag{9}$$

2.1. Error of spectral volume reconstruction. One measurement of the quality of spectral volume reconstruction is the error. Similar to that of polynomial interpolation, the error of spectral volume reconstruction is bounded from below as

$$\|u - u^*\| \le \|u - I_{\Pi_n} u\|, \tag{10}$$

where u^* is the optimal approximating polynomial whose existence is guaranteed by the continuity of $u(x,y)$ [1]. Although it is difficult to determine such optimal approximation for general functions, its existence enables us to evaluate the quality of other approximations. From the linearity of the projection operator I_{Π_n} and the fact that $I_{\Pi_n} f = f\ \forall f \in P^n(S^2)$, one can verify that

$$\|u - I_{\Pi_n} u\| \le (1 + \Lambda(\Pi_n))\, \|u - u^*\|, \tag{11}$$

where

$$\Lambda(\Pi_n) = \|I_{\Pi_n}\| = \max_{(x,y)\in S^2} \sum_{i=1}^{N} |L_i(x,y)| \tag{12}$$

is called the Lebesgue constant of the operator I_{Π_n}.

From the manner of computing $I_{\Pi_n} u$ as previously described, one can show that the Lebesgue constant only depends on the partition Π_n when the approximation space is fixed.

LEMMA 1. [illegible]

[illegible]. From the definition of the Lebesgue constant (cf. (12)), it is enough to show that the same cardinal basis functions will be obtained for two different basis sets of the approximation space.

Choose another basis set, $(q_1(x,y),\ldots,q_N(x,y)) = (p_1(x,y),\ldots,p_N(x,y))\cdot T$, where T is a constant nonsingular matrix. According to (6), the new reconstruction matrix is

$$\tilde{A} = A \cdot T.$$

Therefore the new cardinal basis is

$$\tilde{L} = (q_1,\ldots,q_N)\cdot\left(\tilde{A}\right)^{-1} = (p_1,\ldots,p_N)\cdot T\cdot T^{-1}A^{-1} = L,$$

which proves the lemma. □

According to (11) and (10), the magnitude of the Lebesgue constant reflects how close the spectral volume reconstruction is to the optimal polynomial approximation. Therefore, spectral volume reconstructions with small Lebesgue constant are preferred. For simplicity, we consider only partitions with subcells being convex polygons with straight edges. However, this might keep us from obtaining spectral volume reconstructions with Lebesgue constant as small as those of polynomial interpolation in [10] and [9].

Another important issue is the work load when the spectral volume reconstruction procedure is used in solving partial differential equations. As shown in [6, 4], the work load is roughly proportional to the total number of edges of the partition. Hence the optimal partition should have a minimum number of edges and lead to the smallest Lebesgue constant. If one wants to optimize the partition, one needs to minimize both the number of edges and the Lebesgue constant at the same time, which apparently is not an easy task. We made no such effort in this paper.

We also want to emphasize that it is necessary that the reconstruction matrix A is well conditioned for high-order spectral volume reconstructions because of the finite precision of computers. For polynomial interpolation on a triangle, this is usually achieved by choosing the Dubiner basis [2] instead of the notorious monomials, provided the nodal set is good.

3. Partitions from Voronoi diagram and its variants. In this section, we describe a few systematic techniques to partition a one- and two-dimensional simplex by using the Voronoi diagram and its variants.

3.1. The Voronoi diagram and its variants. The following definition is a generalization of the two-dimensional Voronoi diagram [3, 11].

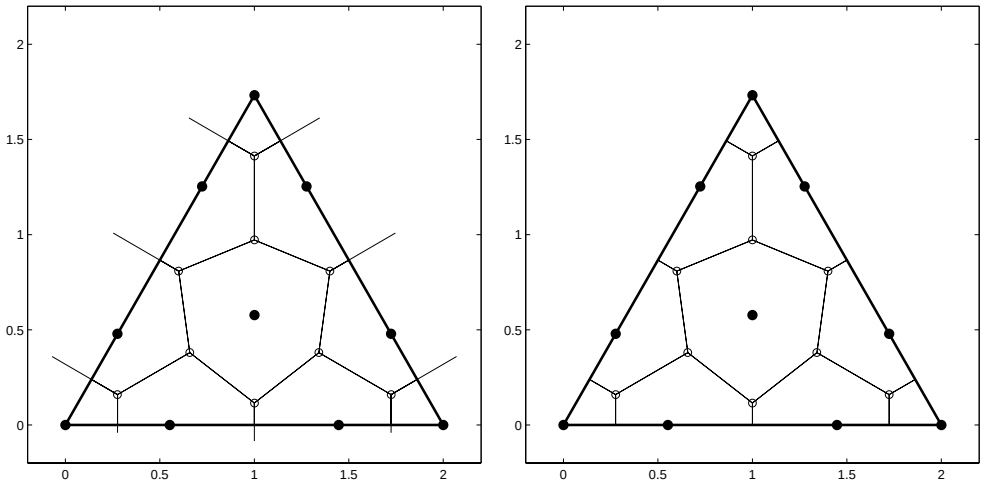


FIG. 1. *Partition of a triangle from the Voronoi diagram of given points on the triangle. On the left graph, the dark dots are the input points, the circles are the Voronoi vertices, and the thin solid lines are the Voronoi edges. The right graph displays the partition.*

DEFINITION 2 (Voronoi diagram). N $S = \{p_1, \ldots, p_N\}$ R^m R^m N $\{V_1, \ldots, V_N\}$ $p_i \in V_i$ $i = 1, \ldots, N$ V_i R^m p_i S.

$$V_i = \{x \in R^m \ : \ |x - p_i| \leq |x - q|, \forall q \in S - p_i\}, \tag{13}$$

$|x-y|$ x y

All Voronoi cells and faces form a cell complex whose vertices and edges are called and . The unbounded edges are also called . When the input points, $\{p_1, \ldots, p_N\}$, are in the plane ($m = 2$), we can bound the number of the Voronoi vertices and Voronoi edges by the following theorem [3, 11].

THEOREM 3. $N \geq 3$ N $2N-5$ $3N-6$

When the input points are on a triangle, it is straightforward to generate a partition of the triangle from the Voronoi diagram, e.g., Figure 1. Thus, we can partition a triangle by choosing the input points and computing the Voronoi diagram. Furthermore, each Voronoi vertex (the circles in Figure 1) is the circumcenter of one triangle with vertices being three input points. One can get a few variants of the Voronoi diagram, and thus different partitions, by replacing each Voronoi vertex (circumcenter) with the corresponding incenter, centroid, or any other point related to that triangle. We recall that for a triangle the circumcenter is the center of its circumcircle, the incenter is the center of its incircle, and the centroid is the intersection of the triangle's three triangle medians.

3.2. Partitions of S^1. S^1 is simply a line segment. Without loss of generality, let $S^1 = [-1, 1]$. One can generate a partition from any given input points set as follows. Suppose $-1 = x_0 < x_1 < \cdots < x_n = 1$ are the input points. Take $y_0 = x_0, y_{n+1} = x_n$ and $y_i = (x_i + x_{i-1})/2, i = 1, \ldots, n$. The points $\{y_i\}$ then

TABLE 1
Lebesgue constants for one-dimensional spectral volume reconstruction and interpolation. n: the order of spectral volume reconstruction or interpolation.

n	Λ^{LGL}	$\Lambda_{\mathrm{V}}^{\mathrm{LGL}}$	Λ^{CGL}	$\Lambda_{\mathrm{V}}^{\mathrm{CGL}}$	$\Lambda_{\mathrm{V}}^{\mathrm{Eq}}$
2	1.2500	2.6667	1.2500	2.6667	3.3333
3	1.5000	3.0299	1.6667	2.8095	5.3333
4	1.6359	3.2702	1.7987	2.8607	8.5333
5	1.7786	3.4451	1.9889	2.8846	13.8666
6	1.8737	3.5800	2.0825	2.8976	23.0095
7	1.9724	3.6881	2.2022	2.9054	39.0095
8	2.0456	3.7773	2.2747	2.9106	67.4539
9	2.1210	3.8526	2.3619	2.9141	118.6539
10	2.1805	3.9172	2.4210	2.9439	211.7448
11	2.2415	3.9735	2.4894	3.0627	382.4115
12	2.2917	4.0230	2.5393	3.1503	697.4884
13	2.3428	4.0671	2.5957	3.2478	1282.6313
14	2.3862	4.1067	2.6388	3.3229	2374.8979
15	2.4303	4.1424	2.6867	3.4054	4422.8979
16	2.4684	4.1749	2.7247	3.4709	8227.9568
17	2.5072	4.2047	2.7664	3.5422	15559.7345
18	2.5412	4.2320	2.8003	3.6004	29356.7872
19	2.5758	4.2573	2.8371	3.6631	55571.1872
20	2.6066	4.2807	2.8678	3.7154	105503.3776

define a partition: $V_i = [y_i, y_{i+1}], i = 0, \ldots, n$. A different points set will yield a different partition. Table 1 includes the Lebesgue constants corresponding to the Legendre–Gauss–Lobatto (LGL) and Chebyshev–Gauss–Lobatto (CGL) quadrature points. Note that the LGL points are actually the Fekete points in the interval [12]. Λ^{LGL} and Λ^{CGL} represent the Lebesgue constants of polynomial interpolation with the LGL and CGL points. $\Lambda_{\mathrm{V}}^{\mathrm{LGL}}$ and $\Lambda_{\mathrm{V}}^{\mathrm{CGL}}$ represent the Lebesgue constants for the reconstruction on the partitions with LGL and CGL points being the input. $\Lambda_{\mathrm{V}}^{\mathrm{Eq}}$ denotes the Lebesgue constant for spectral volume reconstruction on the uniform mesh; i.e., all subcells have the same size.

Note that the Lebesgue constants of spectral volume reconstruction on the partitions from both LGL and CGL points are less than twice those of polynomial interpolation, which are very close to the Lebesgue constants of the optimal nodal set [9]. In summary,

$$\Lambda_{\mathrm{V}}^{\mathrm{Eq}} > \Lambda_{\mathrm{V}}^{\mathrm{LGL}} > \Lambda_{\mathrm{V}}^{\mathrm{CGL}} > \Lambda^{\mathrm{CGL}} > \Lambda^{\mathrm{LGL}}. \tag{14}$$

3.3. Partitions of S^2. We study only the standard equilateral triangle S^2 (cf. Figure 1) because any other triangle can be obtained from a linear transformation of this triangle. The linear transformation will not change the Lebesgue constant of spectral volume reconstruction, which will be shown below.

LEMMA 4. [illegible] S^2 [illegible] $\Pi_n = \{C_1, \ldots, C_N\}$ [illegible] $\tilde{S}$ [illegible] $\xi - \eta$ [illegible] $\tilde{S}$ $\tilde{\Pi}_n = \{\tilde{C}_1, \ldots, \tilde{C}_N\}$ [illegible] S^2 [illegible] $\tilde{S}$ [illegible] Π_n [illegible] $\tilde{S}$ [illegible] S^2 [illegible]. Denote $\tilde{V}_i$ as the area of subcell $\tilde{C}_i$. Use $\tilde{A}$ and $\tilde{L}$ to represent the reconstruction matrix and cardinal basis set on $\tilde{S}$. For S^2, the same notation will be used as in section 2.

From the definition of the Lebesgue constant, it suffices to show that

$$\tilde{L}(\xi,\eta) = L(x,y) \qquad \forall (\xi,\eta) \in \tilde{S} \text{ or } (x,y) \in S^2.$$

According to (8),

$$\begin{cases} L(x,y) = (p_1(x,y),\ldots,p_N(x,y))\, A^{-1}, \\ \tilde{L}(\xi,\eta) = (p_1(\xi,\eta),\ldots,p_N(\xi,\eta))\, \tilde{A}^{-1}. \end{cases}$$

Since the transformation $(\xi,\eta) \leftarrow (x,y)$ is linear, there exists a constant matrix T such that

$$(p_1(\xi,\eta),\ldots,p_N(\xi,\eta)) = (p_1(x,y),\ldots,p_N(x,y)) \cdot T. \tag{15}$$

T is also nonsingular as the transformation is nonsingular. (The nonsingularity of T can be proved by evaluating (15) at N distinct points.)

Plug (15) into the reconstruction matrix

$$\tilde{A} = \begin{pmatrix} \frac{1}{\tilde{V}_1}\int_{\tilde{C}_1} \tilde{p}_1(\xi,\eta)d\tilde{V} & \cdots & \frac{1}{\tilde{V}_1}\int_{\tilde{C}_1} \tilde{p}_N(\xi,\eta)d\tilde{V} \\ \cdots & \cdots & \cdots \\ \frac{1}{\tilde{V}_N}\int_{\tilde{C}_N} \tilde{p}_1(\xi,\eta)d\tilde{V} & \cdots & \frac{1}{\tilde{V}_N}\int_{\tilde{C}_N} \tilde{p}_N(\xi,\eta)d\tilde{V} \end{pmatrix}$$

to obtain

$$\tilde{A} = A \cdot T.$$

Hence,

$$\begin{aligned} \tilde{L}(\xi,\eta) &= (p_1(\xi,\eta),\ldots,p_n(\xi,\eta))\, \tilde{A}^{-1} \\ &= (p_1(x,y),\ldots,p_N(x,y)) \cdot T \cdot (A \cdot T)^{-1} = L(x,y). \end{aligned}$$ □

The first kind of partitions, denoted as $\Pi^{\mathrm{F}}_{\mathrm{out}}$, are from the Voronoi diagram[1] of the two-dimensional Fekete points [10] on the triangle. As previously mentioned, each Voronoi vertex is the circumcenter of one triangle with vertices being three input points. Therefore, there might be some Voronoi vertices which are outside the big equilateral triangle. When this happens, the partition is not as obvious as shown in Figure 1. But as demonstrated in Figure 2, we can obtain a partition by replacing each outside Voronoi vertex with the intersection point of the corresponding Voronoi ray and the edge of the big equilateral triangle. With this strategy, we generate partitions up to 14th order (Figure 3). The Lebesgue constants of spectral volume reconstruction are listed in Table 2. The number of distinct edges is bounded as follows.

THEOREM 5. [illegible] $\Pi^{\mathrm{F}}_{\mathrm{out}}$ [illegible]. For the nth order partition $\Pi^{\mathrm{F}}_{\mathrm{out}}$, according to Theorem 3, the Voronoi diagram contributes at most $(3N-6)$ to the total number of edges, where $N = \binom{2+n}{2}$ (cf. (1)) is the total number of input points on the triangle. In addition, there are

[1]Fortune's code is used to compute the two-dimensional Voronoi diagram.

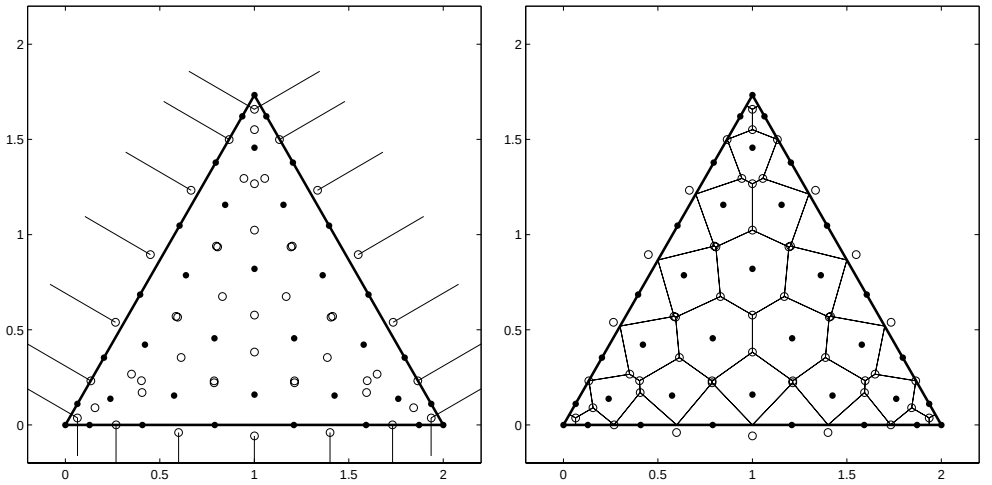


FIG. 2. *The seventh order partition from the Voronoi diagram of given Fekete points. On the left graph, the thin solid lines are the Voronoi rays; the circles are the Voronoi vertices. The right graph is the partition. The outside Voronoi vertices are also plotted for clarity.*

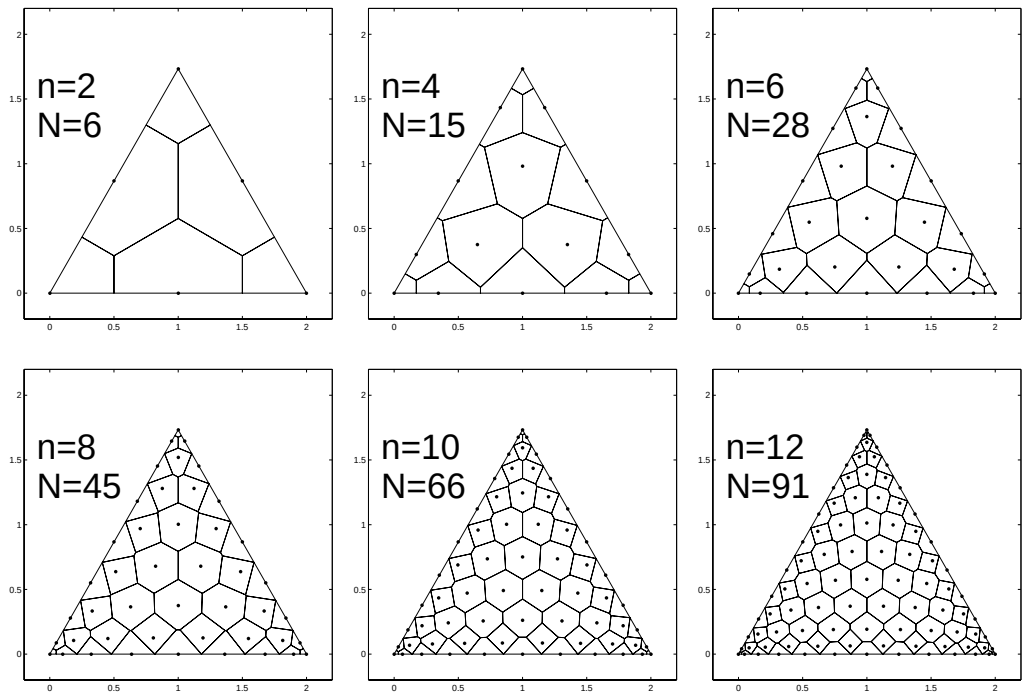


FIG. 3. *Partition* $\Pi^{\mathrm{F}}_{\mathrm{out}}$. *n: order of spectral volume reconstruction; N: number of subcells; ".": input Fekete points on the triangle.*

TABLE 2

Lebesgue constants for two-dimensional spectral volume reconstruction and interpolation. Λ^{F} *stands for the Lebesgue constant for the interpolations with Fekete points.* $\Lambda^{\mathrm{F}}_{\mathrm{out}}$, $\Lambda^{\mathrm{F}}_{\mathrm{in}}$, $\Lambda^{\mathrm{F}}_{\mathrm{mass}}$, *and* $\Lambda^{\mathrm{Eq}}_{\mathrm{V}}$ *represent the Lebesgue constants for the spectral volume reconstruction on the partition* $\Pi^{\mathrm{F}}_{\mathrm{out}}$, $\Pi^{\mathrm{F}}_{\mathrm{in}}$, $\Pi^{\mathrm{F}}_{\mathrm{mass}}$, *and* $\Pi^{\mathrm{Eq}}_{\mathrm{V}}$, *respectively.*

n	N_n	Λ^{F}	$\Lambda^{\mathrm{F}}_{\mathrm{out}}$	$\Lambda^{\mathrm{F}}_{\mathrm{in}}$	$\Lambda^{\mathrm{F}}_{\mathrm{mass}}$	$\Lambda^{\mathrm{Eq}}_{\mathrm{V}}$
2	6	1.6600	4.1673	4.1673	4.1673	4.167
3	10	2.1053	4.8374	4.8999	4.8797	6.031
4	15	2.7227	6.5407	5.6978	5.7719	8.768
5	21	3.5950	9.5786	7.2547	7.3822	12.911
6	28	4.1706	12.3091	8.3994	8.5399	19.384
7	36	4.9271	17.1220	10.1359	10.3084	29.774
8	45	5.8785	21.1783	21.3010	11.8001	46.859
9	55	6.8006	27.4297	24.8328	13.9493	75.523
10	66	7.9620	33.2994	28.0529	16.2751	124.448
11	78	9.4765	42.1511	33.6964	19.6389	209.165
12	91	11.0552	52.5173	39.8890	23.3799	357.725
13	105	13.2040	67.9809	49.9275	29.3272	620.707
14	120	15.9693	87.1305	61.5641	36.1131	1090.979

$3(n+1)$ edges which lie on the edges of the big equilateral triangle. Thus, the total number of edges of the nth partition $\Pi^{\mathrm{F}}_{\mathrm{out}}$ is at most

$$3N - 6 + 3(n+1) = \frac{3}{2}(n^2 + 5n). \tag{16}$$

Consider an arbitrary partition leading to an nth-order spectral volume reconstruction. Denote X as the number of edges of the partition which are on the edges of the big equilateral triangle, and Y as the number of remaining edges (called , since they are inside the big equilateral triangle). Clearly, $X = 3(n+1)$. Since an nth-order reconstruction needs N polygons, each of which has at least three edges, one can show that

$$X + 2Y \geq 3N,$$

where the coefficient before Y is due to the fact that each inside edge belongs to two polygons. Thus, the minimum number of distinct edges for any nth-order partition satisfies

$$X + Y \geq X + \frac{3N - X}{2} = 3(n+1) + \frac{3N - 3(n+1)}{2} = \frac{3}{4}(n^2 + 5n + 4).$$

Comparing the above equation with (16) proves the theorem. □

However, this simple usage of the Voronoi diagram does not yield very small Lebesgue constants, as shown in Figure 7, for high-order spectral volume reconstructions.

We derive the second kind of partitions (denoted as $\Pi^{\mathrm{F}}_{\mathrm{in}}$) from one variant of the Voronoi diagram in which each Voronoi vertex is replaced by the corresponding incenter. By doing that, the structure of the partitions is more similar to the structure of the input points in the sense of layered structures and concentration near the edges (see Figures 3 and 4). This is due to the fact that the incenter of a triangle is always inside the triangle. We believe that the layered structure and the concentration near the edges of the subcells are crucial for the partition to produce small Lebesgue

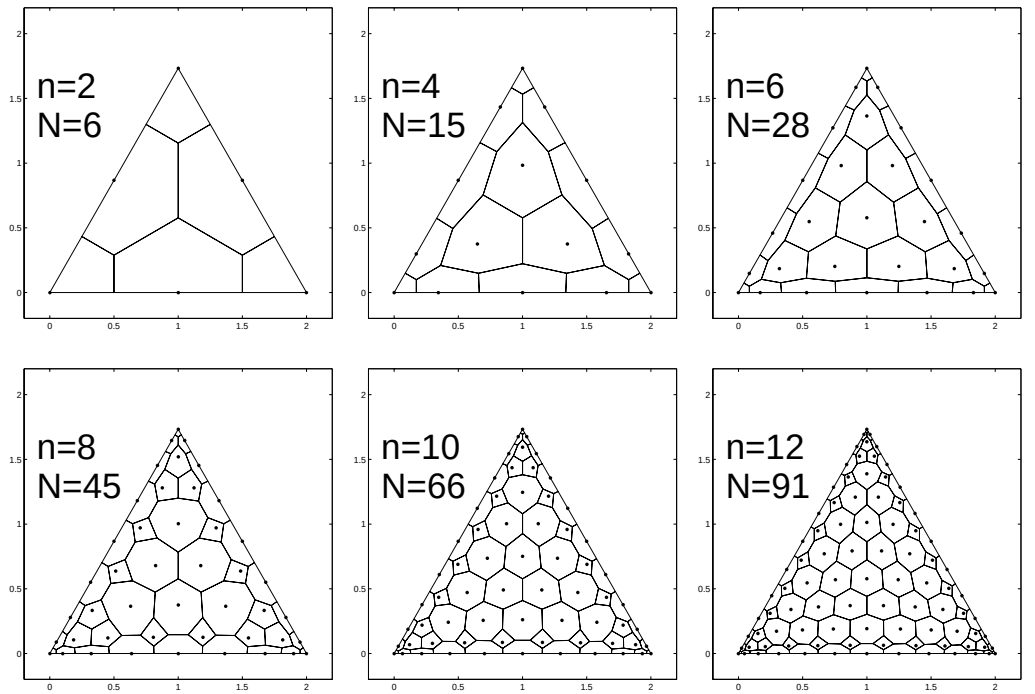


FIG. 4. *Partition* $\Pi^{\mathrm{F}}_{\mathrm{in}}$. *n: order of spectral volume reconstruction; N: number of subcells; ".": input Fekete points on the triangle.*

constants. As expected, the partition $\Pi^{\mathrm{F}}_{\mathrm{in}}$ produces smaller Lebesgue constants than $\Pi^{\mathrm{F}}_{\mathrm{out}}$ for most cases of the spectral volume reconstructions of order up to 14 (Table 2).

Unfortunately, as shown in Figure 7, there is a sudden increase in the Lebesgue constants of the 8th or higher order $\Pi^{\mathrm{F}}_{\mathrm{in}}$ partitions. By examining Figure 4 more carefully, we notice that the layered structure is a bit "distorted" in the place close to the edges of the big triangle. The distortion is responsible for the sudden increase of the Lebesgue constants.

The Fekete points set itself has a very nice structure (Figure 5), based on which we come up with the third kind of partitions, denoted as $\Pi^{\mathrm{F}}_{\mathrm{mass}}$. There are three steps to build the partition, which are demonstrated in Figure 5. At first, we concatenate the input points layer by layer (see the left graph of Figure 5). Then we construct a triangular mesh as shown in the middle graph of Figure 5. Finally, for each input point inside the big triangle, we construct a polygon by connecting the centroids (or any other points) of the small triangles that share the input point (see the right graph of Figure 5). When an input point is on the edge of the big triangle, one can construct a polygon containing it by using two more points on the edges of the big equilateral triangle. Some examples of the partitions are displayed in Figure 6. We also tried to use the incenter instead of centroid of those small triangles. It leads to almost the same Lebesgue constants as the centroid does.

The same upper bound holds for the number of edges of the partitions $\Pi^{\mathrm{F}}_{\mathrm{in}}$ and $\Pi^{\mathrm{F}}_{\mathrm{mass}}$.

THEOREM 6. [illegible] $\Pi^{\mathrm{F}}_{\mathrm{in}}$ [illegible] $\Pi^{\mathrm{F}}_{\mathrm{mass}}$ [illegible]

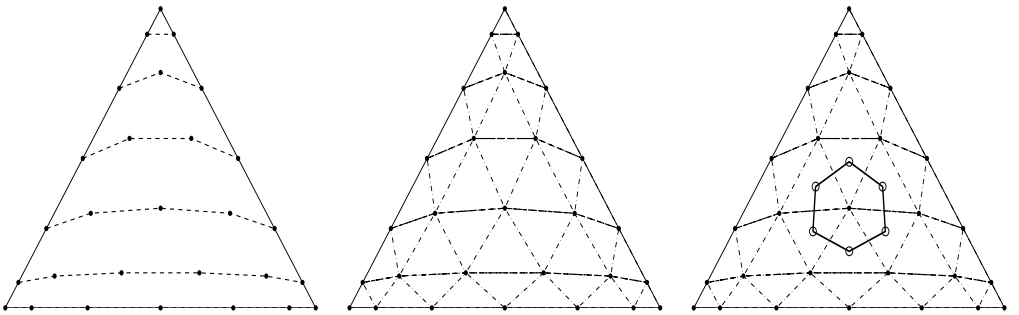


FIG. 5. *Three steps to construct the partition* $\Pi^{\mathrm{F}}_{\mathrm{out}}$. *".": input Fekete points. Left: the layered structure of Fekete points. Middle: the triangular mesh. Right: the method to build the polygon (thick line) for one input point. "o": vertices of the polygon.*

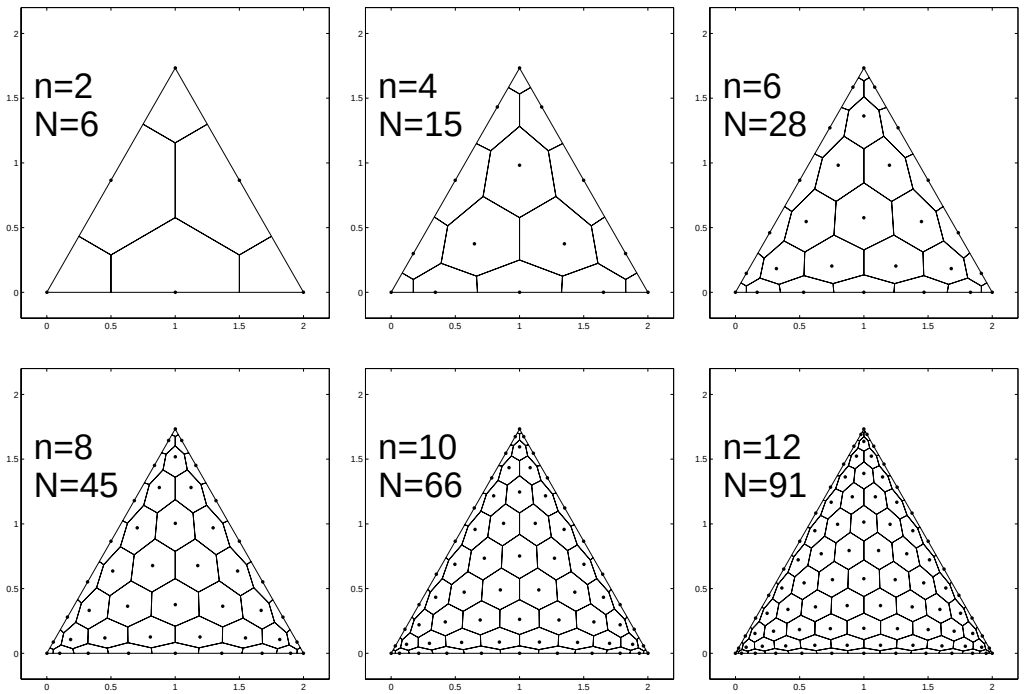


FIG. 6. *Partition* $\Pi^{\mathrm{F}}_{\mathrm{mass}}$. *n: order of spectral volume reconstruction; N: number of subcells; ".": input Fekete points on the triangle.*

The proof is omitted as it is basically the same as that of Theorem 5.

For comparison, we also compute the Lebesgue constants of the partition $\Pi^{\mathrm{Eq}}_{\mathrm{V}}$, which is from the Voronoi diagram of equispaced (in the area coordinate system) points on the equilateral triangle. The Lebesgue constants for all partitions are listed in Table 2. Table 3 contains the 2-norm condition numbers of the reconstruction matrix when the Dubiner basis is used. Figure 7 displays the ratios of the Lebesgue constants from the above partitions to those of the polynomial interpolations based on the Fekete points.

TABLE 3
The 2-norm condition numbers of the spectral volume reconstruction matrix on the two-dimensional partitions.

n	N_n	$\Pi^{\mathrm{F}}_{\mathrm{out}}$	$\Pi^{\mathrm{F}}_{\mathrm{in}}$	$\Pi^{\mathrm{F}}_{\mathrm{mass}}$	$\Pi^{\mathrm{Eq}}_{\mathrm{V}}$
2	6	1.8556	1.8556	1.8556	1.8556
3	10	3.4387	3.2403	3.2541	2.7936
4	15	4.4622	4.1734	4.1895	3.2358
5	21	5.9947	5.2375	5.2674	3.8526
6	28	8.0695	6.8737	6.9193	4.7738
7	36	9.8388	8.5045	8.5158	6.2232
8	45	12.0599	14.2730	10.9258	8.5089
9	55	15.0423	16.9076	13.6832	12.3057
10	66	19.2063	21.3786	17.7896	18.5947
11	78	25.3897	26.9520	23.3292	29.2915
12	91	34.1437	35.6501	31.4112	47.5801
13	105	47.1800	47.8854	42.8654	79.2277
14	120	65.8074	66.3107	59.6859	134.6209

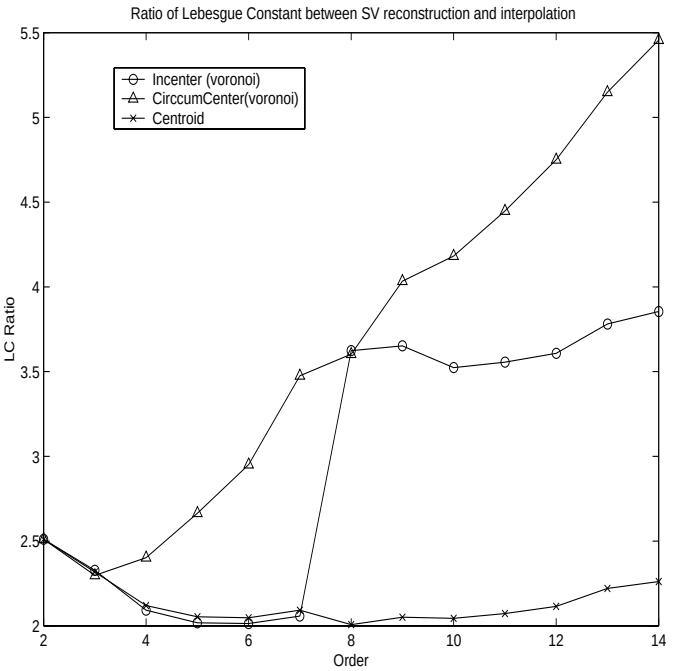


FIG. 7. *The ratio of the Lebesgue constants of two-dimensional spectral volume reconstructions to those of the interpolation on Fekete points. "△": $\Pi^{\mathrm{F}}_{\mathrm{out}}$ partition; "o": $\Pi^{\mathrm{F}}_{\mathrm{in}}$ partition; "x": $\Pi^{\mathrm{F}}_{\mathrm{mass}}$ partition.*

4. Conclusions. We have developed several systematic techniques to partition a one- and two-dimensional simplex by using the well-known Voronoi diagram and its variants. The resulting partitions have layered structure and the subcells concentrate near the edges. These two properties are found to be crucial for the partitions which lead to spectral volume reconstructions with small Lebesgue constants.

The spectral volume reconstructions on those partitions have small Lebesgue constants, one of which is roughly twice the Lebesgue constant of the same order interpolation based on the almost optimal nodal sets. The total number of edges (the total work when being used in the spectral volume method) of the partitions is shown to be at most twice the minimum number of edges of all partitions for the reconstructions of the same order accuracy. When using the Dubiner basis, the spectral volume reconstruction matrix is very well conditioned. All of these suggest that the partitions are a good choice for the spectral volume methods and other numerical methods which rely on reconstructions from cell averages.

Acknowledgments. The author thanks Prof. Franco P. Preparata for many helpful suggestions and greatly appreciates the guidance and support of Prof. David Gottlieb and Prof. Jan S. Hesthaven. The author would also like to thank Prof. Z. J. Wang; this project arose from his seminar.

REFERENCES

[1] P. J. Davis, *Interpolation and Approximation*, Dover, New York, 1975.
[2] M. Dubiner, *Spectral methods on triangles and other domains*, J. Sci. Comput., 6 (1991), pp. 345–390.
[3] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
[4] Z. J. Wang, *Spectral (finite) volume method for conservation laws on unstructured grids: Basic formulation*, J. Comput. Phys., 178 (2002), pp. 210–251.
[5] M. Zhang and C.-W. Shu, *An analysis of and a comparison between the discontinuous Galerkin and the spectral finite volume methods*, Comput. & Fluids, 34 (2005), pp. 581–592.
[6] Z. J. Wang and Y. Liu, *Spectral (finite) volume method for conservation laws on unstructured grids* II: *Extension to two-dimensional scalar equation*, J. Comput. Phys., 179 (2002), pp. 665–697.
[7] Q. Chen and I. Babuška, *Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle*, Comput. Methods Appl. Mech. Engrg., 128 (1995), pp. 405–417.
[8] Z. J. Wang and Y. Liu, *Spectral (finite) volume method for conservation laws on unstructured grids* III: *One-dimensional systems and partition optimization*, J. Sci. Comput., 20 (2004), pp. 137–157.
[9] J. S. Hesthaven, *From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex*, SIAM J. Numer. Anal., 35 (1998), pp. 655–676.
[10] M. A. Taylor, B. A. Wingate, and R. E. Vincent, *An algorithm for computing Fekete points in the triangle*, SIAM J. Numer. Anal., 38 (2000), pp. 1707–1720.
[11] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, 2nd ed., Springer-Verlag, New York, 2000.
[12] L. Fêjer, *Bestimmung derjenigen abszissen eines intervalles, für welche die quadratsumme der grundfunktionen der lagrangeschen interpolation im intervalle* $[-1, +1]$ *ein möglichst kleine maximum besitzt*, Ann. Scuola Norm. Sup. Pisa, 1 (1932), pp. 3–16.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1471–1492
© 2006 Society for Industrial and Applied Mathematics

A NOVEL MULTIGRID BASED PRECONDITIONER FOR HETEROGENEOUS HELMHOLTZ PROBLEMS*

Y. A. ERLANGGA†, C. W. OOSTERLEE†, AND C. VUIK†

Abstract. An iterative solution method, in the form of a preconditioner for a Krylov subspace method, is presented for the Helmholtz equation. The preconditioner is based on a Helmholtz-type differential operator with a complex term. A multigrid iteration is used for approximately inverting the preconditioner. The choice of multigrid components for the corresponding preconditioning matrix with a complex diagonal is validated with Fourier analysis. Multigrid analysis results are verified by numerical experiments. High wavenumber Helmholtz problems in heterogeneous media are solved indicating the performance of the preconditioner.

Key words. Helmholtz equation, nonconstant high wavenumber, complex multigrid preconditioner, Fourier analysis

AMS subject classifications. 65N55, 65F10, 65N22, 78A45, 76Q05

DOI. 10.1137/040615195

1. Introduction.

In this paper we present a novel preconditioner for high wavenumber Helmholtz problems in heterogeneous media. The preconditioner is based on the Helmholtz operator, where an imaginary term is added. This preconditioner can be handled by multigrid. This is somewhat surprising as multigrid, without enhancements, has convergence troubles for the original Helmholtz operator at high wavenumbers.

A part of this paper is therefore reserved for the analysis of the multigrid method for Helmholtz problems with a complex zeroth order term. This is done, for constant wavenumbers, by means of Fourier analysis. The preconditioned system leads to a favorably clustered spectrum for a Krylov subspace convergence acceleration. As the preconditioner is not based on a regular splitting of the original Helmholtz problem, it must be used in the setting of Krylov subspace methods. The particular example presented can be viewed as a generalization of the work by Bayliss, Goldstein, and Turkel [3] from the 1980s, where the Laplacian was used as a preconditioner for Helmholtz problems. This work has been generalized by Laird and Giles [17], proposing a Helmholtz preconditioner with a $+$ sign in front of the Helmholtz term. In [13] we have proposed a preconditioner with a purely imaginary shift added to the Laplacian. The method here is an improvement of that method.

In this paper we benefit from Fourier analysis in several ways. First of all, for idealized (homogeneous boundary conditions, constant coefficients) versions of the preconditioned system it is possible to visualize its spectrum for different values of the wavenumber, as Fourier analysis provides all eigenvalues. Second, for analyzing multigrid algorithms quantitatively, Fourier smoothing, two-, and three-grid analysis [6, 7, 23, 24, 30] are the tools of choice.

*Received by the editors September 16, 2004; accepted for publication (in revised form) May 19, 2005; published electronically January 27, 2006. The research is financially supported by Dutch Ministry of Economic Affairs project BTS01044.

http://www.siam.org/journals/sisc/27-4/61519.html

†Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands (y.a.erlangga@math.tudelft.nl, c.w.oosterlee@math.tudelft.nl, c.vuik@math.tudelft.nl).

TABLE 1
Number of grid points employed, related to the wavenumber, so that $kh = 0.625$.

k	40	50	80	100	150	200	500	600
h	1/64	1/80	1/128	1/160	1/240	1/320	1/800	1/960

The outline of this paper is as follows. In section 2 the Helmholtz problem is introduced and the convergence difficulties of multigrid for this equation are detailed. The new preconditioner is introduced in section 3, where multigrid components for Helmholtz problems with a complex term (smoothing, operator-dependent prolongation) are presented. Fourier analysis to obtain quantitative performance estimates of components and methods is performed in section 4. Numerical experiments on two-dimensional high wavenumber heterogeneous Helmholtz problems are presented in section 5.

2. Helmholtz equation, standard multigrid. Consider the Helmholtz equation for a wave problem in a heterogeneous medium

$$\mathcal{A}\phi := -\partial_{xx}\phi - \partial_{yy}\phi - (1-\alpha i)k^2(x,y)\phi = g(x,y) \quad \text{in } \Omega \subset \mathbb{R}^2. \tag{1}$$

Here, $\phi = \phi(x,y)$ represents the solution, usually a pressure field, and g represents the source term. The medium is barely attenuative if $0 \le \alpha << 1$, with α indicating the fraction of damping in the medium ($i = \sqrt{-1}$, the imaginary unit). In geophysical applications, which are our main interest, this damping can be set up to 5% ($\alpha = 0.05$). The wavenumber $k = \omega_f/c$ is space-dependent because of a spatially dependent speed of sound $c(x,y)$ in a heterogeneous medium. With $\omega_f := 2\pi f$ the angular frequency (f is the frequency), wavelength ℓ is defined by $\ell = c/f$. The number of wavelengths in a domain of size L equals L/ℓ. n_w, the number of points per wavelength, is typically chosen to be 10–12 points. Wavenumber k can be large.

The dimensionless wavenumber $\overline{k}$ on a nondimensional $[0,1]^2$ domain is defined by $\overline{k} = 2\pi f L/c$. A dimensionless discretization step reads $h = \ell/(n_w L)$, and therefore for the angular frequency one finds $\omega_f = 2\pi/(n_w h) = 2\pi L/\ell$. With domain size $L = 1$, an accuracy requirement for second order discretizations is that $kh \le \pi/5 (\approx 0.63)$ for $n_w = 10$ points per wavelength, and $kh \le 0.53$ with $n_w = 12$ points per wavelength. In Table 1, the number of grid points used for several wavenumbers k is displayed. For each combination we have $kh = 0.625$. These mesh refinements assume a linear connection between k and h. However, in order to avoid a reduction of accuracy for a second order scheme due to the so-called [illegible] [26, 14] k^2h^3 should be set constant. As for an iterative solution method, keeping kh constant is more severe; we stay with kh as in Table 1. In this paper we emphasize the iterative solution rather than the accuracy of the discretization.

Typically, boundary conditions at the boundary $\Gamma = \partial\Omega$ are in the form of first- or second-order absorbing boundary conditions or of a perfectly matched layer (PML). We use approximate radiation (or nonreflecting) boundary conditions at an artificial boundary. The well-known second-order radiation boundary condition [12], to avoid unphysical reflections at boundaries, reads

$$\mathcal{A}_\Gamma\phi := \frac{\partial\phi}{\partial\nu} - ik\phi - \frac{i}{2k}\frac{\partial^2\phi}{\partial\tau^2} = 0 \quad \text{on } \Gamma, \tag{2}$$

with ν the outward normal direction to the boundary and τ pointing in the tangential direction. At the cornerpoints the suggestions in [2] to avoid corner reflections

have been adopted. Whereas these conditions are commonly applied for problems in homogeneous media, they are less obvious for inhomogeneous media with discontinuous wavenumbers at the boundaries. The reason is that these discontinuities may act as unphysical scatterers at the boundaries. This can be avoided by appropriately increasing damping, modeled by the imaginary part in (2). Another natural approach with an inhomogeneous medium at the boundaries may be the use of the PML.

If a discretization is applied to (1) and (2), a linear system of the form

$$A\boldsymbol{\phi} = \boldsymbol{g}, \qquad A \in \mathbb{C}^{N\times N}, \qquad \boldsymbol{\phi}, \boldsymbol{g} \in \mathbb{C}^N, \tag{3}$$

is obtained, where N is the number of unknowns in the computational domain Ω_h. Matrix A has complex components due to the discrete boundary operator (2) and the damping term in (1). A is in general symmetric with eigenvalues in the left and right half-plane, non-Hermitian, and, because of the accuracy requirements, also large for high wavenumbers. However, A is also sparse; its sparsity pattern depends on the discretization method used.

We consider here, in stencil notation, the well-known $O(h^2)$ 5-point discretization stencil:

$$A_h \stackrel{\wedge}{=} \frac{1}{h^2}\begin{bmatrix} & -1 & \\ -1 & 4-(kh)^2(1-\alpha i) & -1 \\ & -1 & \end{bmatrix}. \tag{4}$$

We use matrix and stencil notation simultaneously: Matrix A (3) relates to the discretization of (1), (2), and discrete operator A_h (4) relates to the discretization of (1). The discrete solution is represented by $\boldsymbol{\phi}$ and ϕ_h, respectively. The eigenvalues (for constant k-problems with homogeneous Dirichlet boundary conditions)

$$\begin{aligned} \lambda_h^{\ell,m} &= \widetilde{\lambda}_h^{\ell,m} - k^2(1-\alpha i) \\ &\equiv \frac{2}{h^2}(2-\cos\ell\pi h - \cos m\pi h) - k^2(1-\alpha i) \quad (\ell, m = 1, 2, \ldots, \sqrt{N}-1) \end{aligned} \tag{5}$$

are not equal to zero as long as $k^2(1-\alpha i)$ is not equal to any of the eigenvalues of the corresponding discrete Laplacian $\widetilde{\lambda}_h^{\ell,m}$. Otherwise, the matrix is singular, and its null-space is spanned by the eigenfunctions

$$v_h^{\ell,m} = \sin\ell\pi x \sin m\pi y, \tag{6}$$

with ℓ, m for which $\lambda_h^{\ell,m} = 0$.

2.1. Multigrid convergence for the Helmholtz equation. Textbook multigrid methods are typically set up so that a smoothing method reduces high frequency components of an error between the numerical approximation and the exact discrete solution, and a coarse grid correction handles the low frequency error components. Whereas such methods are easily defined for elliptic Poisson-like equations, this is not the case for the Helmholtz equation without any damping in (1), $\alpha = 0$. Depending on the particular value of k^2, this equation gives rise to both smoothing and coarse grid correction difficulties. The matrix has eigenvalues in only the right half-plane as long as k^2 is less than the smallest eigenvalue of the Laplacian, $\widetilde{\lambda}_h^{1,1}$. For $k^2 > \widetilde{\lambda}_h^{1,1}$, the matrix does not have only positive eigenvalues. Pointwise Jacobi iteration with underrelaxation does not [illegible] in that case, but since its smoothing properties are

satisfactory, the multigrid convergence will deteriorate only gradually for increasing k^2. By the time k^2 approaches the 6th eigenvalue $\widetilde{\lambda}_h^{\ell,m}$ ($k^2 \approx 150$), the standard multigrid method diverges. The Jacobi relaxation now diverges for smooth eigenfrequencies $v_h^{\ell,m}$ with $\widetilde{\lambda}_h^{\ell,m} < k^2$. Consequently, the multigrid method will still converge as long as the coarsest level used is fine enough to represent these smooth eigenfrequencies sufficiently. So, the coarsest level chosen limits the convergence. When k^2 gets larger more variables need to be represented on the coarsest level for standard multigrid convergence. Eventually, this does not result in an $\mathcal{O}(N)$ iterative method.

In addition to this feature, the Helmholtz equation also brings a multigrid coarse grid correction difficulty. Eigenvalues close to the origin may undergo a sign change after discretization on a coarser grid. If a sign change occurs, the coarse grid solution does not give a convergence acceleration to the finer grid problem but gives a severe convergence degradation (or even divergence) instead. In [11] this phenomenon is analyzed and a remedy for the coarse grid correction related to these problematic eigenvalues is proposed. The efficient treatment in [11] is that the multigrid method is combined with Krylov subspace iteration methods. GMRES is proposed as a smoother and as a cure for the problematic coarse grid correction.

Standard multigrid will also fail for k^2-values very close to eigenvalues. In that case subspace correction techniques should be employed [9].

An advanced multigrid based solution method for the Helmholtz equation is the wave-ray multigrid method [8]. The method has been adapted for a first-order system least-squares version of the Helmholtz equation in [18]. Wave-ray multigrid has been developed for Helmholtz problems with constant or smoothly varying wavenumbers. A thorough overview for the numerical solution of the Helmholtz equation is presented in [25].

3. Shifted Laplacian preconditioner. To solve (3), iterative methods based on the Krylov subspace are of interest. In particular, we choose preconditioned Bi-CGSTAB. In [13], Bi-CGSTAB is preferred over other Krylov subspace methods as the convergence for Helmholtz problems is reported typically faster than that of GMRES. We have also tested advanced versions such as Bi-CGSTAB(2) [22] and GMRESR [28], but Bi-CGSTAB remains the method of choice, especially for the Helmholtz equation without damping ($\alpha = 0$). A preconditioner $M \in \mathbb{C}^{N\times N}$ for A is developed such that the preconditioned system

$$AM^{-1}\boldsymbol{\psi} = \boldsymbol{g}, \qquad \boldsymbol{\psi} = M\boldsymbol{\phi}, \tag{7}$$

has better spectral properties than the original system. The preconditioner M proposed here is based on the following operator:

$$\mathcal{M} \equiv -\partial_{xx} - \partial_{yy} - (\beta_1 - \beta_2 i)k^2(x,y), \qquad \beta_1, \beta_2 \in \mathbb{R}, \tag{8}$$

with (β_1, β_2) parameters that can be chosen freely and with i the imaginary unit. Boundary conditions are set identical to those for the original Helmholtz problem (2).

A large imaginary value for the "wavenumber" physically corresponds to adding some form of damping for preconditioning. In the time domain the heat equation is sometimes used as the preconditioner for the Laplacian. Also the Jacobi iteration for the Laplacian can be interpreted as a time stepping procedure for the heat equation. For the wave equation such an iteration is a less common approach. A large imaginary Helmholtz term can be seen as a time-dependent term that is transformed to Fourier space.

If one is interested in only the interior problem without any damping term and real-valued boundary conditions, the original problem would be real-valued. Introducing a complex shift then introduces complex arithmetic into a real problem. Besides the extra work, it necessitates possibly different iterative solvers. Here, however, we are interested in geophysics applications with outgoing waves.

The basic choice in this paper is $(\beta_1, \beta_2) = (1, 1)$. Tuning of multigrid components is especially necessary for $\beta_2 < 1$, for example, for $\beta_2 = 0.5$, to be presented below. In [13] we have proposed a positive [illegible] shift $(\beta_1, \beta_2) = (0, 1)$ to the Laplacian for a satisfactory convergence. Preconditioner (8) is an improvement of this preconditioner with $\beta_1 = 1$.

We perform Fourier analysis to visualize the effect of the choice of (β_1, β_2) in the preconditioner on the clustering of the eigenvalues of the preconditioned system. For this we consider operator (7) with homogeneous Dirichlet boundary conditions, wavenumber k constant, and a discrete version of Helmholtz operator (1), A_h, and of preconditioner (8), M_h. This particular choice of the boundary conditions especially simplifies the analysis. For radiation boundary conditions the Helmholtz operator is nonnormal. Hence eigenvalue analysis alone would not be sufficient for analyzing preconditioned Krylov subspace methods. Here, we perform the analysis as a first indication of what we can expect from the solver to be developed and concentrate on the eigenvalues.

For both A_h and M_h we choose the 5-point stencil, as in (4). The components (6) are eigenfunctions of these discrete operators with constant coefficients. With these eigenfunctions $A_h M_h^{-1}$ is diagonalizable and the eigenvalues are easily determined. In the first tests we do not include damping in A_h, $\alpha = 0$ in (1), (4).

Figure 1 presents spectra of $A_h M_h^{-1}$ for $(\beta_1, \beta_2) = (0, 0)$ (Laplacian preconditioner), $(\beta_1, \beta_2) = (-1, 0)$ (Laird preconditioner [17]), $(\beta_1, \beta_2) = (0, 1)$ (preconditioner from [13]), $(\beta_1, \beta_2) = (1, 1)$ (basic parameter choice), $(\beta_1, \beta_2) = (1, 0.5)$, and $(\beta_1, \beta_2) = (1, 0.3)$ (more advanced parameters). The results are for $k = 40$ ($k^2 = 1600$) and $h = 1/64$. Similar eigenvalue distributions are observed for finer grids.

From the spectra presented with the new preconditioner, the lower pictures of Figure 1 are favorable as their real parts vary between 0 and 1. The Laplacian preconditioner in Figure 1(a) exhibits large isolated eigenvalues; for the Laird preconditioner the eigenvalues in Figure 1(b) are distributed between -1 and 1 on the real axis. The preconditioners with complex Helmholtz terms give rise to a curved spectrum. Whereas the real part of the spectrum in Figure 1(c) still includes a part of the negative real axis, this is not the case for the (β_1, β_2)-preconditioners with $\beta_1 = 1$. The difference between Figures 1(d), 1(e), and 1(f) is that, with a smaller value of β_2, fewer outliers close to the origin are observed. This is favorable for the convergence of the preconditioned Krylov method. The approximate inversion of the preconditioner itself by multigrid, however, will be shown to be harder for smaller values of β_2. In Figure 2 the spectra for $k = 100 (k^2 = 10^4)$ are presented on a grid with $h = 1/160$ for $\beta_1 = 1$ and β_2 varying between 1 and 0.3. The spectra are very similar to those in Figure 1. More eigenvalues lie, however, in the vicinity of the origin due to the higher wavenumber and the correspondingly finer grid. Figure 3 presents the distribution of eigenvalues for the case that 5% damping ($\alpha = 0.05$) is set in $\mathcal{A}$. Parameters in the preconditioner are $(\beta_1, \beta_2) = (1, 0.5)$. Again the 5-point stencil as in (4) is used for discretization. Figure 3(a) presents the spectrum for $k = 40, h = 1/64$, and Figure 3(b) presents the spectrum for $k = 100, h = 1/160$. An interesting observation is that now the eigenvalues move away from the origin into the right half-plane. This

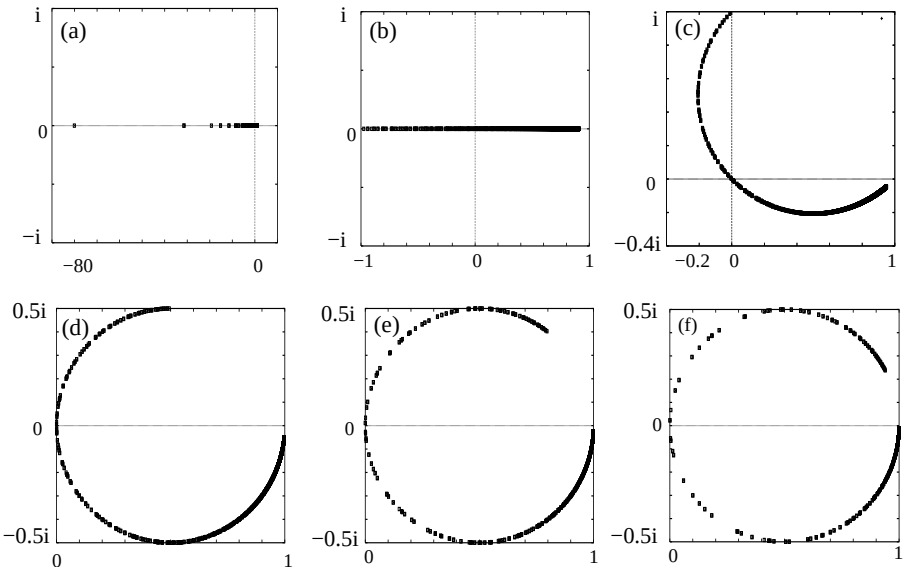


FIG. 1. *Spectral pictures of* $A_h M_h^{-1}$ *with* $\alpha = 0$ *and different values of* (β_1, β_2) *in* (8). (a) $(\beta_1, \beta_2) = (0, 0)$, (b) $(-1, 0)$, (c) $(0, 1)$, (d) $(1, 1)$, (e) $(1, 0.5)$, *and* (f) $(1, 0.3)$.

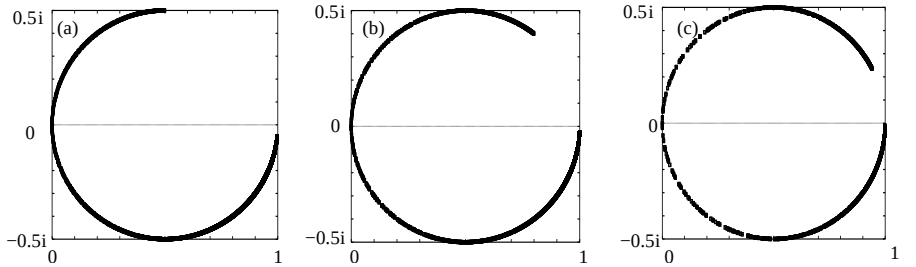


FIG. 2. *Spectral pictures of* $A_h M_h^{-1}$ *for* $k = 100$, $h = 1/160$, *and* $\alpha = 0$; (a) $(\beta_1, \beta_2) = (1, 1)$, (b) $(\beta_1, \beta_2) = (1, 0.5)$, *and* (c) $(\beta_1, \beta_2) = (1, 0.3)$.

is beneficial for iterative solution methods. From the spectra in Figure 3 it is expected that the Bi-CGSTAB (and GMRES) convergence in the case of damping will be considerably faster than for the undamped case.

4. Multigrid for the preconditioner.

4.1. Multigrid components. Geometric multigrid converges satisfactorily for the Helmholtz operator (8) for certain choices of β_1 and β_2 (assumed in [15], see also [16]). In this section, we detail the multigrid components that can be specified for approximately inverting a discrete version of $\mathcal{M}$ in (8). We consider a 5-point discretization and denote the equation for the preconditioner by $M_h \phi_h = \psi_h$. Standard multigrid coarsening, i.e., doubling the mesh size h in every direction, is chosen.

For smoothing the pointwise Jacobi relaxation with underrelaxation (ω-JAC) is chosen. This smoother is well parallelizable, which is an important aspect for our research (w.r.t. a generalization to three dimensions). In principle, one can choose the underrelaxation parameter $\omega \in \mathbb{C}$, but the Fourier analysis indicates that there is no real benefit for the problems considered. So, we choose $\omega \in \mathbb{R}$.

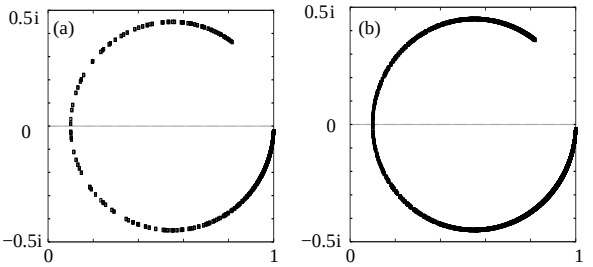


FIG. 3. *Spectral pictures of AM^{-1} with 5 % damping in A and $(\beta_1, \beta_2) = (1, 0.5)$; (a) $k = 40, h = 1/64$ and (b) $k = 100, h = 1/160$.*

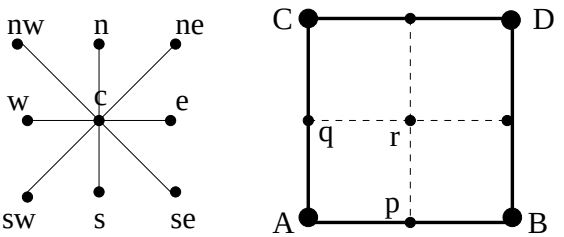


FIG. 4. *Left: Nine point stencil with numbering. Right: Coarse grid cell and four fine cells, (Coarse grid indices designated by capital letters and fine grid indices designated by lower case letters).*

The coarse grid correction components are also based on established operators. For the discrete coarse grid operators $M_{2h}, M_{4h}, \ldots$, the Galerkin coarse grid operator is used:

$$M_{2h} := R_h^{2h} M_h P_{2h}^h, \quad M_{4h} := R_{2h}^{4h} M_{2h} P_{4h}^{2h}, \quad \text{etc.}$$

In the Fourier analysis to follow, this discretization will be compared to a direct coarse grid discretization of (1). The Galerkin coarse grid discretization is a natural choice for heterogeneous problems. Also with boundary conditions containing first and second derivatives, it is convenient to choose the Galerkin coarse grid discretization as it defines the appropriate coarse grid boundary stencils automatically. The transfer operators used in building the coarse grid operators are the same as those used for transferring coarse and fine grid quantities to fine and coarse grids, respectively. The prolongation operator considered is an operator-dependent interpolation based on de Zeeuw's transfer operators [31]. Originally, this prolongation was set up for general (possibly unsymmetric) real-valued matrices with a splitting of matrix M into a symmetric and an antisymmetric part, $M_s = \frac{1}{2}(M + M^T)$, $M_t = M - M_s$ in [31]. However, since the discretization here leads to a complex symmetric matrix, the prolongation is adapted and briefly explained for such matrices with nine diagonals. The numbering in a stencil for the explanation of the prolongation is as shown in Figure 4 (left side). The right side of Figure 4 shows one coarse and four fine grid cells with indices for the explanation of the interpolation weights. Capital letters denote coarse grid points and lower case letters denote fine grid points. Operator element m_p^w, for example, denotes the west element of operator M_h at point p on the fine grid. The corrections from the coarse to the fine grid are obtained by interpolation among nearest coarse grid neighbors. The operator-dependent interpolation weights,

w, to determine the fine grid correction quantities e_h are derived with the following formulas.

- For fine grid points p in Figure 4, $e_{h,p} = w_A e_{H,A} + w_B e_{H,B}$.
  $w_A = \min(1, \max(0, w_w))$; $w_B = \min(1, \max(0, w_e))$,

where

$$d_w = \max(|m_p^{sw} + m_p^w + m_p^{nw}|, |m_p^{sw}|, |m_p^{nw}|), \tag{9}$$

$$d_e = \max(|m_p^{se} + m_p^s + m_p^{ne}|, |m_p^{se}|, |m_p^{ne}|), \tag{10}$$

$$w_w = \frac{d_w}{d_w + d_e}, \qquad w_e = \frac{d_e}{d_w + d_e}. \tag{11}$$

- For fine grid points q in Figure 4, $e_{h,q} = w_A e_{H,A} + w_C e_{H,C}$.
  $w_A = \min(1, \max(0, w_s))$; $w_C = \min(1, \max(0, w_n))$,

with

$$d_n = \max(|m_q^{nw} + m_q^n + m_q^{ne}|, |m_q^{nw}|, |m_q^{ne}|), \tag{12}$$

$$d_s = \max(|m_q^{sw} + m_q^s + m_q^{se}|, |m_q^{sw}|, |m_q^{se}|), \tag{13}$$

$$w_s = \frac{d_s}{d_s + d_n}, \qquad w_n = \frac{d_n}{d_s + d_n}. \tag{14}$$

On the remaining points the prolongation is defined as follows:

$$\text{On fine grid points that are also coarse points, } e_h(A) = e_{2h}(A). \tag{15}$$

$$\text{On points } r,\ e_h(r) \text{ is determined so that } \quad M_h P_{2h}^h e_{2h} = 0 \text{ at } r. \tag{16}$$

The interpolation weights are the same as in [31] but are specially tailored to the symmetric complex Helmholtz equation, i.e., the unsymmetric components in [31] have been removed. $|.|$ denotes the modulus, in this case, leading to real-valued interpolation weights. As for symmetric problems with jumping coefficients, the prolongation operator by de Zeeuw [31] is very similar to the original operator-dependent prolongation in [1]. In [1], for d_w, for example, the lumped sum of three elements, $m_p^{sw} + m_p^w + m_p^{nw}$, is chosen. For satisfactory convergence it is, however, important to consider the [illegible] of the operator elements, as in (9), (10), (12), and (13), in the definition of the interpolation weights. This prolongation is also valid at boundaries.

The full weighting operator is employed as the restriction operator. So, we do not choose the adjoint of the prolongation operator, which is commonly used but is not absolutely necessary, as already stated in [1] (an example where the restriction is not the adjoint of the prolongation operator has been given in [10]). We choose the combination of a full weighting restriction and the operator-dependent interpolation, as it brings a robust convergence for a variety of Helmholtz problems with constant and nonconstant coefficients. For constant coefficients and mildly varying wavenumbers, bilinear interpolation also gives very satisfactory convergence results, but for strongly varying coefficients, as in the Marmousi problem discussed in section 5.3, a robust and efficient convergence on different grid sizes and for many frequencies is observed for the combination of the transfer operators chosen.

4.2. Fourier analysis. Fourier smoothing and two-grid analysis, two classical multigrid analysis tools, have been used for quantitative estimates of the smoothing properties and of the other multigrid components in a two-grid method [5, 6, 7, 23, 24]. Consider a discretization of (7), (8), $M_h \phi_h = \psi_h$, where ϕ_h represents the exact discrete solution. The error $w_h^l = \phi_h^l - \phi_h$ after the lth iteration is transformed by a

cycle as

$$w_h^{l+1} = T_h^{2h}\, w_h^l, \qquad T_h^{2h} = S_h^{\nu_2} K_h^{2h} S_h^{\nu_1}, \qquad K_h^{2h} = I_h - P_{2h}^h (M_{2h})^{-1} R_h^{2h} M_h. \tag{17}$$

M_h, M_{2h} correspond to discretizations of (8) on the h-, $2h$-grid, S_h is the smoothing operator on the fine grid, and I_h is the identity operator. ν_l $(l = 1, 2)$ represents the number of pre- and postsmoothing steps, and R_h^{2h} and P_{2h}^h denote the restriction and prolongation operator, respectively. In the analysis we assume an equidistant grid with $\sqrt{N}$ points in each direction. The $O(h^2)$-discrete complex Helmholtz operator from (8) with constant wavenumber and Dirichlet boundary conditions belongs to the class of symmetric stencils. For these stencils it is possible to apply Fourier analysis on the basis of discrete sine-eigenfunctions $v_h^{\ell,m}$, $\ell, m = 1, \ldots, \sqrt{N} - 1$ (6), instead of the local Fourier analysis with exponential functions. For problems with symmetric stencils and homogeneous Dirichlet boundary conditions, this analysis can predict h-dependent convergence factors. From the discussion of multigrid methods for the original Helmholtz equation, it seems necessary to gain insight into the h-dependency of the multigrid methods developed also for the complex Helmholtz operator. (The definition of the operator-dependent prolongation and the Galerkin coarse grid stencils in section 4.1 also leads to symmetric operators that can be analyzed within this framework.)

For the pointwise Jacobi smoother, the $v_h^{\ell,m}$ (6) are also eigenfunctions of the smoothing operator. This is not true for the two-grid iteration operator T_h^{2h}. However, 4-dimensional linearly independent spaces, the

$$E_h^{\ell,m} = \left[v_h^{\ell,m},\ v_h^{\sqrt{N}-\ell,\sqrt{N}-m},\ -v_h^{\sqrt{N}-\ell,m},\ -v_h^{\ell,\sqrt{N}-m} \right] \quad \text{for} \quad \ell, m = 1, \ldots, \tfrac{\sqrt{N}}{2} \tag{18}$$

are invariant under these operators. One can show [23, 24] that

$$\begin{aligned} &M_h : \operatorname{span}[v_h^{\ell,m}] \to \operatorname{span}[v_h^{\ell,m}], \quad (M_{2h})^{-1} : \operatorname{span}[v_{2h}^{\ell,m}] \to \operatorname{span}[v_{2h}^{\ell,m}], \\ &S_h : \operatorname{span}[v_h^{\ell,m}] \to \operatorname{span}[v_h^{\ell,m}], \\ &R_h^{2h} : E_h^{\ell,m} \to \operatorname{span}[v_{2h}^{\ell,m}], \quad P_{2h}^h : \operatorname{span}[v_{2h}^{\ell,m}] \to E_h^{\ell,m}, \end{aligned}$$

and $T_h^{2h} : E_h^{\ell,m} \to E_h^{\ell,m}$ $(\ell, m = 1, \ldots, \frac{\sqrt{N}}{2})$. Therefore, the representation of T_h^{2h} with respect to $E_h^{\ell,m}$ leads to a block-diagonal matrix, $\widetilde{T}_h^{2h}$,

$$T_h^{2h} \triangleq \left[\widehat{T}_h^{2h}(\ell, m) \right]_{\ell,m=1,\ldots,\frac{\sqrt{N}}{2}} =: \widetilde{T}_h^{2h} \ . \tag{19}$$

Here the blocks $\widehat{T}_h^{2h}(\ell, m)$ are 4×4 matrices if $\ell, m < \frac{\sqrt{N}}{2}$, and are 2×2 (1×1) matrices if either $\ell = \frac{\sqrt{N}}{2}$ or $m = \frac{\sqrt{N}}{2}$ $(\ell = \frac{\sqrt{N}}{2}$ and $m = \frac{\sqrt{N}}{2})$. The two-grid convergence factor is defined as

$$\rho_{2g} := \max_{1 \le \ell, m \le \frac{\sqrt{N}}{2}} \rho\Big(\widehat{T}_h^{2h}(\ell, m) \Big) \ . \tag{20}$$

Thus, the spectral radii of at most 4×4 matrices $\widehat{T}_h^{2h}(\ell, m)$ have to be determined, and their maximum with respect to ℓ and m has to be found.

The definition of the smoothing factor μ is closely related. The smoothing factor measures the reduction of high frequency error components by an iterative method.

It is based on a coarse grid correction operator that annihilates the low frequency error components completely and keeps the high frequency components unchanged. K_h^{2h} is replaced by a projection operator Q_h^{2h} mapping onto the space of high frequencies, i.e., a block-diagonal matrix with $\widehat{Q}_h^{2h}$ at most 4×4-diagonal blocks defined by $\mathrm{diag}(0,1,1,1)$. So, μ is computed as ρ_{2g} (20) with $\widehat{K}_h^{2h}$ in $\widehat{T}_h^{2h}$ replaced by $\widehat{Q}_h^{2h}$.

Recently, three-grid Fourier analysis was proposed in [30]. An issue that can be analyzed in some more detail with a third grid is the coarse grid correction. If a large difference occurs between the two-grid and the three-grid convergence factors, ρ_{2g} and ρ_{3g}, this is an indication for a problematic coarse grid correction. For the complex Helmholtz preconditioner it is important to analyze the coarse grid correction carefully. The error transformation by a three-grid cycle is given by

$$\begin{aligned} & w_h^{l+1} = T_h^{4h}\, w_h^l \quad \text{with} \\ T_h^{4h} = S_h^{\nu_2} K_h^{4h} S_h^{\nu_1} \quad & \text{and} \quad K_h^{4h} = I_h - P_{2h}^h (I_{2h} - (T_{2h}^{4h})^\gamma)(M_{2h})^{-1} R_h^{2h} M_h. \end{aligned} \tag{21}$$

Here T_{2h}^{4h}, defined by (17), reads $T_{2h}^{4h} = S_{2h}^{\nu_2}(I_{2h} - P_{4h}^{2h}(M_{4h})^{-1}R_{4h}^{2h})S_{2h}^{\nu_1}$. M_{4h} corresponds to $4h$-grid discretization of (8); S_{2h} is the smoothing operator, and I_{2h} is the identity on the $2h$-grid; and R_{2h}^{4h} and P_{4h}^{2h} are transfer operators between the different grids. The $2h$-equation is solved approximately in a three-grid cycle (21) by performing γ two-grid iterations T_{2h}^{4h} with zero initial approximation; see also [23, 30].

The three-grid analysis is a recursive application of the two-grid analysis. Four frequencies are coupled not only in the transition from the h- to the $2h$-grid but also in the transition from the $2h$- to the $4h$-grid. Thus the three-grid error transformation operator couples 16 Fourier frequencies. As a consequence, T_h^{4h} is unitarily equivalent to a block-diagonal matrix $\widetilde{T}_h^{4h}$ with at most 16×16 blocks, $\widehat{T}_h^{4h}(\ell, m)$. The block matrices are composed of the Fourier symbols from the two-grid analysis, which is due to the recursive application of the two-grid analysis. One may compute the three-grid factor ρ_{3g} as the supremum of the spectral radii from the 16×16 block matrices, $\widehat{T}_h^{4h}(\ell, m)$.

For more details about the three-grid analysis, we refer to [30]. Three-grid Fourier analysis software, based on the exponential functions, is freely available; see http://www.mgnet.org/mgnet-codes-wienands.html.

4.3. Fourier analysis and multigrid results. We first compare the numerical multigrid convergence with asymptotic convergence factors $\mu, \rho_{2g}, \rho_{3g}$ from Fourier analysis. For this, we consider here solely the preconditioner $\mathcal{M}$ (8). (The behavior of the complete solution method will be considered in the next section.) Wavenumber k is taken as a constant here and a square domain with an equidistant grid is used. The second-order boundary conditions (2) are set in the numerical experiments to mimic reality.

An interesting aspect is that almost identical convergence factors are obtained, both from the analysis and from the actual experiments, for constant values of kh. They are set as in Table 1. The results are validated from $k = 40$ up to $k = 600$, the highest wavenumber tested is ($k^2 = 3.6 \times 10^5$). During testing the following abbreviations are used: "ω-JAC" is the Jacobi smoother with underrelaxation, "Galerkin" is the Galerkin coarse grid discretization, and "direct" is a direct coarse grid discretization of the PDE. "Direct" has not been implemented in the numerical code, but it can be used in the analysis framework.

Multigrid coarsening is continued until fewer than 10×10 points are processed on the coarsest grid. The number of levels is h- and therefore also k-dependent, as

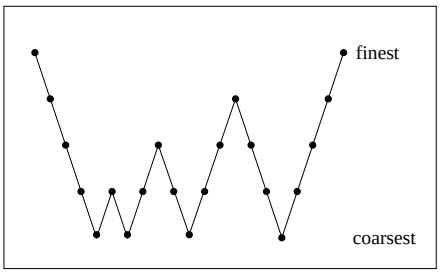


FIG. 5. *An F-cycle for five grids.*

TABLE 2

Comparison of asymptotic convergence from Fourier analysis with numerical multigrid convergence, $(\beta_1, \beta_2) = (0, 1)$. μ *is the smoothing factor;* ρ_{2g}, ρ_{3g} *are the two- and three-grid convergence factors from Fourier analysis; and* ρ_h *is the numerical multigrid convergence factor. The smoother is* ω*-JAC with* $\omega = 0.8$.

(ν_1, ν_2)	μ	ρ_{2g}	ρ_{3g}, $\gamma = 2$	ρ_h, F-cycle
(1,0)	0.60	0.60	0.60	0.58
(1,1)	0.36	0.36	0.36	0.34

kh is kept constant on the finest grid, and varies between 5 and 9 grids.

The F-cycle (see Figure 5) is always used in the numerical tests; the V-cycle's performance was generally too poor; and the W-cycle is considered too expensive on the very fine grids processed at high wavenumbers. The F-cycle often shows the robustness of the W-cycle at the efficiency of the V-cycle. In the three-grid analysis, $\gamma = 2$, the W-cycle analysis is used.

. . . . The Fourier analysis applied directly to the Helmholtz equation (1) with $\alpha = 0$ and the specified mesh sizes gives a satisfactory smoothing factor, but the two- and three-grid analysis convergence factors and also the actual multigrid results show a strong divergence, as expected.

. . . . $(\beta_1, \beta_2) = (0, 1)$ We start with $(\beta_1, \beta_2) = (0, 1)$, as in [13]. This case is not of the highest interest as a preconditioner, as the Bi-CGSTAB convergence for the corresponding preconditioned system is worse than with $\beta_1 = 1$ (shown in the next section). This case $(\beta_1, \beta_2) = (0, 1)$ serves as a reference for the comparison between Fourier analysis and numerical convergence. The underrelaxation parameter ω is set to $\omega = 0.8$, as this is the optimal choice for the Laplacian [24]. The agreement between the smoothing two- and three-grid Fourier analysis results with one and two smoothing iterations and the numerical convergence is excellent, presented in Table 2. The results obtained are very similar to the convergence factors for the Laplacian with ω-JAC.

. . . . For the case $(\beta_1, \beta_2) = (0, 1)$, one can adopt the well-known multigrid components: direct PDE coarse grid discretization and red-black Gauss–Seidel relaxation. This gives $\rho_{3g} = 0.16$ for $\gamma = 1$ and $\rho_{3g} = 0.08$ for $\gamma = 2$ with two smoothing iterations, very similar to the Laplacian situation. Red-black Gauss–Seidel relaxation is, however, not as robust as the ω-JAC relaxation for the $\beta_1 = 1$ cases. Furthermore, the cost in CPU time on a Linux PC of one red-black Gauss–Seidel iteration is about twice that of a Jacobi iteration.

. . . . $(\beta_1, \beta_2) = (1, 1)$ The second test is for $(\beta_1, \beta_2) = (1, 1)$. In this test we employ ω-JAC smoothing with $\omega = 0.7$ in an F(1,1)-cycle ($\nu_1 = \nu_2 = 1$). It is necessary to adapt the relaxation parameter ω for satisfactory numerical convergence. The

TABLE 3

Comparison of convergence $(\beta_1, \beta_2) = (1, 1)$ with Fourier analysis convergence ($\gamma = 1$), ω-JAC, $\omega = 0.7$, and $F(1,1)$-cycle. Coarse grid discretizations are compared. (The direct discretization has not been implemented).

Coarse discr.	μ	ρ_{2g}	ρ_{3g}, $\gamma = 2$	ρ_h, F(1,1)
Galerkin	0.47	0.47	0.47	0.45
Direct	0.47	0.47	0.47	-

TABLE 4

Fourier analysis convergence factors compared to multigrid convergence $(\beta_1, \beta_2) = (1, 0.5)$. The smoother is ω-JAC with $\omega = 0.5$. (The direct discretization has not been implemented).

Coarse discr.	μ	ρ_{2g}	ρ_{3g}, $\gamma = 2$	ρ_h, F(1,1)
Galerkin	0.60	0.60	0.60	0.61
Direct	0.60	0.60	0.60	-

performance of the ω-JAC smoother is not sensitive with respect to choosing somewhat smaller values of ω. We compare the Galerkin discretization with the direct coarse grid PDE discretization. Analysis results with two smoothing iterations are shown in Table 3, and they are compared to the numerical F(1,1) multigrid convergence.

Convergence factors well below 0.5 are obtained with the F(1,1)-cycle and ω-JAC relaxation with $\omega = 0.7$. The Fourier analysis results with the Galerkin coarse grid discretization are very similar to those obtained with a direct coarse grid PDE discretization.

$(\beta_1, \beta_2) = (1, 0.5)$ The preconditioner of choice in this paper is based on the parameters $(\beta_1, \beta_2) = (1, 0.5)$. For this parameter set it is possible to define a converging multigrid iteration by means of an F(1,1)-cycle, ω-JAC relaxation with $\omega = 0.5$, and a Galerkin coarse grid discretization. The underrelaxation parameter needs to be adapted for a robust convergence for a variety of heterogeneous Helmholtz problems. For values $\beta_2 < 0.5$ it is very difficult to define a satisfactory converging multigrid F(1,1)-cycle with the components at hand. They are therefore not considered.

Table 4 compares the Galerkin with the direct PDE coarse grid discretization. Also here, the operator-dependent interpolation and full weighting restriction are chosen, and two smoothing iterations are applied. The smoothing factors and two- and three-grid factors are very similar, which is an indication for the proper choice of coarse grid correction components for the problems under investigation. The numerical convergence with the F(1,1)-cycle is again very similar to the Fourier results.

In the following three remarks, we explain the satisfactory convergence of a standard multigrid method for the complex Helmholtz equation and $\beta_1 = 1$ with some heuristic arguments. The remarks show that, when β_2 is chosen small but such that the multigrid method still converges, the coarse grid stencils do not represent the fine grid problem well. In particular, the main diagonal operator elements in (22) on different coarse grids are smaller than the off-diagonal elements. This will be the reason why the V-cycle does not perform as well as the F-cycle and why damped Jacobi relaxation is a more robust smoother than red-black Gauss–Seidel relaxation for the case $\beta_2 = 0.5$.

The Fourier symbol of ω-JAC for the complex Helmholtz equation reads

TABLE 5
Smoothing factors μ_h for ω-JAC on different coarse grids and various (β_1, β_2)-values.

(β_1, β_2)	ω in ω-JAC	h			
		1/64	1/32	1/16	1/8
$(1, 0)$	0.7	0.47	0.75	2.31	0.18
$(0, 1)$	0.8	0.36	0.32	0.13	0.05
$(1, 1)$	0.7	0.47	0.56	0.35	0.13
$(1, 0.5)$	0.5	0.60	0.77	0.81	0.32

$$S_h = 1 - \frac{\omega}{4 - (\beta_1 - \beta_2 i)(hk)^2} \left(4 - (\beta_1 - \beta_2 i)(hk)^2 - 2\cos \ell\pi h - 2\cos m\pi h\right),$$
$$\ell, m = 1, \ldots, \sqrt{N} - 1.$$

We consider the case $k = 40, h = 1/64$ and take ω as in the previous experiments. Table 5 presents smoothing factors on four consecutive grids for $(\beta_1, \beta_2) = (1, 0)$ (the original Helmholtz equation) and for $(\beta_1, \beta_2) = (0, 1), (1, 1)$, and $(1, 0.5)$. For simplicity, a direct PDE discretization on the coarse grids has been used. From Table 5, one confirms that for $h = 1/16, \omega$-JAC diverges for the original Helmholtz operator (also found with other relaxation parameters). This is in accordance with the remarks in [9, 11] that smoothing problems do not occur on the very fine or the very coarse grids but do occur on the intermediate grids. Furthermore, it can be observed that the $(\beta_1, \beta_2) = (0, 1)$-preconditioner resembles a Laplacian-type situation, with excellent smoothing factors on all grids. The preconditioners with $\beta_1 = 1, \beta_2 \neq 0$ give smoothing factors less than one on every grid. The (1,1)-preconditioner exhibits better smoothing factors than the set $(\beta_1, \beta_2) = (1, 0.5)$, which represents a limit case for which smoothing factors are still below one.

Some insight into the coarse grid correction can be gained from the so-called "simplified coarse grid analysis" or first-differential-approximation analysis [7, 9, 24]. As in [11] we apply this analysis for a one-dimensional (1D) Helmholtz operator. Assuming that transfer operators do not have any effect on the lowest frequencies, the quantity $1 - \lambda_h^\ell / \lambda_{2h}^{2\ell}$ (ℓ small) gives some insight into the relation between the discrete fine and coarse grid operators. This quantity should be close to zero and is an indication of the suitability of a coarse grid operator in a multigrid method. For the original 1D Helmholtz equation and $\alpha = 0$ (no damping), this quantity reads [11]

$$1 - \lambda_h^\ell / \lambda_{2h}^{2\ell} = \frac{\sin^4(\ell h\pi/2)}{\sin^2(\ell h\pi/2)\cos^2(\ell h\pi/2) - (kh/2)^2}, \qquad \ell = 1, \ldots, N.$$

It may give rise to a problematic coarse grid correction in the range where

$$\sin^2(\ell h\pi/2)\cos^2(\ell h\pi/2) \approx (kh/2)^2$$

and ℓ is associated with a smooth mode. For a 1D version of the [illegible] Helmholtz operator, this quantity reads

$$1 - \lambda_h^\ell / \lambda_{2h}^{2\ell} = \frac{\sin^4(\ell h\pi/2)}{\sin^2(\ell h\pi/2)\cos^2(\ell h\pi/2) - (kh/2)^2(\beta_1 - \beta_2 i)}$$
$$= \frac{\sin^4(\ell h\pi/2)\left(\sin^2(\ell h\pi/2)\cos^2(\ell h\pi/2) - (kh/2)^2(\beta_1 + \beta_2 i)\right)}{\left(\sin^2(\ell h\pi/2)\cos^2(\ell h\pi/2) - (kh/2)^2\beta_1\right)^2 + (kh/2)^2\beta_2^2},$$
$$\ell = 1, \ldots, N.$$

This expression for the complex Helmholtz operator is close to zero for the (β_1, β_2)-sets under consideration: the denominator does not reach zero, and the numerator contains the term $\sin^4 \ell h\pi/2$ which is very small for smooth eigenmodes.

When a Galerkin coarse grid discretization is used, it is difficult to gain insight into the coarse grid correction, as the coarse grid stencil elements are constructed with nontrivial formulas. Therefore, we discuss here for the case $(\beta_1, \beta_2) = (1, 0.5)$ two coarse grid discretizations. With $h = 1/64, k = 40$, $\alpha = 0$ in (4), we obtain by direct PDE discretization similar coarse grid stencils as the fine grid stencil with grid sizes $2h$ or $4h$, respectively. In that case, only the central stencil element contains an imaginary contribution. When the Galerkin coarse grid operator is employed, the imaginary part is distributed over all entries. With operator-dependent interpolation and full weighting restriction we find

$$A_{2h} \triangleq \begin{bmatrix} -282.9 + 15.3i & -665.8 + 80.6i & -282.9 + 15.3i \\ -665.8 + 80.6i & 2164.5 + 461.2i & -665.8 + 80.6i \\ -282.9 + 15.3i & -665.8 + 80.6i & -282.9 + 15.3i \end{bmatrix},$$

$$A_{4h} \triangleq \begin{bmatrix} -129.5 + 43.0i & -290.1 + 135.0i & -129.5 + 43.0i \\ -290.1 + 135.0i & -101.4 + 483.2i & -290.1 + 135.0i \\ -129.5 + 43.0i & -290.1 + 135.0i & -129.5 + 43.0i \end{bmatrix}. \tag{22}$$

The h-ellipticity measures are 0.28 and 0.18, indicating the suitability of the stencils for pointwise smoothing [7, 24]. For the direct PDE discretization, the h-ellipticity measures are 0.13 and 0.45 for the $2h$- and $4h$-discretizations, respectively. The fact that these qualitative measures are not close to zero means that pointwise smoothers can be constructed for these stencils. From these complicated coarse grid stencils it is, however, difficult to judge between the different smoothers, relaxation parameters, etc., but the three-grid Fourier analysis helps to some extent. We obtain very satisfactory multigrid convergence with simple multigrid components, although the coarse grid discretization (22) seems awkward. At least it does not spoil the h-independent multigrid convergence. One merely needs to choose the underrelaxation parameter in the smoother with some care.

4.4. Multigrid for the preconditioner. One multigrid iteration is taken for approximating the inverse of the operator in (8). After some experimentation it was found that it is sufficient to employ a multigrid iteration with a convergence factor $\rho_h \approx 0.6$ for the preconditioner. To some extend this can also be observed qualitatively from spectral pictures obtained by Fourier analysis (again, constant k, Dirichlet boundary conditions). Starting with a regular splitting of M_h,

$$C_h \phi_h^{l+1} = (C_h - M_h)\phi_h^l + \psi_h, \quad \text{or} \quad \phi_h^{l+1} = (I_h - C_h^{-1} M_h)\phi_h^l + C_h^{-1}\psi_h. \tag{23}$$

This splitting is considered to represent a multigrid iteration, with iteration matrix $(I_h - C_h^{-1}M_h)$ and C_h^{-1} an [illegible] of M_h^{-1}. T_h^{2h} in (17) represents the two-grid version of a multigrid iteration matrix. Therefore, we equate $T_h^{2h} = I_h - C_h^{-1}M_h$. Matrix $\widetilde{T}_h^{2h}$ in (19) is a block matrix related to T_h^{2h}: $\widetilde{T}_h^{2h} = U_h T_h^{2h} U_h^{-1}$, where U_h is a unitary matrix with four consecutive rows defined by the orthogonal eigenvectors related to (6). U_h transforms the two-grid iteration matrix into the block-diagonal matrix $\widetilde{T}_h^{2h}$. Clearly,

$$\widetilde{T}_h^{2h} = I_h - U_h C_h^{-1} M_h U_h^{-1}, \text{ and}$$
$$U_h C_h^{-1} M_h U_h^{-1} = U_h C_h^{-1} U_h^{-1} U_h M_h U_h^{-1} =: \widetilde{C}_h^{-1} \widetilde{M}_h$$

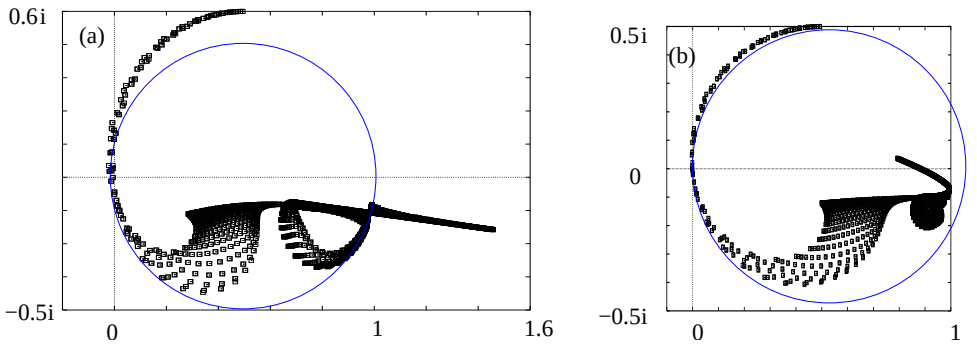


FIG. 6. *Spectral pictures of the preconditioned system with one two-grid iteration used for preconditioning* $(\beta_1, \beta_2) = (1, 1)$, $k = 40, h = 1/64$*:* (a) *one* ω*-JAC relaxation and* (b) *two* ω*-JAC relaxations,* $\omega = 0.7$*. (The eigenvalues with the exact inversion lie at the circles.)*

is in block-diagonal form. We have $\widetilde{C}_h^{-1}\widetilde{M}_h\widetilde{M}_h^{-1} = (I_h - \widetilde{T}_h^{2h})\widetilde{M}_h^{-1}$. So, the expression for the block-diagonal form $\widetilde{A}_h\widetilde{C}_h^{-1}$ ($\widetilde{C}_h^{-1}$ is the approximation of $\widetilde{M}_h^{-1}$) from (7) reads

$$\widetilde{A}_h\widetilde{C}_h^{-1} = \widetilde{A}_h(I_h - \widetilde{T}_h^{2h})\widetilde{M}_h^{-1}. \tag{24}$$

As all the symbols of the operators in the right-hand side of (24) can be formed easily with Fourier two-grid analysis, the corresponding eigenvalues can be visualized for various multigrid cycles. These spectra can be compared to those in Figure 1, where operator M_h from (8) is inverted exactly. Figure 6, for example, presents the spectrum of the $(\beta_1, \beta_2) = (1, 1)$-preconditioned system where a two-grid iteration is used for preconditioning for wavenumber $k = 40$ ($h = 1/64$). Figure 6(a) shows the spectrum for one ω-JAC ($\omega = 0.7$) smoothing iteration for which $\rho_{2g} \approx 0.7$, whereas Figure 6(b) shows the two-grid spectral picture with two ω-JAC smoothing iterations, $\nu_1 + \nu_2 = 2$, operator-dependent interpolation, full weighting restriction, and Galerkin coarse grid discretization ($\rho_{2g} = 0.45$). Figure 6(b) shows a spectrum that coincides well with the spectrum related to the exact inversion in Figure 1(d), whereas in Figure 6(a) eigenvalues are also outside the circle obtained with the exact inversion.

Figure 7 presents the spectra with a two-grid iteration for the $(\beta_1, \beta_2) = (1, 0.5)$-preconditioner and Galerkin coarsening, with ω-JAC relaxation ($\omega = 0.5$). Figure 7(a) is for $\nu = 1$; Figure 7(b) is for $\nu = 2$. Also for this approximate inversion of the preconditioner the spectrum obtained in Figure 7(b) compares well with the exact inversion in Figure 1(e), indicating that one multigrid iteration with two ω-JAC smoothing steps may be sufficient for approximating M_h^{-1}.

Indeed, a numerical comparison between inverting the preconditioner by several multigrid iterations versus by only one multigrid iteration did not lead to substantially different numbers of Krylov subspace iterations for solving the Helmholtz problem.

5. Applications. In this section the overall solution method, preconditioned Bi-CGSTAB for the indefinite heterogeneous Helmholtz problems (1) with the complex Helmholtz (β_1, β_2)-preconditioner, is evaluated. One multigrid F(1,1)-cycle is used for approximately inverting the preconditioner equation with the complex Helmholtz operator. Three problems of increasing difficulty are discussed.

5.1. Constant wavenumber. For constant wavenumbers k the Bi-CGSTAB convergence for the Helmholtz equation with the three preconditioners is presented.

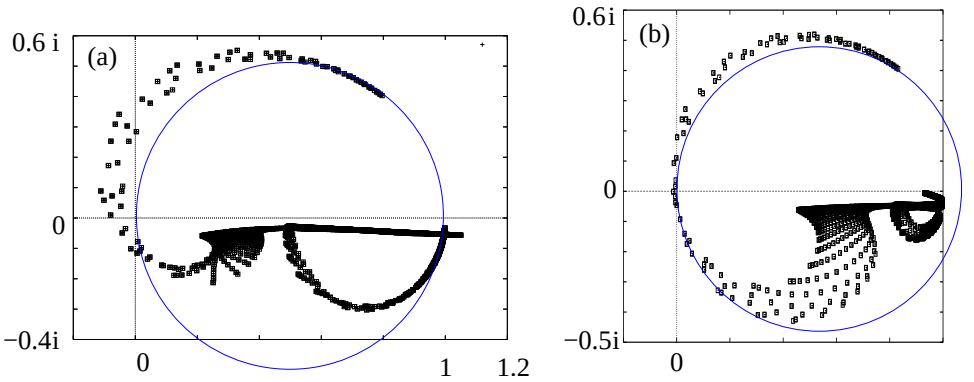


FIG. 7. *Spectral pictures of preconditioned system with one two-grid iteration used for preconditioning* $(\beta_1, \beta_2) = (1, 0.5)$*,* $k = 40, h = 1/64$*:* (a) *one* ω*-JAC relaxation and* (b) *two* ω*-JAC relaxations,* $\omega = 0.5$*. (The eigenvalues with exact inversion lie at the circles.)*

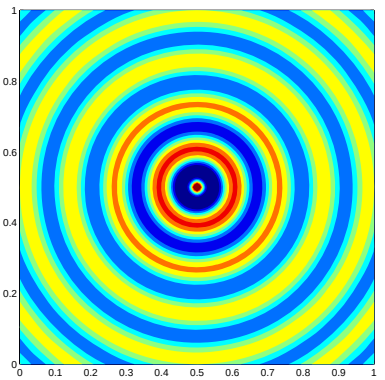


FIG. 8. *Numerical solution at* $k = 50$ *for the model problem with* k *constant.*

We consider a square domain $\Omega = (0, 1)^2$. A point source is located at the center of the domain. The solution satisfies the second-order conditions (2). In these experiments the finest grid size for each wavenumber is as shown in Table 1. The numerical solution corresponding to $k = 50$ is presented in Figure 8. Unphysical reflections at the boundaries are not present due to the boundary treatment.

A zero initial guess has been used during the computations. The Bi-CGSTAB iteration is terminated as soon as the initial residual is reduced by 7 orders of magnitude. Note that each Bi-CGSTAB iteration involves two preconditioning steps.

For all three preconditioners, $(\beta_1, \beta_2) = (0, 1), (1, 1)$, and $(1, 0.5)$, the method chosen to approximately invert the preconditioner consists of one multigrid F(1,1)-cycle with ω-JAC, operator-dependent interpolation plus full weighting as the transfer operators, and a Galerkin coarse grid discretization. The only difference is the value of the underrelaxation parameter in ω-JAC, which is $\omega = 0.8$ for $(\beta_1, \beta_2) = (0, 1)$, $\omega = 0.7$ for $(\beta_1, \beta_2) = (1, 1)$, and $\omega = 0.5$ for $(\beta_1, \beta_2) = (1, 0.5)$. The results for different values of k and $(\beta_1, \beta_2) = (0, 1)$ are presented in the upper part of Table 6. In the middle part of Table 6, the Bi-CGSTAB convergence with the $(\beta_1, \beta_2) = (1, 1)$-preconditioner is presented. In the lower lines of Table 6 the $(\beta_1, \beta_2) = (1, 0.5)$-preconditioner is employed. Next to the results for the Helmholtz equation without

TABLE 6

Number of preconditioned Bi-CGSTAB iterations and CPU time in seconds (in parentheses) to reduce the initial residual by 7 orders. Damping parameter α is varied in the Helmholtz problem.

		k				
(β_1, β_2)	α from (1)	40	50	80	100	150
(0, 1)	$\alpha = 0$	57 (0.44)	73 (0.92)	112 (4.3)	126 (7.7)	188 (28.5)
	2.5% damping	48 (0.38)	61 (0.77)	84 (3.3)	93 (5.6)	121 (18.5)
	5% damping	45 (0.35)	55 (0.70)	69 (2.7)	75 (4.7)	97 (14.9)
(1,1)	$\alpha = 0$	36 (0.30)	39 (0.51)	54 (2.2)	74 (4.5)	90 (13.9)
	2.5% damping	33 (0.27)	37 (0.48)	44 (1.8)	51 (3.2)	61 (9.6)
	5% damping	28 (0.24)	30 (0.39)	36 (1.5)	41 (2.6)	49 (7.5)
(1,0.5)	$\alpha = 0$	26 (0.21)	31 (0.40)	44 (1.8)	52 (3.3)	73 (10.8)
	2.5% damping	24 (0.20)	26 (0.35)	33 (1.4)	39 (2.5)	47 (7.3)
	5% damping	21 (0.18)	23 (0.32)	28 (1.2)	32 (2.1)	37 (5.8)

TABLE 7

High wavenumbers, number of Bi-CGSTAB iterations, and CPU time in seconds (in parentheses) needed to reduce the initial residual by 7 orders with and without damping in the Helmholtz problem.

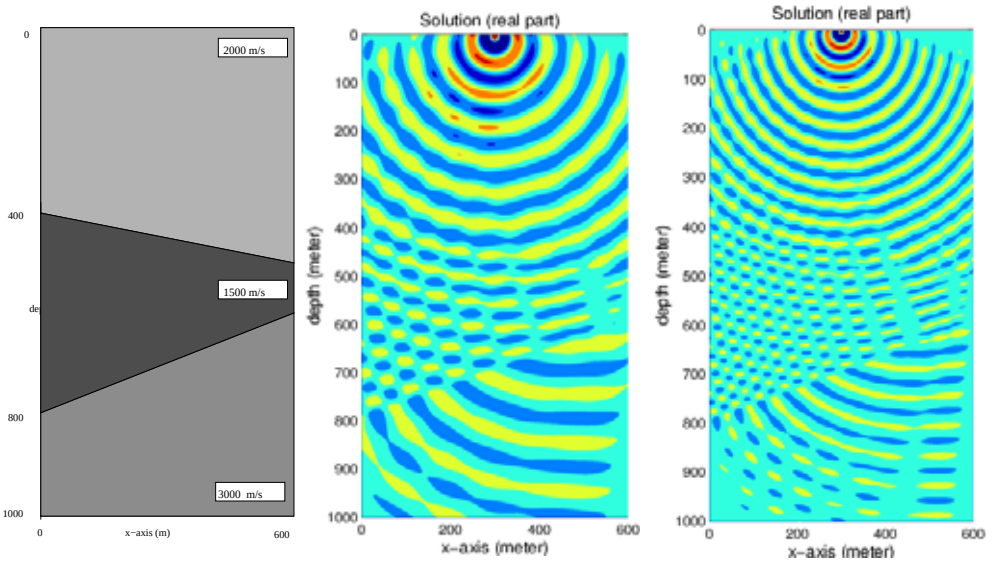
		k		
(β_1, β_2)	α in (1)	200	500	600
(1,1)	$\alpha = 0$	114 (30.8)	291 (515)	352 (890)
	2.5 % damping	74 (20.2)	125 (227)	145 (372)
	5 % damping	56 (15.5)	95 (174)	80 (205)
(1,0.5)	$\alpha = 0$	92 (25.4)	250 (425)	298 (726)
	2.5 % damping	57 (15.2)	91 (164)	102 (252)
	5 % damping	44 (11.9)	64 (115)	66 (165)

any damping ($\alpha = 0$), we also show the convergence with 2.5% ($\alpha = 0.025$) and 5% ($\alpha = 0.05$) damping. The number of Bi-CGSTAB iterations are presented as well as the CPU time on a Pentium 4 PC with 2.4 Ghz and 2 Gb RAM. From the results in Table 6 we conclude that the preferred methods among the choices are the preconditioners with $\beta_1 = 1$. This was already expected from the spectra in Figure 1. Fastest convergence is obtained for $(\beta_1, \beta_2) = (1, 0.5)$. The components of the multigrid iteration for this preconditioner have been validated with the help of the Fourier analysis.

Table 6 shows that the Bi-CGSTAB convergence with some damping in the Helmholtz problem is considerably faster than for $\alpha = 0$. This was already expected from the spectra in Figure 3. Furthermore, the number of iterations in the case of damping grows only slowly for increasing wavenumbers, especially for the $(\beta_1, \beta_2) = (1, 0.5)$-preconditioner.

The difference between the two preconditioners with $\beta_1 = 1$ is more pronounced if we compute higher wavenumbers. The Bi-CGSTAB convergence and CPU time for the higher wavenumbers, without and with damping in the Helmholtz problem, are presented in Table 7. Also for the higher wavenumbers damping in the Helmholtz problem by means of $\alpha \neq 0$ improves the convergence significantly. Very satisfactory convergence is found for high wavenumbers on fine grids.

5.2. The wedge model. A problem of intermediate difficulty is the wedge model. It is used to evaluate the preconditioner's behavior for a simple heterogeneous medium. The problem is adopted from [19]. The domain is defined to be a rectangle of dimension $600 \times 1000\ m^2$. The second-order boundary conditions (2) are set, and a point source is located at the center of the upper surface (which is assigned



(a) (b) (c)

FIG. 9. *Wedge problem:* (a) *Problem geometry with velocity profile indicated,* (b) *real part of numerical solution at* 30 *Hz, and* (c) *real part of numerical solution at* 50 *Hz.*

to be $y = 0$) with frequency, $f = \overline{k}c/(2\pi)L$, varying from 10 to 60 Hz (where c is the speed of sound). The corresponding values of the local dimensionless wavenumbers $\overline{k}$ vary between 20 (smallest for 10 Hz) and 240 (biggest for 60 Hz). For the problem at 10 Hz approximately 18 points per wavelength are used. Figure 9(a) presents the domain, the wedge, and the variation of c in the medium. The variation of c is due to the different local properties of the medium. The real part of the numerical solution for the wedge problem at 30 Hz and 50 Hz is plotted in Figures 9(b) and 9(c).

In the preconditioner (8) wavenumber $k(x, y)$ is chosen as in the original problem. Also the boundary conditions in the preconditioner are as for the original problem. The number of Bi-CGSTAB iterations with one multigrid iteration for the preconditioner with $(\beta_1, \beta_2) = (0, 1), (1, 1)$, and $(1, 0.5)$ are displayed in Table 8 for frequencies ranging from 10 to 60 Hz on corresponding grid sizes. Results with and without damping in the Helmholtz problem are presented. The only difference in the multigrid methods for the preconditioner is the value of the relaxation parameter: for $(\beta_1, \beta_2) = (0, 1)$ $\omega = 0.8$, for $(\beta_1, \beta_2) = (1, 1)$ $\omega = 0.7$, and for $(\beta_1, \beta_2) = (1, 0.5)$ $\omega = 0.5$. A zero initial guess has been used as a starting approximation. The convergence results for $(\beta_1, \beta_2) = (1, 0.5)$ are best, also without any damping in the original problem. The convergence with the (1,0.5)-preconditioner is about 1.5 times faster than with the (1,1)-preconditioner and about 3 times faster than with the (0,1)-preconditioner. The Bi-CGSTAB convergence for the wedge problem for $\alpha = 0$ and different frequencies are also visualized for $(\beta_1, \beta_2) = (1, 0.5)$ in Figure 10.

5.3. The Marmousi problem. This example is a part of the full Marmousi problem which mimics subsurface geology [4]; see also [19]. The domain is rectangular with a dimension of $6000 \times 1600\ m^2$. A point source is placed at the center of the upper surface. The values for the speed of sound c are irregularly structured throughout the

TABLE 8

Bi-CGSTAB convergence for the wedge problem with and without damping and the three multigrid based (β_1, β_2)-preconditioners compared. The number of Bi-CGSTAB iterations and the CPU time in seconds (in parentheses) are shown.

f (Hz)	Grid	Damping	(β_1, β_2) (0,1)	(1,1)	(1,0.5)
10	75 × 125	0.0%	52 (1.2)	30 (0.67)	19 (0.42)
		2.5%	48 (1.1)	27 (0.62)	17 (0.39)
		5.0%	42 (0.91)	25 (0.57)	16 (0.38)
20	149 × 249	0.0%	91 (8.8)	45 (4.5)	27 (2.8)
		2.5%	75 (7.2)	39 (4.0)	23 (2.4)
		5.0%	65 (6.3)	35 (3.5)	20 (2.1)
30	232 × 386	0.0%	128 (30.6)	64 (15.8)	37 (9.4)
		2.5%	94 (22.8)	49 (12.3)	29 (7.5)
		5.0%	86 (21.0)	42 (10.7)	25 (6.6)
40	301 × 501	0.0%	161 (66.1)	80 (33.5)	49 (20.8)
		2.5%	116 (48.0)	60 (25.4)	35 (15.2)
		5.0%	91 (37.9)	46 (19.8)	28 (12.4)
50	376 × 626	0.0%	205 (134.5)	98 (65.5)	58 (38.7)
		2.5%	135 (89.0)	67 (45.5)	37 (24.8)
		5.0%	99 (66.5)	54 (37.1)	32 (22.0)
60	481 × 801	0.0%	232 (247.3)	118 (127.6)	66 (71.9)
		2.5%	147 (159.1)	74 (81.1)	42 (47.1)
		5.0%	110 (119.6)	58 (64.5)	32 (36.7)

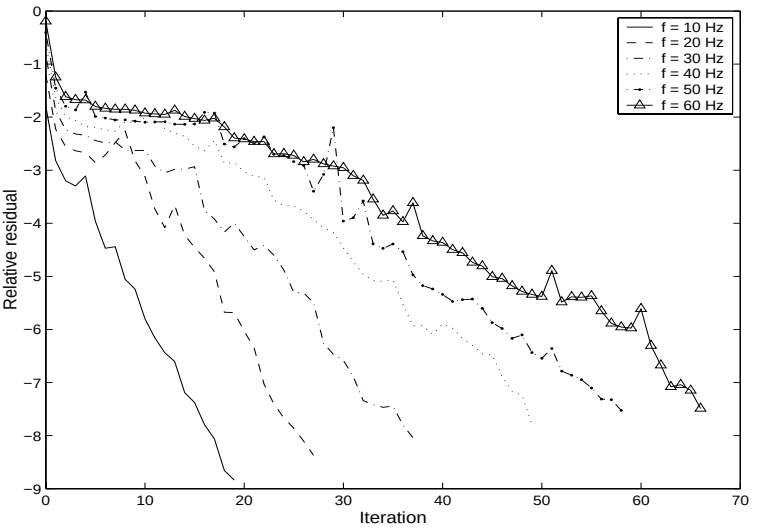


FIG. 10. *Bi-CGSTAB convergence plot for $(\beta_1, \beta_2) = (1, 0.5)$ for the wedge problem at different frequencies, $\alpha = 0$.*

domain; see Figure 11(a). The minimum number of points per wavelength equals 17. The frequency is varied between 1 and 30 Hz.

Preconditioning consists of one multigrid iteration for the complex Helmholtz equation with the multigrid components prescribed. The underrelaxation parameter in ω-JAC is varied as usual depending on (β_1, β_2). In the preconditioner again the wavenumbers $k(x, y)$ are as in the original problem. Also the boundary conditions are as in the original problem. Table 9 presents the number of Bi-CGSTAB iterations to solve the indefinite Helmholtz–Marmousi problem with the CPU times required shown

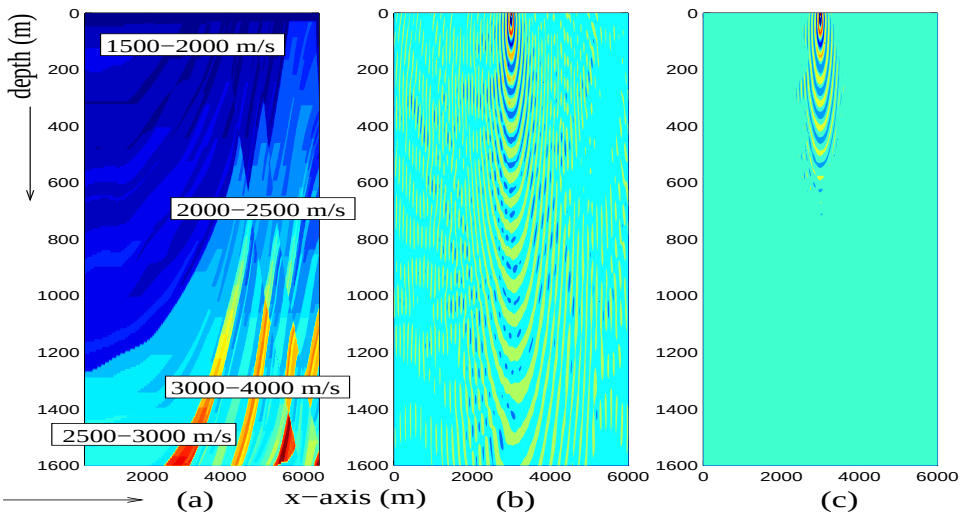


FIG. 11. *Marmousi problem (not to scale).* (a) *Velocity distribution in meter/s,* (b) *real part of the solution for* $f = 20$ *Hz, no damping, and* (c) *real part of the solution for* $f = 20$ *Hz,* 2.5% *damping.*

TABLE 9

Bi-CGSTAB convergence for the Marmousi problem with and without damping and the three multigrid based (β_1, β_2)*-preconditioners are presented. The number of Bi-CGSTAB iterations and the CPU time in seconds (in parentheses) are shown.*

f (Hz)	Grid	Damping	(β_1, β_2) (0,1)	(1,1)	(1,0.5)
1	751 × 201	0.0%	74 (37.5)	54 (27.6)	38 (19.7)
		2.5%	67 (34.1)	55 (28.2)	32 (17.0)
		5.0%	64 (32.5)	53 (27.3)	31 (16.5)
10	751 × 201	0.0%	180 (89.2)	84 (42.4)	47 (24.2)
		2.5%	119 (59.4)	59 (30.1)	33 (17.5)
		5.0%	96 (48.3)	48 (24.8)	28 (15.0)
20	1501 × 401	0.0%	414 (832.3)	168 (308.7)	104 (212.1)
		2.5%	203 (410.8)	88 (179.9)	55 (115.3)
		5.0%	145 (294.7)	64 (133.4)	37 (79.3)
30	2001 × 534	0.0%	458 (1724.8)	211 (799.4)	136 (519.4)
		2.5%	197 (745.6)	94 (361.1)	58 (226.8)
		5.0%	119 (455.3)	61 (238.4)	38 (151.9)

in parentheses. Results are presented for $\alpha = 0, 0.025$, and 0.05. A zero initial guess has been used. The $(\beta_1, \beta_2) = (1, 0.5)$-preconditioner shows a satisfactory and robust convergence for this problem with irregularly varying wavenumbers. For $\alpha = 0.05$ the number of iterations increases only very slowly for increasing frequencies. With the $(\beta_1, \beta_2) = (1, 0.5)$-preconditioner the CPU time is reduced by a factor of 3 compared to the performance of the $(\beta_1, \beta_2) = (0, 1)$-preconditioner for the challenging problems. The difference with the $(\beta_1, \beta_2) = (1, 1)$ is less pronounced but still significant.

The real parts of the solutions at 20 Hz for $\alpha = 0$ and $\alpha = 0.025$ are presented in Figures 11(b) and 11(c). The effect of damping of the solution is significant, as can be deduced from these global pictures. However, in the actual geophysics applications, some damping is present. Figure 11(c) may therefore be a more realistic solution for the real application. An adaptation of the solution method presented to a variant

of the Helmholtz problem in which the damping parameter α is varying [illegible] is an easy generalization in our preconditioner.

6. Conclusions. In this paper a complex Helmholtz preconditioner has been proposed for handling indefinite Helmholtz problems in heterogeneous media. In the preconditioner we advocate the use of a Helmholtz operator with a negative real term, as in the original Helmholtz problem, plus a positive imaginary Helmholtz part. Multigrid is proposed for approximately inverting the complex Helmholtz operator in the preconditioner. As the original Helmholtz problems are defined on square domains with structured grids (a common choice for many geophysical applications), a geometric multigrid method based on cartesian grids can be defined for the complex Helmholtz preconditioner. Extension of the geometric multigrid to the complex Helmholtz operator is straightforward. We can employ very similar multigrid components as those for the Laplacian, with some variation of the underrelaxation parameter in the point-wise Jacobi smoother, and an operator-dependent prolongation operator to deal with highly varying wavenumbers. These components are validated by Fourier analysis tools. The smallest size of the β_2-parameter in front of the imaginary Helmholtz term in the preconditioner, for which the multigrid method can be successfully employed, has been determined. After a parameter study all parameters appearing in the method are fixed (i.e., $\beta_1 = 1, \beta_2 = 0.5, \omega = 0.5$ in ω-JAC).

Bi-CGSTAB, preconditioned with a multigrid iteration for the complex Helmholtz operator, proves to be an efficient and robust iterative solution method to solve heterogeneous high wavenumber Helmholtz problems. The applications ranged from constant wavenumber to irregular heterogeneity structures in a medium. The multigrid components have been chosen such that the solution method is well parallelizable. The method proposed and the corresponding analysis are easily generalized to three dimensions. This is currently being done.

Acknowledgments. The use of Roman Wienands freely available Fourier analysis software *LFA00_scalar* and the particular version *RFA00_scalar* has been extremely helpful and is gratefully acknowledged.

The authors thank Rene-Edouard Plessix and Wim A. Mulder at Shell International Exploration and Production, Rijswijk, The Netherlands, for fruitful discussions and for providing the Marmousi data.

REFERENCES

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy Jr., and J. W. Painter, *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Comput., 2 (1981), pp. 430–454.

[2] A. Bamberger, P. Joly, and J. E. Roberts, *Second-order absorbing boundary conditions for the wave equation: A solution for the corner problem*, SIAM J. Numer. Anal., 27 (1990), pp. 323–352.

[3] A. Bayliss, C. I. Goldstein, and E. Turkel, *An iterative method for Helmholtz equation*, J. Comput. Phys., 49 (1983), pp. 443–457.

[4] A. Bourgeois, M. Bourget, P. Lailly, M. Poulet, P. Ricarte, and R. Versteeg, *Marmousi, model and data*, in The Marmousi Experience: EAEG, Workshop on Practical Aspects of Seismic Data Inversion, 1991, pp. 5–16.

[5] K. Brackenridge, *Multigrid and cyclic reduction applied to the Helmholtz equation*, in Proceedings of the 6th Annual Copper Mountain Conference on Multigrid Methods (CP3224), N. D. Melson, T. A. Manteuffel, and S. F. Mc Cormick, eds., NASA, Hampton, VA, 1993, pp. 31–41.

[6] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[7] A. Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, GMD-Studie 85, Gesellech. Math. Datenverarbeitung, Sankt Augustin, Germany, 1984.
[8] A. Brandt and I. Livshits, *Wave-ray multigrid methods for standing wave equations*, Electron. Trans. Numer. Analysis, 6 (1997), pp. 162–181.
[9] A. Brandt and S. Ta'asan, *Multigrid method for nearly singular and slightly indefinite problems*, in Proceedings EMG'85 Cologne, Multigrid Methods II, W. Hackbusch, and U. Trottenberg, eds., Springer, Berlin, 1986, pp. 99–121.
[10] J. E. Dendy Jr., *Blackbox multigrid for nonsymmetric problems*, Appl. Math. Comput., 13 (1983), pp. 261–283.
[11] H. R. Elman, O. G. Ernst, and D. P. O'Leary, *A multigrid method enhanced by Krylov subspace iteration for discrete Helmholtz equations*, SIAM J. Sci. Comput., 23 (2001), pp. 1291–1315.
[12] B. Engquist and A. Majda, *Absorbing boundary conditions for the numerical simulation of waves*, Math. Comp., 31 (1997), pp. 629–651.
[13] Y. A. Erlangga, C. Vuik, and C. W. Oosterlee, *On a class of preconditioners for the Helmholtz equation*, Appl. Numer. Math., 50 (2004), pp. 409–425.
[14] F. Ihlenburg and I. Babuska, *Finite element solution to the Helmholtz equation with high wave numbers*, Comput. Math. Appl., 30 (1995), pp. 9–37.
[15] S. Kim and S. Kim, *Multigrid simulation for high-frequency solutions of the Helmholtz problem in heterogeneous media*, SIAM J. Sci. Comput., 24 (2002), pp. 684–701.
[16] D. Lahaye, H. De Gersem, S. Vandewalle, and K. Hameyer, *Algebraic multigrid for complex symmetric systems*, IEEE Trans. Magn., 36 (2000), pp. 1535–1538.
[17] A. L. Laird and M. B. Giles, *Preconditioned Iterative Solution of the 2D Helmholtz Equation*, report 02/12, Oxford Computer Laboratory, Oxford, UK, 2002.
[18] B. Lee, T. A. Manteuffel, S. F. McCormick, and J. Ruge, *First-order system least-squares for the Helmholtz equation*, SIAM J. Sci. Comput., 21 (2000), pp. 1927–1949.
[19] R. E. Plessix and W. A. Mulder, *Separation-of-variables as a preconditioner for an iterative Helmholtz solver*, Appl. Numer. Math., 44 (2003), pp. 385–400.
[20] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear system*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[21] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[22] G. L. G. Sleijpen and D. R. Fokkema, *BiCGstab(ℓ) for Linear Equations involving Unsymmetric Matrices with Complex Spectrum*, Electron. Trans. Numer. Anal., 1 (1993), pp. 11–32.
[23] K. Stüben and U. Trottenberg, *Multigrid methods: fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, W. Hackbusch and U. Trottenberg, eds., Springer, Berlin, Germany, 1982, pp. 1–176.
[24] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[25] E. Turkel, *Numerical difficulties solving time harmonic equations*, in Multiscale Computational Methods in Chemistry and Physics, A. Brandt, et. al., eds., IOS Press, Tokyo, 2001, pp. 319–337.
[26] A. Bayliss, C. I. Goldstein, and E. Turkel, *On accuracy conditions for the numerical computation of waves*, J. Comput. Phys., 59 (1985), pp. 396–404.
[27] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput., 13 (1992), pp. 631–644.
[28] H. A. Van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[29] R. Wienands, C. W. Oosterlee, and T. Washio, *Fourier analysis of GMRES(m) preconditioned by multigrid*, SIAM J. Sci. Comput., 22 (2000), pp. 582–603.
[30] R. Wienands and C. W. Oosterlee, *On three-grid Fourier analysis for multigrid*, SIAM J. Sci. Comput., 23 (2001), pp. 651–671.
[31] P. M. de Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 33 (1990), pp. 1–27.

SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1493–1512
© 2006 Society for Industrial and Applied Mathematics

THE METHOD OF FUNDAMENTAL SOLUTIONS FOR STATIONARY HEAT CONDUCTION PROBLEMS IN ROTATIONALLY SYMMETRIC DOMAINS*

YIORGOS-SOKRATIS SMYRLIS† AND ANDREAS KARAGEORGHIS†

Abstract. We propose an efficient boundary collocation method for the solution of certain two- and three-dimensional problems of steady-state heat conduction in isotropic bimaterials. In particular, in two dimensions we consider the case where a circular region composed of one material is coated with an annular region of another material. In three dimensions, we examine the corresponding case for axisymmetric domains. The proposed method involves the use of a domain decomposition technique in conjunction with a matrix decomposition algorithm. The circulant structure of the matrices appearing in this method is exploited by using fast Fourier transforms. The method is tested numerically on several problems.

Key words. heat conduction in layered materials, method of fundamental solutions, Laplace equation, axisymmetric domains

AMS subject classifications. Primary 35J25, 65N12, 65N38; Secondary 65N15, 65T50

DOI. 10.1137/040615213

1. Introduction.

We consider steady-state heat conduction in an isotropic bimaterial. The bimaterial is decomposed into two subdomains $\mathfrak{O}_1$ and $\mathfrak{O}_2$, with boundaries $\partial\mathfrak{O}_1$ and $\partial\mathfrak{O}_2$, respectively, which are separated by an interface $\mathcal{S} = \partial\mathfrak{O}_1$. The two materials in $\mathfrak{O}_1$ and $\mathfrak{O}_2$ are characterized by the thermal conductivity constants k_1 and k_2, respectively. The temperature distribution in each subdomain satisfies the Laplace equation. On the interface $\mathcal{S}$ we have continuity of the temperature

$$u_1 = u_2$$

and continuity of the flux

$$k_1 \frac{\partial u_1}{\partial n} = k_2 \frac{\partial u_2}{\partial n},$$

where $\partial/\partial n$ denotes the normal derivative in the outward direction to $\mathfrak{O}_1$ on the interface. Examples and details about the modeling of such problems may be found in [18, 24].

The aim of this study is to develop efficient algorithms for the numerical solution of such problems in the case when $\overline{\mathfrak{O}}_1 \subset \mathfrak{O}_2$ and both domains are axisymmetric about the same axis. In two dimensions this means that we consider the case where one material occupies a circular region and another material occupies a concentric annular region surrounding it. In three dimensions, we examine the case where one material occupies an axisymmetric region and is coated by another material occupying a region which is axisymmetric about the same axis.

*Received by the editors September 16, 2004; accepted for publication (in revised form) May 24, 2005; published electronically January 27, 2006.
http://www.siam.org/journals/sisc/27-4/61521.html
†Department of Mathematics and Statistics, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus (smyrlis@ucy.ac.cy, andreask@ucy.ac.cy). This work was supported by University of Cyprus grant 8037-3/312-21005 21014. Part of the second author's work was undertaken while the author was a visiting professor in the Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, CO 80401.

The numerical method that we use is the method of fundamental solutions (MFS) which is a boundary collocation meshless technique. In recent years, the MFS has become very popular, primarily because of the ease with which it can be implemented. It belongs to the family of Trefftz-type methods (see [4]), and details about its implementation and its various applications can be found in the survey articles [6, 7, 11] and the books [10, 14].

We use a domain decomposition approach and a matrix decomposition algorithm. The domain is decomposed in a natural way into two subdomains, each containing a different material. Domain decomposition MFS applications can be found in [1], where heat conduction in layered materials is considered, and in [2, 9], where linear elasticity problems are studied. In the current work, in both the two- and the three-dimensional cases, the MFS discretization leads to the solution of linear systems whose coefficient matrices have [illegible] structures which can be exploited by using fast Fourier transforms (FFTs). The circulant and/or block circulant structure of the matrices appearing when the MFS is applied to axisymmetric problems has been exploited in [8, 22]. The idea of exploiting the FFTs to solve axisymmetric integral equations was introduced in [12]; see also [16]. The circulant structure of the coefficient matrices also arises from the application of other boundary methods such as the boundary element method (BEM), and this has been exploited in the past (see, e.g., [3, 15]). Details of applying a multiregion BEM approach to heat conduction problems in layered materials are presented in [19].

This paper is organized as follows. In section 2, we present the MFS formulation in the two-dimensional case. In section 3, we present the MFS formulation in the three-dimensional case. In section 4, we test numerically the proposed algorithms. Finally, in section 5, we give some concluding remarks.

2. The two-dimensional case.

2.1. The problem.

We consider the bimaterial problem in $\mathbb{R}^2$,

$$\Delta u_1 = 0 \quad \text{in} \quad B_{\varrho_1}, \tag{2.1a}$$

$$\Delta u_2 = 0 \quad \text{in} \quad A_{\varrho_1,\varrho_2}, \tag{2.1b}$$

subject to the boundary condition

$$u_2 = f \quad \text{on} \quad S_{\varrho_2} \tag{2.1c}$$

and the interface conditions

$$u_1 = u_2 \quad \text{and} \tag{2.1d}$$

$$k_1 \frac{\partial u_1}{\partial n} = k_2 \frac{\partial u_2}{\partial n} \quad \text{on} \quad S_{\varrho_1}. \tag{2.1e}$$

Here,

$$B_\varrho = \{\boldsymbol{x} : |\boldsymbol{x}| < \varrho\}, \quad S_\varrho = \{\boldsymbol{x} : |\boldsymbol{x}| = \varrho\},$$

and

$$A_{\varrho_1,\varrho_2} = \{\boldsymbol{x} \in \mathbb{R}^2 : \varrho_1 < |\boldsymbol{x}| < \varrho_2\}.$$

The coefficients k_1, k_2 in (2.1e) are the thermal conductivity constants. Δ denotes the Laplacian and f is a given function. The boundary of A_{ϱ_1,ϱ_2} is $\partial A_{\varrho_1,\varrho_2} = S_{\varrho_1} \cup S_{\varrho_2}$. In (2.1e), $\partial/\partial n$ denotes the normal derivative in the outward direction to B_{ϱ_1} on S_{ϱ_1}.

2.2. MFS formulation. We treat the two subdomains separately: in B_{ϱ_1} we approximate u_1 at a point P by

$$u_1^N(\boldsymbol{c}, \boldsymbol{Q}; P) \;=\; \sum_{\nu=1}^{N} c_\nu \varphi(P, Q_\nu),$$

and in A_{ϱ_1,ϱ_2} we approximate u_2 at a point P by

$$u_2^N(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{S}, \boldsymbol{T}; P) \;=\; \sum_{\nu=1}^{N} d_\nu \varphi(P, S_\nu) + \sum_{\nu=1}^{N} e_\nu \varphi(P, T_\nu),$$

where the $2N$-dimensional vectors $\boldsymbol{Q}$, $\boldsymbol{S}$, and $\boldsymbol{T}$ contain the coordinates of the [illegible]. In particular, the singularities described by $\boldsymbol{Q}$ lie outside $\overline{B}_{\varrho_1}$, whereas the singularities described by $\boldsymbol{S}$ and $\boldsymbol{T}$ lie outside the annulus $\overline{A}_{\varrho_1,\varrho_2}$.

The function $\varphi(P, Q)$ is a fundamental solution of the Laplacian Δ, i.e.,

$$-\Delta\, \varphi(\cdot, Q) \;=\; \delta_Q, \tag{2.2}$$

where δ_Q is the Dirac unit mass at Q. Equation (2.2) is satisfied in the sense of distributions. In particular, we take

$$\varphi(P, Q) \;=\; -\frac{1}{2\pi} \log |P - Q|,$$

with $|P - Q|$ denoting the distance between the points P and Q. The vectors $\boldsymbol{c} = (c_1, \ldots, c_N)$, $\boldsymbol{d} = (d_1, \ldots, d_N)$, and $\boldsymbol{e} = (e_1, \ldots, e_N)$ contain the coefficients of the fundamental solutions.

Specifically, the singularities $\boldsymbol{Q} = (Q_1, \ldots, Q_N)$ are placed on the circle of radius R_1, i.e., $S_{R_1} = \{\boldsymbol{x} : |\boldsymbol{x}| = R_1\}$, where $R_1 > \varrho_1$, at the points

$$Q_j \;=\; (R_1 \cos \vartheta_j, R_1 \sin \vartheta_j), \quad \vartheta_j = \frac{2\pi(j - 1 + \alpha)}{N}, \quad j = 1, \ldots, N,$$

and the singularities $\boldsymbol{S} = (S_1, \ldots, S_N)$ and $\boldsymbol{T} = (T_1, \ldots, T_N)$ are placed on the circles $S_{R_2} = \{\boldsymbol{x} : |\boldsymbol{x}| = R_2\}$, and $S_{R_3} = \{\boldsymbol{x} : |\boldsymbol{x}| = R_3\}$, respectively, with $R_2 < \varrho_1 < \varrho_2 < R_3$, at the points

$$S_j \;=\; (R_2 \cos \psi_j, R_2 \sin \psi_j), \quad T_j \;=\; (R_3 \cos \psi_j, R_3 \sin \psi_j), \quad \psi_j = \frac{2\pi(j - 1 + \beta)}{N},$$

$j = 1, \ldots, N$. The parameters α and β, where $\alpha, \beta \in [-\frac{1}{2}, \frac{1}{2}]$, describe rotations of the singularities. It has been observed that such rotations improve the approximation significantly, especially when the singularities are close to the boundary. (See [20, 21, 23, 25]).

Two sets of equally spaced collocation (or observation) points $\boldsymbol{P}^1 = (P_1^1, \ldots, P_N^1)$ and $\boldsymbol{P}^2 = (P_1^2, \ldots, P_N^2)$ are taken on the circles S_{ϱ_1} and S_{ϱ_2}, respectively:

$$P_j^1 \;=\; (\varrho_1 \cos \chi_j, \varrho_1 \sin \chi_j), \quad P_j^2 \;=\; (\varrho_2 \cos \chi_j, \varrho_2 \sin \chi_j), \quad \chi_j = \frac{2\pi(j - 1)}{N},$$

where $j = 1, \dots, N$. The vectors of coefficients $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{e}$ are determined from the collocation equations

$$u_1^N(\boldsymbol{c}, \boldsymbol{Q}; P_j^1) = u_2^N(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{S}, \boldsymbol{T}; P_j^1), \tag{2.3a}$$

$$k_1 \frac{\partial}{\partial n} u_1^N(\boldsymbol{c}, \boldsymbol{Q}; P_j^1) = k_2 \frac{\partial}{\partial n} u_2^N(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{S}, \boldsymbol{T}; P_j^1), \tag{2.3b}$$

$$u_2^N(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{S}, \boldsymbol{T}; P_j^2) = f(P_j^2), \tag{2.3c}$$

$j = 1, \dots, N$. Clearly, (2.3a), (2.3b), and (2.3c) correspond to (2.1d), (2.1e), and (2.1c), respectively. These yield the $3N \times 3N$ system

$$H \left(\begin{array}{c} \boldsymbol{c} \\ \hline \boldsymbol{d} \\ \hline \boldsymbol{e} \end{array} \right) = \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \boldsymbol{f} \end{array} \right), \tag{2.4}$$

where

$$H = \left(\begin{array}{c|c|c} \mathcal{A}(\boldsymbol{P}^1, \boldsymbol{Q}) & -\mathcal{A}(\boldsymbol{P}^1, \boldsymbol{S}) & -\mathcal{A}(\boldsymbol{P}^1, \boldsymbol{T}) \\ \hline k_1 \mathcal{B}(\boldsymbol{P}^1, \boldsymbol{Q}) & -k_2 \mathcal{B}(\boldsymbol{P}^1, \boldsymbol{S}) & -k_2 \mathcal{B}(\boldsymbol{P}^1, \boldsymbol{T}) \\ \hline 0 & \mathcal{A}(\boldsymbol{P}^2, \boldsymbol{S}) & \mathcal{A}(\boldsymbol{P}^2, \boldsymbol{T}) \end{array} \right) = \left(\begin{array}{c|c|c} H_{11} & H_{12} & H_{13} \\ \hline H_{21} & H_{22} & H_{23} \\ \hline H_{31} & H_{32} & H_{33} \end{array} \right)$$

and $\boldsymbol{f} = \big(f(P_1^2), \dots, f(P_N^2)\big)$, with

$$\big(\mathcal{A}(\boldsymbol{V}, \boldsymbol{W})\big)_{\mu\nu} = -\frac{1}{2\pi} \log |V_\mu - W_\nu| \quad \text{and}$$
$$\big(\mathcal{B}(\boldsymbol{V}, \boldsymbol{W})\big)_{\mu\nu} = -\frac{\partial}{\partial n_{V_\mu}} \frac{1}{2\pi} \log |V_\mu - W_\nu|$$

for $\boldsymbol{V} = \big(V_1, \dots, V_N\big)$, $\boldsymbol{W} = \big(W_1, \dots, W_N\big)$, and $\mu, \nu = 1, \dots, N$.

2.3. Matrix decomposition algorithm. All the blocks H_{ij}, $i, j = 1, 2, 3$, of the coefficient matrix H in system (2.4) are [illegible]. A square matrix G is circulant (see [5]) if it has the form

$$G = \begin{pmatrix} g_1 & g_2 & \cdots & g_N \\ g_N & g_1 & \cdots & g_{N-1} \\ \cdots & \cdots & \cdots & \cdots \\ g_2 & g_3 & \cdots & g_1 \end{pmatrix} \tag{2.5}$$

and is denoted by $G = \operatorname{circ}(g_1, g_2, \dots, g_N)$. Circulant matrices are diagonalizable by

$$G = U^* D U,$$

where

$$D = \operatorname{diag}\big(\lambda_1, \dots, \lambda_N\big)$$

is a diagonal matrix whose diagonal elements are the eigenvalues of the matrix G, given by (see [5])

$$\lambda_j = \sum_{\ell=1}^{N} \omega^{(j-1)(\ell-1)} g_\ell \tag{2.6}$$

and

$$U^* = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2N-2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)(N-1)} \end{pmatrix}, \tag{2.7}$$

with $\omega = \mathrm{e}^{\frac{2\pi \mathrm{i}}{N}}$. The matrix U^*, known as the Fourier matrix, is unitary.

It is obvious from (2.6) that the λ_j can be obtained from the g_ℓ via discrete Fourier transforms. The blocks $H_{ij},\ i,j = 1,2,3$, of the system matrix H can therefore be diagonalized by the Fourier matrix so that (2.4) may be written as

$$\big(I_3 \otimes U\big) H \big(I_3 \otimes U^*\big)\big(I_3 \otimes U\big) \left(\begin{array}{c} \boldsymbol{c} \\ \hline \boldsymbol{d} \\ \hline \boldsymbol{e} \end{array}\right) = \big(I_3 \otimes U\big) \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \boldsymbol{f} \end{array}\right), \tag{2.8}$$

where I_3 is the 3×3 identity matrix and $\otimes$ is the [illegible].[1]

Thus system (2.8) can be written as

$$\left(\begin{array}{c|c|c} D_{11} & D_{12} & D_{13} \\ \hline D_{21} & D_{22} & D_{23} \\ \hline D_{31} & D_{32} & D_{33} \end{array}\right) \left(\begin{array}{c} \hat{\boldsymbol{c}} \\ \hline \hat{\boldsymbol{d}} \\ \hline \hat{\boldsymbol{e}} \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \hat{\boldsymbol{f}} \end{array}\right), \tag{2.9}$$

where

$$\hat{\boldsymbol{c}} = U\boldsymbol{c},\ \ \hat{\boldsymbol{d}} = U\boldsymbol{d},\ \ \hat{\boldsymbol{e}} = U\boldsymbol{e},\ \ \hat{\boldsymbol{f}} = U\boldsymbol{f},$$

and

$$D_{ij} = UH_{ij}U^* = \operatorname{diag}\big(\lambda_{ij}^1, \ldots, \lambda_{ij}^N\big), \quad i,j = 1,2,3.$$

The elements λ_{ij}^ℓ can be computed from (2.6), with the appropriate choice of g_ℓ's.

Thus the linear system (2.9) reduces to N independent 3×3 systems:

$$\begin{pmatrix} \lambda_{11}^\ell & \lambda_{12}^\ell & \lambda_{13}^\ell \\ \lambda_{21}^\ell & \lambda_{22}^\ell & \lambda_{23}^\ell \\ \lambda_{31}^\ell & \lambda_{32}^\ell & \lambda_{33}^\ell \end{pmatrix} \begin{pmatrix} \hat{c}_\ell \\ \hat{d}_\ell \\ \hat{e}_\ell \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \hat{f}_\ell \end{pmatrix}, \qquad \ell = 1, \ldots, N. \tag{2.10}$$

Finally, system (2.4) can be solved efficiently using the following algorithm:

[1] The tensor (or Kronecker) product of the $m \times n$ matrix V and the $\ell \times k$ matrix W is the $m\,\ell \times n\,k$ matrix

$$V \otimes W = \begin{pmatrix} v_{11}\,W & v_{12}\,W & \cdots & v_{1n}\,W \\ v_{21}\,W & v_{22}\,W & \cdots & v_{2n}\,W \\ \cdots & \cdots & \cdots & \cdots \\ v_{m1}\,W & v_{m2}\,W & \cdots & v_{mn}\,W \end{pmatrix}.$$

Basic properties are (i) $V \otimes (W \otimes Z) = (V \otimes W) \otimes Z$, (ii) $(V_1 \otimes W_1)\,(V_2 \otimes W_2) = (V_1 V_2) \otimes (W_1 W_2)$, and (iii) $V \otimes (W + Z) = V \otimes W + V \otimes Z$. A comprehensive list of properties of the tensor product can be found in [5, pp. 22–23] or [17, pp. 597–598].

Step 1: Compute $\hat{\boldsymbol{f}} = U\boldsymbol{f}$.
Step 2: Evaluate the (diagonal) elements the diagonal matrices $D_{\mu\nu}$, $\mu, \nu = 1, 2, 3$, using formula (2.6).
Step 3: Solve the N systems of order 3 in (2.10) to determine the vectors $\hat{\boldsymbol{c}}$, $\hat{\boldsymbol{d}}$, and $\hat{\boldsymbol{e}}$.
Step 4: Compute the vectors $\boldsymbol{c} = U^*\hat{\boldsymbol{c}}$, $\boldsymbol{d} = U^*\hat{\boldsymbol{d}}$, and $\boldsymbol{e} = U^*\hat{\boldsymbol{e}}$.

(i) In Steps 1 and 4, because of the form of the matrices U and U^*, the operations can be carried out via FFTs at a cost of $\mathcal{O}(N \log N)$ operations.

(ii) FFTs can also be used for the evaluation of the diagonal matrices in Step 2 at a cost of $\mathcal{O}(N \log N)$.

(iii) In Step 3, we need to solve N complex linear systems of order 3 which can be done at a cost of $\mathcal{O}(N)$ operations.

3. The three-dimensional case.

3.1. The problem. We consider the bimaterial problem in $\mathbb{R}^3$,

$$\Delta u_1 = 0 \quad \text{in} \quad \Omega_1, \tag{3.1a}$$

$$\Delta u_2 = 0 \quad \text{in} \quad \Omega_2 \setminus \overline{\Omega}_1, \tag{3.1b}$$

where $\overline{\Omega}_1 \subset \Omega_2$, subject to the boundary condition

$$u_2 = f \quad \text{on} \quad \partial\Omega_2 \tag{3.1c}$$

and the interface conditions

$$u_1 = u_2 \quad \text{and} \tag{3.1d}$$

$$k_1 \frac{\partial u_1}{\partial n} = k_2 \frac{\partial u_2}{\partial n} \quad \text{on} \quad \partial\Omega_1. \tag{3.1e}$$

The regions $\Omega_1, \Omega_2 \subset \mathbb{R}^3$ are axisymmetric, which means that they are formed by rotating regions $\Omega_1', \Omega_2' \subset \mathbb{R}^2$ ($\Omega_1' \subset \Omega_2'$), respectively, about the z-axis.

3.2. MFS formulation. As in the two-dimensional case, we treat the two subdomains separately: in Ω_1 we approximate u_1 at a point P by

$$u_1^{M,N}(\boldsymbol{c}, \boldsymbol{Q}; P) = \sum_{\mu=1}^{M} \sum_{\nu=1}^{N} c_{\mu,\nu}\, \Phi(P, Q_{\mu,\nu})$$

and in $\Omega_2 \setminus \overline{\Omega}_1$, u_2 at a point P is approximated by

$$u_2^{M,N}(\boldsymbol{d}, \boldsymbol{e}, \boldsymbol{S}, \boldsymbol{T}; P) = \sum_{\mu=1}^{M} \sum_{\nu=1}^{N} d_{\mu,\nu} \Phi(P, S_{\mu,\nu}) + \sum_{\mu=1}^{M} \sum_{\nu=1}^{N} e_{\mu,\nu} \Phi(P, T_{\mu,\nu}).$$

Here

$$\begin{aligned} \boldsymbol{c} &= (c_{11}, c_{12}, \ldots, c_{1N}, \ldots, c_{M1}, \ldots, c_{MN})^T, \\ \boldsymbol{d} &= (d_{11}, d_{12}, \ldots, d_{1N}, \ldots, d_{M1}, \ldots, d_{MN})^T, \\ \boldsymbol{e} &= (e_{11}, e_{12}, \ldots, e_{1N}, \ldots, e_{M1}, \ldots, e_{MN})^T, \end{aligned}$$

and $\boldsymbol{Q}$, $\boldsymbol{S}$, $\boldsymbol{T}$ are $3MN$-vectors containing the coordinates of the singularities $Q_{\mu,\nu}$, $S_{\mu,\nu}$, $T_{\mu,\nu}$, $\mu = 1,\dots,M$, $\nu = 1,\dots,N$. The singularities described by $\boldsymbol{Q}$ lie outside $\overline{\Omega}_1$, whereas the singularities described by $\boldsymbol{S}$ and $\boldsymbol{T}$ lie outside $\overline{\Omega}_2 \setminus \Omega_1$. More specifically, the singularities $\boldsymbol{S}$ lie in Ω_1 and the singularities $\boldsymbol{T}$ lie outside $\overline{\Omega}_2$.

The function $\Phi(P,Q)$ is a fundamental solution of Laplace's equation in $\mathbb{R}^3$ given by

$$\Phi(P,Q) = \frac{1}{4\pi\,|P-Q|}$$

with $|P-Q|$, as before, denoting the distance between the points P and Q.

Two sets of MN collocation points $\{P^1_{m,n}\}^{M,N}_{m=1,n=1}$ and $\{P^2_{m,n}\}^{M,N}_{m=1,n=1}$ are chosen on $\partial\Omega_1$ and $\partial\Omega_2$, respectively, in the following way.

We first choose N points on each of the boundaries $\partial\Omega_1'$ and $\partial\Omega_2'$. These are described by their cartesian coordinates $\left(r_n^{P^1}, z_n^{P^1}\right)$, $n = 1,\dots,N$, and $\left(r_n^{P^2}, z_n^{P^2}\right)$, $n = 1,\dots,N$, in Ω_1' and Ω_2', respectively. Here $r_n^{P^i}$ denotes the horizontal distance from the axis of rotation, and $z_n^{P^i}$ denotes the z-coordinate. The points on $\partial\Omega_i$, $i = 1,2$, are taken to be

$$x_{P^i_{m,n}} = r_n^{P^i}\cos\varphi_m,\quad y_{P^i_{m,n}} = r_n^{P^i}\sin\varphi_m,\quad z_{P^i_{m,n}} = z_n^{P^i}, \tag{3.2}$$

where $\varphi_m = \frac{2(m-1)\pi}{M}, m = 1,\dots,M$.

Similarly, we choose three sets of MN singularities $\{Q_{\mu,\nu}\}^{M,N}_{\mu=1,\nu=1}$, $\{S_{\mu,\nu}\}^{M,N}_{\mu=1,\nu=1}$, and $\{T_{\mu,\nu}\}^{M,N}_{\mu=1,\nu=1}$ given by

$$\begin{aligned}
x_{Q_{\mu,\nu}} &= r_\nu^Q\cos\psi_\mu, \quad y_{Q_{\mu,\nu}} = r_\nu^Q\sin\psi_\mu, \quad z_{Q_{\mu,\nu}} = z_\nu^Q,\\
x_{S_{\mu,\nu}} &= r_\nu^S\cos\chi_\mu, \quad y_{S_{\mu,\nu}} = r_\nu^S\sin\chi_\mu, \quad z_{S_{\mu,\nu}} = z_\nu^S,\\
x_{T_{\mu,\nu}} &= r_\nu^T\cos\chi_\mu, \quad y_{T_{\mu,\nu}} = r_\nu^T\sin\chi_\mu, \quad z_{T_{\mu,\nu}} = z_\nu^T,
\end{aligned} \tag{3.3}$$

where

$$\psi_\mu = \frac{2(\mu-1+\alpha)\pi}{M}, \qquad \chi_\mu = \frac{2(\mu-1+\beta)\pi}{M}, \qquad \mu = 1,\dots,M.$$

The parameters $\alpha,\beta \in [-\frac{1}{2},\frac{1}{2}]$ prescribe the rotations of the singularities in the azimuthal direction.

The coefficients $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{e}$ are determined so that the interface and boundary conditions are satisfied at the boundary points $\{P^1_{m,n}\}^{M,N}_{m=1,n=1}$ and $\{P^2_{m,n}\}^{M,N}_{m=1,n=1}$, respectively,

$$u_1^{M,N}\left(\boldsymbol{c},\boldsymbol{Q};P^1_{mn}\right) - u_2^{M,N}\left(\boldsymbol{d},\boldsymbol{e},\boldsymbol{S},\boldsymbol{T};P^1_{mn}\right) = 0,$$

$$k_1\frac{\partial}{\partial n}u_1^{M,N}\left(\boldsymbol{c},\boldsymbol{Q};P^1_{mn}\right) - k_2\frac{\partial}{\partial n}u_2^{M,N}\left(\boldsymbol{d},\boldsymbol{e},\boldsymbol{S},\boldsymbol{T};P^1_{mn}\right) = 0,$$

$$u_2^{M,N}(\boldsymbol{d},\boldsymbol{e},\boldsymbol{S},\boldsymbol{T};P^2_{m,n}) = f(P^2_{m,n}),$$

$m = 1,\dots,M$, $n = 1,\dots,N$. These equations yield a $3MN \times 3MN$ linear system of the form

$$H\left(\begin{array}{c}\boldsymbol{c}\\ \hline \boldsymbol{d}\\ \hline \boldsymbol{e}\end{array}\right) = \left(\begin{array}{c}0\\ \hline 0\\ \hline \boldsymbol{f}\end{array}\right), \tag{3.4}$$

where

$$H = \left(\begin{array}{c|c|c} \mathcal{A}(\boldsymbol{P}^1,\boldsymbol{Q}) & -\mathcal{A}(\boldsymbol{P}^1,\boldsymbol{S}) & -\mathcal{A}(\boldsymbol{P}^1,\boldsymbol{T}) \\ \hline k_1\mathcal{B}(\boldsymbol{P}^1,\boldsymbol{Q}) & -k_2\mathcal{B}(\boldsymbol{P}^1,\boldsymbol{S}) & -k_2\mathcal{B}(\boldsymbol{P}^1,\boldsymbol{T}) \\ \hline 0 & \mathcal{A}(\boldsymbol{P}^2,\boldsymbol{S}) & \mathcal{A}(\boldsymbol{P}^2,\boldsymbol{T}) \end{array}\right) = \left(\begin{array}{c|c|c} H_{11} & H_{12} & H_{13} \\ \hline H_{21} & H_{22} & H_{23} \\ \hline H_{31} & H_{32} & H_{33} \end{array}\right).$$

Here $\boldsymbol{f} = \big(f_{mn}\big)_{m=1,n=1}^{M,N}$ and the $MN \times MN$ matrices are given by

$$\big(\mathcal{A}(\boldsymbol{V},\boldsymbol{W})\big)_{(m-1)N+n,(\mu-1)N+\nu} = \frac{1}{4\pi|V_{m,n} - W_{\mu,\nu}|},$$

$$\big(\mathcal{B}(\boldsymbol{V},\boldsymbol{W})\big)_{(m-1)N+n,(\mu-1)N+\nu} = \frac{\partial}{\partial n_{V_{m,n}}}\left\{\frac{1}{4\pi|V_{m,n} - W_{\mu,\nu}|}\right\}$$

for

$$\boldsymbol{V} = (V_{11}, V_{12}, \ldots, V_{1N}, \ldots, V_{M1}, \ldots, V_{MN})^T,$$
$$\boldsymbol{W} = (W_{11}, W_{12}, \ldots, W_{1N}, \ldots, W_{M1}, \ldots, W_{MN})^T$$

and $m, \mu = 1, \ldots, M$ and $n, \nu = 1, \ldots, N$.

3.3. Matrix decomposition algorithm. Each of the nine blocks H_{ij}, $i, j = 1, 2, 3$, of the coefficient matrix H in (3.4) has a [illegible] structure [8, 22] of the form

$$H_{ij} = \left(\begin{array}{c|c|c|c} H_{ij}^1 & H_{ij}^2 & \cdots & H_{ij}^M \\ \hline H_{ij}^M & H_{ij}^1 & \cdots & H_{ij}^{M-1} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline H_{ij}^2 & H_{ij}^3 & \cdots & H_{ij}^1 \end{array}\right),$$

where the matrices H_{ij}^ℓ, $\ell = 1, \ldots, M$, $i, j = 1, 2, 3$, are $N \times N$ matrices defined by

$$\big(H_{11}^\ell\big)_{n,\nu} = \frac{1}{4\pi|P_{1,n}^1 - Q_{\ell,\nu}|}, \quad \big(H_{12}^\ell\big)_{n,\nu} = -\frac{1}{4\pi|P_{1,n}^1 - S_{\ell,\nu}|},$$

$$\big(H_{13}^\ell\big)_{n,\nu} = -\frac{1}{4\pi|P_{1,n}^1 - T_{\ell,\nu}|}, \quad \big(H_{21}^\ell\big)_{n,\nu} = k_1\frac{\partial}{\partial n_{P_{1,n}^1}}\left\{\frac{1}{4\pi|P_{1,n}^1 - Q_{\ell,\nu}|}\right\},$$

$$\big(H_{22}^\ell\big)_{n,\nu} = -k_2\frac{\partial}{\partial n_{P_{1,n}^1}}\left\{\frac{1}{4\pi|P_{1,n}^1 - S_{\ell,\nu}|}\right\}, \quad \big(H_{23}^\ell\big)_{n,\nu} = -k_2\frac{\partial}{\partial n_{P_{1,n}^1}}\left\{\frac{1}{4\pi|P_{1,n}^1 - T_{\ell,\nu}|}\right\},$$

$$\big(H_{31}^\ell\big)_{n,\nu} = 0, \quad \big(H_{32}^\ell\big)_{n,\nu} = \frac{1}{4\pi|P_{1,n}^2 - S_{\ell,\nu}|}, \quad \big(H_{33}^\ell\big)_{n,\nu} = \frac{1}{4\pi|P_{1,n}^2 - T_{\ell,\nu}|},$$

$n, \nu = 1, \ldots, N$. Because of the block circulant structure of the matrices H_{ij}, following a technique similar to the ones employed in [8, 22], for every $i, j = 1, 2, 3$, we have that

$$H_{ij} = \sum_{\ell=1}^{N} \mathcal{P}^{\ell-1} \otimes H_{ij}^\ell,$$

where the matrix $\mathcal{P}$ is the $M \times M$ permutation matrix $\mathcal{P} = \mathrm{circ}(0, 1, 0, \ldots, 0)$ and $\mathcal{P}^0 = I_M$. Thus

$$H = \left(\begin{array}{c|c|c} H_{11} & H_{12} & H_{13} \\ \hline H_{21} & H_{22} & H_{23} \\ \hline H_{31} & H_{32} & H_{33} \end{array}\right)$$

$$= \left(\begin{array}{c|c|c} \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{11}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{12}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{13}^{\ell} \\ \hline \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{21}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{22}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{23}^{\ell} \\ \hline \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{31}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{32}^{\ell} & \sum_{\ell=1}^{M} \mathcal{P}^{\ell-1} \otimes H_{33}^{\ell} \end{array}\right)$$

$$= \sum_{\ell=1}^{M} \left(\begin{array}{c|c|c} \mathcal{P}^{\ell-1} \otimes H_{11}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{12}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{13}^{\ell} \\ \hline \mathcal{P}^{\ell-1} \otimes H_{21}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{22}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{23}^{\ell} \\ \hline \mathcal{P}^{\ell-1} \otimes H_{31}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{32}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{33}^{\ell} \end{array}\right). \tag{3.5}$$

From [5], the matrices $\mathcal{P}^{\ell}$ can be diagonalized as

$$\mathcal{P}^{\ell} = U^* E^{\ell} U,$$

where U is defined from (2.7) and $E = \mathrm{diag}\big(1, \omega, \omega^2, \ldots, \omega^{M-1}\big)$, with $\omega = \mathrm{e}^{2\pi \mathrm{i}/M}$. Premultiplying system (3.4) by

$$\left(\begin{array}{c|c|c} U & 0 & 0 \\ \hline 0 & U & 0 \\ \hline 0 & 0 & U \end{array}\right) \otimes I_N = I_3 \otimes U \otimes I_N$$

yields

$$\big(I_3 \otimes U \otimes I_N\big)\, H\, \big(I_3 \otimes U^* \otimes I_N\big)\big(I_3 \otimes U \otimes I_N\big) \left(\begin{array}{c} \boldsymbol{c} \\ \hline \boldsymbol{d} \\ \hline \boldsymbol{e} \end{array}\right) = \big(I_3 \otimes U \otimes I_N\big) \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \boldsymbol{f} \end{array}\right)$$

(using the fact that $\big(I_3 \otimes U^* \otimes I_N\big)\big(I_3 \otimes U \otimes I_N\big) = I_{3MN}$) or

$$\tilde{H} \left(\begin{array}{c} \tilde{\boldsymbol{c}} \\ \hline \tilde{\boldsymbol{d}} \\ \hline \tilde{\boldsymbol{e}} \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \tilde{\boldsymbol{f}} \end{array}\right), \tag{3.6}$$

where

$$\tilde{H} = \big(I_3 \otimes U \otimes I_N\big)\, H\, \big(I_3 \otimes U^* \otimes I_N\big)$$

and

$$\tilde{\boldsymbol{c}} = \big(U \otimes I_N\big)\, \boldsymbol{c}, \quad \tilde{\boldsymbol{d}} = \big(U \otimes I_N\big)\, \boldsymbol{d}, \quad \tilde{\boldsymbol{e}} = \big(U \otimes I_N\big)\, \boldsymbol{e}, \quad \tilde{\boldsymbol{f}} = \big(U \otimes I_N\big)\, \boldsymbol{f}.$$

From (3.5)

$$\tilde{H} = \sum_{\ell=1}^{M} \big(I_3 \otimes U \otimes I_N\big) \left(\begin{array}{c|c|c} \mathcal{P}^{\ell-1} \otimes H_{11}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{12}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{13}^{\ell} \\ \hline \mathcal{P}^{\ell-1} \otimes H_{21}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{22}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{23}^{\ell} \\ \hline \mathcal{P}^{\ell-1} \otimes H_{31}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{32}^{\ell} & \mathcal{P}^{\ell-1} \otimes H_{33}^{\ell} \end{array}\right) \big(I_3 \otimes U^* \otimes I_N\big),$$

and using the fact that

$$\big(U \otimes I_N\big)\big(\mathcal{P}^{\ell-1} \otimes H_{ij}^{\ell}\big)\big(U^* \otimes I_N\big) \;=\; E^{\ell-1} \otimes H_{ij}^{\ell},$$

we obtain

$$\tilde{H} = \sum_{\ell=1}^{M} \left(\begin{array}{c|c|c} E^{\ell\,-1} \otimes H_{11}^{\ell} & E^{\ell\,-1} \otimes H_{12}^{\ell} & E^{\ell\,-1} \otimes H_{13}^{\ell} \\ \hline E^{\ell\,-1} \otimes H_{21}^{\ell} & E^{\ell\,-1} \otimes H_{22}^{\ell} & E^{\ell\,-1} \otimes H_{23}^{\ell} \\ \hline E^{\ell\,-1} \otimes H_{31}^{\ell} & E^{\ell\,-1} \otimes H_{32}^{\ell} & E^{\ell\,-1} \otimes H_{33}^{\ell} \end{array}\right).$$

Thus, system (3.6) can be written as

$$\sum_{\ell=1}^{M} \left(\begin{array}{c|c|c} E^{\ell\,-1} \otimes H_{11}^{\ell} & E^{\ell\,-1} \otimes H_{12}^{\ell} & E^{\ell\,-1} \otimes H_{13}^{\ell} \\ \hline E^{\ell\,-1} \otimes H_{21}^{\ell} & E^{\ell\,-1} \otimes H_{22}^{\ell} & E^{\ell\,-1} \otimes H_{23}^{\ell} \\ \hline E^{\ell\,-1} \otimes H_{31}^{\ell} & E^{\ell\,-1} \otimes H_{32}^{\ell} & E^{\ell\,-1} \otimes H_{33}^{\ell} \end{array}\right) \left(\begin{array}{c} \tilde{\boldsymbol{c}} \\ \hline \tilde{\boldsymbol{d}} \\ \hline \tilde{\boldsymbol{e}} \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \tilde{\boldsymbol{f}} \end{array}\right).$$

Since the matrix E is diagonal, each block $E^{\ell-1} \otimes H_{ij}^{\ell}$ is block diagonal, i.e.,

$$E^{\ell-1} \otimes H_{ij}^{\ell} \;=\; \operatorname{diag}\big(H_{ij}^{\ell}, \omega^{\ell-1} H_{ij}^{\ell}, \ldots, \omega^{(\ell-1)(M-1)} H_{ij}^{\ell}\big),$$

as are the blocks of the matrix $\tilde{H}$:

$$\begin{aligned} \sum_{\ell=1}^{M} E^{\ell-1} \otimes H_{ij}^{\ell} &= \operatorname{diag}\left(\sum_{\ell=1}^{M} H_{ij}^{\ell}, \sum_{\ell=1}^{M} \omega^{\ell-1} H_{ij}^{\ell}, \ldots, \sum_{\ell=1}^{M} \omega^{(\ell-1)(M-1)} H_{ij}^{\ell} \right) \\ &= \operatorname{diag}\big(\tilde{H}_{ij}^{1}, \tilde{H}_{ij}^{2}, \ldots, \tilde{H}_{ij}^{M}\big), \end{aligned}$$

where

$$\tilde{H}_{ij}^{m} \;=\; \sum_{\ell=1}^{M} \omega^{(\ell-1)(m-1)} H_{ij}^{\ell}, \quad m = 1, \ldots, M, \quad i, j = 1, 2, 3. \tag{3.7}$$

System (3.4) thus reduces to the M independent $3N \times 3N$ systems:

$$\left(\begin{array}{c|c|c} \tilde{H}_{11}^{m} & \tilde{H}_{12}^{m} & \tilde{H}_{13}^{m} \\ \hline \tilde{H}_{21}^{m} & \tilde{H}_{22}^{m} & \tilde{H}_{23}^{m} \\ \hline \tilde{H}_{31}^{m} & \tilde{H}_{32}^{m} & \tilde{H}_{33}^{m} \end{array}\right) \left(\begin{array}{c} \tilde{\boldsymbol{c}}_m \\ \hline \tilde{\boldsymbol{d}}_m \\ \hline \tilde{\boldsymbol{e}}_m \end{array}\right) = \left(\begin{array}{c} 0 \\ \hline 0 \\ \hline \tilde{\boldsymbol{f}}_m \end{array}\right), \quad m = 1, \ldots, M, \tag{3.8}$$

where

$$\begin{aligned} \tilde{\boldsymbol{c}}_m &= \big(\tilde{c}_{m1}, \tilde{c}_{m2}, \cdots, \tilde{c}_{mN}\big)^T, \quad \tilde{\boldsymbol{d}}_m = \big(\tilde{d}_{m1}, \tilde{d}_{m2}, \cdots, \tilde{d}_{mN}\big)^T, \\ \tilde{\boldsymbol{e}}_m &= \big(\tilde{e}_{m1}, \tilde{e}_{m2}, \cdots, \tilde{e}_{mN}\big)^T, \quad \tilde{\boldsymbol{f}}_m = \big(\tilde{f}_{m1}, \tilde{f}_{m2}, \cdots, \tilde{f}_{mN}\big)^T. \end{aligned}$$

Once $\tilde{\boldsymbol{c}}$, $\tilde{\boldsymbol{d}}$, and $\tilde{\boldsymbol{e}}$ are computed, the vectors $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{e}$ can be calculated from

$$\boldsymbol{c} = \big(U^* \otimes I_N\big)\, \tilde{\boldsymbol{c}}, \quad \boldsymbol{d} = \big(U^* \otimes I_N\big)\, \tilde{\boldsymbol{d}}, \quad \boldsymbol{e} = \big(U^* \otimes I_N\big)\, \tilde{\boldsymbol{e}}. \tag{3.9}$$

System (3.8) can thus be solved efficiently using the following algorithm:

Step 1: Compute $\tilde{\boldsymbol{f}} = (U \otimes I_N)\, \boldsymbol{f}$.

Step 2: Construct the matrices $\tilde{H}_{ij}^{m}$, $i, j = 1, 2, 3$, $m = 1, \ldots, M$, from formula (3.7).

Step 3: Solve the M systems of order $3N$ in (3.8).
Step 4: Compute $\boldsymbol{c}$, $\boldsymbol{d}$, and $\boldsymbol{e}$ from (3.9).

(i) In Steps 1 and 4, because of the form of the matrices U and U^*, the operations can be carried out via FFTs and inverse FFTs at a cost of $\mathcal{O}(NM\log M)$.

(ii) In Step 2, for each $r,s=1,\ldots,N$, we need to perform an M-dimensional inverse FFT, in order to compute the entries $(\tilde{H}_{ij}^1)_{rs},\ldots,(\tilde{H}_{ij}^M)_{rs}$. This can be done at a cost of $\mathcal{O}(N^2\,M\log M)$.

(iii) In Step 3, we need to solve M complex linear systems of order $3N$. This can be done using an LU-factorization with partial pivoting at a cost of $\mathcal{O}(MN^3)$ operations.

4. Numerical experiments.

4.1. Two-dimensional case. In this subsection, we test numerically the algorithms proposed in section 2 in the case of problem (2.1). Since both the exact solutions u_1, u_2 and approximate solutions u_1^N, u_2^N are harmonic in $\mathfrak{O}_1=B_{\varrho_1}$ and $\mathfrak{O}_2=A_{\varrho_1,\varrho_2}$, respectively, the maximum principle applies to their difference, and thus

$$\sup_{\boldsymbol{x}\in\mathfrak{O}_j}|u_j^N(\boldsymbol{x})-u_j(\boldsymbol{x})| \;=\; \sup_{\boldsymbol{x}\in\partial\mathfrak{O}_j}|u_j^N(\boldsymbol{x})-u_j(\boldsymbol{x})|, \qquad j=1,2.$$

In all of our experiments, we calculated the maximum relative error of the problem

$$E \;=\; \max\{E_1,E_2\}, \tag{4.1a}$$

where

$$E_j \;=\; \frac{\sup_{\boldsymbol{x}\in\partial\mathfrak{O}_j}\left|u_j^N(\boldsymbol{x})-u_j(\boldsymbol{x})\right|}{\sup_{\boldsymbol{x}\in\partial\mathfrak{O}_j}\left|u_j(\boldsymbol{x})\right|}, \qquad j=1,2. \tag{4.1b}$$

The maximum relative error was calculated on uniform grids on the boundaries of the two regions, namely, at the points $\{V_\ell^j\}_{\ell=1,\ldots,L}^{j=1,2}$ given by

$$V_\ell^j \;=\; \big(\varrho_j\cos\vartheta_\ell,\varrho_j\sin\vartheta_\ell\big), \qquad \vartheta_\ell=\frac{2\pi}{L}(\ell-1).$$

In most experiments, the parameter L was taken to be equal to 1001.

In our numerical experiments we took the boundary data in (2.1c) to represent Fourier frequencies, namely, $f=\varrho_2^m\,\mathrm{e}^{\mathrm{i}m\vartheta}$, where m is a nonnegative integer. The exact solution of this problem, for $m>0$, is of the form

$$u_1(r,\vartheta) \;=\; A\,r^m\,\mathrm{e}^{\mathrm{i}m\vartheta}, \quad u_2(r,\vartheta) \;=\; B\,r^m\,\mathrm{e}^{\mathrm{i}m\vartheta}+\Gamma\,r^{-m}\,\mathrm{e}^{\mathrm{i}m\vartheta},$$

where A, B, and Γ can be uniquely determined from (2.1c)–(2.1e),

$$A=\frac{2k_2\,\varrho_2^{2m}}{k_1(\varrho_2^{2m}-\varrho_1^{2m})+k_2(\varrho_1^{2m}+\varrho_2^{2m})},$$

$$B=\frac{\varrho_2^{2m}(k_1+k_2)}{k_1(\varrho_2^{2m}-\varrho_1^{2m})+k_2(\varrho_1^{2m}+\varrho_2^{2m})},$$

$$\Gamma=\frac{k_2-k_1}{k_1(\varrho_2^{2m}-\varrho_1^{2m})+k_2(\varrho_1^{2m}+\varrho_2^{2m})}.$$

When $m=0$, the solution is $u_1=u_2=1$. We also chose $\varrho_1=1$, $\varrho_2=3$, $k_2/k_1=2$.

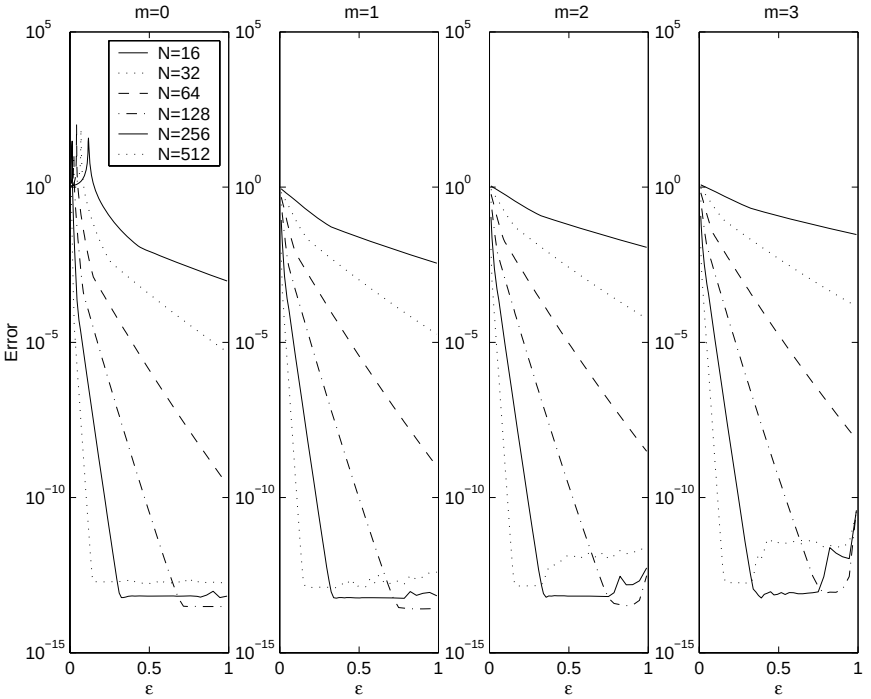


FIG. 1. *Semilog-plot of the error versus* ε *in Example* 1 *for different values of* N *when* $f = \varrho_2^m \mathrm{e}^{\mathrm{i}m\vartheta}$, $m = 0,\ 1,\ 2,\ 3$.

The FFT operations were performed using the NAG[2] routines `C06FPF`, `C06FQF`, and `C06FRF`. All numerical experiments were carried out on an IBM RS6000 (375 MHz) workstation.

Figure 1. We present the logarithm of E, as defined by (4.1a) and (4.1b), versus the distance ε of the three circles S_{R_j}, $j = 1, 2, 3$ describing the pseudoboundaries from the corresponding boundaries, S_{ϱ_1} and S_{ϱ_2}. That is,

$$R_1 = \varrho_1 + \varepsilon, \quad R_2 = \varrho_1 - \varepsilon, \quad R_3 = \varrho_2 + \varepsilon.$$

Here we consider the unrotated case, i.e., $\alpha = \beta = 0$. We consider the cases $m = 0, 1, 2, 3$ with $N = 16, 32, 64, 128, 256$ and 512.

We observe that, for a fixed m and a fixed N, as $\varepsilon \in (0, 1)$ grows, the accuracy improves exponentially fast. Further, for fixed ε and fixed m, the error decays exponentially as N grows. Also, for fixed N and ε, as m increases, the error increases.

The cusps in the case $m = 0$ occur because the coefficient matrix in (2.10), when $\ell = 1$, becomes singular for the values of ε where the cusps occur. When $\ell \neq 1$, no such singularity was observed. Since only the solution of system (2.10) for $\ell = m + 1$ contributes to the MFS solution corresponding to m, this phenomenon is not observed for $m \neq 0$. The analyses of similar observations, in the case of the Dirichlet problem in the disk and the annulus, are reported in [23, 25], respectively.

Note that as N and ε increase, the errors become close to machine accuracy.

[2]Numerical Analysis Group (NAG) Library Mark 20, NAG Ltd, Wilkinson House, Jordan Hill Road, Oxford, UK, 2001.

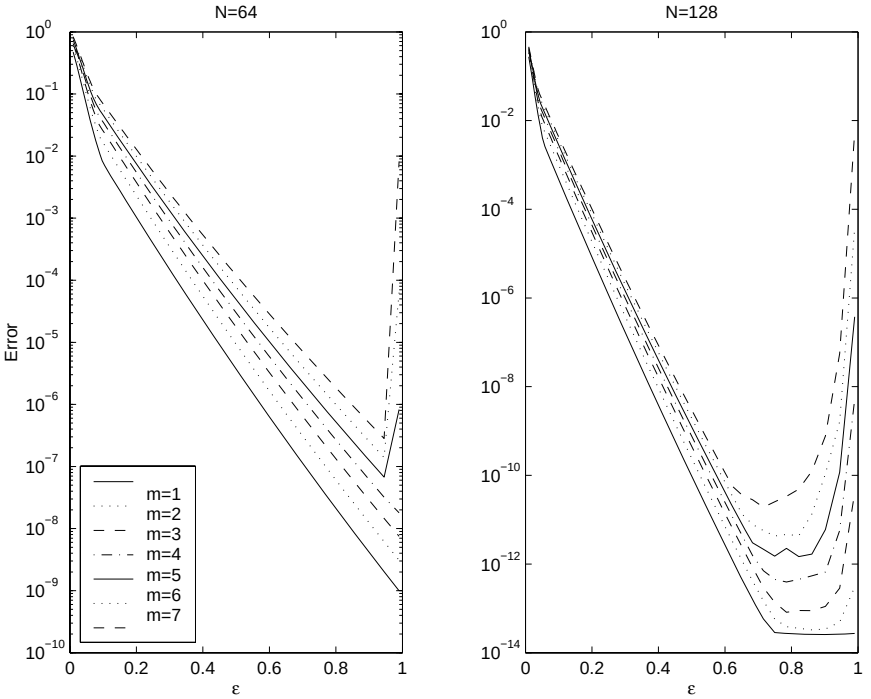


FIG. 2. *Semilog-plot of the error versus* ε *in Example* 1 *for different values of* m *when* $f = \varrho_2^m \mathrm{e}^{\mathrm{i}m\vartheta}$ *and* $N = 64, 128$.

Figure 2. In this case we show the dependence of the logarithm of E on ε for $m = 1, \ldots, 7$, for the cases $N = 64, 128$.

We observe that in both cases, as m increases, the accuracy of the approximation deteriorates. Similar phenomena were observed in previous studies, in particular, in [13, 23, 25]. In [23], in the case of the Dirichlet problem in a disk of radius ϱ, with the singularities placed on a circle of radius R, it is shown that the error behaves like

$$E = \mathcal{O}\left(\left(\frac{\varrho}{R}\right)^{N-2m}\right).$$

Further, we observe that for large m there is a deterioration of the accuracy for $\varepsilon \approx 1$. This is due to ill-conditioning resulting from the fact that the radius of S_{R_2} is close to zero.

Figure 3. We present the dependence of the error on the angular parameters α and β, when $m = 1$, $\varepsilon = .005, .01, .02, .04, .08, .16$. The two subplots correspond to the cases $N = 64$ and $N = 128$. For simplicity we took $\alpha = \beta$. Because of symmetry we need to consider only $\alpha \in [0, \frac{1}{2}]$.

We observe that, for the values of ε presented, as ε increases, the error E is optimized for $\alpha \approx \frac{1}{4}$. However, the improvement in the accuracy of the MFS approximation is more visible for smaller values of ε. In particular, for $\varepsilon = .005$, the accuracy of the approximation improves by a factor of ten at $\alpha \approx .1$, which is where the minimum is reached. Further, for larger values of ε the improvement due to rotation is negligible. These phenomena were also observed in previous studies [20, 21].

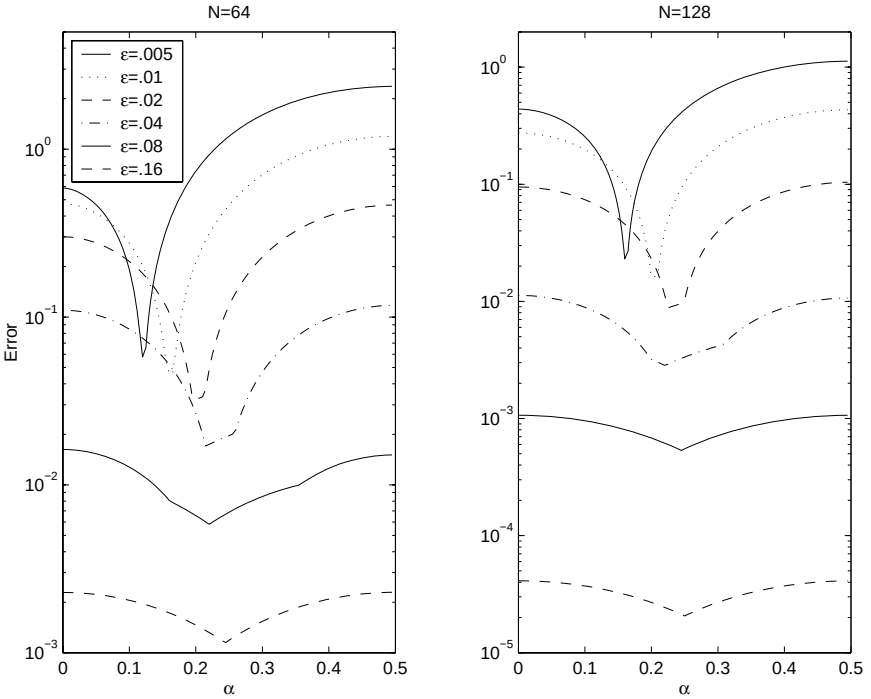


FIG. 3. *Semilog-plot of the error versus the angular parameter α when $f = \varrho_2 e^{\mathrm{i}\vartheta}$ for various values of ε and $N = 64, 128$. The parameter β is taken to be equal to α.*

4.2. Three-dimensional case. In this subsection we test numerically the algorithm proposed in section 3 for (3.1e). In the three-dimensional case it is difficult to construct an exact solution for (3.1e), and therefore, in order to test the convergence of the algorithm, we examine, as an estimate of the error, the supremum norm of the differences $u_i^{M,N} - u_i^{2M,2N}$, $i = 1, 2$, for various values of M and N. Since these differences are harmonic, from the maximum principle, it is sufficient to consider them on the boundaries $\partial\Omega_i$, $i = 1, 2$. We therefore calculate

$$E^{M,N} = \max\left\{E_1^{M,N}, E_2^{M,N}\right\}, \tag{4.2a}$$

where

$$E_1^{M,N} = \sup_{x\in\partial\Omega_1} \left|u_1^{M,N}(\boldsymbol{x}) - u_1^{2M,2N}(\boldsymbol{x})\right|, \tag{4.2b}$$

$$E_2^{M,N} = \sup_{x\in\partial\Omega_1\cup\partial\Omega_2} \left|u_2^{M,N}(\boldsymbol{x}) - u_2^{2M,2N}(\boldsymbol{x})\right|. \tag{4.2c}$$

This quantity was calculated on prescribed grids on the boundaries of the two regions which will be specified for each case.

We consider two examples of boundary data in (3.1c), namely,

1. $f = \cosh(.03x)\cosh(.04y)\cos(.05z)$.
2. $f = (9.5 - x^2 - y^2 - z^2)^{-1}$.

In all numerical experiments performed in this study, we took $k_2/k_1 = 2$.

4.2.1. Spherical domains. We consider the case where Ω_1 and Ω_2 are the spheres of radii ϱ_1 and ϱ_2 ($\varrho_1 < \varrho_2$), respectively, that is,

$$\Omega_j = \{(x,y,z) \in \mathbb{R}^3 : x^2 + y^2 + z^2 < \varrho_j^2\}, \quad j = 1, 2.$$

The singularities $\{Q_{\mu,\nu}\}_{\mu=1,\nu=1}^{M,N}$, $\{S_{\mu,\nu}\}_{\mu=1,\nu=1}^{M,N}$, and $\{T_{\mu,\nu}\}_{\mu=1,\nu=1}^{M,N}$ are placed on the spherical surfaces $\partial\Omega_Q$, $\partial\Omega_S$, and $\partial\Omega_T$, centered at the origin, of radii R_Q, R_S, and R_T, respectively, with $R_S < \varrho_1 < R_T$ and $\varrho_2 < R_Q$. The coordinates of the singularities are given by (3.3) with

$$\begin{aligned} r_\nu^Q &= R_Q \sin\vartheta_\nu, \;\; z_\nu^Q = R_Q \cos\vartheta_\nu, \\ r_\nu^S &= R_S \sin\vartheta_\nu, \;\; z_\nu^S = R_S \cos\vartheta_\nu, \\ r_\nu^T &= R_T \sin\vartheta_\nu, \;\; z_\nu^T = R_T \cos\vartheta_\nu, \end{aligned}$$

where $\vartheta_\nu = \frac{\nu\pi}{N+1}$, $\nu = 1, \ldots, N$. The collocation points $\{P_{m,n}^i\}_{m=1,n=1}^{M,N}$ on the boundaries $\partial\Omega_i$, $i = 1, 2$, are given by

$$x_{P_{m,n}^i} = \varrho_i \sin\vartheta_n \cos\varphi_m, \quad y_{P_{m,n}} = \varrho_i \sin\vartheta_n \sin\varphi_m, \quad z_{P_{m,n}^i} = \varrho_i \cos\vartheta_n. \tag{4.3}$$

Note that we avoid the points corresponding to $\vartheta_j = 0$ and $\vartheta_j = \pi$ as they remain invariant under rotation in the φ-direction and would lead to singular matrices.

The error was calculated at the points

$$V_{k,\ell}^i = \varrho_i(\sin\Theta_\ell \cos\Phi_k, \sin\Theta_\ell \sin\Phi_k, \cos\Theta_\ell), \quad i = 1, 2,$$

where $\Theta_\ell = \frac{\pi\ell}{L+1}$ and $\Phi_k = \frac{2\pi(k-1)}{L}$, $k, \ell = 1, \ldots, L$. In most numerical experiments L was taken to be equal to 31.

In our numerical experiments, we considered the case of the spheres of radii 1 and 3, respectively.

Figure 4. We present the logarithm of error, as defined by (4.2c), versus the distance ε of the three spherical surfaces $\partial\Omega_Q$, $\partial\Omega_S$, and $\partial\Omega_T$ from the corresponding boundaries $\partial\Omega_1$, $\partial\Omega_2$. That is,

$$R_Q = \varrho_1 + \varepsilon, \quad R_S = \varrho_1 - \varepsilon, \quad R_T = \varrho_2 + \varepsilon.$$

We consider both Examples 1 and 2 in the unrotated case, i.e., $\alpha = \beta = 0$. We took $M = N = 12, 16, 24, 32, 48$. From this figure we observe that, for both examples, as ε varies on the interval $(10^{-2}, 1)$, the error decreases exponentially fast. Also, the rate of convergence is more rapid as N grows.

Figure 5. We present the dependence of the error on the angular parameters α and β. For simplicity, we considered the case when $\alpha = \beta$. Because of symmetry we need to consider only the case $\alpha \in [0, \frac{1}{2}]$ and examine the behavior of the error for $M = N = 16, 24, 32, 48$ and fixed ε.

We observe that for small ε an optimal value was reached at about $\alpha \approx \frac{1}{4}$. This behavior can be seen in Figure 5 in the case $\varepsilon = \frac{1}{10}$ for both Examples 1 and 2. For larger values of ε, the minimum is no longer visible. Similar phenomena have been observed in previous studies [20, 23].

Figure 6. In Figure 6 we present the logarithm of an estimate of the condition number in the L^∞-norm of the coefficient matrix in system (3.8) for $m = 1$ versus ε, for $N = 8, 12, 16, 24, 32$, and 48. As can be seen from this figure, the conditioning of the matrix deteriorates considerably as ε increases. The estimate of the condition number was calculated using the NAG pair `F07ARF-F07AUF`. The behavior of the condition number of the matrices in the cases $m = 2, 3, \ldots M$ is very similar.

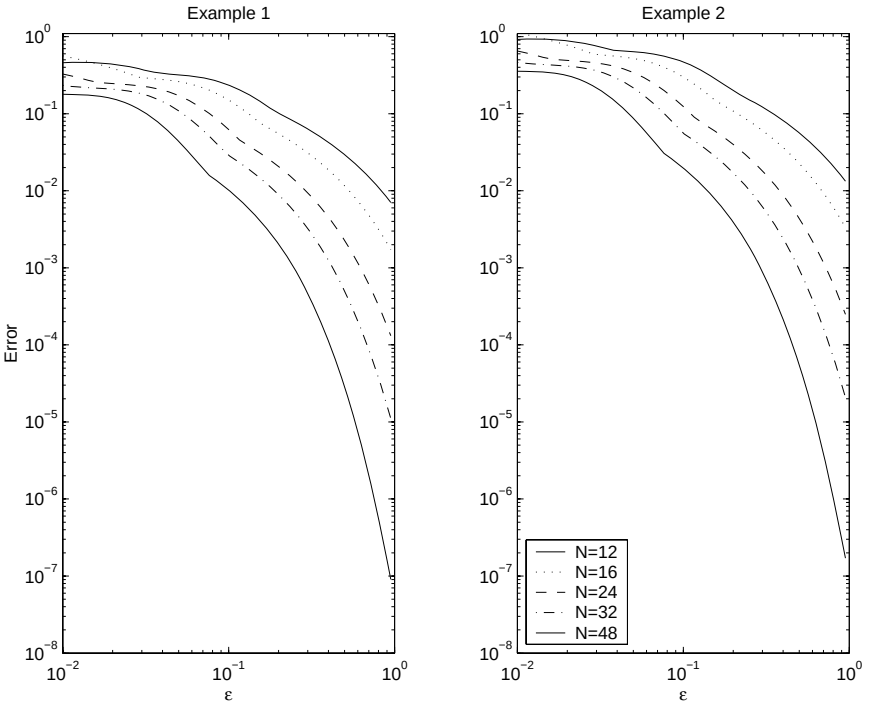


FIG. 4. *Loglog-plot of the error versus ε in Examples* 1 *and* 2 *in the case where Ω_1 and Ω_2 are spheres.*

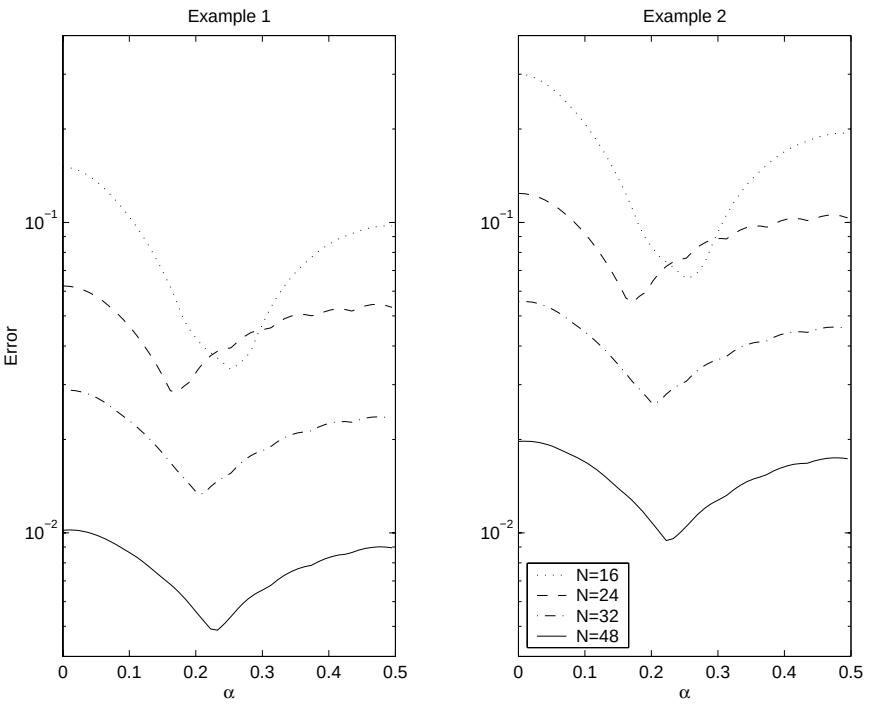


FIG. 5. *Spherical domains: Semilog-plot of the error versus the angular parameter α in Examples* 1 *and* 2 *for $\varepsilon = 1/10$. The parameter β is taken to be equal to α.*

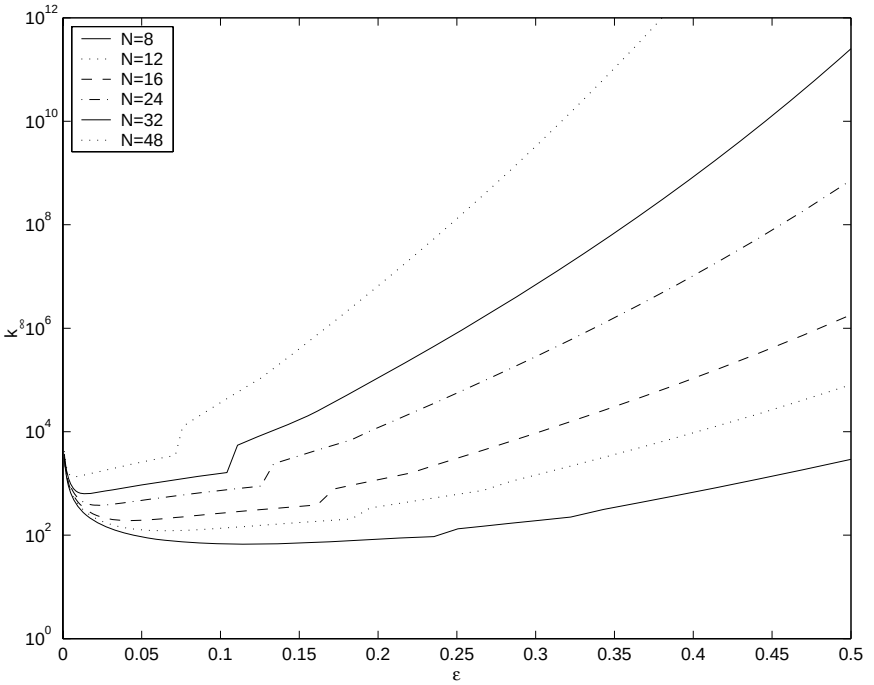


FIG. 6. *Spherical domains: Semilog-plot of the condition number κ_∞ of the matrix in the system* (3.8) *versus* ε.

4.2.2. Cylinder inside a sphere. We consider the case when Ω_1 is the cylinder

$$\Omega_1 = \left\{(x,y,z) \in \mathbb{R}^3 \,:\, x^2+y^2 < \varrho_1^2,\ |z| < h\right\}$$

and Ω_2 is the sphere

$$\Omega_2 = \left\{(x,y,z) \in \mathbb{R}^3 \,:\, x^2+y^2+z^2 < \varrho_2^2\right\}$$

where $\varrho_1^2 + h^2 < \varrho_2^2$. The boundary points $P^2_{\mu,\nu}$ are chosen as in (4.3). The points $P^1_{\mu,\nu}$ are given by (3.2), where $(r_n^{P^1}, z_n^{P^1}) \in \mathbb{R}^2$ are uniformly distributed on the three sides of the rectangle $[0,\varrho_1]\times[-h,h]$ which is the generator of the cylinder.

The singularities $Q_{\mu,\nu}$ and $S_{\mu,\nu}$ are placed on cylinders similar to Ω_1, both at a distance ε from $\partial\Omega_1$. The singularities $T_{\mu,\nu}$ are placed on a sphere of radius R_T, where $R_T - \varrho_2 = \varepsilon > 0$.

In our numerical experiments, we took $h = 1$, $\varrho_1 = 1$, and $\varrho_2 = 3$.

Figure 7. We present the logarithm of the error as defined by (4.2c) versus ε. We apply the method to Example 1 in the unrotated case, and we varied ε in the interval $(10^{-2}, 1)$. We took $M = N = 8, 12, 16, 24, 32, 48$. From this figure, we observe that as ε increases, the error decreases exponentially fast up to about $\varepsilon \approx \frac{1}{2}$. For larger values of ε, the quality of the approximation deteriorates rapidly. For this geometry, the approximation is considerably less accurate than the corresponding approximation in the case of the spherical domains. This is due to the presence of the edges in the cylindrical subdomain. Further, the error in the approximation for this geometry starts deteriorating for smaller values of ε than for the spherical domains.

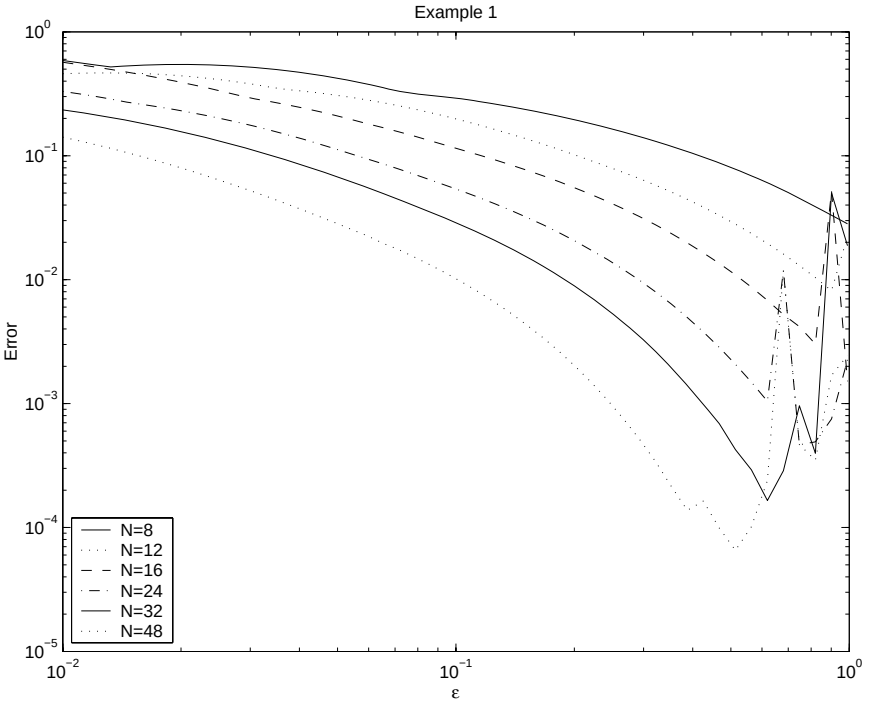


FIG. 7. *Cylinder inside a sphere: Loglog-plot of the error versus ε in Example* 1.

Figure 8. We present the dependence of the error on the angular parameters α and β in Example 1. For simplicity, again, we considered the case when $\alpha = \beta$, and because of symmetry we examined $\alpha \in [0, \frac{1}{2}]$. In the first subplot we took $\varepsilon = \frac{1}{10}$ and $M = N = 16, 24, 32, 48$. As observed previously, there is a tendency for the optimum to be reached at $\alpha = \frac{1}{4}$ and fixed ε. We observed that for small ε an optimal value was reached at about $\alpha \approx \frac{1}{4}$.

In the second subplot we present the corresponding results for $\varepsilon = \frac{1}{2}$. As expected, the accuracy is higher, and similar phenomena to those seen in the first subplot are observed.

5. Concluding remarks. In this paper, we apply a meshless boundary collocation method for the solution of certain heat conduction problems in layered materials in two and three dimensions. In two dimensions, we consider materials in concentric regions whereas in three dimensions we consider axisymmetric domains possessing the same axis of symmetry. We employ a domain decomposition technique which leads to systems whose coefficient matrices exhibit block circulant structures. Matrix decomposition methods in conjunction with the use of FFTs enable us to construct efficient algorithms for the numerical solution of these problems.

The algorithms developed in this study could be easily extended to the solution of bimaterial problems in the same geometries in two and three dimensions but governed by different differential equations such as the Helmholtz equation, the biharmonic equation, or the Cauchy–Navier equations of elasticity, in two and three dimensions. Further, this method is not restricted to dealing with bimaterial problems but could also be applied to multilayered materials in corresponding axisymmetric domains in two and three dimensions.

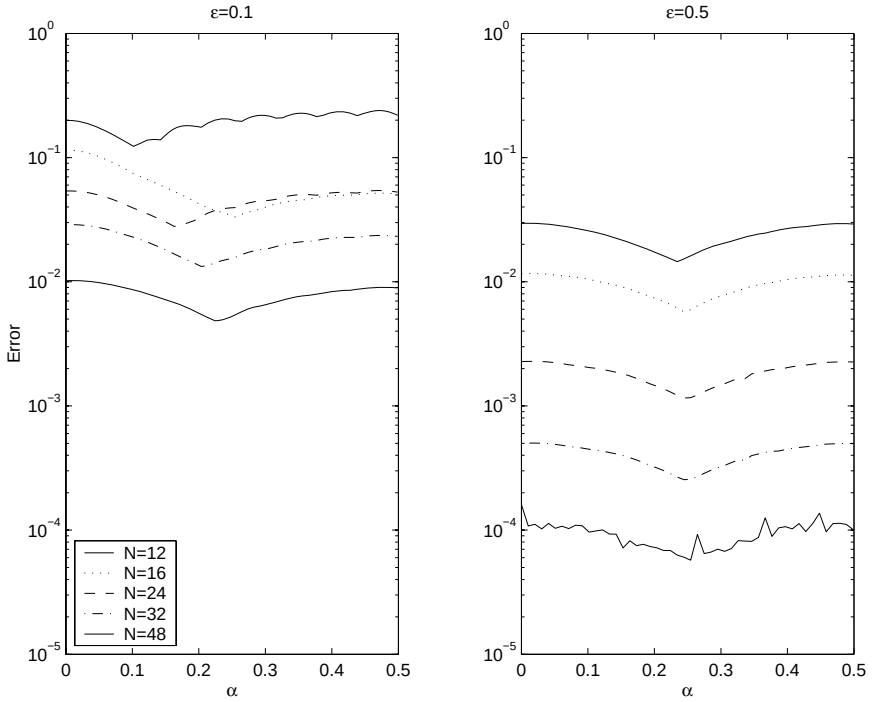


FIG. 8. *Cylinder inside a sphere: Semilog-plot of the error versus the angular parameter* α *for* $\varepsilon = 1/10,\ 1/2$. *The parameter* β *is taken to be equal to* α.

Finally, the block circulant structure of the matrices arising in these problems is preserved for other boundary method discretizations. The current algorithm could thus be used when, for example, the BEM is applied to the type of problems examined in this work.

Acknowledgments. The authors wish to thank Professors John R. Berger, Graeme Fairweather, and Paul A. Martin of the Colorado School of Mines for their helpful suggestions.

REFERENCES

[1] J. R. BERGER AND A. KARAGEORGHIS, *The method of fundamental solutions for heat conduction in layered materials*, Internat. J. Numer. Methods Engrg., 45 (1999), pp. 1681–1694.

[2] J. R. BERGER AND A. KARAGEORGHIS, *The method of fundamental solutions for layered elastic materials*, Engrg. Anal. Boundary Elem., 25 (2001), pp. 877–886.

[3] J. T. CHEN, J. H. LIN, S. R. KUO, AND S. W. CHYUAN, *Boundary element analysis for the Helmholtz eigenvalue problems with a multiply connected domain*, R. Soc. Lond. Proc. Ser. A Math. Phys. Eng. Sci., 457 (2001), pp. 2521–2546.

[4] H. A. CHO, M. A. GOLBERG, A. S. MULESHKOV, AND X. LI, *Trefftz methods for time dependent partial differential equations*, Comput. Mat. Cont., 1 (2004), pp. 1–37.

[5] P. J. DAVIS, *Circulant Matrices*, John Wiley & Sons, New York, Chichester, Brisbane, 1979.

[6] G. FAIRWEATHER AND A. KARAGEORGHIS, *The method of fundamental solutions for elliptic boundary value problems. Numerical treatment of boundary integral equations*, Adv. Comput. Math., 9 (1998), pp. 69–95.

[7] G. FAIRWEATHER, A. KARAGEORGHIS, AND P. A. MARTIN, *The method of fundamental solutions for scattering and radiation problems*, Engrg. Anal. Boundary Elem., 27 (2003), pp. 759–769.

[8] G. Fairweather, A. Karageorghis, and Y.-S. Smyrlis, *A matrix decomposition MFS algorithm for axisymmetric biharmonic problems*, Adv. Comput. Math., 23 (2005), pp. 55–71.
[9] R. T. Fenner, *A force superposition approach to plane elastic stress and strain analysis*, J. Strain Anal., 36 (2001), pp. 517–529.
[10] M. A. Golberg and C. S. Chen, *Discrete Projection Methods for Integral Equations*, Computational Mechanics Publications, Southampton, 1997.
[11] M. A. Golberg and C. S. Chen, *The method of fundamental solutions for potential, Helmholtz and diffusion problems*, in Boundary Integral Methods: Numerical and Mathematical Aspects, Comput. Eng. 1, WIT Press/Computational Mechanics Publications, Boston, MA, 1999, pp. 103–176.
[12] P. Henrici, *Fast Fourier methods in computational complex analysis*, SIAM Rev., 21 (1979), pp. 481–527.
[13] M. Katsurada, *A mathematical study of the charge simulation method. II*, J. Fac. Sci. Univ. Tokyo Sect. IA Math., 36 (1989), pp. 135–162.
[14] J. A. Kołodziej, *Applications of the Boundary Collocation Method in Applied Mechanics*, Wydawnictwo Politechniki Poznańskiej, Poznań, 2001 (in Polish).
[15] S. R. Kuo, J. T. Chen, and C. X. Huang, *Analytical study and numerical experiments for true and spurious eigensolutions of a circular cavity using the real-part dual BEM*, Internat. J. Numer. Methods Engrg., 48 (2000), pp. 1401–1422.
[16] U. Lamp, K.-T. Schleicher, and W. L. Wendland, *The fast Fourier transform and the numerical solution of one-dimensional boundary integral equations*, Numer. Math., 47 (1985), pp. 15–38.
[17] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.
[18] M. N. Özişik, *Boundary Value Problems of Heat Conduction*, International Textbook Company, Scranton, PA, 1968.
[19] J. H. Kane, *Boundary Element Analysis in Engineering Continuum Mechanics*, Prentice Hall, Engelwood Cliffs, NJ, 1994.
[20] Y.-S. Smyrlis and A. Karageorghis, *Some aspects of the method of fundamental solutions for certain harmonic problems*, J. Sci. Comput., 16 (2001), pp. 341–371.
[21] Y.-S. Smyrlis and A. Karageorghis, *Some aspects of the method of fundamental solutions for certain biharmonic problems*, CMES Comput. Model. Eng. Sci., 4 (2003), pp. 535–550.
[22] Y.-S. Smyrlis and A. Karageorghis, *A matrix decomposition MFS algorithm for axisymmetric potential problems*, Engrg. Anal. Boundary Elem., 28 (2004), pp. 463–474.
[23] Y.-S. Smyrlis and A. Karageorghis, *Numerical analysis of the MFS for certain harmonic problems*, M2AN Math. Model. Numer. Anal., 28 (2004), pp. 495–517.
[24] I. Stakgold, *Green's Functions and Boundary Value Problems*, John Wiley & Sons, New York, Chichester, Brisbane, 1979.
[25] T. Tsangaris, Y.-S. Smyrlis, and A. Karageorghis, *Numerical analysis of the MFS for harmonic problems in annular domains*, Numer. Methods Partial Differential Equations, to appear.

SPECIAL ISSUE: 2004 COPPER MOUNTAIN CONFERENCE

The biennial Copper Mountain Conference on Iterative Methods was held March 28–April 2, 2004, in Copper Mountain, Colorado. In addition to the traditional management services provided by Front Range Scientific Computations, Inc., Howard Elman (University of Maryland) and Panayot Vassilevski (Center for Applied Scientific Computing at LLNL) served as co-chairs representing the conference's new organizing institutions. The Conference gratefully acknowledges support from all three major national labs—Los Alamos, Lawrence Livermore, and Sandia—as well as support from the Department of Energy, IBM, and the National Science Foundation.

Traditionally, the Copper Mountain Conference is organized in cooperation with the Society for Industrial and Applied Mathematics (SIAM). This special issue was open to the public and was advertised in advance on the web sites of both the [illegible] (SISC) and the Copper Mountain Conference.

There were 35 manuscripts submitted; 15 found their place in this special issue. It is the guest editor-in-chief's opinion that these papers reflect state-of-the-art trends in iterative methods, among them parallel-block preconditioning, various improved versions of GMRES (block, relaxed, and flexible), approximate factorization preconditioners for saddle-point problems, and iterative methods exploiting deflation. But above all, this special issue clearly demonstrates the strong impact of multigrid/multilevel methods as the unique optimal-order iterative methods in various areas of applications. It also demonstrates the search for optimal-order methods or preconditioners on the basis of multilevel ideas in variety of nontraditional areas.

I would like to thank the guest editorial board—Loyce Adams, Xiao-Chuan Cai, Iain Duff, Roland Freund, Anne Greenbaum, Tim Kelley, David Keyes, Tom Manteuffel, Steve McCormick, Ray Tuminaro, Henk van der Vorst, Olof Widlund, and Carol Woodward of the conference program committee—as well as Raymond Chan of the SISC editorial board and the many anonymous referees for their hard work to help meet the strict deadlines.

I would also like to thank the SIAM staff for their constant support and efforts to bring this issue to life. Of course, this issue would not have been possible without the authors' contributions. Some of these submissions also appeared in the student paper competition, a unique event at the Copper Mountain conferences that has proven to be a very successful forum for presenting the latest results in the field. We have had increasingly strong participation from very talented recent Ph.D. holders and current Ph.D. students in this competition.

We thank all the authors for their valuable contributions and hope that this special issue will stimulate further progress in this exciting area of research.

Panayot Vassilevski

[illegible]

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1513–1533
© 2006 Society for Industrial and Applied Mathematics

A NEW BLOCK PARALLEL SOR METHOD AND ITS ANALYSIS*

DEXUAN XIE†

Abstract. As the development of the PSOR method (a point parallel SOR method by mesh domain partitioning proposed in [*SIAM J. Sci. Comput.*, 20 (1999), pp. 2261–2281], this paper introduces a new mesh domain partition and ordering (the multitype partition and ordering), and proposes a new block parallel SOR (BPSOR) method for numerically solving 2-dimensional (2D) or three-dimensional (3D) elliptic boundary problems. A general mathematical analysis shows that the BPSOR method can have the same asymptotic convergence rate as the corresponding sequential block SOR method if the coefficient matrix of the block linear system is "consistently ordered." It also shows that the original sequential ordering can be maintained in the parallel implementation of the BPSOR method so that the BPSOR method can be effectively applied to solve complex elliptic boundary problems. Furthermore, three particular multitype orderings are proposed based on strip and block mesh partitions, which lead to three effective BPSOR methods for solving the five-point like linear systems (in two dimensions) and the seven-point like linear systems (in three dimensions). In addition, it is shown that the PSOR method can be generated from BPSOR if each block equation is solved approximately by only one point SOR iteration. Thus, the PSOR method is well defined without involving any interprocessor data communication operations, and extended to solving three-dimensional (3D) problems. Finally, numerical results are presented which confirm the theoretical results and show that BPSOR has a good parallel performance on a parallel MIMD computer.

Key words. block SOR, PSOR, domain decomposition, parallel computing

AMS subject classifications. 65Y05, 65F10

DOI. 10.1137/040604777

1. Introduction.

The successive overrelaxation (SOR) method [28] is an important classic iterative method for solving large-scale sparse linear systems arising from a finite element or finite difference discretization of the elliptic boundary problem. It is also widely used to produce efficient preconditioners [10, 14, 28] and robust smoothers [10, 22, 25, 27]. Due to the efficiency and simplicity in implementation, the SOR method is often used in scientific computing. For example, in protein simulations, three widely used biomolecular simulation packages, CHARMM [5], DelPha [19], and UHBD [7], include the SOR method as the default solver for the linearized Poisson–Boltzmann equation (an elliptic Dirichlet boundary problem with discontinuous coefficients), which models the electrostatic potential energy of a solvate protein system through treating the solvent and the protein as two different continuum dielectric materials [6, 13, 20]. However, the SOR method using the natural ordering is sequential. It must be parallelized to be implemented on a parallel computer.

Various parallel SOR methods have been developed by using multicolor ordering schemes and domain decomposition techniques [1, 2, 4, 8, 9, 12, 15, 16, 17, 18, 23, 24, 26, 28]. Usually, a parallel SOR method defined by multicolor techniques can have a faster convergence rate but may be more difficult to apply for solving complex elliptic boundary problems than a parallel SOR method constructed from a domain decomposition. To improve the convergence rate of a domain decomposition algorithm, a novel mesh partition strategy was proposed in [26] which led to an efficient parallel

*Received by the editors March 4, 2004; accepted for publication (in revised form) October 19, 2004; published electronically February 3, 2006. This work was supported by the National Science Foundation through grant DMS-0241236.

http://www.siam.org/journals/sisc/27-5/60477.html

†Department of Mathematical Sciences, University of Wisconsin, Milwaukee, WI 53201-0413 (dxie@uwm.edu).

SOR method called the PSOR method. The PSOR method was shown to retain the convergence rate of the sequential SOR method [26]. However, the PSOR method was only considered as a point iterative algorithm for solving 2-dimensional (2D) elliptic boundary value problems, and its definition involved a usage of interprocessor data communication operations.

As the development of the PSOR method, this paper proposes a new domain decomposition scheme called the multitype partition. The idea of the multitype partition can be briefly described as follows: the mesh domain Ω_h is first divided into p subdomains, $\{\Omega_{h\mu}\}_{\mu=1}^{p}$, and then the mesh points of each subdomain are grouped into t types, $\{\Omega_{h,\mu}^{i}\}_{i=1}^{t}$, such that each type is coupled only with different adjacent types and at least one of the adjacent types is located in a neighboring subdomain. Here p is the number of processors, and t is less than or equal to the number of neighboring subdomains. The multitype ordering is then defined by numbering the tp type subdomains $\{\Omega_{h,\mu}^{i}\}$ from $\mu = 1$ to p for $i = 1$ to t, while the original ordering is kept within each type subdomain.

This paper then proposes a new block parallel SOR (BPSOR) method by using the multitype partition. A general mathematical analysis shows that the BPSOR method can have the same convergence rate as the corresponding sequential block SOR method if the coefficient matrix of the block linear system is "consistently ordered." (Note that a consistently ordered matrix in this paper usually refers to the blocks of the matrix being consistently ordered. See Definition 9-2.2 in [11] for details.) This paper also introduces three particular multitype partitions based on strip and block mesh partitioning for solving 5-point like (in two dimensions) and 7-point like (in three dimensions) finite difference equations. The three corresponding BPSOR methods are shown, separately, to have the same convergence rate as the corresponding sequential block SOR method.

Two matrix forms of the BPSOR iterative expression are obtained in this paper. One is formulated in the original sequential ordering, and the other in the multitype ordering. With the matrix form formulated by the multitype ordering, the BPSOR method is shown to be equivalent to the normal block SOR method for solving the reordered block linear system by the multitype ordering. Hence, the BPSOR method can be analyzed in the framework of the classic SOR theory [28]. On the other hand, with the matrix form formulated by the original sequential ordering, the BPSOR method is shown to be an iterative method for solving the original block linear system. This implies that the BPSOR method can be implemented easily in parallel on an MIMD parallel computer without any reordering scheme of the unknowns.

In addition, this paper shows that the point PSOR method can be defined as an inexact BPSOR method in which each block equation is solved approximately by only one point sequential SOR iteration. As an inexact BPSOR method, the PSOR method is now well defined without involving any interprocessor data communication operations and extended to solving 3-dimensional (3D) problems. Since the BPSOR theory is developed based on the work of PSOR in [26], some notation and analysis techniques from [26] are kept in this paper for consistency.

Numerical experiments were made on a parallel MIMD computer, the SGI (Silicon Graphics, Inc.) Origin 2000 at the University of Wisconsin-Milwaukee, which has 16 R12000 400 MHz processors. The test model problem is a 7-point finite difference approximation to the Poisson equation on a unit cubic domain. The parallel program is written in FORTRAN 77 and MPI [3]. Each block equation was solved either by the sparse LDL^T subroutine from the SGI's Scientific Computing Software Library or

simply by the sequential point SOR method. Numerical results confirm that BPSOR has the same convergence rate as the corresponding sequential block SOR method. They also show that BPSOR has good parallel performances. For examples, speedups around 13 to 16 were obtained on the sixteen processors of the SGI Origin 2000 for solving the model problem with $h = 1/65$ and $1/129$. Moreover, numerical results show that BPSOR can significantly improve the convergence rate and parallel performance of the point PSOR method. For example, BPSOR is found to take about 16 times fewer iterations and about 16 times less interprocessor communication time than the point PSOR method for the model problem with grid size $h = 1/65$ on eight processors. Here the number of iterations is determined by an iteration termination rule. Even with the point SOR method as a solver of each block equation, test results show that the total CPU time of BPSOR can be less than that of PSOR. Clearly, the total CPU time of BPSOR can be further reduced if each block equation of BPSOR is solved by a more efficient iterative algorithm (such as the multigrid method).

It is possible to generate efficient parallel preconditioners and robust parallel smoothers from the BPSOR method through a proper selection of a local solver, a mesh domain partition, and an accuracy of local solutions. Also, the multitype ordering can be extended to parallelize the sequential Schwartz multiplicative domain decomposition method [21]. Such research topics will be investigated in the sequential papers.

The remainder of the paper is organized as follows. Section 2 introduces the multitype partition and the BPSOR method. Section 3 analyzes the BPSOR method. Section 4 defines three particular and useful BPSOR methods based on strip and block partitionings. Numerical results are presented in section 5.

2. The multitype partition and the BPSOR method.

2.1. The block linear system by the multitype partition. Consider a large-scale system of linear equations that arises from a finite element or finite difference approximation to a 2D or 3D elliptic Dirichlet boundary problem. Let the mesh domain Ω_h be a set of grid points on which the unknowns of the linear system are defined with a grid size of h. To solve the linear system in parallel on p processors of a multiprocessor computer, the mesh domain Ω_h is divided into p disjoint subdomains $\Omega_{h,\mu}$ with $\mu = 1, 2, \ldots, p$ such that

$$\Omega_h = \Omega_{h,1} \cup \Omega_{h,2} \cup \cdots \cup \Omega_{h,p} \quad \text{and} \quad \Omega_{h,\mu} \cap \Omega_{h,\nu} = \emptyset \quad \text{for } \mu \neq \nu. \tag{2.1}$$

Based on domain decomposition (2.1), the linear system can be formulated into p block linear equations:

$$\sum_{\nu=1}^{p} A_{\mu\nu}\mathcal{U}_\nu = \mathcal{F}_\mu, \quad \mu = 1, 2, \ldots, p, \tag{2.2}$$

where $\mathcal{U}_\nu$ is the vector of unknowns defined on Ω_ν, $\mathcal{F}_\mu$ is a given right-hand side vector on $\Omega_{h,\mu}$, $A_{\mu\mu}$ is a nonsingular matrix defined on $\Omega_{h,\mu}$, and $A_{\mu\nu}$ denotes the matrix that indicates the connection of Ω_μ with Ω_ν. If the matrix $A_{\mu\nu}$ with $\mu \neq \nu$ contains at least one nonzero entry, Ω_μ is said to be connected to Ω_ν. Two subdomains are referred to as two neighboring subdomains if they are connected and the intersection of their closures is nonempty. It is assumed that the matrix $A_{\mu\nu}$ with $\mu \neq \nu$ is nonzero if and only if Ω_μ and Ω_ν are two neighboring subdomains.

According to the connection information of each subdomain with its neighboring subdomains, the grid points of each subdomain can be grouped into t different types

such that each type is connected only to its different adjacent types, and at least one of the adjacent types is located in a neighboring subdomain. Here the number t is less than or equal to the number of neighboring subdomains, and two types are called two adjacent types only if they are connected and have a nonempty closure intersection. If $\Omega_{h,\mu}^i$ denotes the ith type of the μth subdomain, then the multitype partition of the mesh domain Ω_h is defined by

$$\Omega_{h,\mu} = \Omega_{h,\mu}^1 \cup \Omega_{h,\mu}^2 \cup \cdots \cup \Omega_{h,\mu}^t \quad \text{and} \quad \Omega_{h,\mu}^i \cap \Omega_{h,\mu}^j = \emptyset \quad \text{for } i \neq j \tag{2.3}$$

for $\mu = 1, 2, \ldots, p$.

Based on the multitype partition (2.3), a system of tp block linear equations is formulated as

$$\sum_{\nu=1}^{p} \sum_{j=1}^{t} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j = \mathcal{F}_\mu^i, \quad i = 1, 2, \ldots, t \text{ and } \mu = 1, 2, \ldots, p, \tag{2.4}$$

where $\mathcal{U}_\mu^i$ and $\mathcal{F}_\mu^i$ denote the vectors defined on $\Omega_{h,\mu}^i$, $A_{\mu\mu}^{ii}$ is a nonsingular matrix defined on $\Omega_{h,\mu}^i$, and $A_{\mu\nu}^{ij}$ denotes the matrix that indicates the connections of $\Omega_{h,\mu}^i$ with $\Omega_{h,\nu}^j$. In the multitype partition, the submatrix $A_{\mu\nu}^{ij}$ with $\mu \neq \nu$ and $i \neq j$ is nonzero only if $\Omega_{h,\mu}^i$ and $\Omega_{h,\nu}^j$ are two adjacent types. Specifically, $A_{\mu\nu}^{ij} = 0$ for all $i \leq j$ if $\mu < \nu$ and for all $i \geq j$ if $\mu > \nu$ so that the block linear system of (2.4) can be simplified as

$$\sum_{j=1}^{t} A_{\mu\mu}^{ij} \mathcal{U}_\mu^j + \sum_{\nu=\mu+1}^{p} \sum_{j=1}^{i-1} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j + \sum_{\nu=1}^{\mu-1} \sum_{j=i+1}^{t} A_{\mu\nu}^{ij} \mathcal{U}_\nu^j = \mathcal{F}_\mu^i, \tag{2.5}$$

where $\mu = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, t$. Many submatrices $A_{\mu\nu}^{ij}$ in (2.5) may still be zero, but they are retained in (2.5) to include various multitype partitions.

The linear system can be expressed in a matrix equation if a global ordering of the unknowns is given. A natural way to order the tp-type subdomains $\{\Omega_{h,\mu}^i\}$ is to order them from $i = 1$ to t for each value of μ from 1 to p. Such a global ordering will be referred to as the natural ordering.

In the natural ordering, the block linear system (2.5) can be expressed in the matrix form

$$Au = f, \tag{2.6}$$

where u and f are two collective vectors defined by

$$u = \begin{pmatrix} \mathcal{U}_1 \\ \mathcal{U}_2 \\ \vdots \\ \mathcal{U}_p \end{pmatrix} \text{ with } \mathcal{U}_\mu = \begin{pmatrix} \mathcal{U}_\mu^1 \\ \mathcal{U}_\mu^2 \\ \vdots \\ \mathcal{U}_\mu^t \end{pmatrix}, \; f = \begin{pmatrix} \mathcal{F}_1 \\ \mathcal{F}_2 \\ \vdots \\ \mathcal{F}_p \end{pmatrix} \text{ with } \mathcal{F}_\mu = \begin{pmatrix} \mathcal{F}_\mu^1 \\ \mathcal{F}_\mu^2 \\ \vdots \\ \mathcal{F}_\mu^t \end{pmatrix}, \tag{2.7}$$

and A is a $tp \times tp$ partitioned block matrix defined by

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1p} \\ A_{21} & A_{22} & \cdots & A_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pp} \end{bmatrix} \text{ with } A_{\mu\mu} = \begin{bmatrix} A_{\mu\mu}^{11} & A_{\mu\mu}^{12} & \cdots & A_{\mu\mu}^{1t} \\ A_{\mu\mu}^{21} & A_{\mu\mu}^{22} & \cdots & A_{\mu\mu}^{2t} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\mu\mu}^{t1} & A_{\mu\mu}^{t2} & \cdots & A_{\mu\mu}^{tt} \end{bmatrix}, \tag{2.8}$$

and

$$A_{\mu\nu} = \begin{cases} \begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{\mu\nu}^{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{\mu\nu}^{t1} & \cdots & A_{\mu\nu}^{t,t-1} & 0 \end{bmatrix} & \text{for } \mu < \nu, \\ \begin{bmatrix} 0 & A_{\mu\nu}^{12} & \cdots & A_{\mu\nu}^{1t} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & A_{\mu\nu}^{t-1,t} \\ 0 & 0 & \cdots & 0 \end{bmatrix} & \text{for } \mu > \nu. \end{cases} \tag{2.9}$$

Here the block matrix A is assumed to be sparse, symmetric positive definite, or consistently ordered.

2.2. The sequential block SOR method by the *t*-type partition. Let the block matrix A be split into the form

$$A = D - L - U, \tag{2.10}$$

where L and U are the two strictly block lower- and upper-triangular matrices, respectively, and D is the block diagonal matrix. They are defined as

$$D = \operatorname{diag}(D_1, D_2, \ldots, D_p) \text{ with } D_\mu = \operatorname{diag}(A_{\mu\mu}^{11}, A_{\mu\mu}^{22}, \ldots, A_{\mu\mu}^{tt}), \tag{2.11}$$

$$L = -\begin{bmatrix} L_1 & 0 & \cdots & 0 \\ A_{21} & L_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ A_{p1} & \cdots & A_{p(p-1)} & L_p \end{bmatrix}, \quad U = -\begin{bmatrix} U_1 & A_{12} & \cdots & A_{1p} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & U_{p-1} & A_{(p-1)p} \\ 0 & \cdots & 0 & U_p \end{bmatrix},$$

where the abbreviation $\operatorname{diag}(\alpha_1, \alpha_2, \ldots, \alpha_m)$ denotes the diagonal matrix with the diagonal elements $\alpha_1, \alpha_2, \ldots, \alpha_m$, and

$$L_\mu = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ A_{\mu\mu}^{21} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ A_{\mu\mu}^{t1} & \cdots & A_{\mu\mu}^{t,t-1} & 0 \end{bmatrix}, \quad U_\mu = \begin{bmatrix} 0 & A_{\mu\mu}^{12} & \cdots & A_{\mu\mu}^{1t} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{\mu\mu}^{t-1,t} \\ 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{2.12}$$

In terms of the above matrices, the block SOR method by the t-type partition is defined by

$$u^{(k+1)} = M_{SOR} u^{(k)} + \omega[D - \omega L]^{-1} f, \tag{2.13}$$

where $u^{(k)}$ is a block vector in the form

$$u^{(k)} = \begin{pmatrix} \mathcal{U}_1^{(k)} \\ \mathcal{U}_2^{(k)} \\ \vdots \\ \mathcal{U}_p^{(k)} \end{pmatrix} \quad \text{with} \quad \mathcal{U}_\mu^{(k)} = \begin{pmatrix} \mathcal{U}_\mu^{1,(k)} \\ \mathcal{U}_\mu^{2,(k)} \\ \vdots \\ \mathcal{U}_\mu^{t,(k)} \end{pmatrix} \quad \text{for } \mu = 1, 2, \ldots, p, \tag{2.14}$$

$\mathcal{U}_\mu^{i,(k)}$ is the kth iterate on $\Omega_{h,\mu}^i$, and M_{SOR} is the SOR iteration matrix defined by

$$M_{SOR} = [D - \omega L]^{-1}[(1-\omega)D + \omega U]. \tag{2.15}$$

See [28] for the details on the analysis of the SOR method.

However, the block SOR method is a sequential iterative method. In fact, on each type subdomain, Ω_μ^i, the component expression of the block SOR method (2.13) can be written as

$$\begin{aligned}\mathcal{U}_\mu^{i,(k+1)} = \omega(A_{\mu\mu}^{ii})^{-1}\Bigg[\mathcal{F}_\mu^i - \sum_{j=1}^{i-1} A_{\mu\mu}^{ij}\mathcal{U}_\mu^{j,(k+1)} - \sum_{j=i+1}^{t} A_{\mu\mu}^{ij}\mathcal{U}_\mu^{j,(k)} \\ - \sum_{\nu=\mu+1}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)} - \sum_{\nu=1}^{\mu-1}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)}\Bigg] + (1-\omega)\mathcal{U}_\mu^{i,(k)},\end{aligned} \tag{2.16}$$

where $i = 1, 2, \ldots, t$, $\mu = 1, 2, \ldots, p$, and $k = 0, 1, 2, \ldots$. From (2.16) it can be seen that the current new iterate $\mathcal{U}_\mu^{i,(k+1)}$ depends on the previous new iterates $\{\mathcal{U}_\nu^{j,(k+1)}\}_{\nu=1}^{\mu-1}$, which are located on other subdomains, $\{\Omega_{h,\nu}\}_{\nu=1}^{\mu-1}$. Thus, the block SOR method by the multitype partition is sequential in terms of subdomains $\{\Omega_{h,\nu}\}_{\nu=1}^{p}$.

2.3. The BPSOR method. However, the sequential block SOR method by the multitype partition can be easily parallelized by using the multitype ordering, which orders the subdomains $\{\Omega_{h,\mu}^i\}$ from $\mu = 1$ to p for each value of i from 1 to t. In detail, the subdomains $\{\Omega_{h,\mu}^1\}_{\mu=1}^p$ of type 1 is first ordered from $\mu = 1$ to p, then the subdomains $\{\Omega_{h,\mu}^2\}_{\mu=1}^p$ of type 2, and so on, through the subdomains $\{\Omega_{h,\mu}^t\}_{\mu=1}^p$ of type t. Within each type subdomain, the original natural ordering of mesh points is maintained so that each block equation of (2.5) is kept. The multitype ordering with t types will be simply referred to as the t-type ordering.

In the t-type ordering, the iterates $\mathcal{U}_\nu^{j,(k+1)}$ with $j = 1, 2, \ldots, i-1$ and $\nu = \mu, \mu+1, \ldots, p$ will be computed before the calculation of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$. Hence, the p iterates $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$ can be defined by

$$\begin{aligned}\mathcal{U}_\mu^{i,(k+1)} = \omega(A_{\mu\mu}^{ii})^{-1}\Bigg[\mathcal{F}_\mu^i - \sum_{\nu=\mu}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)} - \sum_{\nu=1}^{\mu}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)}\Bigg] \\ + (1-\omega)\mathcal{U}_\mu^{i,(k)} \quad \text{for} \quad \mu = 1, 2, \ldots, p, \quad i = 1, 2, \ldots, t.\end{aligned} \tag{2.17}$$

Here $k = 0, 1, 2, \ldots$, $\mathcal{U}_\nu^{j,(0)}$ is a given initial guess, and $\omega \in (0, 2)$. Since all the iterates on the right-hand side of the above equations are given or computed in the previous steps, the iterative expressions of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{\mu=1}^p$ in (2.17) are completely independent. In addition, these p iterates are defined on p different subdomains, $\{\Omega_{h,\mu}\}_{\mu=1}^p$, respectively. Hence, when each subdomain is assigned to one processor, they can be calculated in parallel on p processors. Therefore, the iterative method defined by (2.17) is a parallel scheme based on the domain decomposition (2.1). For clarity, it will be called the block parallel SOR (BPSOR) method.

A parallel implementation scheme of BPSOR is described in Algorithm 1. Here processor μ contains both subdomain Ω_μ and its ghost boundary, which is a set of the mesh points of the neighboring subdomains that are connected to Ω_μ for $\mu = 1, 2, \ldots, p$.

ALGORITHM 1 (for parallel implementation of BPSOR). u_μ^i $\mathcal{U}_\mu^{i,(k)}$ $\mathcal{U}_\mu^{i,(0)}$ $\mu = 1$ p

1. $u_\mu^i := \mathcal{U}_\mu^{i,(0)}$ $i = 1, 2, \ldots, t$ $k = 1$
2. $\{u_\mu^i\}_{i=1}^t$ i $i = 1, 2, \ldots, t$
   (a) $A_{\mu\mu}^{ii} v_\mu^i = f_\mu^i$ v_μ^i f_μ^i

$$f_\mu^i := \mathcal{F}_\mu^i - \sum_{j=1}^{i-1} \sum_{\nu=\mu}^{p} A_{\mu\nu}^{ij} u_\nu^j - \sum_{j=i+1}^{t} \sum_{\nu=1}^{\mu} A_{\mu\nu}^{ij} u_\nu^j.$$

   (b) $u_\mu^i := \omega v_\mu^i + (1-\omega) u_\mu^i$.
   (c) u_μ^i (i)
3. $k := k + 1$ 2

3. The analysis of the BPSOR method.

3.1. BPSOR as an iterative method using the natural ordering. From Algorithm 1 it can be seen that the BPSOR iterates $\{\mathcal{U}_\mu^{i,(k)}\}$ are calculated from $i = 1$ to t within each processor. Thus, the pt BPSOR iterates $\{\mathcal{U}_\mu^{i,(k)}\}$ can be formulated as a block vector in the natural ordering.

THEOREM 3.1. $\{\mathcal{U}_\mu^{i,(k)}\}$ (2.17) $u^{(k)}$ (2.14)

$$u^{(k+1)} = M_{BPSOR} u^{(k)} + \omega [D - \omega(B + M)]^{-1} f \quad k = 0, 1, 2, \ldots, \tag{3.1}$$

$\omega \in (0, 2)$, M_{BPSOR}

$$M_{BPSOR} = [D - \omega(B + M)]^{-1} [(1 - \omega) D + \omega(C + N)], \tag{3.2}$$

D (2.11) B C

$$B = -\operatorname{diag}(L_1, L_2, \ldots, L_p) \qquad C = -\operatorname{diag}(U_1, U_2, \ldots, U_p), \tag{3.3}$$

M N

$$N = L - B \qquad M = U - C. \tag{3.4}$$

L, U, L_μ U_μ (2.11) (2.12) . With (3.4), the matrices L and U can be expressed by

$$L = N + B \quad \text{and} \quad U = M + C. \tag{3.5}$$

Thus, the regular splitting (2.10) becomes

$$A = D - B - C - M - N. \tag{3.6}$$

An iterative method using the natural ordering means the one for solving the linear system $Au = f$ given in (2.6). With the splitting formula (3.6), an iterative sequence $\{u^{(k)}\}$ using the natural ordering can be defined by

$$Du^{(k+1)} = (B + M)u^{(k+1)} + f + (C + N)u^{(k)}, \quad k = 0, 1, 2, \ldots.$$

Introducing the relaxation parameter ω into the above iterative sequence gives

$$Du^{(k+1)} = \omega[(B + M)u^{(k+1)} + f + (C + N)u^{(k)}] + (1 - \omega)u^{(k)}. \tag{3.7}$$

Solving the above equation for $u^{(k+1)}$ yields the matrix iterative expression (3.1). Expanding the matrix expression (3.1) into the block component expressions in terms of $\{\mathcal{U}_\mu^{i,(k)}\}$ yields the BPSOR iterative expressions in (2.17). Thus, the block iterative sequence defined by (3.7) is exactly the BPSOR method (2.17). Hence, the expression (3.1) is also the matrix iterative expression of the BPSOR method. Therefore, the BPSOR method can be regarded as an iterative method using the natural ordering. This completes the proof of the theorem.

Because of (3.5), the BPSOR iterative expression (3.1) can be changed to the sequential block SOR iterative expression (2.13) by simply switching the positions of N and M. Such a matrix-switch operation is completed in step (c) of Algorithm 1. In this sense, BPSOR can be regarded as a parallel version of the block SOR method (2.13), which is generated from a novel usage of interprocessor communication operations. □

3.2. BPSOR as the block SOR using the multitype ordering. Let P be a permutation matrix defined by $\hat{u} = Pu$ for u in the form (2.7), and $\hat{u}$ in the following form:

$$\hat{u} = \begin{pmatrix} \hat{\mathcal{U}}_1 \\ \hat{\mathcal{U}}_2 \\ \vdots \\ \hat{\mathcal{U}}_t \end{pmatrix} \text{ with } \quad \hat{\mathcal{U}}_i = \begin{pmatrix} \mathcal{U}_1^i \\ \mathcal{U}_2^i \\ \vdots \\ \mathcal{U}_p^i \end{pmatrix} \text{ for } 1 \le i \le t. \tag{3.8}$$

Obviously, by the t-type ordering, the block linear system $Au = f$ in (2.6) can be reordered into an equivalent block linear system. In terms of P, the reordered linear system can be expressed as

$$\hat{A}\hat{u} = \hat{f}, \tag{3.9}$$

where $\hat{A} = PAP^T$, $\hat{u} = Pu$, and $\hat{f} = Pf$. The matrix form of $\hat{A}$ can be found as

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & \cdots & \hat{A}_{1t} \\ \hat{A}_{21} & \hat{A}_{22} & \cdots & \hat{A}_{2t} \\ \vdots & \ddots & \ddots & \vdots \\ \hat{A}_{t1} & \cdots & \hat{A}_{t,t-1} & \hat{A}_{tt} \end{bmatrix} \tag{3.10}$$

with $\hat{A}_{ij}$ being defined by

$$\hat{A}_{ij} = \begin{cases} \operatorname{diag}(A_{11}^{ii}, A_{22}^{ii}, \ldots, A_{pp}^{ii}) & \text{for} \quad i = j, \\ \begin{bmatrix} A_{11}^{ij} & A_{12}^{ij} & \cdots & A_{1p}^{ij} \\ 0 & A_{22}^{ij} & \cdots & A_{2p}^{ij} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & A_{pp}^{ij} \end{bmatrix} & \text{for} \quad i > j, \\ \begin{bmatrix} A_{11}^{ij} & 0 & \cdots & 0 \\ A_{21}^{ij} & A_{22}^{ij} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A_{p1}^{ij} & A_{p2}^{ij} & \cdots & A_{pp}^{ij} \end{bmatrix} & \text{for} \quad i < j. \end{cases} \tag{3.11}$$

THEOREM 3.2. $\{\mathcal{U}_{\mu}^{i,(k)}\}$ (2.17) $\hat{u}^{(k)}$ (3.8) (2.17)

$$\hat{u}^{(k+1)} = \hat{M}_{SOR}\hat{u}^{(k)} + \omega(\hat{D} - \omega\hat{L})^{-1}\hat{f}, \tag{3.12}$$

$\hat{M}_{SOR}$

$$\hat{M}_{SOR} = (\hat{D} - \omega\hat{L})^{-1}[(1-\omega)\hat{D} + \omega\hat{U}], \tag{3.13}$$

$\hat{D}, \hat{L}$ $\hat{U}$

$$\hat{A} = \hat{D} - \hat{L} - \hat{U}, \tag{3.14}$$

$\hat{A}$ (3.10)

. By the splitting of $\hat{A}$ in (3.14), it is easy to find that the block SOR method using the multitype ordering has the matrix iterative expression (3.12). In terms of $\mathcal{U}_{\mu}^{i,(k)}$, the block component expressions of the matrix form (3.12) turn out to be the same as the iterative expressions of the BPSOR method in (2.17). Hence, the BPSOR method is equivalent to the block SOR method using the multitype ordering for solving the reordered linear system (3.9). This completes the proof of the theorem. □

3.3. The relationship between the two iteration matrices. Two matrix iterative expressions, (3.1) and (3.12), have been formulated for the BPSOR method by the natural ordering and the t-type ordering. The expression of (3.1) indicates that the BPSOR method can be easily implemented in parallel since the original natural ordering is maintained. From the expression of (3.12) it implies that the BPSOR method can be analyzed under the framework of the classic SOR theory [28].

THEOREM 3.3. M_{BPSOR} $\hat{M}_{SOR}$ (3.2) (3.13) P (3.8)

$$\hat{M}_{SOR} = PM_{BPSOR}P^T. \tag{3.15}$$

. The key step in proving (3.15) is to show the following two identities:

$$\hat{L} = P(B+M)P^T \quad \text{and} \quad \hat{U} = P(C+N)P^T. \tag{3.16}$$

The proof of the first one is given below.

By multiplying both sides of expression (3.1) by P, (3.1) can be rewritten as

$$\begin{aligned}\hat{u}^{(k+1)} = {} & [\hat{D} - \omega P(B+M)P^T]^{-1}[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} \\ & + \omega[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f},\end{aligned} \tag{3.17}$$

where the identities $P^TP = I$, $Pu^{(k)} = \hat{u}^{(k)}$, $Pf = \hat{f}$, and $PDP^T = \hat{D}$ have been used. The following two expressions of $\hat{u}^{(1)}$ are then produced from (3.17) and (3.12) by setting $\hat{u}^{(0)} = 0$:

$$\hat{u}^{(1)} = \omega[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f} \quad \text{and} \quad \hat{u}^{(1)} = \omega[\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Combining them as one identity yields

$$[\hat{D} - \omega P(B+M)P^T]^{-1}\hat{f} = [\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Since $\hat{f}$ is arbitrary, the above identity follows as $[\hat{D} - \omega P(B+M)P^T]^{-1} = [\hat{D} - \omega\hat{L}]^{-1}$. Simplifying it gives $\hat{L} = P(B+M)P^T$.

To prove the second identity of (3.16), substituting $P(B+M)P^T$ by $\hat{L}$ in (3.17) gives

$$\hat{u}^{(k+1)} = [\hat{D} - \omega\hat{L}]^{-1}[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} + \omega[\hat{D} - \omega\hat{L}]^{-1}\hat{f}.$$

Combining the above expression with (3.12) yields

$$[(1-\omega)\hat{D} + \omega P(C+N)P^T]\hat{u}^{(k)} = [(1-\omega)\hat{D} + \omega U]\hat{u}^{(k)} \quad \text{for all } k \geq 0.$$

Since the above identity holds for any vector of $\hat{u}^{(k)}$, it implies that

$$(1-\omega)\hat{D} + \omega P(C+N)P^T = (1-\omega)\hat{D} + \omega U.$$

Simplifying again gives that $\hat{U} = P(C+N)P^T$.

Finally, with (3.16) and $\hat{D} = PDP^T$,

$$\begin{aligned}\hat{M}_{SOR} &= (\hat{D} - \omega\hat{L})^{-1}[(1-\omega)\hat{D} + \omega\hat{U}] \\ &= (PDP^T - \omega P(B+M)P^T)^{-1}[(1-\omega)PDP^T + \omega P(C+N)P^T] \\ &= P(D - \omega(B+M))^{-1}[(1-\omega)D + \omega(C+N)]P^T = PM_{BPSOR}P^T.\end{aligned} \tag{3.18}$$

This completes the proof of Theorem 3.3. □

3.4. Convergence of the BPSOR method. The main convergence results of the BPSOR method are presented in the following theorem.

THEOREM 3.4. $M_J, M_{SOR},$ M_{BPSOR}

A (2.8)

$u^{(k)}$ $u^* = A^{-1}f$ $k \to \infty$

ω $0 < \omega < 2$

A M_J $\rho(M_J) < 1$

$$\rho(M_{BPSOR}(\omega)) = \rho(M_{SOR}(\omega)) \tag{3.19}$$

$$= \begin{cases} \frac{1}{4}\left[\omega\rho(M_J) + \sqrt{\omega^2\rho(M_J)^2 - 4(\omega-1)}\right]^2 & 0 < \omega \le \omega_b, \\ \omega - 1 & \omega_b \le \omega < 2, \end{cases} \tag{3.20}$$

$$\omega_b = \frac{2}{1 + \sqrt{1 - \rho(M_J)^2}}. \tag{3.21}$$

$\rho(\cdot)$ M_{SOR} M_{BPSOR} (2.15)

(3.2) $M_J = L + U$ L U (2.11)

Since the reordered matrix $\hat{A}$ defined in (3.10) satisfies $\hat{A} = PAP^T$ and A is symmetric positive definite, then $\hat{A}$ is symmetric positive definite. Also, from Theorem 4.1 in [26] it follows that $\hat{A}$ is consistently ordered if A is consistently ordered. Moreover, in Theorem 3.2, the BPSOR method has been shown to be equivalent to the block SOR method for solving the reordered linear system (3.9). Hence, from the classic SOR theory [28] it can be shown that the BPSOR iterates $\hat{u}^{(k)}$ converge to the exact solution $\hat{u}^*$ if and only if the relaxation parameter ω satisfies $0 < \omega < 2$, and both the spectral radius $\rho(\hat{M}_{SOR})$ and the optimal relaxation value $\hat{\omega}_b$ can be expressed as

$$\rho(\hat{M}_{SOR}) = \begin{cases} \frac{1}{4}\left[\omega\rho(\hat{M}_J) + \sqrt{\omega^2\rho(\hat{M}_J)^2 - 4(\omega-1)}\right]^2, & 0 < \omega \le \hat{\omega}_b, \\ \omega - 1, & \hat{\omega}_b \le \omega < 2, \end{cases} \tag{3.22}$$

and

$$\hat{\omega}_b = \frac{2}{1 + \sqrt{1 - \rho(\hat{M}_J)^2}}, \tag{3.23}$$

where $\hat{M}_J = \hat{L} + \hat{U}$ is the Jacobi iteration matrix of the reordered linear system (3.9), and $\hat{u}^* = Pu^*$ with u^* being the exact solution of the original linear system (2.6).

On the other hand, from the classic SOR theory [28] it follows that $\rho(M_{SOR})$ and ω_b have the same expressions as given in (3.20) and (3.21). That $\hat{M}_J = PM_JP^T$ follows easily from (3.16). Thus, $\rho(\hat{M}_J) = \rho(M_J)$. As a result, the expressions (3.22) and (3.23) become the same as the expressions (3.20) and (3.21), resulting in $\rho(\hat{M}_{SOR}) = \rho(M_{SOR})$ for $0 < \omega < 2$. Furthermore, from (3.15) it follows that $\rho(M_{BPSOR}) = \rho(\hat{M}_{SOR})$. Consequently,

$$\rho(M_{BPSOR}) = \rho(\hat{M}_{SOR}) = \rho(M_{SOR}).$$

Finally, the BPSOR iterate $u^{(k)} = P^T\hat{u}^{(k)} \to P^T\hat{u}^* = u^*$ as $k \to \infty$. This completes the proof. □

Two different iterative methods are said to have the same asymptotic rate of convergence if they have the same spectral radius [28]. Hence, (3.19) implies that the BPSOR method has the same asymptotic rate of convergence as the corresponding sequential block SOR method.

3.5. The point PSOR method as an inexact BPSOR scheme. To reduce the computing cost, the block equations of the BPSOR method can be solved approximately by an iterative method. Different iterative solvers with different approximation accuracies lead to different inexact BPSOR methods. In particular, the point PSOR method proposed in [26] can be regarded as an inexact BPSOR method in which each block equation is solved by only one point SOR iteration.

To show the above claim, let $\{\mathcal{U}_\mu^{i,(k)}\}$ be a sequence of the inexact BPSOR iterates in which each block equation is solved by only one point SOR iteration. Then, $\mathcal{U}_\mu^{i,(k+1)}$ is defined by

$$\mathcal{U}_\mu^{i,(k+1)} = \omega(D_\mu^i)^{-1}[f_\mu^i + L_\mu^i \mathcal{U}_\mu^{i,(k+1)} + U_\mu^i \mathcal{U}_\mu^{i,(k)}] + (1-\omega)\mathcal{U}_\mu^{i,(k)}, \tag{3.24}$$

where D_μ^i, L_μ^i, and U_μ^i denote the diagonal, strictly lower-triangular, and strictly upper-triangular matrices satisfying $A_{\mu\mu}^{ii} = D_\mu^i - L_\mu^i - U_\mu^i$, f_μ^i is a vector defined by

$$f_\mu^i = \mathcal{F}_\mu^i - \sum_{\nu=\mu}^{p}\sum_{j=1}^{i-1} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k+1)} - \sum_{\nu=1}^{\mu}\sum_{j=i+1}^{t} A_{\mu\nu}^{ij}\mathcal{U}_\nu^{j,(k)},$$

$\mu = 1, 2, \ldots, p$, and $i = 1, 2, \ldots, t$. In terms of $A_{\mu\nu}$, $\mathcal{U}_\mu^{(k)}$, L_μ, and U_μ, in (2.9), (2.14), and (2.12), the block vector $f_\mu = ((f_\mu^1)^T, (f_\mu^2)^T, \ldots, (f_\mu^t)^T)^T$ can be expressed as

$$f_\mu = \mathcal{F}_\mu - \sum_{\nu=\mu+1}^{p} A_{\mu\nu}\mathcal{U}_\nu^{(k+1)} - L_\mu\mathcal{U}_\mu^{(k+1)} - \sum_{\nu=1}^{\mu-1} A_{\mu\nu}\mathcal{U}_\nu^{(k)} - U_\mu\mathcal{U}_\mu^{(k)}. \tag{3.25}$$

By using (3.25) and the three block diagonal matrices, $\tilde{D}_\mu = \mathrm{diag}(D_\mu^1, D_\mu^2, \ldots, D_\mu^t)$, $\tilde{B}_\mu = \mathrm{diag}(L_\mu^1, L_\mu^2, \ldots, L_\mu^t)$, and $\tilde{C}_\mu = \mathrm{diag}(U_\mu^1, U_\mu^2, \ldots, U_\mu^t)$, the t expressions of $\{\mathcal{U}_\mu^{i,(k+1)}\}_{i=1}^t$ in (3.24) can be formulated as one vector form:

$$\begin{aligned}\mathcal{U}_\mu^{(k+1)} = \omega(D_\mu)^{-1}\Bigg[&\mathcal{F}_\mu + (\tilde{B}_\mu + L_\mu)\mathcal{U}_\mu^{(k+1)} + (\tilde{C}_\mu + U_\mu)\mathcal{U}_\mu^{i,(k)} \\ &- \sum_{\nu=\mu+1}^{p} A_{\mu\nu}\mathcal{U}_\nu^{(k+1)} - \sum_{\nu=1}^{\mu-1} A_{\mu\nu}\mathcal{U}_\nu^{(k)}\Bigg] + (1-\omega)\mathcal{U}_\mu^{(k)}.\end{aligned} \tag{3.26}$$

Further, define three block diagonal matrices as

$$\tilde{D} = \mathrm{diag}(\tilde{D}_1, \tilde{D}_2, \ldots, \tilde{D}_p),\ \tilde{B} = \mathrm{diag}(\tilde{L}_1, \tilde{L}_2, \ldots, \tilde{L}_p),\ \tilde{C} = \mathrm{diag}(\tilde{U}_1, \tilde{U}_2, \ldots, \tilde{U}_p),$$

where $\tilde{L}_\mu = \tilde{B}_\mu + L_\mu$ and $\tilde{U}_\mu = \tilde{C}_\mu + U_\mu$, which are the strictly lower- and upper-triangular matrices satisfying $A_{\mu\mu} = \tilde{D}_\mu - \tilde{L}_\mu - \tilde{U}_\mu$. By using them, the p expressions in (3.26) can be written as the matrix form

$$u^{(k+1)} = \omega\tilde{D}^{-1}[f + (\tilde{B} + M)u^{(k+1)} + (\tilde{C} + N)u^{(k)}] + (1-\omega)u^{(k)},$$

where the matrices M and N are given in (3.4). Solving the above equation for $u^{(k+1)}$ yields the matrix iterative expression of the inexact BPSOR method:

$$\begin{aligned}u^{(k+1)} = &[\tilde{D} - \omega(\tilde{B} + M)]^{-1}[(1-\omega)\tilde{D} + \omega(\tilde{C} + N)]u^{(k)} \\ &+ \omega[\tilde{D} - \omega(\tilde{B} + M)]^{-1}f.\end{aligned} \tag{3.27}$$

On the other hand, from [26] it is known that the point PSOR method has the same matrix expression as the one above. Hence, the inexact BPSOR method is exactly the point PSOR method defined in [26].

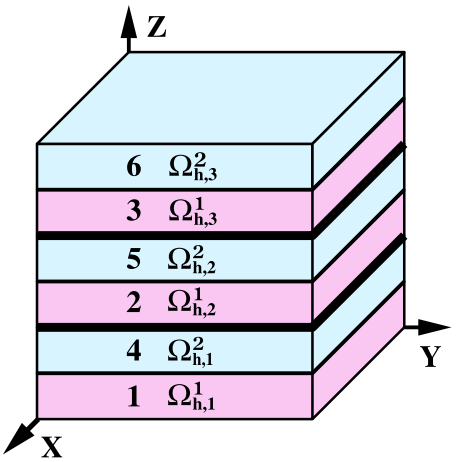


FIG. 4.1. *The 2-type partition based on a 3-strip partitioning of a cubic domain ($p = 3$ and $t = 2$). Each strip contains two types, $\Omega^1_{h,\mu}$ and $\Omega^2_{h,\mu}$ for $\mu = 1, 2, 3$, which are numbered in the 2-type ordering.*

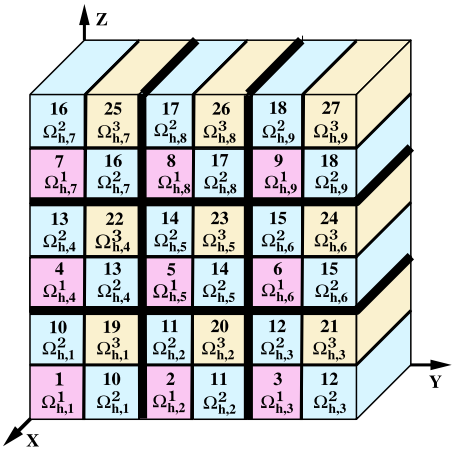


FIG. 4.2. *The 3-type partition based on a 2D 9-block partitioning of a cubic domain ($p = 9$ and $t = 3$). Each block contains three types, $\Omega^1_{h,\mu}$, $\Omega^2_{h,\mu}$, and $\Omega^3_{h,\mu}$ for $\mu = 1, 2, \ldots, 9$, which are numbered in the 3-type ordering.*

4. Three BPSOR methods based on strip and block partitionings.

4.1. The 2-type BPSOR method by strip partitioning. The 2-type BPSOR method is the BPSOR method by the 2-type partition based on a strip partition of the mesh domain Ω_h. For finite element and finite difference equations, the mesh-domain Ω_h is partitioned into p strips along one direction of the space such that each interior strip has only two coupled neighboring strips. Thus, each strip $\Omega_{h,\mu}$ contains only two types, $\Omega^1_{h,\mu}$ and $\Omega^2_{h,\mu}$. An illustration of the 2-type partition and ordering is given in Figure 4.1. Here a cubic domain is partitioned into three strips ($p = 3$), the two types are colored in two different colors, and each type is numbered in the 2-type ordering.

In the case of the 2-type partition ($t = 2$), the block matrix A in (2.8) becomes a tridiagonal block matrix with $A_{ij} = 0$ for $|i - j| \geq 2$. The nonzero off-diagonal submatrices of each row have the following forms:

$$A_{\mu,\mu+1} = \begin{bmatrix} 0 & 0 \\ A^{21}_{\mu,\mu+1} & 0 \end{bmatrix} \quad \text{and} \quad A_{\mu+1,\mu} = \begin{bmatrix} 0 & A^{12}_{\mu+1,\mu} \\ 0 & 0 \end{bmatrix}. \tag{4.1}$$

By setting $A^{12}_{1,0} = A^{21}_{p,p+1} = 0$, the BPSOR method by the strip partitioning can be expressed as

$$\begin{aligned} \mathcal{U}^{1,(k+1)}_\mu &= \omega(A^{11}_{\mu\mu})^{-1}[\mathcal{F}^1_\mu - A^{12}_{\mu,\mu-1}\mathcal{U}^{2,(k)}_{\mu-1} - A^{12}_{\mu,\mu}\mathcal{U}^{2,(k)}_\mu] \\ &\quad + (1-\omega)\mathcal{U}^{1,(k)}_\mu, \quad \mu = 1, 2, \ldots, p \quad \text{for Type 1,} \end{aligned} \tag{4.2}$$

$$\begin{aligned} \mathcal{U}^{2,(k+1)}_\mu &= \omega(A^{22}_{\mu\mu})^{-1}[\mathcal{F}^2_\mu - A^{21}_{\mu,\mu}\mathcal{U}^{1,(k+1)}_\mu - A^{21}_{\mu,\mu+1}\mathcal{U}^{1,(k+1)}_{\mu+1}] \\ &\quad + (1-\omega)\mathcal{U}^{2,(k)}_\mu, \quad \mu = 1, 2, \ldots, p \quad \text{for Type 2.} \end{aligned} \tag{4.3}$$

Obviously, the p iterative expressions of (4.2) are completely independent; so are the p iterative expressions of (4.3). Hence, the 2-type BPSOR method is a parallel iterative scheme by the strip partitioning.

In the 2-type ordering, the reordered matrix $\hat{A}$ of (3.10) becomes a 2×2 block matrix in the form

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} \\ \hat{A}_{21} & \hat{A}_{22} \end{bmatrix},$$

where $\hat{A}_{11} = \text{diag}(A_{11}^{11}, A_{22}^{11}, \ldots, A_{pp}^{11})$, $\hat{A}_{22} = \text{diag}(A_{11}^{22}, A_{22}^{22}, \ldots, A_{pp}^{22})$,

$$\hat{A}_{12} = \begin{bmatrix} A_{11}^{12} & & & \\ A_{21}^{12} & A_{22}^{12} & & \\ & \ddots & \ddots & \\ & & A_{p,p-1}^{12} & A_{pp}^{21} \end{bmatrix}, \text{ and } \hat{A}_{21} = \begin{bmatrix} A_{11}^{21} & A_{12}^{21} & & \\ & A_{22}^{21} & \ddots & \\ & & \ddots & A_{p-1,p}^{21} \\ & & & A_{pp}^{21} \end{bmatrix}.$$

Obviously, the above $\hat{A}$ is consistently ordered since it is a form of block red-black matrix [11]. Hence, the 2-type BPSOR method has the same convergence rate as the corresponding block SOR method.

4.2. The 3-type BPSOR method by 2D block partitioning. The 3-type BPSOR method is the BPSOR method by the 3-type partition based on the 2D block partitioning. For a linear system that arises from a 5-point-like or 7-point-like finite difference approximation to a 2D or 3D elliptic boundary problem, the mesh domain Ω_h is partitioned into p blocks $\Omega_{h,\mu}$ along two dimensions of the space such that each interior block is only connected to the four neighboring blocks from the left, right, top, and bottom sides. Each block is then divided into three types $\Omega_{h,\mu}^1, \Omega_{h,\mu}^2$, and $\Omega_{h,\mu}^3$ for $\mu = 1, 2, \ldots, p$. An illustration of the 3-type partition is given in Figure 4.2. Here a cubic domain is partitioned into nine blocks along the y-axis and z-axis directions, and the types labeled as $\Omega_{h,\mu}^i$ for $\mu = 1, 2, \ldots, 9$ and $i = 1, 2, 3$ are numbered in the 3-type ordering. Note that each second type $\Omega_{h,\mu}^2$ consists of two small blocks since the iterates defined on these two small blocks can be calculated simultaneously without affecting the definition of BPSOR.

In the case of the 3-type partition, the block matrix A given in (2.8) becomes a 5-diagonal block matrix such that the BPSOR method defined in (2.17) can be simplified as

$$\begin{aligned}
\mathcal{U}_\mu^{1,(k+1)} &= \omega(A_{\mu\mu}^{11})^{-1}[\mathcal{F}_\mu^1 - A_{\mu\mu}^{12}\mathcal{U}_\mu^{2,(k)} - A_{\mu,\mu-1}^{12}\mathcal{U}_{\mu-1}^{2,(k)} - A_{\mu,\mu-m}^{12}\mathcal{U}_{\mu-m}^{2,(k)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{1,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 1,} \\
\mathcal{U}_\mu^{2,(k+1)} &= \omega(A_{\mu\mu}^{22})^{-1}[\mathcal{F}_\mu^2 - A_{\mu\mu}^{21}\mathcal{U}_\mu^{1,(k+1)} - A_{\mu\mu}^{23}\mathcal{U}_\mu^{3,(k)} - A_{\mu,\mu-1}^{23}\mathcal{U}_{\mu-1}^{3,(k)} \\
&\quad - A_{\mu,\mu+1}^{21}\mathcal{U}_{\mu+1}^{1,(k+1)} - A_{\mu,\mu+m}^{21}\mathcal{U}_{\mu+m}^{1,(k+1)} - A_{\mu,\mu-m}^{23}\mathcal{U}_{\mu-m}^{3,(k)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{2,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 2,} \\
\mathcal{U}_\mu^{3,(k+1)} &= \omega(A_{\mu\mu}^{33})^{-1}[\mathcal{F}_\mu^3 - A_{\mu\mu}^{32}\mathcal{U}_\mu^{2,(k+1)} - A_{\mu,\mu+1}^{32}\mathcal{U}_{\mu+1}^{2,(k+1)} - A_{\mu,\mu+m}^{32}\mathcal{U}_{\mu+m}^{2,(k+1)}] \\
&\quad + (1-\omega)\mathcal{U}_\mu^{3,(k)}, \quad \mu = 1, 2, \ldots, p, \quad \text{for Type 3,}
\end{aligned} \tag{4.4}$$

where $A_{\mu j}^{12} = A_{\mu j}^{23} = 0$ for $j \leq 0$ and $A_{\mu k}^{21} = A_{\mu k}^{32} = 0$ for $k \geq p$. Obviously, the p expressions of each type in (4.4) are completely independent. Hence, the 3-type BPSOR method is a parallel scheme by the 2D block partitioning.

Under the 3-type ordering, the reordered matrix $\hat{A}$ of (3.10) becomes a 3×3 block matrix in the form

$$\hat{A} = \begin{bmatrix} \hat{A}_{11} & \hat{A}_{12} & 0 \\ \hat{A}_{21} & \hat{A}_{22} & \hat{A}_{23} \\ 0 & \hat{A}_{32} & \hat{A}_{33} \end{bmatrix}, \tag{4.5}$$

where $\hat{A}_{ii} = \mathrm{diag}(A^{ii}_{11}, A^{ii}_{22}, \ldots, A^{ii}_{pp})$ for $i = 1, 2, 3$, $\hat{A}_{12}$, $\hat{A}_{23}, \hat{A}_{21}$, and $\hat{A}_{32}$ have nonzero entries only on the main diagonal, the second off-diagonal, and the mth off-diagonal. That is, $\hat{A}_{12}$ and $\hat{A}_{23}$ have the forms

$$\hat{A}_{12} = \begin{bmatrix} A^{12}_{11} & & & & & \\ A^{12}_{21} & A^{12}_{22} & & & & \\ A^{12}_{m+1,1} & \ddots & \ddots & & & \\ & \ddots & A^{12}_{p-1,p-2} & A^{12}_{p-1,p-1} & & \\ & & A^{12}_{p,p-m} & A^{12}_{p,p-1} & A^{12}_{pp} & \end{bmatrix},$$

$$\hat{A}_{23} = \begin{bmatrix} A^{23}_{11} & & & & & \\ A^{23}_{21} & A^{23}_{22} & & & & \\ A^{23}_{m+1,1} & \ddots & \ddots & & & \\ & \ddots & A^{23}_{p-1,p-2} & A^{23}_{p-1,p-1} & & \\ & & A^{23}_{p,p-m} & A^{23}_{p,p-1} & A^{23}_{pp} & \end{bmatrix};$$

$\hat{A}_{21} = \hat{A}^T_{12}$ and $\hat{A}_{32} = \hat{A}^T_{23}$ if A is symmetric.

By the definition of the consistently ordered matrix as the one given in Definition 9-2.2 of [11], it can be shown that the reordered matrix $\hat{A}$ in (4.5) is consistently ordered. In fact, the index set $S = \{1, 2, \ldots, 3p\}$ since $\hat{A}$ for a $3p\times 3p$ block matrix of $\hat{A}$ has the partition $S = S_1 \cup S_2 \cup S_3$, where $S_1 = \{1, 2, \ldots, p\}$, $S_2 = \{p+1, p+2, \ldots, 2p\}$, and $S_3 = \{2p + 1, 2p + 2, \ldots, 3p\}$, which satisfies the definition of the consistently ordered matrix [11, 28]. That is, if the (i, j)th entry of $\hat{A}$ is nonzero and $i \in S_k$, then $j \in S_{k+1}$ for $j > i$ and $j \in S_{k-1}$ if $j < i$.

Since $\hat{A}$ is consistently ordered, from the classic SOR theory it follows that the 3-type BPSOR method has the same convergence rate as the corresponding block SOR method.

4.3. The 6-type BPSOR method by the 3D block partitioning. The 6-type BPSOR method is the BPSOR method by the 6-type partition based on the 3D block partitioning. For the linear system arising from a 7-point-like finite difference approximation to a 3D elliptic boundary problem, a 3D mesh domain is partitioned into p blocks along the three dimensions of the space such that each block is only connected to its six neighboring blocks that come from the left, right, bottom, top, back, and front sides. Each block is then divided into six types Ω^i_μ with $i = 1, 2, \ldots, 6$ and $\mu = 1, 2, \ldots, p$ as illustrated in Figure 4.3. An example of the 6-type partition is given in Figure 4.4. Here a cubic domain is partitioned into eight blocks along the x-, y-, and z-axis directions. To view the labels and ordering numbers of each type, this 6-type partition is cut into four portions along the x-axis direction so that the 48 types $\Omega^i_{h,\mu}$ with $\mu = 1, 2, \ldots, 8$ and $i = 1, 2, \ldots, 6$ of this 6-type partition can be displayed in the four plots of Figure 4.5.

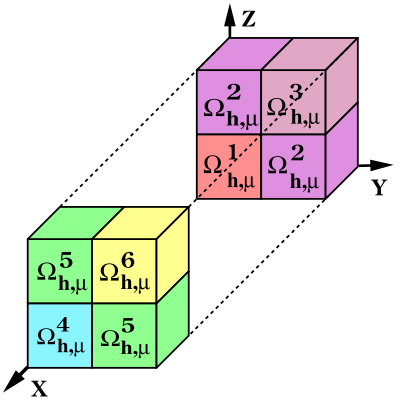


FIG. 4.3. *The six types of each block for the* 6*-type partition in Figure* 4.4*. The six types are labeled as* $\Omega^i_{h,\mu}$ *for* $i = 1, 2, \ldots, 6$*, and the front three types have been pushed away from the back three types to display the labels.*

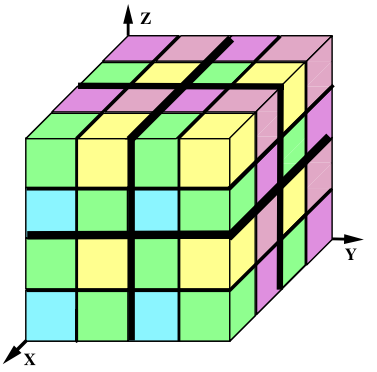


FIG. 4.4. *The* 6*-type partition based on a* $2 \times 2 \times 2$ *block partitioning of a cubic domain. The six types of each block are illustrated in Figure* 4.3*. See Figure* 4.5 *for the* 6*-type ordering.*

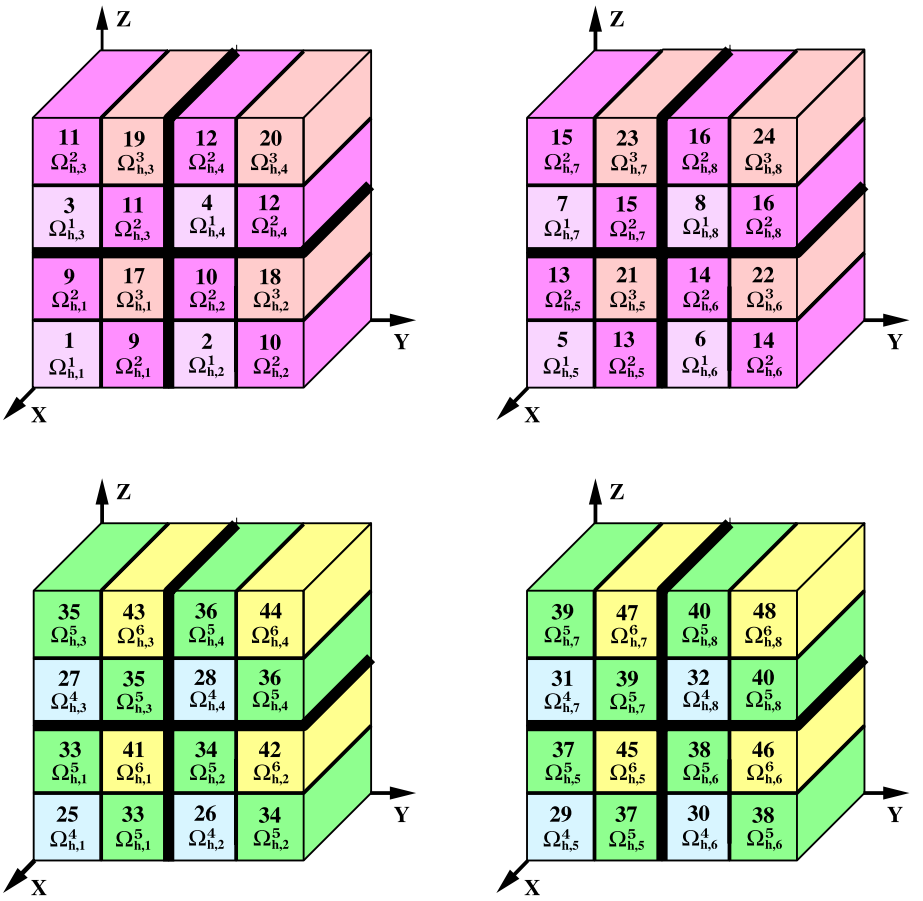


FIG. 4.5. *The* 6*-type ordering and the* 48 *types* $\Omega^i_{h,\mu}$ *with* $\mu = 1, 2, \ldots, 8$ *and* $i = 1, 2, \ldots, 6$ *of the* 6*-type partition in Figure* 4.4*. Here the* 6*-type partition consists of four type layers, which are drawn separately in the four plots to display the label and ordering number of each type.*

In the case of the 6-type partition, the block matrix A in (2.8) is reduced to a seven diagonal block matrix so that the iterative expression of the 6-type BPSOR method can be obtained directly from the general expression (2.17). In the 6-type ordering, the reordered matrix $\hat{A}$ in (3.10) is consistently ordered. In fact, since $\hat{A}$ is a $6p \times 6p$ block matrix, the index set $S = \{1, 2, \ldots, 6p\}$ is considered. It can be found that S has the following partition:

$$S = S_1 \cup S_2 \cup S_3 \cup S_4,$$

where $S_1 = \{1, 2, \ldots, p\}$, $S_2 = \{p+1, p+2, \ldots, 2p, 3p+1, 3p+2, \ldots, 4p\}$, $S_3 = \{2p+1, 2p+2, \ldots, 3p, 4p+1, 4p+2, \ldots, 5p\}$, and $S_4 = \{5p+1, 5p+2, \ldots, 6p\}$. It then can be shown that the partition of S satisfies the definition of a consistently ordered matrix. Since $\hat{A}$ is a consistently ordered matrix, the classic SOR theory follows that the 6-type BPSOR method has the same convergence rate as the corresponding sequential block SOR method.

5. Numerical results. Numerical experiments were made on the SGI Origin 2000 at the University of Wisconsin-Milwaukee, which contains 16 R12000 processors. A linear system for numerical tests comes from a 7-point finite difference approximation to a 3D Poisson boundary value problem on the unit cubic domain with a boundary value of zero. The mesh domain Ω_h consists of the interior mesh points (x_i, y_j, z_k) with $x_i = (i-1)h$, $y_j = (j-1)h$, and $z_k = (k-1)h$ for $i, j, k = 2, 3, \ldots, n$. Here $h = 1/n$. By setting $u_{i,j,k}$ as the numerical approximation to the Poisson solution at mesh point (x_i, y_j, z_k), the linear system is defined as

$$\begin{aligned} 6u_{i,j,k} - u_{i+1,j,k} - u_{i-1,j,k} - u_{i,j+1,k} - u_{i,j-1,k} \\ - u_{i,j,k-1} - u_{i,j,k+1} = h^2 f_{i,j,k}, \end{aligned} \tag{5.1}$$

where $i, j, k = 2, 3, \ldots, n$, $f_{i,j,k}$ denotes the value of the right-hand side function at mesh point (x_i, y_j, z_k), and the boundary values are $u_{i1} = u_{i,n+1} = u_{1i} = u_{n+1,i} = 0$ for $1 \le i \le n+1$. In numerical tests, $f_{i,j,k}$ was set as one.

The parallel program was written in FORTRAN 77 and MPI (the Message Passing Interface) [3]. The program was compiled by using the optimization level $O2$. The CPU time was measured by the MPI function MPI_Wtime, which returns the wall time in seconds. The sequential SOR program in FORTRAN 77 was implemented on one processor of the computer. Its CPU time was measured by the F77 function $etime()$. The iteration termination rule is given by

$$\|f - Au^{(k)}\| < 10^{-6}, \tag{5.2}$$

where $u^{(k)}$ denotes the kth iterate for solving the linear system $Au = f$.

5.1. Solution of each block linear equation. Each BPSOR iteration requires us to solve tp block equations, $A^{ii}_{\mu\mu} v^i_\mu = f^i_\mu$ for $i = 1, 2, \ldots, t$ and $\mu = 1, 2, \ldots, p$. Since each block equation is located within one processor, it can be solved accurately by a sequential direct method or approximately by a sequential iterative method. In this numerical test, the sequential point SOR method, and the sparse LDL^T method from the SGI's Scientific Computing Software Library were used to solve block equations. Here a point SOR iterate $v^{i,(k)}_\mu$ was accepted as an approximate solution of block equation $A^{ii}_{\mu\mu} v^i_\mu = f^i_\mu$ when it met the termination rule: $\|f^i_\mu - A^{ii}_{\mu\mu} v^{i,(k)}_\mu\| \le 10^{-8}$ for all $i = 1, 2, \ldots, t$, $\mu = 1, 2, \ldots, p$. For simplicity, the point SOR method

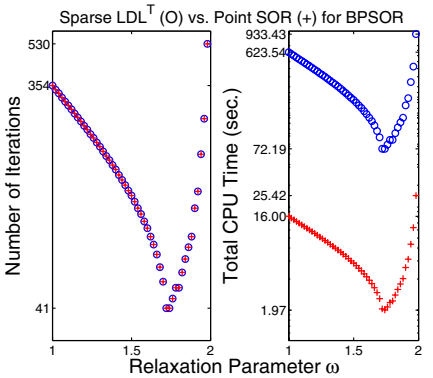


FIG. 5.1. *Comparisons of the convergence and performance of the BPSOR method using the LDL^T direct solver for each block equation with that using the sequential point SOR iterative solver for each block equation. The tests were made on eight processors for the model problem* (5.1) *with* $h = 1/65$.

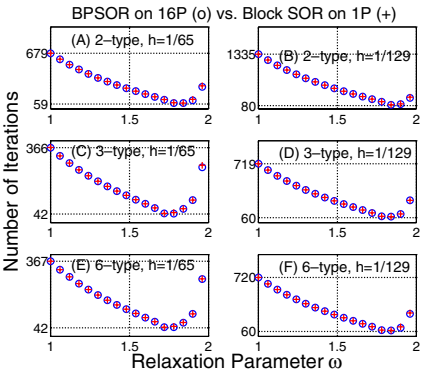


FIG. 5.2. *Comparisons of the convergence rates of the three particular BPSOR methods defined by the* 2*-type,* 3*-type, and* 6*-type partitions on 16 processors with that of the three corresponding sequential block SOR methods on one processor for solving the model problem* (5.1) *with* $h = 1/65$ *and* $h = 1/129$.

used $\omega = 1.54$ (which may not be optimal) for solving all the block equations. The BPSOR method was carried out on eight processors based on an 8-strip partition along the z-axis direction. The numerical results were plotted in Figure 5.1. They show that with the point SOR method, the BPSOR method can have almost the same number of iterations determined by (5.2) as that with the sparse LDL^T direct solver, while can take much less CPU time (about 40 times less in this test).

5.2. Convergence comparisons. Figure 5.2 compares the total numbers of iterations of the three particular BPSOR methods defined by the 2-type, 3-type, and 6-type partitions with that of the three corresponding sequential block SOR methods for solving the model problem (5.1) with $h = 1/65$ and $h = 1/129$, respectively. The BPSOR methods were implemented on the sixteen processors of the SGI Origin 2000. The total number of iterations was determined by the termination rule (5.2). The 2-type, 3-type, and 6-type partitions were constructed based on the strip partitioning of order $1 \times 1 \times 16$, the 2D-block partitioning of $1 \times 4 \times 4$, and the 3D-block partitioning of $2 \times 2 \times 4$, respectively. The size of each block equation was selected to be almost the same. Each block equation was solved approximately by the sequential point SOR method with $\omega = 1.54$ for the 2-type case and $\omega = 1.7$ for the 3-type and 6-type cases. From Figure 5.2 it can be seen that all the three BPSOR methods have almost the same numbers of iterations as the three corresponding sequential block SOR methods. This confirms the conclusion of Theorem 3.4; i.e., the BPSOR method has the same asymptotic rate of convergence as the corresponding sequential block SOR method.

5.3. Parallel performances of BPSOR. Figure 5.3 displays the CPU time data of the above BPSOR and block SOR methods and the speedups of the three BPSOR methods as functions of relaxation parameter ω between 1 and 1.98. The interprocessor data communication time spent by BPSOR were also reported in the figure as dashed lines. Here for each value the speedup was calculated as the ratio of the total CPU time of the sequential block SOR method on one processor to that of the BPSOR method on sixteen processors. From Figure 5.3 it can be seen that all the

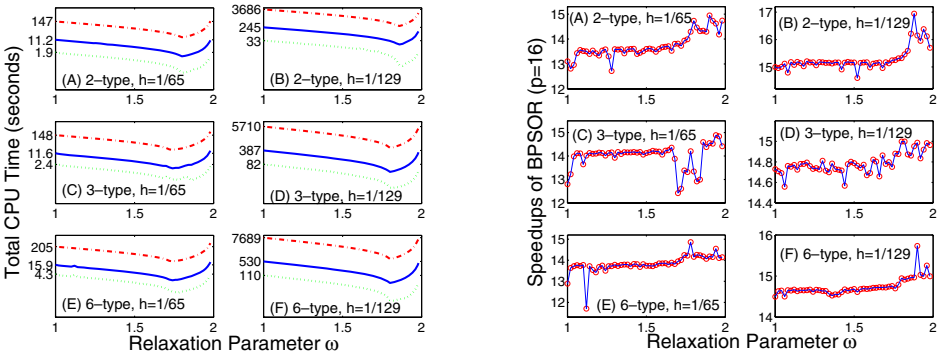


FIG. 5.3. *Performance comparisons between the three particular BPSOR methods* (the solid lines) *defined by the* 2*-type,* 3*-type, and* 6*-type partitions on* 16 *processors and the corresponding three sequential block SOR methods* (the dash-dotted lines) *on one processor for solving the model problem* (5.1) *with* $h = 1/65$ *and* $h = 1/129$. The dotted lines *indicate the interprocessor communication times. The speedups of the BPSOR methods are also presented.*

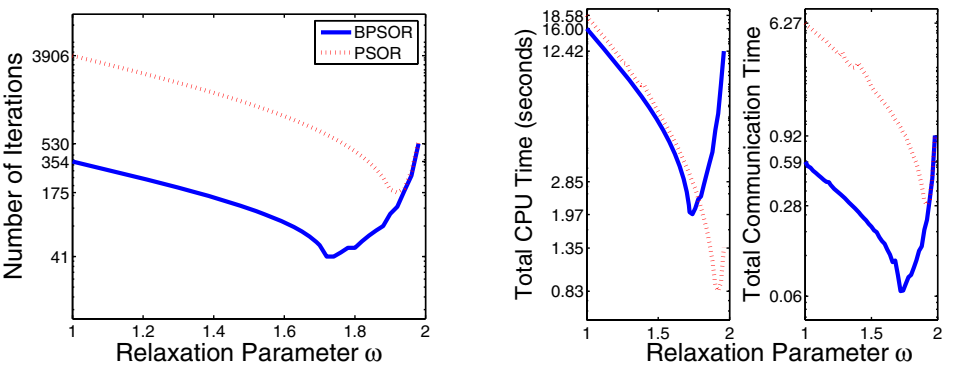


FIG. 5.4. *Performance comparisons of BPSOR with PSOR on eight processors of the SGI Origin 2000 based on an* 8*-strips partition of the cubic mesh domain with* $h = 1/65$.

three BPSOR methods have the speedups between 12 to 16 on the sixteen processors of the SGI Origin 2000, which indicate that the BPSOR methods have good parallel performances.

5.4. Parallel performance comparisons with PSOR. The parallel performance of the BPSOR method was compared with that of the point PSOR method in Figure 5.4. Both BPSOR and PSOR were defined on the same 8-strip partition along the z-axis, and implemented on eight processors of the SGI Origin 2000. The first plot of Figure 5.4 shows that BPSOR took about 10 to 16 times fewer iterations than PSOR from $\omega = 1$ to $\omega = 1.76$ to satisfy the termination rule (5.2), confirming that BPSOR has a much faster rate of convergence than the point PSOR method. As a result, as displayed in the third plot of Figure 5.4, the BPSOR method took about 10 to 16 times less CPU time in interprocessor data communications than the point PSOR method. The second plot of Figure 5.4 also shows that BPSOR took up to 53 percent less total CPU time than PSOR for the values of ω from 1 to 1.76. Furthermore, the first plot shows that the convergence rate of BPSOR is less sensitive

to the value selection of ω than that of the point PSOR method. Obviously, the total performance of BPSOR can be further improved significantly when a more efficient sequential iterative method (such as the multigrid method [22]) is used to solve each block equation.

Acknowledgments. The author thanks Professor Loyce Adams and the anonymous referees for their valuable comments.

REFERENCES

[1] L. M. Adams and H. F. Jordan, *Is SOR color-blind?*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 490–506.
[2] L. M. Adams and J. M. Ortega, *A multi-color SOR method for parallel computation*, Proceedings of the 1982 International Conference on Parallel Processing, Bellaire, MI, 1982, pp. 53–58.
[3] *MPI: A Message-Passing Interface Standard*, University of Tennessee, Knoxville, Tennessee, June 1995.
[4] C. Ashcraft and R. Grimes, *On vectorizing incomplete factorization and SSOR preconditioners*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 122–151.
[5] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus, *CHARMM: A program for macromolecular energy, minimization, and dynamics calculations*, J. Comp. Chem., 4 (1983), pp. 187–217.
[6] C. J. Cramer and D. G. Truhlar, *Implicit solvation models: Equilibria, structure, spectra, and dynamics*, Chem. Rev., 99 (1999), pp. 2161–2200.
[7] M. E. Davis, J. D. Madura, B. A. Luty, and J. A. McCammon, *Electrostatics and diffusion of molecules in solution: Simulations with the University of Houston, Brownian dynamics program*, Comput. Phys. Comm., 62 (1991), pp. 187–197.
[8] I. S. Duff and G. A. Meurant, *The effect of orderings on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–647.
[9] C. Farhat, *Multiprocessors in Computational Mechanics*, Ph.D. thesis, Civil Engineering Department, University of California at Berkeley, Berkeley, CA, 1986.
[10] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations*, Springer-Verlag, New York, 1994.
[11] L. A. Hageman and D. M. Young, *Applied Iterative Methods*, Academic Press, San Diego, 1981.
[12] D. Harrar and J. M. Ortega, *Multicoloring with lots of colors*, in Proceedings of the Third International Conference on Supercomputing, Crete, Greece, 1989, pp. 1–6.
[13] B. Honig and A. Nicholls, *Classical electrostatics in biology and chemistry*, Science, 268 (1995), pp. 1144–1149.
[14] T. Lippert, *Parallel SSOR preconditioner for lattice QCD*, Parallel Comput., 25 (1999), pp. 1357–1370.
[15] R. Melhem, *Towards efficient implementations of preconditioned conjugate methods on vector supercomputers*, Internat. J. Supercomput. Appl., 1 (1987), pp. 70–98.
[16] J. M. Ortega, *Orderings for conjugate gradient preconditionings*, SIAM J. Optim., 1 (1991), pp. 565–582.
[17] E. L. Poole and J. M. Ortega, *Multicolor ICCG methods for vector computers*, SIAM J. Numer. Anal., 24 (1987), pp. 1394–1418.
[18] S. A. Stotland and J. M. Ortega, *Orderings for parallel conjugate gradient preconditioners*, SIAM J. Sci. Comput., 18 (1997), pp. 854–868.
[19] W. Rocchia, E. Alexov, and B. Honig, *Extending the applicability of the nonlinear Poisson-Boltzmann equation: Multiple dielectric constants and multivalent ions*, J. Phys. Chem. B, 105 (2001), pp. 6507–6514.
[20] B. Roux and T. Simonson, *Implicit solvent models*, Biophys. Chem., 78 (1999), pp. 1–20.
[21] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[22] K. Stüen and U. Trottenberg, *Multigrid methods: Fundamental algorithms, model problem analysis and applications*, in Multigrid Methods, Lecture Notes in Math. 960, Springer-Verlag, Berlin, New York, 1982, pp. 1–176.
[23] E. F. van de Velde, *Concurrent Scientific Computing*, Springer-Verlag, New York, 1994.

[24] H. A. van der Vorst *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.
[25] J. Wang, *Convergence analysis without regularity assumptions for multigrid algorithms based on SOR smoothing*, SIAM J. Numer. Anal., 29 (1992), pp. 987–1001.
[26] D. Xie and L. Adams, *New parallel SOR method by domain partitioning*, SIAM J. Sci. Comput., 20 (1999), pp. 2261–2281.
[27] I. Yavneh, *On red-black SOR smoothing in multigrid*, SIAM J. Sci. Comput., 17 (1996), pp. 180–192.
[28] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1534–1554
© 2006 Society for Industrial and Applied Mathematics

PARALLEL ALGEBRAIC MULTIGRIDS FOR STRUCTURAL MECHANICS*

MARIAN BREZINA†, CHARLES TONG‡, AND RICHARD BECKER§

Abstract. This paper presents the results of a comparison of three parallel algebraic multigrid (AMG) preconditioners for structural mechanics applications. In particular, we are interested in investigating both the scalability and robustness of the preconditioners. Numerical results are given for a range of structural mechanics problems with various degrees of difficulty, and they show that algebraic multigrid methods can be applied successfully even to difficult problems in structural mechanics.

Key words. algebraic multigrid, preconditioning, parallel computing

AMS subject classifications. 65F10, 65N20

DOI. 10.1137/040608271

1. Introduction.

Krylov methods with algebraic multigrid preconditioners have been demonstrated to offer fast convergence rates as well as good parallel efficiency for the iterative solution of linear systems arising from discretizations of many elliptic partial differential equations. Two classes of algebraic multigrid (AMG) methods have gained popularity: the classical AMG method, originally conceived in [5] and developed practically by Ruge and Stüben (RS) [22], and smoothed aggregation (SA) multigrid proposed by Vaněk, Mandel, and Brezina [28, 26]. In addition, improvements to SA have been made to handle more difficult problems such as Maxwell's equations [4]. In this paper we investigate another improvement to SA to handle difficult problems in structural mechanics. This improvement involves locally enriching the prolongation operator using low energy eigenvectors of subdomains in a finite element framework. The idea of enriching the coarse space was already recognized in [28] in conjunction with treatment of high-order equations and was later expanded into an adaptive procedure similar to the one presented here by Brezina, Heberton, and Vaněk in [7] as a general principle to eliminate convergence problems due to local irregularities of the finite element discretization. The latter method was, however, limited to two levels. Our implementation generalizes this coarse-space enriching approach to multilevel, replacing the subdomain smoothers with more efficient multilevel

*Received by the editors May 13, 2004; accepted for publication (in revised form) December 17, 2004; published electronically February 3, 2006. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48. This work was performed by an employee of the U.S. Government or under U.S. Government contract. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/27-5/60827.html

†Department of Applied Mathematics, Campus Box 526, University of Colorado at Boulder, Boulder, CO 80309–0526, and Front Range Scientific Computations, Inc. (mbrezina@math.cudenver.edu). This author's work was sponsored by the Department of Energy under grants DE-FC02-01ER25479 and DE-FG02-03ER25574; Lawrence Livermore National Laboratory under contract B533502; Sandia National Laboratories under contract 15268; and the National Science Foundation under VIGRE grant DMS-9810751.

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551 (chtong@llnl.gov).

§New Technology and Engineering Division, Lawrence Livermore National Laboratory, L-129, Box 808, Livermore, CA 94551 (becker13@llnl.gov).

procedure that allows for further coarsening. We call this method GSA, an acronym for generalized smoothed aggregation.

Parallelization issues for AMG preconditioners are also considered in this paper. The application of AMG involves two phases: the setup phase and the solution phase. A typical setup phase consists of coarse grid selection, construction of prolongation operators, and creation of coarse grid matrices. The original coarse grid selection procedures in both RS and SA are sequential in nature. Efforts to parallelize the coarsening step have been reported in [24, 16, 18, 15]. The construction of coarse grid matrices using triple matrix products can be parallelized in a straightforward manner, but perfect scalability is difficult to achieve. Scalability issues in the solution phase are associated mainly with the parallel smoothers, and with solution of the coarse grid problem. In this paper we discuss some of these parallelization issues in detail and make recommendations giving different options.

This paper is organized as follows. In section 2, we give an introduction of the three AMG methods in question. Section 3 is dedicated to expounding the GSA methods and providing theoretical motivation for the method. Section 4 describes in more detail the parallel aspects of the setup and solution phases. Numerical experiments will be presented in section 5, followed by a few afterthoughts in section 6.

2. A brief review of AMG. AMG methods can be used as independent linear solvers for solving linear systems of the form

$$Ax = b, \tag{2.1}$$

where $A \in \Re^{n\times n}$ is a positive definite matrix and $x, b \in \Re^n$. In practice, they are often used as preconditioners in preconditioned conjugate gradient and/or generalized minimal residual methods. A typical V-cycle AMG is given as follows (by calling $\mathrm{MG}(A_0, b, u, 0)$).

ALGORITHM $\mathrm{MG}(A_k, b_k, u_k, k)$.

1. If level k is the coarsest level, $u_k = A_k^{-1} b_k$, return
2. $u_k = S_k(A_k, b_k, u_k)$ /* presmoothing */
3. $r_k = b_k - A_k u_k$ /* residual calculation */
4. $b_{k+1} = R_k * r_k$ /* restriction */
5. $v_{k+1} = 0$
6. $\mathrm{MG}(A_{k+1}, b_{k+1}, v_{k+1}, k+1)$
7. $u_k = u_k + P_k v_{k+1}$ /* coarse grid correction */
8. $u_k = S_k(A_k, b_k, u_k)$ /* post-smoothing. */

Here S_k, R_k, and P_k are the smoothing, restriction, and prolongation operators, respectively.

Application of AMG methods consists of distinct setup and iteration phases. In the setup phase, the transfer operators P_k, R_k are created based on the operator A_k, and the multigrid smoothing operators S_k are set up. Once P_k becomes available, the restriction operator, R_k, is typically defined as $R_k = P_k^T$, and the coarse problem operator is constructed using the Galerkin formula, $A_{k+1} = R_k A_k P_k$. The hybrid Gauss–Seidel smoother [29] is chosen for our numerical study, and its performance aspects will be discussed later.

The main idea of multigrid is based on the complementary roles of the multigrid smoothing and coarse grid correction procedures. In order to have a well-convergent multigrid, the solution error not eliminated well by smoothing must be eliminated well by coarse grid correction. As the prolongation operators define the coarse spaces, they play a critical role in transmitting and eliminating the error modes between

different grid levels. In the following subsections we briefly describe two different ways of selecting the coarse variables and the corresponding prolongation operators.

2.1. The classical AMG. Setting up the coarse grid structure in the classical algebraic multigrid [22] (RS) is guided by two principles.

P1: Errors not efficiently reduced by smoothing must be approximated well by the range of interpolation, and

P2: the coarse grid correction must efficiently eliminate fine grid error in the range of interpolation.

For problems with the M-matrix property P1 is satisfied by a two-pass coarse grid selection and the RS interpolation formula [22], while construction of coarse problems using the Galerkin formula allows P2 to be satisfied.

A typical setup phase for RS is as follows [11].

RS Setup Phase.

1. Set k = 0.
2. Partition the level k set of points Ω^k into disjoint sets C^k and F^k.
   (a) Set $\Omega^{k+1} = C^k$.
   (b) Define interpolation P_k.
3. Set $R_k = P_k^T$ and $A_{k+1} = R_k A_k P_k$.
4. If Ω_{k+1} is small enough, stop. Otherwise, set $k = k + 1$ and go to step 2.

Step 2 begins by forming a set of dependencies for each point i (a point can be a single unknown in the scalar case or a set of unknowns in the system case) in the current grid level defined by

$$S_i = \{j \neq i : -a_{ij} \leq \alpha \max_{k \neq i}(-a_{ik})\}, \tag{2.2}$$

where α is the strength threshold for pruning weak couplings between variables. All points in the domain are chosen either as coarse or fine points. A set of interpolation points C_i is then selected for every fine point i. Since errors are more effectively smoothed in the direction of influence, a suitable criterion for selecting C_i is $C_i = S_i \cap C$ (C is the set of all C points.)

Two criteria for choosing the F and C sets are as follows.

P3: For each $i \in F$, each $j \in S_i$ is either in C or $S_j \cap C_i \neq \emptyset$, and

P4: C is a maximal set with the property that if i and j are both coarse points, then $j \notin S_i$ and $i \notin S_j$.

Criterion P3 says that if i is a fine point, then the points influencing i must either be coarse points or must share a common coarse point with i. P4 is enforced to keep the coarse grid as small as possible. Since it may not be possible to satisfy both criteria simultaneously, a common practice is to enforce P3, guided by P4. This commonly takes the form of a two-pass coarse selection, which proceeds by first choosing a maximal independent set of coarse points, followed by patching up the C set to satisfy P3.

Once the coarse set C has been determined, the interpolation formula is prescribed as

$$(P_k u_{k+1})_i = \begin{cases} (u_{k+1})_l & \text{if } i \in C \text{ and } i \text{ and } l \text{ are the same point,} \\ \sum_{j \in C_i} \omega_{ij} (u_{k+1})_j & \text{if } i \in F. \end{cases}$$

Details of computing the quantities ω_{ij} can be found in [22].

Note that the RS method was originally developed for problems with M-matrix property, so the problems considered in this paper are outside its theoretical scope. However, the method has been successfully applied to a variety of problems violating the M-matrix property, and we demonstrate that it can, with some success, be applied to the problems under our consideration.

2.2. The SA multigrid. The coarse grid construction for SA differs from that of RS in that, instead of choosing the coarse set as a subset of the current grid nodes (a node can be a single unknown in the scalar case or a set of unknowns in the system case), each fine node is assigned to a unique aggregate. Each aggregate thus corresponds to one or more equations in the coarse grid operator. The first coarsening step is thus to form a collection of mutually disjoint aggregates covering all the fine nodes (however, the equations corresponding to the essential boundary conditions are usually left out). The goal is to have uniformly shaped aggregates. The optimal aggregate size for the standard method has a diameter three, i.e., three nodes along the diameter (small aggregates may result in higher overall complexity, but very large aggregates may result in poor convergence rates unless a nonstandard prolongator smoother is applied; cf. [27]). A basic aggregation procedure proceeds as follows [28, 24].

SA STEP (GIVEN A GRAPH REPRESENTING THE NODE CONNECTIVITY).
Phase 1: Form initial set of aggregates: for $i = 1$ to n,
  a. If node i has been aggregated or it is adjacent to an aggregated node, go to the next node.
  b. Otherwise, select it as a root node, and define a new aggregate as node i plus all its strongly coupled neighbors.
Phase 2: Put unaggregated nodes into existing aggregates: for $i = 1$ to n,
  a. If node i has been aggregated, go to the next node.
  b. Otherwise, put it in the aggregate to which one of its neighbors belongs.

Actual implementation can add complexity to this aggregation algorithm. For example, more phases may be needed if aggregate size control is incorporated. Also, different criteria for choosing host aggregates in phase 2 may be prescribed to enhance performance.

The aggregation step is followed by forming the tentative prolongation operator $\tilde{P}_k$, the formula of which, in the simplest case of piecewise constant interpolation, is given as

$$(\tilde{P}_k u_{k+1})_i = \begin{cases} (u_{k+1})_j & \text{if } i \in \text{aggregate}_j, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

For construction of the tentative prolongation in more general cases we refer to [28]. For solving systems of partial differential equations, aggregation is typically performed on the nodes in the computational grid. Three-dimensional elasticity problems, for example, have three unknowns per node corresponding to displacements in the three orthogonal directions. Thus, in the case of structured hexahedral mesh, each aggregate would ideally contain 27 nodes or 81 unknowns. In addition, instead of using piecewise constant interpolation for each aggregate, rigid body modes or near null space vectors can be provided for defining $\tilde{P}_k$. The number of coarse grid equations for each aggregate is then the number of null space vectors provided. For three-dimensional elasticity, six rigid body modes can be computed from the nodal coordinates which are readily available from most applications.

More robust prolongators can be constructed by smoothing the tentative prolongators. For example, applying a damped Jacobi smoother gives

$$P_k = (I - \omega D_k^{-1} A_k)\tilde{P}_k, \tag{2.4}$$

where D_k is the diagonal of A_k, and ω is the damping factor related to the maximum eigenvalue of the scaled matrix $D_k^{-1}A_k$. Convergence theory [26] proves that this prolongation smoother is needed to guarantee convergence rates only mildly dependent on the grid size.

3. Generalized smoothed aggregation. For many applications, the knowledge of a small number of near-kernel components suffices to construct a robust smoothed aggregation solver. However, for difficult problems, it may be beneficial or indeed necessary to generalize the method.

To motivate the GSA method, we first recall the theoretical convergence estimate derived in [27]. Throughout this section we drop the level indices in the notation for the multigrid operators when there is no danger of ambiguity. Let n and m denote the dimensions of the Euclidean spaces associated with the fine and coarse spaces, respectively. We further define $A_S = S^2 A$, where S denotes the prolongator smoother. Then for the two-level smoothed aggregation method employing a presmoother with a linear part given by S and a post-smoother with linear part given by

$$S' = I - \frac{\hat{\omega}}{\varrho(A_S)} A_S, \tag{3.1}$$

the following result holds (cf. [27]).

THEOREM 3.1. C_{apx}

1. Q $\Re^n$ $\text{Range}(\hat{P})$

$$\|(I-Q)u\|^2 \le \frac{C_{\text{apx}}^2}{\bar{\varrho}(A_S)} \|u\|_A^2 \quad \forall u \in \Re^n; \tag{3.2}$$

2. S A $\varrho(S) \le 1$

e_i i $P = S\hat{P}$ $e_i^S = Se_i$

$$\|e_i^S\|_A^2 \le (1-C)^i \|e_0\|_A^2,$$

$$C = \frac{C_{\text{apx}}^{-2}\hat{\omega}(2-\hat{\omega})}{1 + C_{\text{apx}}^{-2}\hat{\omega}(2-\hat{\omega})}, \quad \hat{\omega} \quad (3.1).$$

As assumption 2 of Theorem 3.1 is usually easy to satisfy, our attention is focused on minimizing the constant C_{apx} in (3.2). We note that the left-hand side of (3.2) depends only on the tentative prolongator, while the prolongator smoother is present only in the form of a bound on the spectral radius of A_S in the denominator of the right-hand side.

The smoothed aggregation method thus possesses two mechanisms for ensuring good convergence properties. One strategy leading to improved constant C_{apx} relies on developing more powerful prolongator smoothers S, thus forcing $\varrho(S^2A) \ll \varrho(A)$. This

approach had been investigated in [27], where a two-level method with convergence properties independent of the size of the coarse grid was developed.

Another way to improve C_{apx} is based on appropriately enriching the range of the tentative prolongator $\hat{P}$, thus minimizing the left-hand side in (3.2). This approach can, naturally, be combined with the improvement of S for a compound benefit. Note, however, that in forming the prolongator $P = S\hat{P}$ to be used in the final method, all columns of $\hat{P}$ must be multiplied by S. Hence, the effect of using an improved S is global in nature. The corresponding smoothing of the coarse space basis functions is thus appropriate for correcting global convergence phenomena tied to the differential equation solved, and the use of nonstandard prolongator smoothers may thus be an overkill if slow convergence of the method is rooted in local phenomena. In contrast, due to the disjoint nature of the aggregate decomposition, the enrichment of the range of $\hat{P}$ can be carried out locally one aggregate at a time.

As the emphasis of this paper is on parallel implementation, we are seeking a procedure which would modify the iterative solver locally in the problematic regions. We will thus focus on local enrichment of the range of the tentative prolongator $\hat{P}$ and refer the interested reader to [27] for details on construction of S.

We will consider a decomposition of our computational domain, Ω, into J nonoverlapping subdomains, Ω_i,

$$\Omega = \bigcup_{i=1}^{J} \bar{\Omega}_i, \quad \Omega_i \cap \Omega_j = \emptyset \text{ if } i \neq j. \tag{3.3}$$

We assume that each $\bar{\Omega}_i$ is a connected cluster of elements. Denote by n_i the number of degrees of freedom in $\bar{\Omega}_i$, by A_i the $n_i \times n_i$ local stiffness matrix corresponding to $\bar{\Omega}_i$, and by N_i the zero-one matrix mapping the degrees of freedom associated with $\bar{\Omega}_i$ into the set of global degrees of freedom in matrix A. Obviously, the stiffness matrix A can be assembled from the local stiffness matrices A_i by

$$A = \sum_{i=1}^{J} N_i A_i N_i^T. \tag{3.4}$$

We will utilize a system of large nodal aggregates $\{\mathcal{A}_i\}_{i=1}^{J}$ forming a disjoint covering of the set of all nodes such that all nodes in $\mathcal{A}_i$ lie in $\bar{\Omega}_i$. Such aggregates are easily formed by assigning all the nodes in $\bar{\Omega}_i$ to $\mathcal{A}_i$ and distributing any nodes shared by more than one $\bar{\Omega}_i$ to exactly one of the corresponding aggregates at will.

To formalize this distribution of nodes, for each subdomain $\bar{\Omega}_i$ with number of degrees of freedom n_i, we define a zero-one diagonal matrix I_i of dimension n_i as

$$(I_i)_{dd} = \begin{cases} 1 & \text{if degree of freedom } d \text{ corresponds to a node in } \mathcal{A}_i, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5}$$

We further define for each subdomain $\bar{\Omega}_i$ a set of vectors $\{w_j^{(i)}\}_{j=1}^{m_i}$, $w_j^{(i)} \in \text{Range}(I_i)$, and denote by W_i the $n_i \times m_i$ matrix consisting of orthogonal columns $\{w_j^{(i)}\}_{j=1}^{m_i}$. The tentative prolongator $\hat{P} : \Re^m \mapsto \Re^n$, which is able to represent exactly all functions in $\{w_j^{(i)}\}_{j=1}^{m_i}$ over $\mathcal{A}_i$, can then be defined by

$$\hat{P}x = \sum_{i=1}^{J} N_i W_i x^{(i)}, \tag{3.6}$$

where $x^{(i)}$ denotes the segment of coarse degrees of freedom associated with aggregate $\mathcal{A}_i$.

Theorem 3.1 guarantees that the rate of convergence of the two-level method is bounded by $1 - \frac{C_{\text{apx}}^{-2}}{1+C_{\text{apx}}^{-2}}$ under the assumption that there exists $v \in \Re^m$ such that

$$\|u - \hat{P}v\|_{l^2(\bar{\Omega})}^2 \leq \frac{C_{\text{apx}}^2}{\varrho(S^2A)}\|u\|_A^2 \quad \forall u \in \Re^n. \tag{3.7}$$

Using the sparsity structure of $\hat{P}$ stemming from disjointness of the aggregates, we can write, for any $u \in \Re^n$,

$$\|u - \hat{P}v\|_{l^2(\bar{\Omega})}^2 = \sum_{i=1}^{J} \|u - \hat{P}_{*,i}v_i\|_{l^2(\mathcal{A}_i)}^2,$$

where $\|\cdot\|_{l^2(\mathcal{A}_i)}$ denotes the restriction of the Euclidean norm to the set of degrees of freedom corresponding to the node in the aggregate $\mathcal{A}_i$, and $\hat{P}_{*,i}$ denotes the supercolumn of $\hat{P}$ corresponding to aggregate $\mathcal{A}_i$. Similarly, using (3.4), we write

$$\|u\|_A^2 = \sum_{i=1}^{J} \langle A_i N_i^T u, N_i^T u\rangle.$$

Thus, in order to satisfy (3.7) it suffices to satisfy over all aggregates $\mathcal{A}_i$ the inequality

$$\|u - \hat{P}v\|_{l^2(\mathcal{A}_i)}^2 \leq \frac{C_{\text{apx}}^2}{\varrho(S^2A)}\langle A_i N_i^T u, N_i^T u\rangle \quad \forall u \in \Re^n. \tag{3.8}$$

Assuming a fixed prolongator smoother S, the satisfaction of (3.8) depends solely on the selection of columns of $\hat{P}$. We are thus led to constructing $\hat{P}$ so that

$$\inf_{v_i \in \Re^{m_i}} \|u - \hat{P}_{*,i}v_i\| = \langle (I - Q_i)u, u\rangle_{l^2(\mathcal{A}_i)} \leq \Lambda_i \|u\|_{A_i}^2 \quad \forall u \in \Re^n, \tag{3.9}$$

where $Q_i = W_i(W_i^T W_i)^{-1}W_i^T$ is the $\Re^{n_i}$-orthogonal projection onto the Range(W_i). Equation (3.9) can be reformulated as a generalized eigenvalue problem for $I_i(I-Q_i)I_i$ preconditioned by A_i. The constant Λ_i in (3.9) can be decreased by finding the largest eigenvalue of the generalized eigenvalue problem and adding the corresponding eigenvector to the range of Q_i. This amounts to introducing new columns corresponding to aggregate $\mathcal{A}_i$ into the tentative prolongator $\hat{P}$. This procedure can be repeated until (3.9) is eventually satisfied with a uniform constant $\Lambda \geq \Lambda_i$ for all $i = 1, \ldots, J$.

To reduce cost associated with the solution of the local generalized eigenvalue problem, we instead solve the local eigenvalue problem for A_i. The following lemma [7] justifies this reduction in cost.

LEMMA 3.2. $w_j^{(i)} = I_i\hat{w}_j^{(i)}$ $\hat{w}_j^{(i)}$ A_i A_i $\frac{1}{\Lambda} > 0$

$$\langle I_i(I - Q_i)I_i u, u\rangle_{\Re^{n_i}} \leq \Lambda \|u\|_{A_i}^2 \quad \forall u \in \Re^{n_i}.$$

Note that we only compute the near null space vectors for the subproblems corresponding to subdomains Ω_i created by the domain partitioner (computing the near

null space vectors for the global matrix would be prohibitively expensive). The setup cost is still relatively high, but it can be performed in parallel with no interprocessor communication. To keep cost down, the size of each subdomain is limited to a few thousand nodes.

The coarse-space enriching technique we have thus far described in this section is based on the two-level method proposed in [7], which utilizes aggregates $\mathcal{A}_i$, where each aggregate consists of all the nodes in the subdomain Ω_i. This necessitates the use of special prolongator smoothers and special multigrid relaxation procedure in order to guarantee good convergence. Here we are interested in providing a multilevel generalization. Our current GSA method differs from that in [7] in that a standard aggregation coarsening will be applied on each $\bar{\Omega}_i$. Thus the nodes of Ω_i are subaggregated into a number of small, standard-sized aggregates, $\mathcal{N}_i = \bigcup \mathcal{A}_{ij}$, where $\mathcal{N}_i$ denotes the set of unique nodes corresponding to subdomain $\bar{\Omega}_i$. The eigenmodes of A_i are then restricted to these smaller aggregates, $\mathcal{A}_{ij}$, and the tentative prolongator used in the current GSA can be written as

$$\tilde{P}x = \sum_{i=1}^{J} \sum_{j=1}^{\mu_i} \hat{N}_{ij} W_{ij} x^{(ij)}, \tag{3.10}$$

where W_{ij} denotes restriction of W_i to $\mathcal{A}_{ij}$ and $\hat{N}_{ij}$ is a zero-one matrix mapping the degrees of freedom in $\mathcal{A}_{ij}$ into the set of global degrees of freedom. $x^{(ij)}$ denotes the segment of coarse degrees of freedom associated with aggregate $\mathcal{A}_{ij}$.

As the computation of eigenmodes on each subdomain is performed independently, we do not allow aggregation across the subdomain boundary in order to facilitate alignment. Consequently, the number of unknowns on the coarsest grid is at least $J \times \nu$, where J is the number of subdomains and ν is the desired number of null space vectors. When both J and ν are large, the coarsest problem can become too large for direct solution.

Since the set of coarse grid basis functions generated by $\tilde{P}$ is a refinement of that generated by $\hat{P}$, we trivially obtain

$$\sum_{j=1}^{\mu_i} \min_{x^{(ij)}} \|u - \tilde{P}x^{(ij)}\|^2_{l^2(\mathcal{A}_{ij})} \leq \min_{x^{(i)}} \|u - \hat{P}x^{(i)}\|^2_{l^2(\mathcal{A}_i)} \leq \Lambda \|u\|^2_{A_i}.$$

Thus the standard convergence theory for the smoothed aggregation method [26] can be used to guarantee convergence of the resulting method. As a result of using the standard aggregates, $\tilde{P}$ can approximate well not only the computed set of eigenmodes, $\{w_j^{(i)}\}_{j=1}^{m_i}$ but also a much richer subspace consisting of any modes with locally similar character. This can reduce the number of eigenmodes that need to be computed. Another benefit of using GSA, compared to the method of [7], is the reduced amount of interprocessor communication. The method of [7] was primarily intended for a serial environment. Its multigrid relaxation is based on polynomial smoothing, which makes it suitable for parallel implementation. But its use of the more powerful prolongation smoothing, required to achieve optimal convergence rates, results in larger cross-processor overlap in its coarse-level basis functions. This leads to increased communication necessary in setting up the coarse-level matrices. In contrast, GSA, with its use of the standard prolongation smoother, reduces communication and is certain to result in higher flop rates in a massively parallel environment.

For completeness, we include a summary of the GSA multilevel coarsening procedure below. Let l denote the level index ($l = 0$ corresponds to the finest grid).

Furthermore, let us denote by $\mathcal{N}_i^l$ the set of unique nodes on level l which is associated with subdomain $\bar{\Omega}_i$, and by μ_i^l its number. Symbol $\mathcal{A}_j^l$ denotes the aggregates on level l. Given a decomposition of the domain (3.3) and the corresponding subdomain matrices, A_i, GSA coarsening proceeds as follows:

1. Initialize one large aggregate, $\mathcal{A}_i^0 = \mathcal{N}_i$, per subdomain $\bar{\Omega}_i, i = 1, \ldots, J$, distributing any shared nodes so that the resulting $\mathcal{N}_i$ are mutually disjoint. Set $\mu_i^0 = card(\mathcal{N}_i), i = 1, \ldots, J$.
2. For each local matrix $A_i, i = 1, \ldots, J$, compute a set of m_i eigenmodes, W_i^k, smaller than Λ. Set $l = 0$.
3. Perform standard SA aggregation independently over each $\mathcal{N}_i^l$ to produce $\{\mathcal{A}_{ij}^l\}$ such that $\mathcal{N}_i^l = \bigcup_j \mathcal{A}_{ij}^l$. Split W_i^l into segments, W_{ij}^l, corresponding to aggregates.
4. Construct operator $\tilde{P}_l$ as in (3.10). Simultaneously, compute W_i^{l+1}, the coarse-level representations of W_i^l such that $W_i^l = \tilde{P} W_i^{l+1}$.
5. Construct smoothed prolongator, $P_l = (I - \omega D_l^{-1} A_l)\tilde{P}_l$.
6. Construct coarse-level operator, $A_{l+1} = P_l^T A_l P_l$
7. If $l + 1$ is the coarsest level, stop.
   Otherwise, set $\mathcal{N}_i$ to be the set consisting of global numbers indexing the aggregates on level l, $\mu_i^{l+1} = card(\{\mathcal{A}_{ij}^l\})$, set $l = l + 1$, and go to step 3.

The construction of $\tilde{P}$ in step 4 (and in (3.10)) is modified in practice by orthogonalizing the columns of W_{ij}^l over the local aggregates $\mathcal{A}_{ij}^l$. Such modification does not change the range of the constructed tentative prolongator but produces better-conditioned coarse-level matrices (cf. [28]). Also, it can be conveniently combined with the computation of the coarse-level representation W_i^{l+1} (cf. [26]).

We note that a coarse-space enrichment similar to the one described here has been used in the context of classical AMG in the AMGe class of methods [6, 10] and also by Mandel [20, 21] in the context of two-level domain decomposition suitable for p-version finite elements. Reference [13] describes a coarse-space enrichment for a nonsmoothed aggregation method using very small element agglomerates, under the assumption that the local element matrices are available. The recently introduced class of adaptive smoothed aggregation methods (αSA; cf. [8]) approaches local coarse-space enrichment by run-time evaluation of the error components of the iteration and subsequent incorporation of these modes into its multigrid transfer and coarse operators.

4. Parallel implementation issues. The original coarse-grid selection algorithms for both RS and SA are inherently sequential. In the solution phase, the primary factor for parallel efficiency is the use of parallel and robust smoothers. This section briefly reviews efforts pertaining to these two issues.

4.1. Parallel coarsenings for RS AM. Several parallel coarsening algorithms have been proposed and tested successfully: the parallel RS coarsening [14], the Cleary–Luby–Jones–Plassmann (CLJP) coarsening [11], and the hybrid (or Falgout) coarsening. The parallel RS coarsening first performs independent local coarsening within each processor, followed by coarsening on the processor boundaries. The CLJP coarsening is a form of a parallel maximal independent set algorithm [17, 19] modified to enforce the coarsening criteria. The hybrid coarsening uses the RS coarsening interior to the processors and CLJP coarsening on the boundaries. The relative performance of these algorithms varies dependent on the applications. Since the CLJP algorithm generally gives better results for unstructured grid problems, it is used in

our numerical experiments. For other strategies which have been proved successful in parallelizing RS-based multigrid methods we refer the reader to [18, 15].

4.2. Parallel coarsenings for SA AMG. Reference [24] describes three parallel aggregation algorithms. The coupled aggregation scheme proceeds by having each processor first separating all nodes assigned to it into the interior and border nodes. Border nodes are those which share a grid edge with a nodes on another processor. Subsequently, this scheme aggregates the border nodes before aggregating the interior nodes. This scheme has the disadvantage that some processors may have to wait until other processors have completed aggregating their border nodes. The worst-case scenario would require a wait time proportional to $P^{1/3}$, where P is the number of processors. However, the scheme can give significantly better aggregates than the other approaches considered.

The second parallel aggregation scheme is based on maximally independent set (MIS). In brief, the widely studied parallel MIS algorithms can be applied to the square of the matrix representing grid connectivities. This scheme has the disadvantage that it requires computing the square of a matrix.

The third scheme, which is our choice in our numerical studies, is called the decoupled aggregation scheme. This scheme lets each processor form its set of aggregates independently of one another. It is very simple to implement and it generally gives good performance. However, its performance depends a great deal on the quality of the domain partitioner. Furthermore, not allowing the aggregate to cross the processor boundaries can result in slightly higher overall grid complexity (the total number of grid nodes in the grid hierarchy) and larger coarsest grid size when the number of processors is large. We remedy the latter by triggering a processor aggregation step (forming processor connectivity graph and grouping processors into aggregates) when the coarsest grid is too large.

4.3. Parallel smoothers. Smoothers play a critical role in the AMG solution phase. On sequential computers, Gauss–Seidel (GS) is usually the smoother of choice. On massively parallel computers, however, the choice of smoother is less clear. The GS smoother can be modified for higher degree of parallelism by combining it with the Jacobi smoother. By preserving the sequential nature of GS within each processor and exploiting the parallel nature of Jacobi across processor boundaries, a hybrid GS-Jacobi smoother has been demonstrated to be quite effective on a small number of processors. For massive parallelism, it may suffer from slow convergence rates. Another approach is to improve GS's parallelism by using multicoloring [1]. A limitation of this approach is that there needs to be sufficient number of unknowns on each processor, and it is thus not an effective smoother on the coarse grids. Other efforts include the use of polynomial smoothers [2] and Krylov smoothers preconditioned with Jacobi, block Jacobi, or overlapped Schwarz, all with limited success.

In our numerical experiments, we use a variant of the hybrid GS-Jacobi smoother given in [29]. The idea is to recover fast convergence rates by using underrelaxation in GS and/or damped Jacobi across processors. The relaxation parameters can be computed by a few conjugate gradient iterations.

5. Test problems from structural mechanics. The test problems are taken from the ALE3D [3] code developed at the Lawrence Livermore National Laboratory. The implicit integration algorithm in ALE3D for solid materials is based on an updated Lagrangian displacement-based formulation. A set of nonlinear equations is solved for equilibrium using the state and configuration at the end of a time step.

During the time integration, nonlinearities arise because of the material response and configuration changes.

The basic equations are the momentum equations derived by starting with the local dynamic equilibrium relation in a form suitable for a solid body

$$\begin{aligned} \rho \mathbf{u}_{tt} &= \nabla \cdot \boldsymbol{\sigma} + \rho \mathbf{b}, \\ \mathbf{n} \cdot \boldsymbol{\sigma} &= \mathbf{g}, \end{aligned} \tag{5.1}$$

where $\mathbf{u}$ is the displacement vector, $\boldsymbol{\sigma}$ is the stress tensor, $\mathbf{b}$ is the body force vector per unit mass, ρ is the density, and $\mathbf{g}$ is the surface traction. Applying the weak form and then integrating by parts yield

$$\int_V \rho \mathbf{u}_{tt} \cdot \delta \mathbf{u} dV + \int_V \boldsymbol{\sigma} : (\nabla \delta \mathbf{u})^T dV - \int_S \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{u} dS - \int_V \rho \mathbf{b} \cdot \delta \mathbf{u} dV = 0, \tag{5.2}$$

where the colon signifies a double inner product, V is the volume, and S is the surface of the body. Time integration is performed by using the full Newton–Raphson iteration scheme represented by

$$\frac{dP}{d\mathbf{u}} \cdot d\mathbf{u} = -P, \tag{5.3}$$

where P is the left-hand side of (5.2) and (A being the surface area)

$$dP = \int_V \left[d\boldsymbol{\sigma} - (\nabla d\mathbf{u})^T \cdot \boldsymbol{\sigma} + (\nabla \cdot d\mathbf{u}) \boldsymbol{\sigma} \right] : (\nabla \delta \mathbf{u})^T dV \tag{5.4}$$

$$\begin{aligned} &- \int_S \left[\frac{d\mathbf{g}}{d\mathbf{u}} \cdot d\mathbf{u} + \frac{1}{A} \mathbf{g} \frac{dA}{d\mathbf{u}} \cdot d\mathbf{u} \right] \cdot \delta \mathbf{u} dS \\ &- \int_V \rho \left[\frac{d\mathbf{b}}{d\mathbf{u}} \right] \cdot d\mathbf{u} + \int_V \rho \left[\frac{d\mathbf{u}_{tt}}{d\mathbf{u}} \right] \cdot \delta \mathbf{u} dV. \end{aligned} \tag{5.5}$$

This displacement-based Jacobian equation is solved with the finite element method using linear elements on a hexahedral mesh. At each time step, corrections are added to the previous estimate of the nodal displacements. The nodal displacements are then used to calculate displacement gradients and strain increment tensors that are passed to the constitutive model.

The constitutive relation we use cannot be written simply in terms of stresses and elastic properties. It also depends on current loading. For a forward gradient stress integration, the material stiffness would take the form

$$C_{ijkl} = L_{ijkl} - \frac{L_{ijrs} P_{rs} P_{tu} L_{tukl}}{h + P_{rs} L_{rstu} P_{tu}}, \quad i, j, k, l, r, s, t, u = 1, \ldots, 3, \tag{5.6}$$

where L is the standard fourth order elasticity tensor, P is the derivative of the yield surface with respect to the stress tensor, and h is the material strain hardening.

5.1. The spherical shell problem. The domain for the first problem is an octant of a spherical shell depicted in Figure 1. The shell has three layers having steel on the outside and inside layers and polycarbonate in the middle. Both elasticity and plasticity are present in the constitutive model. The forcing function is the small amount of energy injected into the polycarbonate layer, causing the polycarbonate to expand against the inner and outer layers. Equation (5.6) can be simplified to $C = L$ in this case, as there is no plasticity.

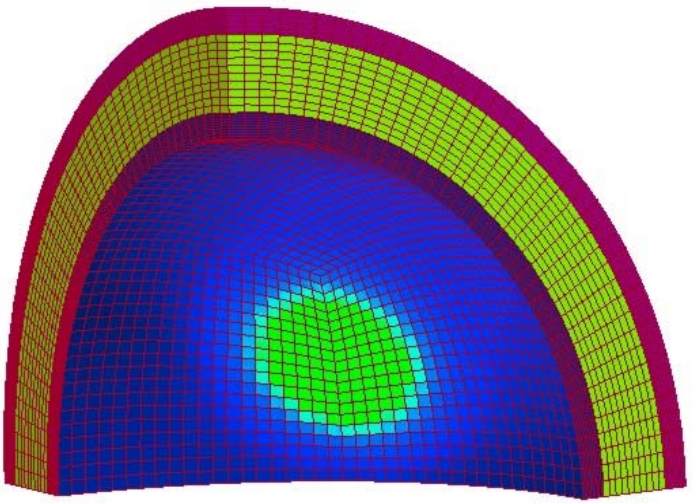


FIG. 1. *The spherical shell problem.*

5.2. The crystal plasticity problem. This test problem (called Tbar and shown in Figure 2) uses a crystal plasticity constitutive model given by

$$C_{ijkl} = L_{ijkl} - \sum_{a=1}^{12} \sum_{b=1}^{12} L_{ijmn} Q^a_{mn} R^b_{rs} L_{rskl}, \tag{5.7}$$

where Q^a and R^b are each second order tensors for each of the 12 slip systems. As a result, the global stiffness matrix is slightly nonsymmetric. Boundary conditions for this test problem are forces applied at one end in the direction of the bar length and the other end is tied down to prohibit displacement in the direction of the bar length.

5.3. The bifurcation problem. This test problem (called Rbar and shown in Figure 3) involves bifurcation which occurs when the force required to stretch the perfectly uniform rod reaches a peak. The load is the axial stress times the cross sectional area. During deformation, the stress first increases linearly with strain as an elastic material. When stretched to a stress level known as the yield point, dislocation mechanisms in the metal allow rearrangement of the atoms and result in permanent deformation of the material. This is plastic flow. During plastic flow the rate of change of stress with strain is typically 2 to 3 orders of magnitude less than during the initial elastic deformation. This rate typically decreases continuously during deformation. Also, during plastic flow the volume of the material is roughly constant leading to a measurable decrease in the bar diameter as it is stretched.

Since the load is the stress times the cross-sectional area, the load rate is comprised of two terms containing the increasing stress rate and the decreasing area rate. For most metals the area rate contribution overtakes the stress rate contribution in the strain range from 5% to 30% extension. This is the peak load and corresponds to the Considère [12] condition. Up to the peak load, energy considerations require that the bar deform uniformly. At the peak load the incremental energy increase applied

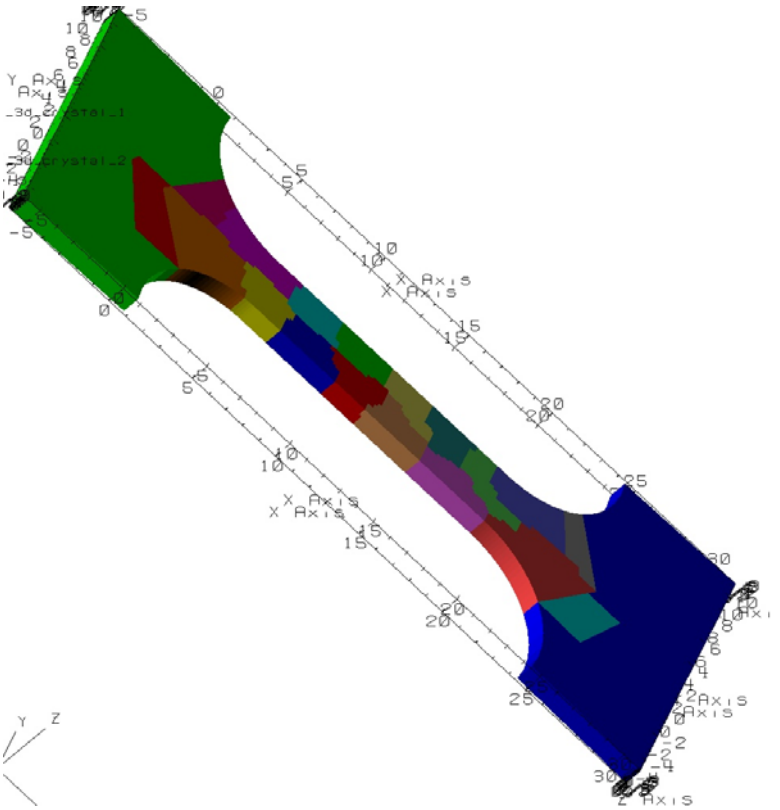


FIG. 2. *The Tbar problem (subdomains in different colors).*

to extend the bar is the same whether it continues stretching uniformly or some nonuniform strain rate pattern emerges. If a nonuniform strain rate pattern emerges, it is not unique.

It is the behavior at this instance of nonunique solution that is challenging to the solver. If everything is geometrically perfect and the low order digits are all the same, the bar can continue to deform uniformly. This has occasionally been seen with direct solvers. If the numerical perturbations are not negligible, a nonuniform deformation pattern will emerge, triggered by the numerical imperfections. The energy increase and the load at the nodes of the finite element mesh are nearly the same regardless of the mode selected.

In the physical experiments, the real materials are not perfect and the effects of the grip ends holding the specimen generally force a nonuniform deformation pattern whereby the diameter reduces faster in one location. The second order work terms at the Considère point are such that these localized deformation modes are favored over continued uniform deformation. Once the localized modes begin, the bar is energetically favored to continue deforming in the localized mode. This eventually leads to what is known as a necking behavior and fracture of the bar (see Figure 4).

6. Numerical experiments. The numerical experiments are run on two IBM SP2 computers (to be called SP2-White and SP2-Blue from this point on) at the Lawrence Livermore National Laboratory. The SP2-White computer has more than

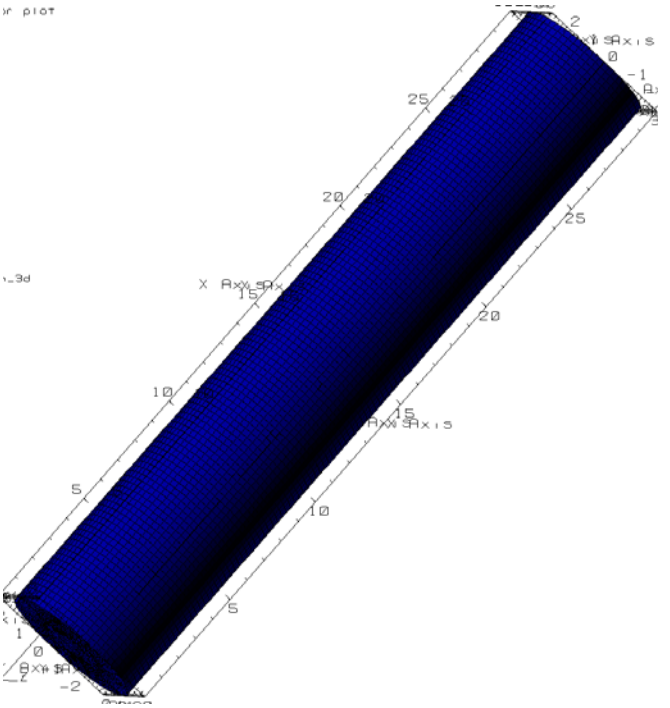


FIG. 3. *The Rbar problem (cylindrical computational domain).*

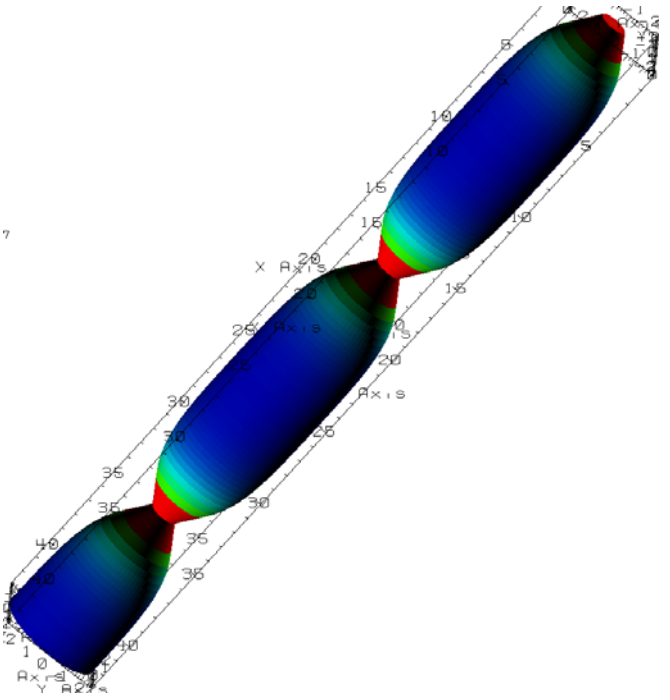


FIG. 4. *The Rbar problem (a deformation configuration).*

500 nodes with 16 gigabyte of main memory per node. Each node has 16 Power3 processors running at 375 MHz each. The SP2-Blue computer has 264 nodes with 4 PowerPC 604e processors and 1.5 gigabytes of memory per node. Due to the scarce availability of large number of processors (1000 or more) and the large disk space requirement for storing the finite element meshes, our numerical experiments are limited to a maximum of 512 processors. For the spherical shell problem, some performance data were collected for RS and SA on 4000 SP2-White processors. The largest run for RS is about 300 millions unknowns which consumes 200 iterations and 2000 seconds for each solve. For SA we were able to successfully run an even larger problem (610 million unknowns, which takes about 500 iterations and 1500 seconds) due to its lower memory requirements.

For both RS and SA, we prescribe one sweep of hybrid SSOR-Jacobi for both pre-smoothing and post-smoothing. For RS, we choose different strength thresholds for best results on different test problems. For SA, the strength threshold is 0.0 and no prolongation smoothing is used for all test problems. In addition, the near null space vectors (the 6 rigid body modes) are computed from the nodal coordinates from our finite element package. For GSA, additional near null space information is calculated from the element stiffness matrices.

6.1. Scalability study for the spherical shell problem. We study the parallel performance of the spherical shell problem on the SP2-White computer using the conjugate gradient method with RS and SA as preconditioners. Each processor has about 100K unknowns, yielding more than 51 million unknowns on 512 processors. The entire simulation takes about 10 linear solves, and the timing results are the averages of all linear solves. Convergence criterion for the conjugate gradient is 10^{-6}. The best strength threshold for RS is around 0.9–0.99 for this problem. We use 0.95 in our experiment. With this threshold the grid and operator complexities (grid complexity is the ratio of the total number of rows in all A_k's with respect to A_0 while the operator complexity is the ratio of the total number of nonzeros in all A_k's with respect to the number of nonzeros in A_0) are approximately 2.1 and 2.5, respectively. The corresponding grid and operator complexities for SA are 1.08 and 1.15, respectively. The timing results and iteration counts are shown in Figures 5 and 6.

These results show that SA is reasonably scalable (that is, the total solution time grows slowly with the number of processors given the same problem size per processor), while RS appears to be somewhat less scalable due mainly to the faster convergence rate deterioration with larger problem size (but RS gives much better solution times than SA when the number of processors is small). For SA, we observe a substantial increase in the solution time going from $P = 128$ to $P = 256$ even though the iteration counts stay about the same. After eliminating the cause attributing to load imbalance, we conclude that the performance deterioration is probably due to higher memory access conflicts for larger test problems.

6.2. Scalability/robustness study for the Tbar problem. Numerical experiments for the Tbar problem are performed on the SP2-Blue computer. Each processor has about 30K unknowns, yielding a total of more than 14 million unknowns for 512 processors. The generalized minimal residual (GMRES [23]) method with a restart size of 200 is used for this problem due to nonsymmetry. The prescribed convergence criterion is 10^{-8}. We use 0.9 as strength threshold for RS. With this threshold the grid and operator complexities are about 2.1 and 3.4, respectively, for all processor configurations. The grid and operator complexities for SA are 1.09

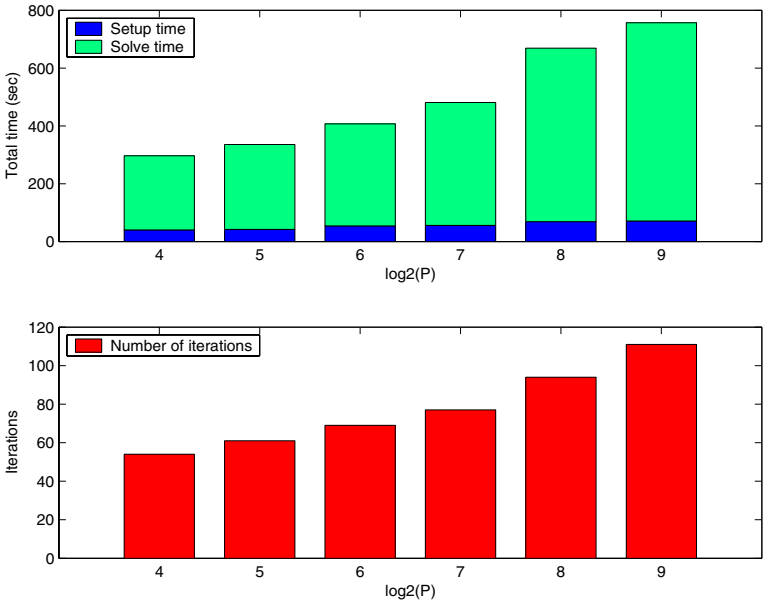


FIG. 5. *Spherical shell problem using RS preconditioner.*

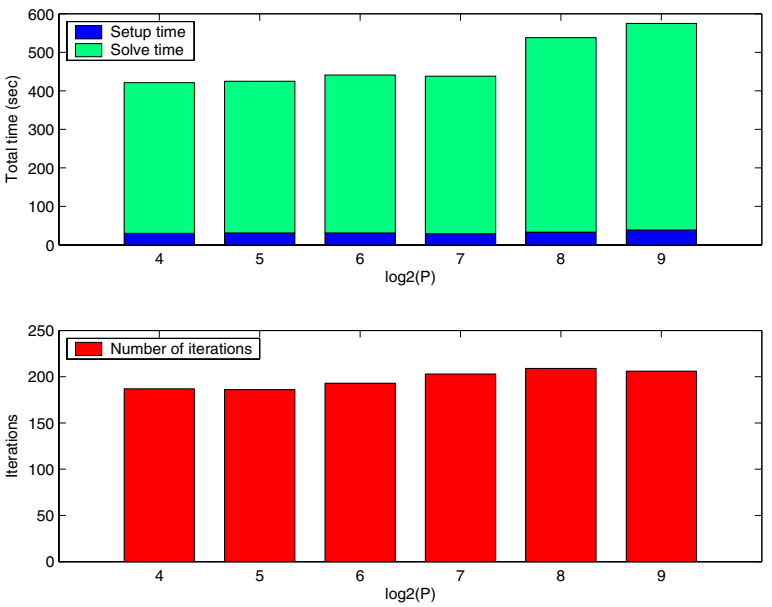


FIG. 6. *Spherical shell problem using SA preconditioner.*

and 1.13, respectively. The numerical results given in Figures 7 and 8 are for the first linear solve. We observe that starting from the second linear solve RS fails to converge (stagnation occurs when the relative residual norm drops to about 10^{-2}), while SA takes many more iterations. We demonstrate in Figures 9 and 10 that GSA helps to improve the robustness of SA.

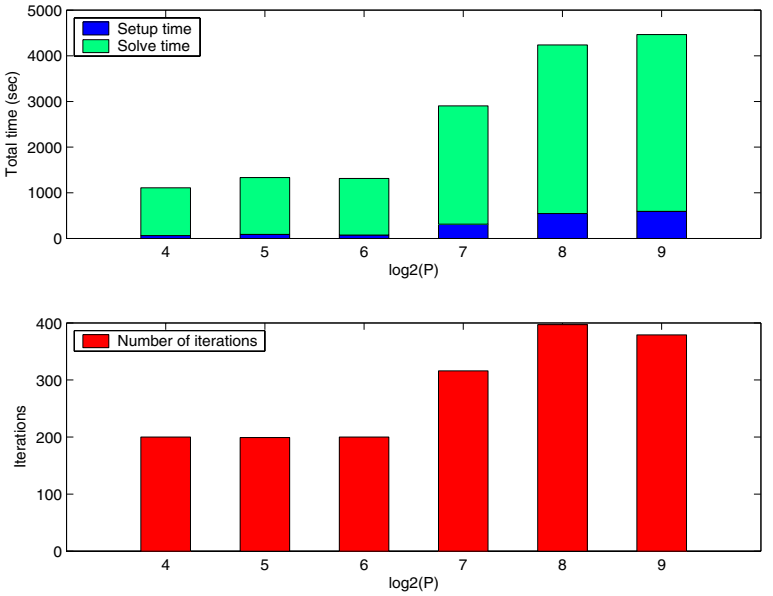


FIG. 7. *Tbar problem using RS preconditioner.*

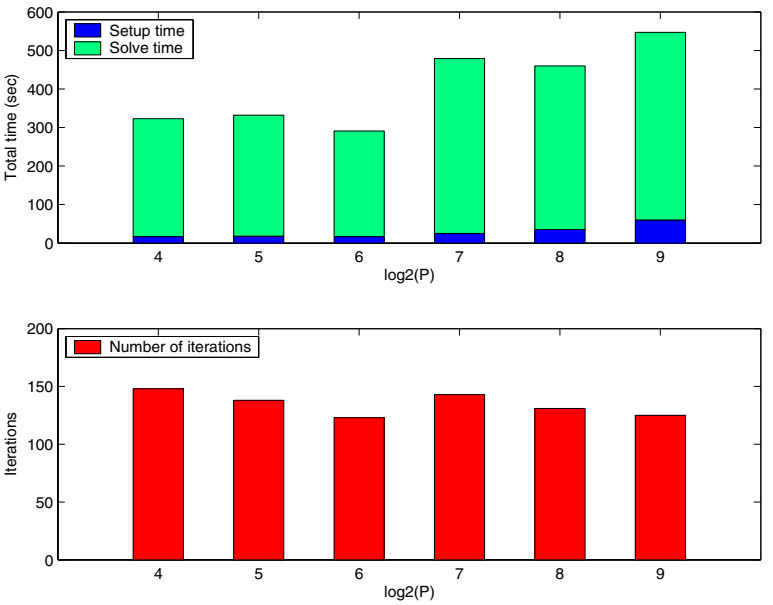


FIG. 8. *Tbar problem using SA preconditioner.*

From Figures 7 and 8, we observe that RS takes many more iterations than SA to converge even for the first linear solve. From examining the convergence history (not shown here) we further notice that the iteration count for RS would have been reduced by a factor of 4 if the convergence criterion is raised to 10^{-6}. We conclude that, for low accuracy runs, RS and SA give comparable performance. Similarly, the

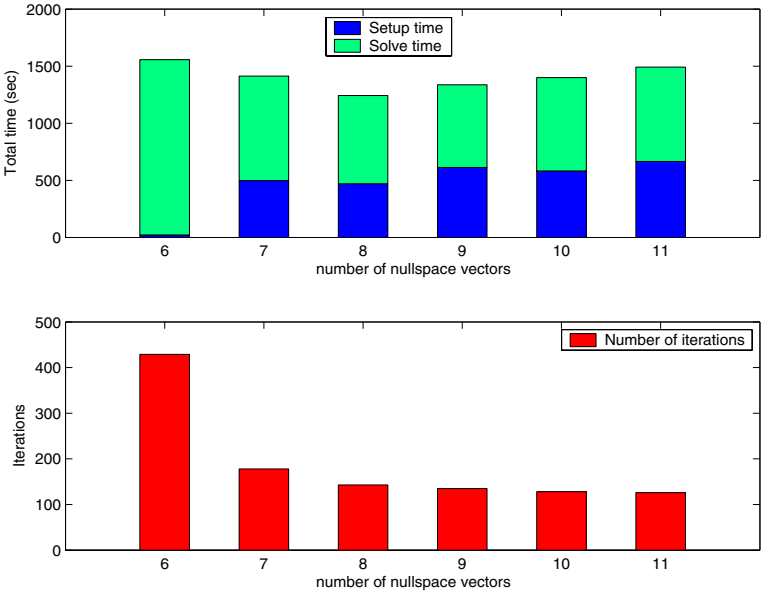


FIG. 9. *Tbar problem using GSA preconditioner.*

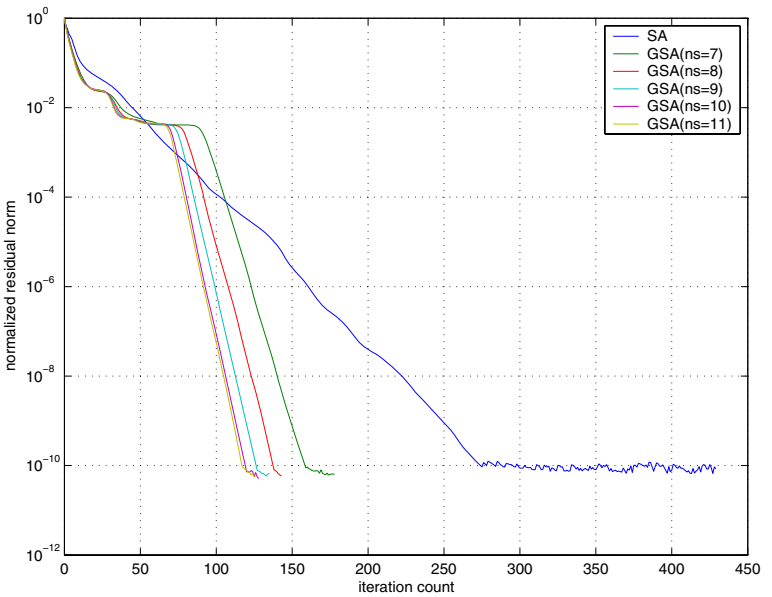


FIG. 10. *Convergence history of Tbar problem using GSA preconditioner.*

residual norm curves in Figure 10 shows that SA would have been competitive with GSA for low accuracy runs. For higher accuracy runs and in later time steps when the effect of crystal plasticity becomes more prominent, GSA performs the best.

6.3. Robustness study for the Rbar problem. For the Rbar problem, our focus is on the robustness of the preconditioners. Since similar scalability behaviors

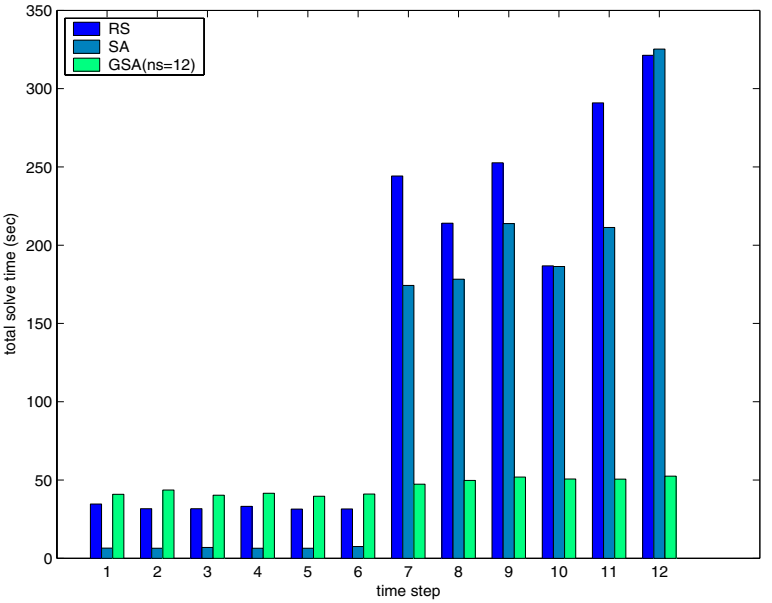


FIG. 11. *Rbar problem with all three preconditioners.*

(as the spherical shell and Tbar problems) are observed here, we deliberately omit the scalability results. The concern for solver robustness stems from the observation that convergence deteriorates rapidly in later time steps when nonuniform deformation occurs. To illustrate the relative performance of the different solvers, we use a test problem with 380K unknowns running on 64 processors of the SP2-Blue computer. Again, we use GMRES(m) with $m = 100$ and convergence criterion 10^{-8}. To reduce the eigendecomposition time in GSA, 128 subdomains are used, with each subdomain having roughly about 3000 unknowns. The number of grid levels and near null space vectors for GSA are 3 and 12 (ns=12), respectively, yielding the coarsest grid size of about 1500.

Timing results for all three AMGs are given in Figure 11 up to the 12th time step. Each time step involves 3 to 6 linear solves, and the data given here are the averages. We observe that in the first 6 time steps when the problem is still well-behaved, SA without enriching the prolongation operator is adequate. Beyond the 12th time step, both RS and SA fail to converge, but GSA continues to march on in time with acceptable convergence rates, as depicted in Figure 12. We observe from Figure 12 that a large fraction of the setup time is in the eigendecomposition. We comment that this setup time can be improved by noting that the first 6 near null space vectors are already available and only 6 additional ones are to be computed. In the current implementation, we have not taken advantage of this fact.

7. Summary and conclusion. In this paper we demonstrate the feasibility of using AMG for solving large-scale structural mechanics problems on massively parallel processors.

For difficult structural mechanics problems, the smoothing and coarsening procedures used by the standard AMG methods may fail to satisfactorily capture all significant low energy modes of the error. Although the standard SA can guarantee approximation of a general set of functions, it requires that these functions be known

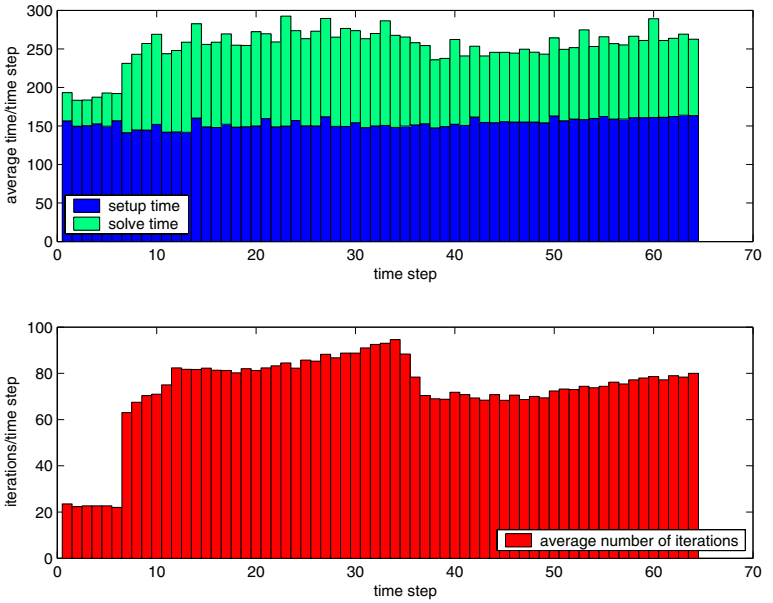


FIG. 12. *Rbar problem using GSA preconditioner.*

before its setup procedure commences. The RS coarsening faces an additional hurdle: even if the set of low energy modes were a priori known, an approximation of a general set of functions is an open problem for RS interpolation. Local prolongation enriching methods such as GSA help to alleviate these problems. Superior convergence has been observed for GSA compared to the RS and SA. Some shortcomings of GSA are the expensive eigendecomposition in the preprocessing step and the potentially large coarsest grid problem. Faster eigensolvers and faster parallel direct solvers are critical for this class of efficient and robust multigrid solvers. Future effort should focus on improving efficiency by reducing memory traffic.

REFERENCES

[1] M. F. ADAMS, *A distributed memory unstructured Gauss-Seidel algorithm for multigrid smoothers,* in ACM/IEEE Proceedings of SC2001: High Performance Networking and Computing, Denver, CO, 2001.

[2] M. F. ADAMS, M. BREZINA, J. J. HU, AND R. S. TUMINARO, *Parallel multigrid smoothing: Polynomial versus GS,* J. Comput. Phys., 188 (2003), pp. 593–610.

[3] *ALE3D Theory Manual,* UCRL-PRES-203313, Lawrence Livermore National Laboratory, Livermore, CA.

[4] P. B. BOCHEV, C. J. GARASI, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *An improved algebraic multigrid method for solving Maxwell's equations,* SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[5] A. BRANDT, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic Multigrid (AMG) for Automatic Multigrid Solutions with Application to Geodetic Computations*, Technical Report, Inst. for Computational Studies, Fort Collins, CO, 1982.

[6] M. BREZINA, A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[7] M. BREZINA, C. HEBERTON, AND P. VANĚK, *An Iterative Method with Convergence Rate Chosen A Priori,* UCD/CCM Report 140, Center for Computational Mathematics, University of Colorado at Denver, Denver, CO, 1999.

[8] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA),* SIAM J. Sci. Comput., 25 (2004), pp. 1896–1920.

[9] M. Brezina and P. Vaněk, *A Black-box Iterative solver based on a two-level Schwarz method,* Computing, 63 (1999), pp. 233–263.

[10] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., 25 (2003), pp. 1–26.

[11] A. J. Cleary, R. D. Falgout, V. E. Henson, and J. E. Jones, *Coarse grid selection for parallel algebraic multigrid,* in Proceedings of the Fifth International Symposium on Solving Irregularly Structured Problems in Parallel, Lecture Notes in Comput. Sci. 1457, Springer-Verlag, New York, 1998, pp. 104–115.

[12] M. Considère, *Die Anwendung von Eisen und Stahl bei Konstruktionen*, Gerold-Verlag, Wien, 1888.

[13] J. Fish and V. Belsky, *Generalized Aggregation Multilevel Solver*, Internat. J. Numer. Methods Engrng., 40 (1997), pp. 4341–4361.

[14] R. D. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners,* in Computational Science (CARS 2002, Part III), Lecture Notes in Comput. Sci. 2331, Springer-Verlag, New York, 2002, pp. 632–641.

[15] G. Haase, M. Kuhn, and S. Reitzinger, *Parallel algebraic multigrid methods on distributed memory computers,* SIAM J. Sci. Comput., 24 (2002), pp. 410–427.

[16] V. E. Henson and U. M. Yang, *BoomerAMG: A parallel algebraic multigrid solver and preconditioner,* Appl. Numer. Math., 41 (2002), pp. 155–177.

[17] M. T. Jones and P. E. Plassmann, *A parallel graph coloring heuristic,* SIAM J. Sci. Comput. 14 (1993), pp. 654–669.

[18] A. Krechel and K. Stüben, *Parallel algebraic multigrid based on subdomain blocking,* Parallel Computing, 27 (2001), pp. 1009–1031.

[19] M. Luby, *A simple parallel algorithm for maximal independent set problem,* SIAM J. Comput., 15 (1986), pp. 1036–1053.

[20] J. Mandel, *Adaptive iterative solvers in finite elements,* in Solving Large-Scale Problems in Mechanics, M. Papadrakakis, ed., John Wiley & Sons, Chichester, UK, 1993, pp. 65–86.

[21] J. Mandel, *Iterative methods for p-version finite elements: Preconditioning thin solids*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 247–257.

[22] J. W. Ruge and K. Stüben, *Algebraic multigrid,* in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math., SIAM, Philadelphia, 1987, pp. 73–130.

[23] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[24] C. H. Tong and R. S. Tuminaro, *Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines,* in Proceedings of Supercomputing, 2000.

[25] P. Vaněk, *Acceleration of convergence of a two-level algorithm by smooth transfer operators*, Appl. Math., 37 (1992), pp. 265–274.

[26] P. Vaněk, M. Brezina, and J. Mandel, *Convergence of algebraic multigrid based on smoothed aggregation,* Numer. Math., 88 (2001), pp. 559–579.

[27] P. Vaněk, M. Brezina, and R. Tezaur, *Two-grid method for linear elasticity on unstructured meshes,* SIAM J. Sci. Comput., 21 (1999), pp. 900–923.

[28] P. Vaněk, J. Mandel, and M. Brezina, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.

[29] U. M. Yang, *On the use of relaxation parameters in hybrid smoothers,* Numer. Linear Algebra Appl., 11 (2004), pp. 155–172.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1555–1572
© 2006 Society for Industrial and Applied Mathematics

APPROXIMATE FACTORIZATION CONSTRAINT PRECONDITIONERS FOR SADDLE-POINT MATRICES*

H. S. DOLLAR† AND A. J. WATHEN†

Abstract. We consider the application of the conjugate gradient method to the solution of large, symmetric indefinite linear systems. Special emphasis is put on the use of constraint preconditioners and a new factorization that can reduce the number of flops required by the preconditioning step. Results concerning the eigenvalues of the preconditioned matrix and its minimum polynomial are given. Numerical experiments validate these conclusions.

Key words. preconditioning, indefinite linear systems, Krylov subspace methods, conjugate gradient method

AMS subject classifications. 65F10, 65F50

DOI. 10.1137/04060768X

1. Introduction.

There are many areas where we wish to solve matrix systems of the form

$$\underbrace{\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} x \\ y \end{bmatrix} = \underbrace{\begin{bmatrix} c \\ d \end{bmatrix}}_{b}, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{m\times n}$. We shall assume that $0 < m \leq n$, B is of full rank, and A is positive definite in the nullspace of B. The symmetric matrix $\mathcal{A}$ is indefinite (i.e., it has both positive and negative eigenvalues.) Such systems may arise in convex quadratic programming problems [13] and saddle-point systems (see [2] and [10, Chapters 7, 9]), as well as many other areas.

Solution of systems of equations of the form (1.1) can be achieved by a number of methods. For large, sparse or structured matrices iterative methods are an attractive option. In particular, Krylov subspace methods apply techniques that involve orthogonal projections onto subspaces of the form

$$\mathcal{K}(\mathcal{A}, b) \equiv \text{span}\{b, \mathcal{A}b, \mathcal{A}^2b, \ldots, \mathcal{A}^{n-1}b, \ldots\}.$$

The conjugate gradient method (CG), minimum residual method (MINRES), and generalized minimal residual method (GMRES) are all common iterative Krylov subspace methods. The CG method is used for symmetric, positive definite matrices, MINRES for symmetric and possibly indefinite matrices, and GMRES for unsymmetric matrices [12].

MINRES and GMRES will find the solution of the linear system (1.1) within $n + m$ iterations in exact arithmetic, but CG may fail because of the indefiniteness of the system. GMRES is just an expensive way to implement MINRES when $\mathcal{A}$ is symmetric as here. For very large systems this upper bound on the number of

*Received by the editors May 3, 2004; accepted for publication (in revised form) December 21, 2004; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-5/60768.html
†Oxford University Computing Laboratory, Numerical Analysis Group, Wolfson Building, Parks Road, Oxford, OX1 3QD, UK (hsd@comlab.ox.ac.uk, wathen@comlab.ox.ac.uk). The work of the first author was supported by an EPSRC Doctoral Training Account.

iterations is not helpful and such methods would generally not be employed unless many fewer iterations were required in practice; it is often advantageous to use a preconditioner $\mathcal{P}$ with such iterative methods. The preconditioner should reduce the number of iterations required for convergence but not significantly increase the amount of computation required at each iteration [12, Chapter 13].

Keller, Gould, and Wathen [8] investigated the use of a preconditioner of the form

$$\mathcal{P} = \begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix}, \tag{1.2}$$

where G approximates but is not the same as A. They were able to prove various results about the eigenvalues and eigenvectors for the preconditioned systems $\mathcal{P}^{-1}\mathcal{A}$, where $\mathcal{A}$ and $\mathcal{P}$ are defined in (1.1) and (1.2), respectively; $\mathcal{P}$ is called a [illegible]

The system (1.1) is symmetric and indefinite, implying that we are unable to apply the CG method directly in a robust way since it might fail [3, section 6.3]. Gould, Hribar, and Nocedal [7] show how the CG method can still be used when solving quadratic programming problems provided a preconditioner of the form (1.2) is employed. We extend their method to the general problem of solving systems of the form (1.1) where a constraint preconditioner of the form (1.2) is also used in sections 2 and 3. In section 4 we introduce a new factorization for the preconditioner $\mathcal{P}$. This previously unpublished factorization was recently developed by Wil Schilders (Tech. Univ. Eindhoven/Philips) [11]. We shall refer to this as the Schilders factorization. In section 5 we give numerical examples to demonstrate the effectiveness of the use of this method. A related but different factorization is the variable-reduction method [4, Chapter 5].

2. CG method for the reduced system. We wish to solve the system (1.1) to find $[x^{*T}y^{*T}]^T$. Let Z be an $n \times (n-m)$ matrix spanning the nullspace of B; then $BZ = 0$. The columns of B^T together with the columns of Z span $\mathbb{R}^n$ and any solution x^* of linear equations $Bx = d$ can be written as

$$x^* = B^T x^*_{_B} + Z x^*_{_Z}. \tag{2.1}$$

Substituting this into (1.1) gives

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} B^T x^*_{_B} + Z x^*_{_Z} \\ y^* \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}. \tag{2.2}$$

Let us split the matrix $\mathcal{A}$ into a block 3×3 structure where each corner block is of dimension $m \times m$. We can also expand the vector $[x^{*T}y^{*T}]^T$ into a matrix-vector product, $\mathcal{H}\rho^*$. Let $Z = [Z_1^T Z_2^T]^T$. Expression (2.2) then becomes

$$\begin{bmatrix} A_{1,1} & A_{1,2} & B_1^T \\ A_{2,1} & A_{2,2} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix} \underbrace{\begin{bmatrix} B_1^T & Z_1 & 0 \\ B_2^T & Z_2 & 0 \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{H}} \underbrace{\begin{bmatrix} x^*_{_B} \\ x^*_{_Z} \\ y^* \end{bmatrix}}_{\rho^*} = \begin{bmatrix} c_1 \\ c_2 \\ d \end{bmatrix}. \tag{2.3}$$

To maintain symmetry of our system we premultiply (2.3) by $\mathcal{H}^T$. Multiplying the matrix expression $\mathcal{H}^T\mathcal{A}\mathcal{H}$ and simplifying, we obtain the linear system

$$\begin{bmatrix} BAB^T & BAZ & BB^T \\ Z^TAB^T & Z^TAZ & 0 \\ BB^T & 0 & 0 \end{bmatrix} \begin{bmatrix} x^*_{_B} \\ x^*_{_Z} \\ y^* \end{bmatrix} = \begin{bmatrix} Bc \\ Z^Tc \\ d \end{bmatrix}. \tag{2.4}$$

We observe that an $m \times m$ system determines x_B^*:

$$BB^T x_B^* = d. \tag{2.5}$$

Since B is of full rank, BB^T is symmetric, positive definite. We could therefore solve this system using the Cholesky factorization of BB^T if the dimension of the system is small enough [5, p. 143]. From (2.4), having found x_B^*, we can find x_Z^* by solving

$$A_{ZZ} x_Z^* = -c_Z, \tag{2.6}$$

where

$$A_{ZZ} = Z^T A Z, \qquad c_Z = Z^T (A B^T x_B^* - c).$$

The matrix A is symmetric, positive definite in the nullspace of B; hence $Z^T A Z$ is symmetric, positive definite. Anticipating our technique, we can apply the CG method to compute an approximate solution to the system (2.6). Substituting this into (2.1) will give us an approximate solution for x^*. Using (2.4) with (2.1) we obtain a system that can be solved to give an approximate solution for y^*:

$$BB^T y^* = B(c - Ax^*). \tag{2.7}$$

If we used a Cholesky factorization to find x_B^*, then this same factorization could be employed to solve (2.7).

Let us consider the practical application of the CG method to the system (2.6). As we noted in section 1, the use of preconditioning can improve the rate of convergence of the CG iteration. Let us assume that a preconditioner W_{ZZ} is given, where W_{ZZ} is a symmetric, positive definite matrix of dimension $n - m$. Let us consider the class of preconditioners of the form $W_{ZZ} = Z^T G Z$, where G is a symmetric matrix such that $Z^T G Z$ is positive definite. The preconditioned CG method can then be applied to the $(n - m)$-dimensional reduced system $A_{ZZ} x_Z^* = -c_Z$ (see [7, Algorithm 2.1]).

3. CG method for the full system. Explicit use of Algorithm 2.1 from [7] would require knowledge of Z and solution of systems involving W_{ZZ}. The algorithm may, however, be written to make explicit use of the constraint preconditioner without the need for Z at all. Let us define the scaled projection matrix

$$P = Z(Z^T G Z)^{-1} Z^T. \tag{3.1}$$

We will later see that P is independent of the choice of nullspace basis Z.

ALGORITHM 3.1 (preconditioned CG in expanded form). [illegible] x [illegible] $Bx = d$ [illegible] $r = Ax - c$ $g = Pr$ [illegible] $p = -g$ [illegible]

$$\begin{aligned} x &\leftarrow x + \alpha p, \quad \alpha = r^T g / p^T A p, \\ r^+ &= r + \alpha A p, \\ g^+ &= P r^+, \\ p &\leftarrow -g^+ + \beta p, \quad \beta = (r^+)^T g^+ / r^T g, \\ g &\leftarrow g^+, \qquad r \leftarrow r^+. \end{aligned}$$

Note the definition of g^+ via the projection step $g^+ = Pr^+$. Following the terminology of [7], the vector g^+ will be called the [illegible]. It is defined to

be in the nullspace of B. We now wish to be able to apply the projection operator $Z(Z^TGZ)^{-1}Z^T$ without a representation of the nullspace basis Z. If G is nonsingular, then P can be expressed as

$$P = G^{-1}(I - B^T(BG^{-1}B^T)^{-1}BG^{-1}). \tag{3.2}$$

We can find g^+ by solving the system

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix} \tag{3.3}$$

whenever $z^TGz \neq 0$ for all nonzero z for which $Bz = 0$ [4, section 5.4.1]. The idea in Algorithm 3.2 below is to replace the projection step with the solution of (3.3) to define the same g^+: this is why a constraint preconditioner of the form (1.2) is needed. A preconditioner which is not of constraint form would not suffice.

Discrepancy in the magnitudes of g^+ and r^+ can cause numerical difficulties for which Gould, Hribar, and Nocedal [7] suggest using a residual update strategy that redefines r^+ so that its norm is closer to that of g^+. This dramatically reduces the roundoff errors in the projection operation in practice.

ALGORITHM 3.2 (preconditioned CG with residual update (PCG)).

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} g \\ v \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix},$$

$y = v$ $r \leftarrow r - B^Ty$

$g^+ = Pr^+$ (3.3)

$r \leftarrow r^+$ $r \leftarrow r^+ - B^Tv^+$

The key point is that Algorithm 3.2 does not require the computation of any nullspace basis but is a CG procedure for a symmetric, positive definite system which yields the solution of the indefinite system (1.1).

We would therefore like to be able to solve the systems of the form (3.3) efficiently. If we were to use Gaussian elimination to solve this system then, in general, the number of flops required would be $\mathcal{O}((n+m)^3)$ [5, p. 98]. Although this may be reduced depending on the sparsity, it could be impractical for large systems. In the following section we introduce a different factorization for matrices of the form $\mathcal{P}$ in (1.2). If $2m \leq n$, then we can use this factorization to solve (3.3) in $\mathcal{O}((n-m)^3)$ flops. If $2m > n$, then the number of flops required is $\mathcal{O}(m^3)$.

4. Schilders' factorization for preconditioning step. The preconditioning step of solving (3.3) in Algorithm 3.2 involves a constraint preconditioner of the same form of $\mathcal{P}$ given in (1.2). Let us split $\mathcal{P}$ into a block 3×3 structure as we did for the matrix $\mathcal{A}$ in (2.3). Suppose we choose matrices $L_1 \in \mathbb{R}^{m\times m}$ and $L_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ such that L_2 is nonsingular and assume that B_1 is nonsingular; then we can factorize $\mathcal{P}$ in the following manner:

$$\mathcal{P} = \underbrace{\begin{bmatrix} {B_1}^T & 0 & L_1 \\ {B_2}^T & {L_2} & E \\ 0 & 0 & I \end{bmatrix}}_{\mathcal{P}_1} \underbrace{\begin{bmatrix} D_1 & 0 & I \\ 0 & D_2 & 0 \\ I & 0 & 0 \end{bmatrix}}_{\mathcal{P}_2} \underbrace{\begin{bmatrix} B_1 & B_2 & 0 \\ 0 & {L_2}^T & 0 \\ {L_1}^T & E^T & I \end{bmatrix}}_{\mathcal{P}_3}, \tag{4.1}$$

where

$$D_1 = {B_1}^{-T} G_{1,1} {B_1}^{-1} - {L_1}^T {B_1}^{-1} - {B_1}^{-T} L_1, \tag{4.2}$$

$$D_2 = {L_2}^{-1} (G_{2,2} - {B_2}^T D_1 B_2 - E B_2 - {B_2}^T E^T) {L_2}^{-T}, \tag{4.3}$$

$$E = G_{2,1} {B_1}^{-1} - {B_2}^T D_1 - {B_2}^T {L_1}^T {B_1}^{-1}. \tag{4.4}$$

We should emphasize that ensuring that B_1 is nonsingular by permutation of columns, if necessary, is an important and nontrivial practical issue.

Equations (4.2), (4.3), and (4.4) are such that we can choose the matrix G and use this to define D_1, D_2, and E, or we could choose matrices D_1, D_2, and E that then define the matrix G. If we choose to use the second option, then we need to make sure that G satisfies several criteria set out in sections 2 and 3. These are

- G is nonsingular, and
- $Z^T G Z$ is symmetric, positive definite.

Since $\mathcal{P}_1 (= \mathcal{P}_3^T)$ is nonsingular, G will certainly satisfy these criteria if $\mathcal{P}_2$ is symmetric, positive definite. This is equivalent to the simple condition that D_2 is symmetric, positive definite.

In using the factorization (4.1) to solve (3.3), we take $L_2 = I$ for simplicity. We then need to calculate the `lu` factorization of $B_1 \in \mathbb{R}^{m \times m}$ and the Cholesky factorization of $D_2 \in \mathbb{R}^{(n-m) \times (n-m)}$ once and then use these factored forms in each iteration. Note that we need only choose D_2 to be symmetric, positive definite, so we can effectively choose how much work is involved in the factorization of D_2: the choice $D_2 = I$ will give a less effective preconditioner but renders this Cholesky factorization trivial. If i iterations are carried out in Algorithm 3.2, then there are $i + 1$ solves carried out using the factorization (4.1). For general B, D_1, D_2, E, L_1, and L_2, the total number of flops required by PCG is given by

$$no.\quad flops. \sim \frac{2}{3}(2(n-m)^3 + m^3) + 2(i+1)(13m^2 + 3n^2).$$

However, if D_2 and L_2 are diagonal, then the number of flops required reduces to

$$no.\quad flops. \sim \frac{2}{3} m^3 + 4m(i+1)(5m + 3n).$$

We would like to balance this reduction in the number of flops with the possible increase in the number of iterations carried out.

Keller, Gould, and Wathen give a bound on the dimension of the Krylov subspace $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$. Proof of the following theorem can be found in [8].

THEOREM 4.1. [illegible] $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$ [illegible]

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix},$$

[illegible] $A \in \mathbb{R}^{n \times n}$ [illegible] $B \in \mathbb{R}^{m \times n}$ [illegible] Z [illegible] $n \times (n-m)$ [illegible] B [illegible] $\mathcal{A}$ [illegible]

$$\mathcal{P} = \begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix},$$

[illegible] $G \in \mathbb{R}^{n \times n}$ [illegible] $G \neq A$ [illegible] $B \in \mathbb{R}^{m \times n}$ [illegible]

- $\mathcal{P}^{-1}\mathcal{A}$
  1. 1 $2m$
  2. $n-m$ λ $Z^TAZx_z = \lambda Z^TGZx_z$
- $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ $n-m+2$

Let us consider what happens if $G_{1,1} = A_{1,1}$, $G_{1,2} = {G_{2,1}}^T = A_{1,2}$, and we choose some matrix D_2 which is symmetric, positive definite. Equation (4.3) can be rearranged to give an explicit expression for $G_{2,2}$: this is given in Theorem 4.2. We assume that $G_{2,2} \neq A_{2,2}$. By choosing D_2 and using the factorization (4.1) to solve (3.3), we will never have to explicitly form the matrix $\mathcal{P}$.

Theorem 4.1 reveals that the preconditioned system $\mathcal{P}^{-1}\mathcal{A}$ has an eigenvalue at 1 with multiplicity $2m$, and $n-m$ eigenvalues which are defined by the generalized eigenvalue problem $Z^TAZx_z = \lambda Z^TGZx_z$. Noting that B_1 is nonsingular, we observe that $Z = [-{B_2}^T{B_1}^{-T} \;\; I]^T{L_2}^{-T}$ is an $n \times (n-m)$ basis for the nullspace of B. Using the above values of $G_{1,1}$, $G_{1,2}$, $G_{2,1}$, and $G_{2,2}$, it is straightforward to show that the $n-m$ eigenvalues can be defined by

$$\widehat{D}_2 x_z = \lambda D_2 x_z, \tag{4.5}$$

where

$$\widehat{D}_2 = {L_2}^{-1}(A_{2,2} - {B_2}^T D_1 B_2 - EB_2 - {B_2}^T E^T){L_2}^{-T}. \tag{4.6}$$

In the optimization community, $\widehat{D}_2$ is called the reduced Hessian.

Note that both D_2 and $\widehat{D}_2$ are symmetric, positive definite, so the eigenvalues are all real. Using the preconditioning matrix $\mathcal{P}$ with $G = [A_{1,1} \;\; A_{1,2};\; A_{2,1} \;\; G_{2,2}]$, we can improve on the Krylov subspace dimension given in Theorem 4.1.

THEOREM 4.2. $\mathcal{A} \in \mathbb{R}^{(n+m)\times(n+m)}$

$$\mathcal{A} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix},$$

$A \in \mathbb{R}^{n\times n}$ $B \in \mathbb{R}^{m\times n}$ m $A = [A_{1,1} \;\; A_{1,2};\; A_{2,1} \;\; A_{2,2}]$ $B = [B_1 \;\; B_2]$ $A_{1,1}, B_1 \in \mathbb{R}^{m\times m}$ $A_{1,2} \in \mathbb{R}^{m\times(n-m)}$ $A_{2,1}, B_2 \in \mathbb{R}^{(n-m)\times m}$ $A_{2,2} \in \mathbb{R}^{(n-m)\times(n-m)}$ $m < n$ $\mathcal{A}$ $L_1 \in \mathbb{R}^{m\times m}$ $L_2, D_2 \in \mathbb{R}^{(n-m)\times(n-m)}$ L_2 D_2 $\mathcal{A}$

$$\mathcal{P} = \begin{bmatrix} A_{1,1} & A_{1,2} & B_1^T \\ A_{2,1} & G_{2,2} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix},$$

$$G_{2,2} = L_2 D_2 {L_2}^T - {B_2}^T {B_1}^{-T} A_{1,1} {B_1}^{-1} B_2 + A_{2,1} {B_1}^{-1} B_2 + {B_2}^T {B_1}^{-T} A_{1,2};$$

$\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ $n-m+1$

. By assumption, B_1 is nonsingular so we can use Schilders' factorization (4.1) to express both $\mathcal{A}$ and $\mathcal{P}$. Using these factorizations we can express $\mathcal{P}^{-1}\mathcal{A}$ as

$$\mathcal{P}^{-1}\mathcal{A} = \begin{bmatrix} I & \Theta & 0 \\ 0 & {L_2}^{-T} {D_2}^{-1} \widehat{D}_2 {L_2}^T & 0 \\ 0 & \Upsilon & I \end{bmatrix}, \tag{4.7}$$

where D_1, $\widehat{D}_2$, and E are given by (4.2), (4.6), and (4.4), respectively, and

$$\Theta = {B_1}^{-1} B_2 (I - {L_2}^{-T} {D_2}^{-1} \widehat{D}_2 {L_2}^T), \tag{4.8}$$

$$\Upsilon = (E^T - {L_1}^T {B_1}^{-1} B_2)(I - {L_2}^{-T} {D_2}^{-1} \widehat{D}_2 {L_2}^T). \tag{4.9}$$

From the eigenvalues of the preconditioned system, it is evident that the characteristic polynomial of the preconditioned system is of the form

$$(\mathcal{P}^{-1}\mathcal{A} - I)^{2m} \prod_{i=1}^{n-m} (\mathcal{P}^{-1}\mathcal{A} - \lambda_i I).$$

To prove the upper bound on the dimension of the Krylov subspace we need to show that the degree of the minimum polynomial is less than or equal to $n - m + 1$. Expanding the polynomial $\prod_{i=1}^{n-m}(\mathcal{P}^{-1}\mathcal{A} - \lambda_i I)$ of degree $n - m$, we obtain a matrix of the form

$$\begin{bmatrix} \prod_{i=1}^{n-m}(1-\lambda_i) I & f_{n-m}(\Theta) & 0 \\ 0 & \prod_{i=1}^{n-m}(S - \lambda_i I) & 0 \\ 0 & f_{n-m}(\Upsilon) & \prod_{i=1}^{n-m}(1-\lambda_i) I \end{bmatrix}, \tag{4.10}$$

where $S = {L_2}^{-T} {D_2}^{-1} \widehat{D}_2 {L_2}^T$ and $f_{n-m}(\cdot)$ is defined by the recursive formula

$$f_{n-m}(\Gamma) = \Gamma \prod_{i=1}^{n-m-1} (1-\lambda_i) I + f_{n-m-1}(\Gamma)(S - \lambda_{n-m} I)$$

with base case $f_1(\Gamma) = \Gamma$. Now let us premultiply the matrix (4.10) by $\mathcal{P}^{-1}\mathcal{A} - I$ to give

$$\begin{bmatrix} 0 & \Theta \prod_{i=1}^{n-m}(S - \lambda_i I) & 0 \\ 0 & (S - I) \prod_{i=1}^{n-m}(S - \lambda_i I) & 0 \\ 0 & \Upsilon \prod_{i=1}^{n-m}(S - \lambda_i I) & 0 \end{bmatrix}. \tag{4.11}$$

We note that the $(1,2)$, $(2,2)$, and $(3,2)$ entries are in fact zero, since the λ_i $(i = 1, \ldots, n-m)$ are eigenvalues of S, which is similar to a symmetric matrix and thus diagonalizable. Hence, the degree of the minimum polynomial of $\mathcal{P}^{-1}\mathcal{A}$ is less than or equal to $n - m + 1$. ∎

If at least one of the eigenvalues defined by (4.5) has multiplicity greater than one, then we can sharpen this bound.

THEOREM 4.3. [illegible] $\mathcal{A}$ [illegible] $\mathcal{P}$ [illegible] 4.2 [illegible] $\alpha \in \mathbb{R}$ [illegible] $1 \le j < n - m$ [illegible] (4.5) [illegible]

$$\begin{aligned} \lambda_i &\neq \alpha, \quad i = 1, \ldots, j, \\ \lambda_i &= \alpha \quad i = j+1, \ldots, n-m; \end{aligned}$$

[illegible] $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ [illegible] $j + 2$ [illegible]. Following the proof of Theorem 4.2 we can show that

$$\prod_{i=1}^{j+1} (\mathcal{P}^{-1}\mathcal{A} - \lambda_i I) = \begin{bmatrix} \prod_{i=1}^{j+1}(1-\lambda_i) I & f_{j+1}(\Theta) & 0 \\ 0 & \prod_{i=1}^{j+1}(S - \lambda_i I) & 0 \\ 0 & f_{j+1}(\Upsilon) & \prod_{i=1}^{j+1}(1-\lambda_i) I \end{bmatrix}, \tag{4.12}$$

where Θ, Υ, S, and $f_{j+1}(\cdot)$ are as defined previously. Now let us premultiply the matrix (4.12) by $\mathcal{P}^{-1}\mathcal{A} - I$ to give

$$\begin{bmatrix} 0 & \Theta \prod_{i=1}^{j+1}(S-\lambda_i I) & 0 \\ 0 & (S-I)\prod_{i=1}^{j+1}(S-\lambda_i I) & 0 \\ 0 & \Upsilon \prod_{i=1}^{j+1}(S-\lambda_i I) & 0 \end{bmatrix}. \tag{4.13}$$

As in the proof of Theorem 4.2, we note that the $(1,2)$, $(2,2)$, and $(3,2)$ entries are in fact zero. Hence, the degree of the minimum polynomial of $\mathcal{P}^{-1}\mathcal{A}$ is less than or equal to $j+2$. ∎

THEOREM 4.4. [illegible] $\mathcal{A}$ [illegible] $\mathcal{P}$ [illegible] 4.2 [illegible] $A_{2,2}$ [illegible] $D_2 = L_2^{-1}A_{2,2}L_2^{-T}$ [illegible] $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ [illegible] $\min(2m+2, n-m+1)$ [illegible]. Let us expand the difference $\widehat{D_2} - D_2$:

$$\begin{aligned} \widehat{D_2} - D_2 &= L_2^{-1}(A_{2,2} + B_2^T B_1^{-T} A_{1,1} B_1^{-1} B_2 - A_{2,1}B_1^{-1}B_2 - B_2^T B_1^{-T} A_{1,2})L_2^{-T} \\ &\quad - L_2^{-1} A_{2,2} L_2^{-T} \\ &= L_2^{-1}(B_2^T B_1^{-T}(A_{1,1}B_1^{-1}B_2 - A_{1,2}) - A_{2,1}B_1^{-1}B_2)L_2^{-T}. \end{aligned}$$

Now,

$$\begin{aligned} rank(\widehat{D_2} - D_2) &\le \min\,[\min[rank(B_2), \min[rank(A_{1,1}), rank(B_2)] + rank(A_{1,2})] \\ &\quad + \min[rank(A_{2,1}), rank(B_2)], n-m] \\ &\le \min(2m, n-m). \end{aligned}$$

Since D_2 is, by assumption, positive definite, we may write $D_2 = WW^T$ for some nonsingular W. Thus

$$W^{-1}\widehat{D_2}W^{-T} = I + W^{-1}L_2^{-1}(B_2^T B_1^{-T}(A_{1,1}B_1^{-1}B_2 - A_{1,2}) - A_{2,1}B_1^{-1}B_2)L_2^{-T}W^{-T}$$

differs from the identity matrix by a matrix of rank at most $\min(2m, n-m)$, and hence there are at most $\min(2m, n-m)$ nonunit eigenvalues defined by the generalized eigenvalue problem (4.5). We can therefore arrange the eigenvalues such that there exists $1 \le j \le \min(2m, n-m)$ and

$$\begin{aligned} \lambda_i &\ne 1, \quad i = 1, \ldots, j, \\ \lambda_i &= 1, \quad i = j+1, \ldots, n-m. \end{aligned}$$

From Theorems 4.2 and 4.3 we conclude that the dimension of the Krylov subspace $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ is at most $\min(2m+2, n-m+1)$. ∎

5. Numerical examples. We apply Algorithm 3.2 to solve a selection of problems from the CUTEr collection [6], and also a set of Stokes saddle-point problems generated using the `ifiss` incompressible flow software associated with the book [2]. Any simple bounds in the CUTEr problems are removed to create a problem of the form

$$\begin{aligned} &\min_x f(x) = \tfrac{1}{2}x^T A x + s^T x \\ &\quad \text{subject to } Bx = t. \end{aligned}$$

For both sets of problems, the matrix $A \in \mathbb{R}^{n\times n}$ is symmetric and $B \in \mathbb{R}^{m\times n}$ is the full row rank matrix of linear constraints; A is positive definite in the nullspace

TABLE 5.1
Values of m, n, nonzeros in A, and nonzeros in B for the test problems.

Problem	m	n	nz(A)	nz(B)
CONT-050	2401	2597	2597	12005
CONT-101	10098	10197	2700	49599
CONT-300	90298	90597	90597	448799
CONT1-10	9801	10197	10197	49005
CVXQP1	5000	10000	69968	14998
CVXQP2	2500	10000	69968	7499
CVXQP3	7500	10000	69968	22497
DUAL1	1	85	7031	85
DUAL2	1	96	8920	96
DUAL3	1	111	12105	111
DUAL4	1	75	5523	75
KSIP	1021	1001	20	21002
MOSARQP1	3200	700	2590	4122
PRIMALC1	9	239	229	2079
PRIMALC2	7	238	230	1624
PRIMALC5	8	295	286	2304
PRIMALC8	8	528	519	4168
STOKES1	288	2178	28418	10452
STOKES2	1088	8450	122206	44306
STOKES3	769	5890	82582	30483
STOKES4	2945	23042	345234	126799
STOKES5	2112	5890	82582	30064
STOKES6	8448	23042	345234	129124
STOKES7	255	2178	28568	3826
STOKES8	1023	8450	122506	16771

of B (in fact on the whole space for the Stokes problems), and vectors x, s, and t have appropriate dimensions. A permuted `lu` factorization of B^T is used to find a permutation matrix, M, such that the first m columns of BM are linearly independent [5, Chapter 3]. We then set $B \leftarrow BM$, $A \leftarrow M^T AM$, $s \leftarrow M^T s$, and $x \leftarrow M^T x$. The assumption about the nonsingularity of B_1 will now hold.

The dimensions of the problems and number of nonzeros in the associated matrices A and B are given in Table 5.1. Tables 5.2–5.6 show the results of using Algorithm 3.2 with different preconditioners and a low tolerance. The results of using the preconditioner $\mathcal{P} = [I\ B^T; B\ 0]$ are shown in Table 5.2. This preconditioner is of the general form considered in [8] (see also [9]). Another possible choice for G is to use the diagonal entries of A [1]. The results of using this preconditioner are shown in Table 5.3. In Tables 5.4 and 5.5 we use preconditioners of the form considered in Theorems 4.2 and 4.4, respectively. The final preconditioner that we try will generally reproduce less of the matrix A in G than the previous two do, but we hope that this will be compensated by the iterations being "cheaper" to apply. The results are given in Table 5.6.

We compare the number of iterations and the amount of CPU time taken when we use an `lu` factorization of the preconditioner $\mathcal{P}$, and when we use the Schilders' factorization for $\mathcal{P}$. In both cases, we carry out the factorization once and then use the factored forms in each iteration. When producing Schilders' factorization, we set $L_1 = 0$ and $L_2 = I$. At present, we have not carried out substantial research into how we should choose the values of L_1 and L_2. In an attempt to maintain sparsity, an approximate minimum degree preordering strategy is applied before carrying out any `lu` factorizations. Threshold pivoting is also used during the `lu` factorization stages: the default threshold value in MATLAB is used.

TABLE 5.2
Solution statistics: Comparisons between iterative PCG solvers when using `lu` *and Schilders' factorizations in the preconditioning step with* $tol = 10^{-2}$, $G = I$.

	LU				Schilders'				
Problem	Its	FTime	ITime	Total	Its	FTime	ITime	Total	Digits
CONT-050	1	1.18	0.76	2.03	1	6.95	0.45	7.49	13
CONT-101	1	21.43	6.53	28.73	1	123.38	2.91	127.06	9
CONT-300	—	—	—	—	—	—	—	—	—
CONT1-10	1	20.59	6.42	27.80	1	128.72	2.89	132.41	13
CVXQP1	2	1.93	1.36	3.40	2	74.49	2.16	76.75	14
CVXQP2	2	0.04	0.11	0.17	2	0.10	0.13	0.24	15
CVXQP3	2	3.19	3.10	6.63	2	23.96	1.02	25.32	15
DUAL1	5	0.00	0.00	0.00	5	0.01	0.01	0.01	15
DUAL2	4	0.00	0.00	0.01	4	0.01	0.01	0.02	13
DUAL3	5	0.00	0.01	0.01	5	0.01	0.01	0.02	14
DUAL4	4	0.00	0.00	0.00	4	0.01	0.00	0.01	15
KSIP	17	0.34	0.91	2.11	18	0.03	0.87	1.63	5
MOSARQP1	4	0.03	0.06	0.11	4	0.08	0.07	0.17	15
PRIMALC1	2	0.00	0.01	0.01	2	0.08	0.02	0.10	12
PRIMALC2	1	0.00	0.00	0.01	1	0.07	0.02	0.09	14
PRIMALC5	0	0.00	2.12	2.13	0	0.16	22.07	22.23	15
PRIMALC8	2	0.01	7.76	7.77	2	0.61	153.51	154.13	14
STOKES1	7	0.05	0.07	0.13	7	15.04	3.68	18.73	14
STOKES2	8	0.27	0.34	0.69	8	932.91	116.33	1049.33	13
STOKES3	2	0.26	0.12	0.44	2	280.59	16.54	297.18	11
STOKES4	2	0.95	0.56	1.83	—	—	—	—	—
STOKES5	2	0.20	0.26	0.62	2	69.71	6.01	75.87	12
STOKES6	2	1.29	1.54	3.84	—	—	—	—	—
STOKES7	9	0.02	0.07	0.10	9	10.48	0.32	10.80	14
STOKES8	10	0.09	0.40	0.51	10	662.66	5.04	667.73	13

	LU			Schilders'		
Problem	nz($L(\mathcal{P})$)	nz($U(\mathcal{P})$)	Total	nz($L(D_2)$)	nz($U(D_2)$)	Total
CONT-050	630887	607170	1238057	19306	19306	155305
CONT-101	5263734	6407090	11670824	4950	4950	700946
CONT-300	—	—	—	—	—	—
CONT1-10	5195636	6075386	11271022	78606	78606	833857
CVXQP1	708449	801586	1510035	1229895	1241503	2519154
CVXQP2	22168	25467	47635	17038	17038	46791
CVXQP3	1682000	1267180	2949180	257477	259336	578439
DUAL1	171	171	342	3570	3570	7227
DUAL2	193	193	386	4560	4560	9218
DUAL3	223	223	446	6105	6105	12323
DUAL4	151	151	302	2775	2775	5627
KSIP	23214	23214	46428	210	210	23424
MOSARQP1	24109	26831	50940	32887	32887	71296
PRIMALC1	2363	4424	6787	26565	26565	55227
PRIMALC2	1890	3500	5390	26796	26796	55230
PRIMALC5	2635	4923	7558	41321	41328	84969
PRIMALC8	4732	8884	13616	135460	135460	275104
STOKES1	21079	21079	42158	1335628	1334125	2683050
STOKES2	105503	105503	211006	23447464	23461548	46971065
STOKES3	63029	63029	126058	10908746	10877987	21825155
STOKES4	326595	326595	653190	—	—	—
STOKES5	135816	136414	272230	4010012	4050659	8095603
STOKES6	746422	745095	1491517	—	—	—
STOKES7	9155	9154	18309	1036995	1033457	2074914
STOKES8	44741	44741	89482	20095193	20083780	40198321

TABLE 5.3
Solution statistics: Comparisons between iterative PCG solvers when using `lu` *and Schilders' factorizations in the preconditioning step with* $tol = 10^{-2}$, $G = \mathrm{diag}(A)$.

	LU				Schilders'				
Problem	Its	FTime	ITime	Total	Its	FTime	ITime	Total	Digits
CONT-050	1	1.28	0.74	2.12	1	6.98	0.45	7.53	13
CONT-101	1	10.38	5.17	16.33	1	32.47	2.84	36.09	10
CONT-300	—	—	—	—	—	—	—	—	—
CONT1-10	1	18.70	6.28	25.78	1	128.01	2.92	131.71	13
CVXQP1	1	1.61	0.93	2.64	1	74.61	1.47	76.18	11
CVXQP2	1	0.04	0.08	0.14	1	0.10	0.10	0.22	15
CVXQP3	1	3.45	2.40	6.20	1	23.14	0.80	24.28	13
DUAL1	5	0.00	0.00	0.00	5	0.01	0.00	0.01	12
DUAL2	4	0.00	0.00	0.01	4	0.01	0.01	0.02	14
DUAL3	5	0.00	0.01	0.01	5	0.01	0.01	0.02	13
DUAL4	4	0.00	0.00	0.00	4	0.01	0.00	0.01	13
KSIP	1	0.34	0.73	1.92	1	0.01	0.73	1.46	15
MOSARQP1	2	0.02	0.03	0.07	2	0.02	0.04	0.08	15
PRIMALC1	0	0.00	4.52	4.52	0	0.00	1.15	1.16	15
PRIMALC2	0	0.00	4.23	4.24	0	0.00	1.00	1.01	15
PRIMALC5	0	0.00	6.87	6.88	0	0.00	1.63	1.63	15
PRIMALC8	0	0.01	28.54	28.55	0	0.00	5.15	5.15	15
STOKES1	5	0.05	0.05	0.12	5	14.79	3.75	18.56	14
STOKES2	5	0.27	0.25	0.61	5	926.89	77.75	1004.72	15
STOKES3	2	0.34	0.12	0.54	2	284.06	17.13	301.24	12
STOKES4	2	0.88	0.55	1.74	—	—	—	—	—
STOKES5	2	0.21	0.27	0.63	2	67.73	6.54	74.43	12
STOKES6	2	1.30	1.55	3.85	—	—	—	—	—
STOKES7	4	0.02	0.04	0.06	4	10.51	0.16	10.67	15
STOKES8	4	0.09	0.19	0.31	4	663.26	2.35	665.64	13

	LU			Schilders'		
Problem	nz($L(\mathcal{P})$)	nz($U(\mathcal{P})$)	Total	nz($L(D_2)$)	nz($U(D_2)$)	Total
CONT-050	620089	639618	1259707	19306	19306	155305
CONT-101	4255279	2928002	7183281	4950	4950	693449
CONT-300	—	—	—	—	—	—
CONT1-10	5135085	5655789	10790874	78606	78606	833857
CVXQP1	723131	577861	1300992	1228384	1232830	2508970
CVXQP2	21880	23198	45078	16921	16713	46349
CVXQP3	1901464	1069000	2970464	257803	259757	579186
DUAL1	171	171	342	3570	3570	7227
DUAL2	193	193	386	4560	4560	9218
DUAL3	223	223	446	6105	6105	12323
DUAL4	151	151	302	2775	2775	5627
KSIP	22022	22024	44046	210	210	22424
MOSARQP1	7289	7380	14669	5145	5145	15462
PRIMALC1	248	4387	4635	230	229	2547
PRIMALC2	245	3478	3723	231	230	2092
PRIMALC5	303	4894	5197	287	286	2885
PRIMALC8	536	8855	9391	520	519	5215
STOKES1	21079	21079	42158	1335797	1334073	2683167
STOKES2	105503	105503	211006	23447428	23461508	46970989
STOKES3	63029	63029	126058	10908667	10878627	21825716
STOKES4	326595	326595	653190	—	—	—
STOKES5	135821	136395	272216	3972246	4017853	8025031
STOKES6	747054	745028	1492082	—	—	—
STOKES7	9155	9154	18309	1039584	1035774	2079820
STOKES8	44741	44741	89482	20085901	20077317	40182566

TABLE 5.4

Solution statistics: Comparisons between iterative PCG solvers when using `lu` *and Schilders' factorizations in the preconditioning step with tol* $= 10^{-2}$, $G_{1,1} = A_{1,1}$, $G_{1,2} = A_{1,2}$, *and* $D_2 = \mathrm{diag}(\widehat{D}_2)$.

	LU				Schilders'				
Problem	Its	FTime	ITime	Total	Its	FTime	ITime	Total	Digits
CONT-050	1	9.63	0.66	10.39	1	6.98	0.44	7.51	12
CONT-101	1	43.41	4.52	48.69	1	32.57	2.87	36.22	12
CONT-300	—	—	—	—	—	—	—	—	—
CONT1-10	1	159.98	5.70	166.48	1	128.90	2.91	132.61	12
CVXQP1	12	743.04	245.24	988.39	11	33.01	1.29	34.40	11
CVXQP2	3	125.36	22.92	148.29	3	0.56	0.15	0.72	14
CVXQP3	5	644.13	105.32	749.79	5	24.08	1.07	25.50	6
DUAL1	6	0.01	0.01	0.01	6	0.00	0.01	0.01	8
DUAL2	5	0.02	0.01	0.03	5	0.01	0.01	0.01	8
DUAL3	6	0.03	0.02	0.04	6	0.01	0.01	0.02	8
DUAL4	4	0.01	0.00	0.02	4	0.01	0.00	0.01	9
KSIP	2	0.08	0.73	1.53	2	0.01	0.74	1.70	11
MOSARQP1	2	0.05	0.04	0.11	2	0.02	0.04	0.07	15
PRIMALC1	0	0.06	4.54	4.60	0	0.00	1.17	1.17	15
PRIMALC2	0	0.00	4.22	4.23	0	0.00	1.01	1.01	15
PRIMALC5	0	0.01	6.84	6.85	0	0.00	1.62	1.62	15
PRIMALC8	0	0.01	28.59	28.60	0	0.00	5.15	5.16	15
STOKES1	52	25.97	50.53	76.51	52	4.09	0.58	4.68	2
STOKES2	—	—	—	—	187	237.68	11.64	249.41	—
STOKES3	45	453.50	374.39	827.94	46	36.96	1.61	38.65	2
STOKES4	—	—	—	—	—	—	—	—	—
STOKES5	35	182.53	179.95	362.63	34	8.40	1.53	10.08	3
STOKES6	—	—	—	—	—	—	—	—	—
STOKES7	8	15.63	6.25	21.89	8	1.47	0.06	1.53	14
STOKES8	11	943.87	167.57	1111.47	11	27.20	0.36	27.58	13

	LU			Schilders'		
Problem	$\mathrm{nz}(L(\mathcal{P}))$	$\mathrm{nz}(U(\mathcal{P}))$	Total	$\mathrm{nz}(L(D_2))$	$\mathrm{nz}(U(D_2))$	Total
CONT-050	595768	649016	1244784	196	196	117085
CONT-101	3681821	2776506	6458327	99	99	683747
CONT-300	—	—	—	—	—	—
CONT1-10	4867901	5674469	10542370	396	396	677437
CVXQP1	25787124	39299255	65086379	5000	5000	91004
CVXQP2	8243264	10825817	19069081	7500	7500	43683
CVXQP3	24645462	44499396	69144858	2500	2500	113266
DUAL1	3604	3634	7238	84	84	337
DUAL2	4579	4544	9123	95	95	377
DUAL3	6140	6103	12243	110	110	441
DUAL4	2882	2805	5687	74	74	297
KSIP	22024	22024	44048	20	20	22044
MOSARQP1	19264	19276	38540	2500	2500	10181
PRIMALC1	248	4387	4635	230	229	2547
PRIMALC2	245	3478	3723	231	230	2092
PRIMALC5	303	4894	5197	287	286	2885
PRIMALC8	536	8855	9391	520	519	5215
STOKES1	1684985	2104142	3789127	1890	1890	19875
STOKES2	—	—	—	7362	7362	87160
STOKES3	13281099	16195569	29476668	5121	5121	55944
STOKES4	—	—	—	—	—	—
STOKES5	7621013	13040444	20661457	3778	3778	62434
STOKES6	—	—	—	—	—	—
STOKES7	1308824	1706168	3014992	1923	1923	11678
STOKES8	23764396	31623689	55388085	7427	7427	48126

TABLE 5.5

Solution statistics: Comparisons between iterative PCG solvers when using `lu` *and Schilders' factorizations in the preconditioning step with tol* $= 10^{-2}$, $G_{1,1} = A_{1,1}$, $G_{1,2} = G_{2,1}^T = A_{1,2}$, *and* $D_2 = A_{2,2}$.

	LU				Schilders'				
Problem	Its	FTime	ITime	Total	Its	FTime	ITime	Total	Digits
CONT-050	1	8.35	0.66	9.12	1	0.06	0.45	0.60	11
CONT-101	1	41.44	4.50	46.71	1	0.45	2.84	4.06	6
CONT-300	—	—	—	—	1	14.48	60.19	98.69	—
CONT1-10	1	147.41	5.66	153.84	1	0.44	2.89	4.11	12
CVXQP1	401	952.69	9531.04	10483.85	134	0.09	15.05	15.24	3
CVXQP2	0	226.36	10.12	236.50	1	1.80	1.50	3.32	0
CVXQP3	29	587.99	522.23	1110.56	18	0.08	2.77	3.19	1
DUAL1	1	0.01	0.00	0.01	1	0.00	0.00	0.01	11
DUAL2	1	0.02	0.00	0.02	1	0.01	0.00	0.01	12
DUAL3	2	0.02	0.01	0.03	2	0.01	0.00	0.01	9
DUAL4	2	0.01	0.00	0.01	2	0.00	0.00	0.01	10
KSIP	0	0.07	6.60	7.40	0	0.00	1.59	2.31	15
MOSARQP1	0	0.04	172.38	172.44	0	0.01	142.10	142.13	15
PRIMALC1	0	0.01	4.24	4.25	0	0.00	1.14	1.15	15
PRIMALC2	0	0.00	4.00	4.01	0	0.00	1.00	1.00	15
PRIMALC5	0	0.01	6.55	6.55	0	0.00	1.71	1.71	15
PRIMALC8	0	0.01	28.06	28.07	0	0.00	5.10	5.10	15
STOKES1	91	23.21	91.12	114.35	92	0.05	1.22	1.28	2
STOKES2	—	—	—	—	560	0.31	39.42	39.82	—
STOKES3	79	450.28	688.39	1138.72	75	0.17	3.17	3.39	3
STOKES4	—	—	—	—	234	1.02	48.17	49.47	—
STOKES5	70	174.76	364.39	539.31	63	0.13	3.03	3.31	3
STOKES6	—	—	—	—	136	0.79	28.75	30.59	—
STOKES7	14	14.35	11.17	25.53	14	0.05	0.14	0.20	11
STOKES8	28	834.47	434.78	1269.28	28	0.28	1.25	1.55	7

	LU			Schilders'		
Problem	nz($L(\mathcal{P})$)	nz($U(\mathcal{P})$)	Total	nz($L(D_2)$)	nz($U(D_2)$)	Total
CONT-050	595768	648968	1244736	196	196	117085
CONT-101	3690797	2775736	6466533	99	99	683747
CONT-300	—	—	—	299	299	10794780
CONT1-10	4867901	5800598	10668499	396	396	677437
CVXQP1	31909093	44477738	76386831	23310	23734	128048
CVXQP2	14908190	16628011	31536201	847610	572256	1448549
CVXQP3	24497857	41965765	66463622	4926	4926	118118
DUAL1	3739	3739	7478	3568	3568	7305
DUAL2	4747	4747	9494	4554	4554	9295
DUAL3	6326	6326	12652	6104	6104	12429
DUAL4	2924	2924	5848	2775	2775	5699
KSIP	22024	22023	44047	20	19	22043
MOSARQP1	9821	8304	18125	2536	2186	9903
PRIMALC1	248	4387	4635	230	229	2547
PRIMALC2	245	3478	3723	231	230	2092
PRIMALC5	303	4894	5197	287	286	2885
PRIMALC8	536	8855	9391	520	519	5215
STOKES1	1684987	2104662	3789649	28475	28475	73045
STOKES2	—	—	—	190490	190490	453416
STOKES3	13362756	16195611	29558367	99430	99430	244562
STOKES4	—	—	—	620644	620715	1439851
STOKES5	7617212	13057170	20674382	79728	79728	214334
STOKES6	—	—	—	439216	452489	1122908
STOKES7	1314702	1707861	3022563	29666	29666	67164
STOKES8	23715838	31936839	55652677	177137	177137	387546

TABLE 5.6

Solution statistics: Comparisons between iterative PCG solvers when using `lu` *and Schilders' factorizations in the preconditioning step with* $tol = 10^{-2}$, $G_{1,1} = 0$, $G_{1,2} = 0$, *and* $D_2 = A_{2,2}$.

	LU				Schilders'				
Problem	Its	FTime	ITime	Total	Its	FTime	ITime	Total	Digits
CONT-050	1	0.72	0.38	1.20	1	0.06	0.43	0.59	11
CONT-101	1	5.39	2.50	8.66	1	0.46	2.85	4.09	10
CONT-300	1	324.00	98.27	446.28	1	14.59	59.78	98.46	14
CONT1-10	1	8.24	3.00	12.05	1	0.45	2.84	4.07	10
CVXQP1	119	29.27	330.01	359.39	104	0.09	10.20	10.39	2
CVXQP2	3	7.73	4.53	12.28	0	1.78	0.99	2.79	0
CVXQP3	16	7.21	18.37	25.91	16	0.07	2.11	2.52	4
DUAL1	1	0.01	0.00	0.01	1	0.00	0.00	0.01	15
DUAL2	1	0.01	0.00	0.02	1	0.01	0.00	0.01	15
DUAL3	2	0.02	0.00	0.02	2	0.01	0.00	0.01	14
DUAL4	2	0.01	0.00	0.01	2	0.00	0.00	0.01	12
KSIP	0	0.34	6.67	7.74	0	0.00	1.59	2.32	15
MOSARQP1	0	0.03	149.10	149.14	0	0.01	136.80	136.82	15
PRIMALC1	0	0.01	4.35	4.36	0	0.00	1.15	1.15	15
PRIMALC2	0	0.00	4.03	4.03	0	0.00	1.01	1.01	15
PRIMALC5	0	0.01	6.85	6.85	0	0.00	1.62	1.62	15
PRIMALC8	0	0.01	28.28	28.29	0	0.00	5.18	5.18	15
STOKES1	75	0.21	3.05	3.28	79	0.05	1.02	1.08	2
STOKES2	240	4.75	259.71	264.55	254	0.30	16.79	17.18	2
STOKES3	65	0.81	8.52	9.38	65	0.16	2.59	2.80	3
STOKES4	179	9.79	211.73	221.81	163	0.98	31.93	33.21	5
STOKES5	58	1.48	20.53	22.15	59	0.12	2.51	2.78	4
STOKES6	133	45.24	543.96	590.21	128	0.69	23.81	25.49	2
STOKES7	14	0.10	0.31	0.41	14	0.05	0.13	0.18	14
STOKES8	28	0.70	4.05	4.77	28	0.27	1.15	1.45	10

	LU			Schilders'		
Problem	nz($L(\mathcal{P})$)	nz($U(\mathcal{P})$)	Total	nz($L(D_2)$)	nz($U(D_2)$)	Total
CONT-050	293011	411916	704927	196	196	114684
CONT-101	1770989	2230310	4001299	99	99	681146
CONT-300	45690125	62556439	108246564	299	299	10704482
CONT1-10	2194946	3280161	5475107	396	396	667636
CVXQP1	4346056	3841931	8187987	23310	23734	89800
CVXQP2	1856579	1343030	3199609	847610	572256	1430081
CVXQP3	1831354	1842046	3673400	4926	4926	63978
DUAL1	3570	3738	7308	3568	3568	7222
DUAL2	4556	4746	9302	4554	4554	9205
DUAL3	6106	6326	12432	6104	6104	12320
DUAL4	2777	2925	5702	2775	2775	5626
KSIP	22022	22023	44045	20	19	22042
MOSARQP1	7325	7066	14391	2536	2186	9544
PRIMALC1	248	4387	4635	230	229	2547
PRIMALC2	245	3478	3723	231	230	2092
PRIMALC5	303	4894	5197	287	286	2885
PRIMALC8	536	8855	9391	520	519	5215
STOKES1	83694	181989	265683	28475	28475	69959
STOKES2	2329914	1548050	3877964	190490	190490	441945
STOKES3	258848	797838	1056686	99430	99430	236513
STOKES4	2486613	6207846	8694459	620644	620715	1409233
STOKES5	768367	542274	1310641	79728	79728	192276
STOKES6	8697371	6909772	15607143	439216	452489	1032170
STOKES7	50998	58625	109623	29666	29666	63539
STOKES8	340761	379503	720264	177137	177137	372599

TABLE 5.7

Solution statistics: Comparison of different preconditioners with the preferable factorization method and tol $= 10^{-6}$. (a) $G = I$, (b) $G = \mathrm{diag}(A)$, (c) $G_{1,1} = A_{1,1}$, $G_{1,2} = G_{2,1}^T = A_{1,2}$, *and* $D_2 = A_{2,2}$, (d) $G_{1,1} = 0$, $G_{1,2} = G_{2,1}^T = 0$, *and* $D_2 = A_{2,2}$.

	LU (a)		LU (b)		Schilders' (c)		Schilders' (d)	
Name	Its	Time	Its	Time	Its	Time	Its	Time
CONT-050	3	2.68	1	2.06	4	1.13	4	1.09
CONT-101	7	45.98	1	16.21	2	5.14	2	5.11
CONT-300	—	—	—	—	4	155.35	4	153.56
CONT1-10	1	27.66	1	26.17	4	7.36	4	7.12
CVXQP1	16	9.04	9	5.98	773	85.96	601	58.43
CVXQP2	18	0.72	9	0.41	1	3.44	0	2.81
CVXQP3	12	15.93	6	11.35	161	21.60	141	15.79
DUAL1	38	0.02	37	0.02	3	0.01	3	0.01
DUAL2	22	0.02	21	0.02	4	0.02	3	0.02
DUAL3	24	0.03	24	0.03	2	0.02	2	0.02
DUAL4	14	0.01	13	0.01	3	0.01	3	0.01
KSIP	20	2.13	1	1.93	0	2.33	0	2.28
MOSARQP1	21	0.28	5	0.08	0	142.22	0	136.41
PRIMALC1	6	1.34	0	4.36	0	1.16	0	1.14
PRIMALC2	2	1.27	0	4.02	0	1.01	0	1.00
PRIMALC5	0	2.13	0	6.59	0	1.64	0	1.64
PRIMALC8	2	7.88	0	28.52	0	5.17	0	5.10
STOKES1	36	0.32	32	0.30	298	4.08	257	3.21
STOKES2	67	2.50	57	2.21	3275	230.97	1575	104.44
STOKES3	23	0.90	21	0.84	397	16.66	330	13.08
STOKES4	38	5.06	34	4.66	2093	425.20	1460	283.37
STOKES5	23	1.45	21	1.36	333	15.56	309	12.42
STOKES6	37	10.90	35	10.38	1382	281.50	1261	226.31
STOKES7	35	0.28	23	0.20	45	0.49	45	0.44
STOKES8	77	2.91	38	1.49	116	5.33	115	4.73

The number of iterations carried out is given in the column headed "Its." We also measure the CPU time taken in carrying out the factorization of the preconditioners (FTime) and the CPU time then taken by Algorithm 3.2 when using these factored forms (ITime). The time spent finding an initial x which satisfies $Bx = d$ is excluded from the value given in Itime. The total CPU time used to solve the problem is given by "Total." Each problem is solved ten times and the mean of the CPU times recorded. The minimum values of $\frac{1}{2}x^T Ax + s^T x$ calculated are also compared for all the problems, although this is not immediately applicable to the Stokes problems. The number of digits of agreement in these values are given in "Digits." We observe that for the majority of the CUTEr problems, the differences in rounding errors of the two factorization methods do not greatly affect the calculated minimum values. We note that for some of the Stokes problems at least one of the solutions is rather inaccurate, though we have no easy way of finding out which! An important factor in how well a certain factorization will perform is its sparsity; in particular, the number of nonzeros in the `lu` factors of $\mathcal{P}$, and the number of nonzeros in the factors of D_2. These values are also given for each of our preconditioners, and the total amount of storage required for the various blocks required. Note that the matrices D_1 and E should not be generated when using Schilders' factorization.

The diagonal form of D_2 used in the examples of Table 5.4 produces a significant

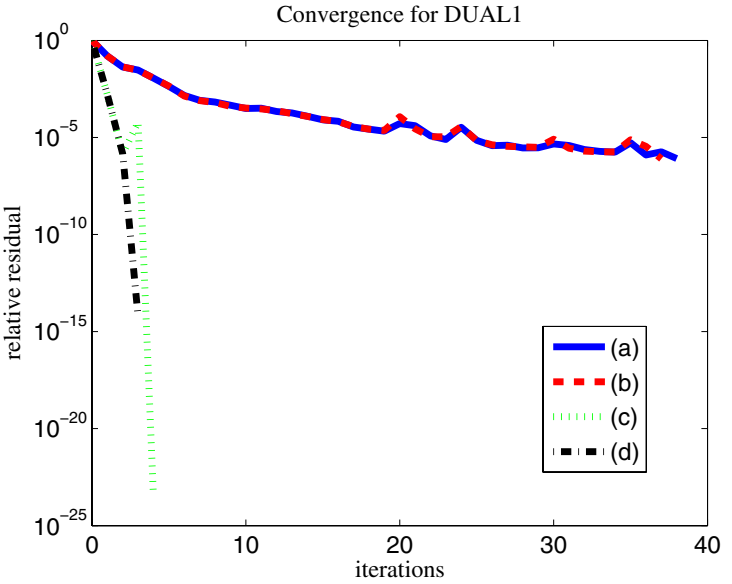


FIG. 5.1. *Convergence of relative residuals $r^T g$ with the different preconditioning strategies from Table* 5.7 *for the problem DUAL*1.

drop in the CPU times when we use the Schilders' factorization instead of the `lu` factorization of $\mathcal{P}$. However, this choice of D_2 is not so practical because we have to form the matrix $\widehat{D}_2$, as defined in (4.6), and then use the diagonal. If $A_{2,2}$ is positive definite and sparse enough for us to consider factorizing it, then we can set $D_2 = A_{2,2}$ with $G_{1,1} = A_{1,1}$ and $G_{1,2} = G_{2,1}^T = A_{1,2}$. From Theorem 4.4 we know that the dimension of the Krylov subspace $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ is at most $\min(n-m+1, 2m+2)$. The results of using such a preconditioner are shown in Table 5.5. In particular, note the four DUAL* problems. For each of these $m = 1$, so the upper bound on the dimension of the Krylov subspace $\mathcal{K}(\mathcal{P}^{-1}\mathcal{A}, b)$ is at most 4. Our results support this claim.

All tests were performed on a dual Intel Xeon 3.20GHz machine with hyper-threading and 2GB of RAM. It was running Fedora Core 2 (Linux kernel 2.6.8) and all programs were written in MATLAB 7.0. We terminate the iteration when the value of $r^T g$ is reduced by at least a factor of 10^{-2}. If no result is given for a problem, then the machine we used ran out of memory during the factorization stage. This is indicated by — in the results tables. Consider the test problem CONT-300. We were unable to solve it by choosing G explicitly, as in Tables 5.2 and 5.3, and then factoring it because of the machine running out of memory. However, when we used Schilders' factorization with $D_2 = A_{2,2}$, our algorithm successfully terminated, as in Tables 5.5 and 5.6. We were also able to factor $\mathcal{P}$ with an `lu` factorization for $G_{1,1} = 0$, $G_{1,2} = G_{2,1}^T = 0$, and $D_2 = A_{2,2}$, but the factorization time was over 300 times larger than using Schilders' factorization, resulting in the total CPU time being at least 38 times larger.

It is clear that, for the majority of the problems, factoring $\mathcal{P}$ with an `lu` factorization is preferable when we choose G as in Tables 5.2 and 5.3. However, it is generally preferable to use Schilders' factorization for the other choices of preconditioner, as we would expect. Table 5.7 shows results for the factorization of preference with four of the preconditioners that we have considered. The iteration process is terminated

when $r^T g$ has been reduced by a factor of 10^{-6}. The number of iterations required and the total CPU time is recorded. We observe that Schilders' factorization with $G_{1,1} = 0$, $G_{1,2} = 0$, and $D_2 = A_{2,2}$ is preferable, or of equal preference, for twelve out of the seventeen CUTEr test problems. However, setting $G = \mathrm{diag}(A)$ and then carrying out an `lu` factorization is better for the Stokes problems tested because the number of iterations required is greatly reduced. The convergence of the relative residuals of $r^T g$ are plotted in Figure 5.1 for the problem DUAL1. The line labeled "(c)" corresponds to a preconditioner which satisfies the assumptions of Theorem 4.4. The plot supports this theorem.

6. Conclusion. In this paper, we have discussed the use of the PCG method for solving indefinite linear systems. We have investigated the use of a new factorization for constraint preconditioners and shown how this can be used in incomplete forms to decrease the number of flops required overall by the preconditioned iterative method. We obtained an upper bound on the number of iterations required to solve systems of the form (1.1) by means of appropriate Krylov subspace methods by using a minimum polynomial argument.

We have provided computational evidence that the PCG method works well for nontrivial quadratic programming problems; however, it would not be the method of choice for the Stokes saddle-point problems tested. Indeed, problem-based preconditioned iterative methods which have optimal computational complexity exist for these problems (see [2, Chapter 6]).

In some cases, the use of Schilders' factorization to define a preconditioner can speed up the solution time by a factor of 5 compared to using the `lu` factorization of a preconditioner for which G has been chosen. For larger dimensional problems, we have shown an example where Schilders' factorization approach enables fast solution but the `lu` factorization with popular choices of preconditioner fails because of memory costs.

The cost of finding m linearly independent columns in the matrix B via a permuted `lu` factorization is often higher than we would like. We plan to investigate different strategies for this in the future, and how these may change the number of iterations required to solve the problems.

Acknowledgments. The authors would like to thank Nick Gould and Wil Schilders for their helpful input during the early stages of this research.

REFERENCES

[1] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Comput. Optim. Appl., 28 (2004), pp. 149–171.

[2] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics*, Oxford University Press, Oxford, UK, to appear.

[3] B. Fischer, *Polynomial Based Iteration Methods for Symmetric Linear Systems*, Adv. Numer. Math., John Wiley & Sons Ltd., Chichester, UK; B. G. Teubner, Stuttgart, Germany, 1996.

[4] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1981.

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Stud. Math. Sci., Johns Hopkins University Press, Baltimore, MD, 1996.

[6] N. I. M. Gould, D. Orban, and Ph. L. Toint, *CUTEr (and SifDec), a Constrained and Unconstrained Testing Environment, Revisited*, Tech. Report TR/PA/01/04, CERFACS, Toulouse, France, 2001.

[7] N. I. M. Gould, M. E. Hribar, and J. Nocedal, *On the solution of equality constrained quadratic programming problems arising in optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 1376–1395.
[8] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1300–1317.
[9] L. Lukšan and J. Vlček, *Indefinitely preconditioned inexact Newton method for large sparse equality constrained non-linear programming problems*, Numer. Linear Algebra Appl., 5 (1998), pp. 219–247.
[10] A. Quarteroni and A. Valli, *Numerical Approximation of Partial Differential Equations*, Springer Series in Computational Mathematics 23, Springer-Verlag, Berlin, 1994.
[11] W. Schilders, *private communication*, 2002.
[12] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge Monographs on Applied and Computationl Mathematics, Cambridge University Press, Cambridge, UK, 2003.
[13] S. J. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, PA, 1997.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1573–1593
© 2006 Society for Industrial and Applied Mathematics

ADAPTIVE MULTISCALE REDISTRIBUTION FOR VECTOR QUANTIZATION*

YAIR KOREN† AND IRAD YAVNEH†

Abstract. Vector quantization is a classical problem that appears in many fields. Unfortunately, the quantization problem is generally nonconvex, and therefore affords many local minima. The main problem is finding an initial approximation which is close to a "good" local minimum. Once such an approximation is found, the Lloyd–Max method may be used to reach a local minimum near it. In recent years, much improvement has been made with respect to reducing the computational costs of quantization algorithms, whereas the task of finding better initial approximations received somewhat less attention. We present a novel multiscale iterative scheme for the quantization problem. The scheme is based on redistributing the representation levels among aggregates of decision regions at changing scales. The rule governing the redistribution relies on the so-called point density function and on the number of representation levels in each aggregate. Our method focuses on achieving better local minima than those achieved by other contemporary methods such as LBG. When quantizing signals with sparse and patchy histograms, as may occur in color images, for example, the improvement in distortion relative to LBG may be arbitrarily large.

Key words. quantization, Lloyd–Max, LBG, point density

AMS subject classifications. 62H30, 65H10, 65K10, 65U05, 68U10

DOI. 10.1137/040607769

1. Introduction.

Quantization [11, 13] is the process of representing continuum with only a finite number of representatives or representing an initially rich amount of discrete data with a lesser amount of representatives. Rounding off real numbers to the nearest integer is a simple form of scalar quantization. Representing color images with a lesser amount of colors is an example of vector quantization.

In general, many digital signal processing problems require some form of quantization. In particular, as the world moves toward an increased use of digital media, the digital processing of sound, images, and video involves an increasing amount of quantization. Some problems that would benefit from better quantization algorithms are analogue-to-digital conversion, sampling in general, and compression [16, 15].

Quantization and Voronoi tessellations are closely related and have a variety of applications to more than just signal processing and telecommunication. For example, distribution of resources and clustering algorithms [14, 19, 18] are, in essence, quantization algorithms that are used in various fields such as statistical analysis, pattern recognition, learning theory, computer graphics, and combinatorial chemistry [5]. A review on Voronoi tessellations as well as additional applications, such as quadrature rules, finite difference schemes, statistics, cellular biology, and the cellular behavior of animals [17], can be found in [7].

1.1. Formal definition.

Stating the problem more formally, let Ω be some input domain and let $p : \Omega \to \mathbb{R}$ be the probability density function of some random process over Ω. Let n be a known positive integer. A quantizer $q : \Omega \to \Omega$ is defined

*Received by the editors May 5, 2004; accepted for publication (in revised form) December 21, 2004; published electronically February 3, 2006. This research was supported by the Israeli Science Foundation under grant 48/02 and by the Israeli Ministry of Science.
http://www.siam.org/journals/sisc/27-5/60776.html
†Department of Computer Science, Technion—Israel Institute of Technology, Haifa 32000, Israel (yair_k@cs.technion.ac.il, irad@cs.technion.ac.il).

by n representation levels $R = \{\vec{r}_i\}_{i=1}^n$ and n decision regions $\{D_i\}_{i=1}^n$, where for all $i, r_i \in D_i$, and $\cup_i D_i = \Omega$. That is, $q(\Omega)$ is a piecewise constant approximation of Ω, where all $\vec{x} \in D_i$ are represented by $\vec{r}_i$. An optimal quantizer achieves minimal distortion, defined as the expectation of the quantization error raised to some power t,

$$\mathcal{D}(q) = E[\|\vec{x} - q(\vec{x})\|_2^t] = \sum_{i=1}^{n} \int_{D_i} \|\vec{x} - \vec{r}_i\|_2^t \, p(\vec{x}) \, d\vec{x}. \tag{1.1}$$

Here, $\|\cdot\|_2$ is the L_2 norm. In the discrete case, where $p(\vec{x})$ is nonzero at only a finite number of points $\{\vec{x}_i\}_{i=1}^N$, a similar functional is minimized:

$$\mathcal{D}(q) = E[\|\vec{x} - q(\vec{x})\|_2^t] = \sum_{i=1}^{n} \left\{ \sum_{\vec{x}_j \in D_i} \|\vec{x}_j - \vec{r}_i\|_2^t p(\vec{x}_j) \right\}. \tag{1.2}$$

Henceforth, we permit $p(\vec{x})$ to be any nonnegative function, not necessarily a distribution density function of unit mass. We also set $t = 2$, yielding the ubiquitous mean square error (MSE) measure. In the theoretical background section we show that our algorithms are equally applicable for any t.

1.2. Previous work. In recent years, many methods have been presented that reduce the complexity of finding a solution, e.g., solving multiple 1-D problems [1], recursively splitting the domain in two [25, 16, 30], and accelerating known algorithms [32, 27]. These methods are usually greedy and/or heuristic and involve a complexity/distortion tradeoff. That is, lower complexity algorithms generally arrive at solutions with higher distortion. Wu [31] generalizes the algorithms described in [25, 16, 30] and attempts to arrive at a good local minimum by making an observation about a constant number of consecutive splits and by using dynamic programming.

1.3. The Lloyd–Max iterative process. Necessary conditions for a minimum of (1.1) are

$$\begin{aligned} \frac{\partial}{\partial \vec{r}_i} \mathcal{D}(q) &= 0, \qquad 1 \le i \le n, \\ \frac{\partial}{\partial D_i} \mathcal{D}(q) &= 0, \qquad 1 \le i \le n. \end{aligned} \tag{1.3}$$

When $t = 2$, these conditions yield

$$\begin{aligned} \vec{r}_i &= \frac{\int_{D_i} \vec{x} p(\vec{x}) d\vec{x}}{\int_{D_i} p(\vec{x}) d\vec{x}}, \qquad 1 \le i \le n, \\ D_i &= \mathcal{V}(\vec{r}_i), \qquad\qquad 1 \le i \le n. \end{aligned} \tag{1.4}$$

Here, $\mathcal{V}(\vec{r}_i)$ is the Voronoi cell corresponding to $\vec{r}_i$, i.e., the set of all points in Ω that are closer to $\vec{r}_i$ than to any other representation level $\vec{r}_j$, $j \neq i$. Lloyd [22] and Max [24] independently developed (1.4) and proposed "ping-ponging" back and forth

between the two optimality equations,

$$D_i^\ell = \mathcal{V}((\vec{r}_i)^\ell), \qquad \ell = 0, 1, 2, \ldots, \tag{1.5}$$

$$(\vec{r}_i)^{\ell+1} = \frac{\int_{D_i^\ell} \vec{x} p(\vec{x}) d\vec{x}}{\int_{D_i^\ell} p(\vec{x}) d\vec{x}}, \qquad \ell = 0, 1, 2, \ldots, \tag{1.6}$$

that is, optimizing the D_i's according to the $\vec{r}_i$'s (computing the Voronoi cells of the $\vec{r}_i$'s) followed by optimizing the $\vec{r}_i$'s according to the new D_i's (computing the centers of mass of the D_i's), and so on. $\vec{r}^0$ is an initial guess at the solution. This process converges monotonically to a solution that satisfies (1.4) (see [22]).

When applying Lloyd–Max iterations (1.5), (1.6) in the discrete case (1.2) one need not actually compute the Voronoi diagram, which takes $\theta(n \log n + n^{\lceil k/2 \rceil})$ operations for a k-dimensional domain Ω. For high dimensions it is both sufficient and efficient to determine, for each $\vec{x} \in \Omega$, which representation level $\vec{r}_i$ is closest to $\vec{x}$. The naive method of inspecting all n possibilities for each $\vec{x} \in \Omega$ and choosing the closest one takes $O(kNn)$ operations, where $N = |\{\vec{x} : p(\vec{x}) \neq 0\}|$. If a k-d tree [9] is used to find best matches, then the expected computation time of a Lloyd–Max iteration is $O(kN \log n)$.

For scalar quantization ($k = 1$), Fleischer showed that a sufficient condition for a unique minimum of the distortion is that $\log(p(x))$ be concave [8]. For vector quantization, we are unaware of a similar result.

Since the Lloyd–Max iterative process requires an initial guess at the solution in order to start iterating, many heuristic methods, such as k-means [23] and Linde–Buzo–Gray (LBG) [21], have been proposed to find a good initial approximation. The LBG algorithm [21] is a "grid refinement" type of process that starts by finding a solution (using the generalized Lloyd algorithm) for two $\vec{r}_i$'s and then splitting each $\vec{r}_i$ into two new $\vec{r}_i$'s. This process is repeated recursively until the number of $\vec{r}_i$'s equals n. The generalized Lloyd algorithm is the iterative process (1.5), (1.6) with the reduction in distortion used as a stopping criterion. That is, the process is stopped at the iteration in which the ratio between the current distortion and the previous iteration's distortion exceeds some threshold. In what follows, we shall compare our method with LBG.

A multigrid [4, 29] approach for scalar (1-D) quantization was presented in [20]. Our approach was basically a multigrid full approximation scheme algorithm [3] with the Lloyd–Max iteration (1.5), (1.6) used as relaxation. The benefits are three-fold. First, much faster convergence is achieved. The convergence rates of our method seem independent of the number of representation levels sought. In contrast, both Lloyd–Max and LBG converge at a rate that is quadratically dependent on the number of representation levels. Second, the speed up alleviates the need for a convergence/stopping criterion such as the one used for LBG. This is because reaching machine accuracy requires only several iterations when using our method. Third, our method sometimes arrives at better minima than other methods.

The vector quantization problem is generally nonconvex and therefore affords multiple local minima [12]. Figure 1.1 depicts several local minima for the simple case, where $p(x, y) \equiv 1$ and four representation levels are sought. We consider the acceleration of convergence less important than converging to a solution of lower distortion. This paper presents methods for both scalar and vector quantization to overcome the multiple local minima problem. Our methods arrive at solutions of lower distortion and do so in less time compared to competing algorithms.

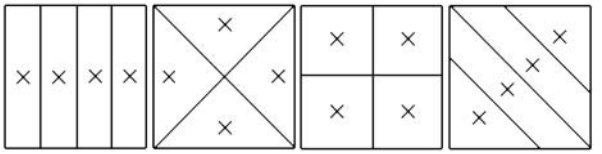


FIG. 1.1. *Multiple local minima for $p(x, y) \equiv 1$. These are all local minima, as the Lloyd–Max criteria* (1.4) *are satisfied. That is, the representation levels are the centers of mass of the decision regions and the decision regions are the Voronoi diagram of the representation levels.*

This paper is organized as follows. Section 2 presents the theoretical background upon which our algorithms are built. Section 3 provides a detailed description of our proposed adaptive multiscale redistribution (AMR) algorithms for vector quantization. Section 4 includes results of our vector quantization algorithms and a comparison with LBG. Section 5 presents a conclusion and further discussion.

2. Theoretical background. Much theoretical research and analysis have been conducted on the problem of quantization [2, 22, 33, 26, 10, 28]. In this section, in order to maintain a self-contained presentation, we recount Gersho's theoretical results concerning optimality criteria for vector quantizers [10]. The optimality criteria are valid for general values of t in the distortion definition (1.1). Therefore, the algorithms derived in the following section are also valid for any t. Gersho's results are true in the large n limit. Nevertheless, our algorithms, which are based on Gersho's results, perform extremely well for small n as well. Examples are presented in section 4.

In his dissertation [33], Zador proves that the [illegible] structure of an asymptotically optimal quantizer can be that of an optimal quantizer for a uniform distribution. Therefore, we henceforth assume that $p(\vec{x})$ is uniform, whereas in what follows we shall apply the optimality criteria derived in this section to nonuniform $p(\vec{x})$ as well.

In 1979, Gersho [10] made the now widely accepted conjecture that, for a uniform distribution $p(\vec{x})$ in any dimension k, the optimal decision regions are all congruent to some k-dimensional convex polytope (precluding the decision regions touching the boundaries of Ω). The Voronoi diagram in this case is referred to as a [illegible] of the k-dimensional domain Ω. Gersho's conjecture has been proven for one (see [6, p. 59]) and two [28] dimensions, where the appropriate tessellation-generating polytopes are the section and the regular hexagon, respectively.

Henceforth, we assume that Gersho's conjecture is correct. Under this assumption, an analysis of the distortion of lattice quantizers is within reach. The class of admissible polytopes in $\mathbb{R}^k$, denoted H_k, is defined as follows. A convex polytope H is admissible, i.e., in H_k, if it generates a tessellation of $\mathbb{R}^k$ that is a Voronoi diagram with respect to the centers of mass of the polytopes comprising this tessellation.

The [illegible] of a polytope H is defined as

$$I(H) = \int_H \|\vec{x} - \mathtt{com}(H)\|^t \, d\vec{x} \Big/ V(H)^{1+t/k}, \tag{2.1}$$

where $V(H)$ is the volume of H and $\mathtt{com}(H)$ is the center of mass of H, or more formally, the point y in H that minimizes $\int_H \|x - y\|^t dx$. Note that $I(\alpha H) = I(H)$ for all $\alpha > 0$, where $\alpha H = \{\alpha\vec{x} : \vec{x} \in H\}$. Intuitively, the normalized inertia measures the quality of a polytope's shape, disregarding size, with respect to distortion minimization.

An optimal polytope, H^*, is an admissible polytope that minimizes the normalized inertia, i.e.,

$$I(H^*) = \inf_{H \in H_k} I(H). \tag{2.2}$$

Given a quantizer q_n having n representation levels, a piecewise constant [illegible] [22] for q_n is defined as

$$\lambda_n(\vec{x}) = (nV(D_i))^{-1} = \frac{1}{n \int_{D_i} d\vec{y}}, \tag{2.3}$$

where D_i is the decision region containing $\vec{x}$, and $V(D_i)$ is D_i's volume. $\lambda_n(\vec{x})\Delta V(\vec{x})$ approximates the fraction of representation levels in the volume element $\Delta V(\vec{x})$. As n approaches infinity, $\lambda_n(\vec{x})$ approximates closely a continuous density function, denoted $\lambda(\vec{x})$. $\lambda(\vec{x})$ has unit mass, i.e., $\int_\Omega \lambda(\vec{x})d\vec{x} = 1$, since $\int_{D_i} \lambda_n(\vec{x})d\vec{x} = 1/n$ for any i and n. Integrating $\lambda(\vec{x})$ over a set $S \in \Omega$ yields the fraction of representation levels in S, whereas integrating $\lambda_n(\vec{x})$ over S approximates this fraction. That is,

$$\frac{|\{\vec{r}_i \,:\, \vec{r}_i \in S\}|}{n} \approx \int_S \lambda_n(\vec{x})d\vec{x}. \tag{2.4}$$

When the number of representation levels is large we may assume that $p(\vec{x}) \approx p(\vec{r}_i)$ for all $\vec{x} \in D_i$; that is, we may assume that the distribution is approximately uniform inside each decision region. Under this assumption, the probability may be removed from under the integral in the distortion definition (1.1), yielding

$$\mathcal{D}(q_n) = \sum_{i=1}^{n} p(\vec{r}_i) \int_{D_i} \|\vec{x} - \vec{r}_i\|^t \, d\vec{x}. \tag{2.5}$$

Plugging in the normalized inertia, we obtain

$$\mathcal{D}(q_n) = \sum_{i=1}^{n} p(\vec{r}_i) \, I(H^*) \, [V(D_i)]^{1+\frac{t}{k}}, \tag{2.6}$$

since, under Gersho's conjecture, the minimal distortion is achieved by decision regions that are similar to H^*. According to (2.3), $V(D_i)$ may be expressed by $\lambda_n(\vec{x})$:

$$\mathcal{D}(q_n) = n^{-(t/k)} I(H^*) \sum_{i=1}^{n} p(\vec{r}_i) \, [\lambda_n(\vec{r}_i)]^{-(t/k)} \, V(D_i). \tag{2.7}$$

As n approaches infinity, the above sum is approximated by the integral

$$\mathcal{D}(q_n) \approx n^{-(t/k)} I(H^*) \int_\Omega \frac{p(\vec{x})}{[\lambda(\vec{x})]^{t/k}} d\vec{x}. \tag{2.8}$$

Equation (2.8) is an approximation of the distortion that is independent of the representation levels and the decision regions. In fact, the only unknown in (2.8) is $\lambda(\vec{x})$. Therefore, the minimal distortion is achieved by optimally choosing $\lambda(\vec{x})$, i.e., by minimizing the integral.

Observe that, by Hölder's inequality,

$$\int_\Omega \left\{ \left[p\left(\vec{x}\right) \lambda(\vec{x})^{-(t/k)} \right]^{\frac{1}{1+t/k}} \right\} \left\{ \lambda(\vec{x})^{\frac{t/k}{1+t/k}} \right\} d\vec{x} \tag{2.9}$$

$$\leq \left\{ \int_\Omega p\left(\vec{x}\right) \lambda(\vec{x})^{-(t/k)} d\vec{x} \right\}^{\frac{1}{1+t/k}} \left\{ \int_\Omega \lambda\left(\vec{x}\right) d\vec{x} \right\}^{\frac{t/k}{1+t/k}},$$

with equality if and only if the right-hand side integrands are proportional, i.e., if and only if $p(\vec{x})[\lambda(\vec{x})]^{-(t/k)} \propto \lambda(\vec{x})$.

As $\lambda(\vec{x})$ has unit mass, we obtain from (2.9)

$$\int_\Omega \frac{p(\vec{x})}{\left[\lambda\left(\vec{x}\right)\right]^{t/k}} d\vec{x} \geq \left\{ \int_\Omega \left[p\left(\vec{x}\right)\right]^{\frac{1}{1+t/k}} d\vec{x} \right\}^{1+t/k}. \tag{2.10}$$

The left-hand side of (2.10) is evidently minimized when equal to the right-hand side. By (2.8), (2.9), and (2.10) the distortion is therefore minimized if and only if

$$\lambda(x) \propto [p(x)]^{k/(k+t)}. \tag{2.11}$$

In what follows we construct algorithms based on the two following corollaries, appearing in Gersho's paper [10].

COROLLARY 2.1. [illegible]

$$\int_S \lambda(\vec{x}) d\vec{x} \approx \frac{\int_S p^{\frac{k}{k+t}}(\vec{x}) d\vec{x}}{\int_\Omega p^{\frac{k}{k+t}}\left(\vec{x}\right) d\vec{x}} \tag{2.12}$$

[illegible] S [illegible] Ω [illegible]. This is a direct result of the previous derivations. The right-hand side denominator assures unit volume for $\lambda(\vec{x})$. □

COROLLARY 2.2. [illegible] q [illegible] (1.1) [illegible]. Using (2.3) and (2.11), one may express both $p(\vec{r}_i)$ and $V(D_i)$ by $\lambda(\vec{x})$ in (2.7). This yields the stated result, as each term in the sum in (2.7) is reduced to the same constant, namely, $1/n$ times the proportion constant from (2.11) □

The property indicated by Corollary 2.2 was observed for $k = 1$ by Panter and Dite [26] and for $k = 2$ by Toth [28].

3. Adaptive multiscale redistribution algorithms. We now exploit the theoretical results described above within a novel AMR algorithm. We begin by defining three methods for building a quantizer. Our ultimate quantization algorithm involves concatenating these three methods.

The first method is an extremely quick way of attaining a good initial set of n representation levels. This method heavily relies on (2.12) and achieves a distortion that usually is nearly as low as our final result. The complexity of obtaining this initial guess is proportional to $\log(n)$ Lloyd–Max relaxations, where n is the number of representation levels sought. Stopping this iterative initial guess phase after two to three iterations usually results in a good initial approximation, which may be even better than LBG's final result. From our tests, LBG [21] also seems to require $O(\log(n))$ iterations in order to arrive at an initial guess, but LBG's guess is usually not as good as the one provided by our method, as shall be demonstrated in section 4. In

addition, stopping LBG before $O(\log(n))$ Lloyd–Max iterations have been performed usually results in a poor initial approximation.

The second method is used to improve an initial guess. The improvement is based on Corollary 2.2 and is performed by moving representation levels from low distortion areas to areas with higher distortion.

The last method is simply applying Lloyd–Max iterations (1.5), (1.6) to the best approximation found to date, thus reaching a local minimum of the quantization distortion. We propose a quantization algorithm that is made up of these three steps. First, find an initial guess. Second, improve the initial guess. Third, relax until a local minimum has been reached.

3.1. The general approach. In both acquiring an initial guess and improving it, we use the following AMR approach. A hierarchy of scales is defined. Scale m, denoted R^m, is a partition of the set of representation levels R into m disjoint aggregates. That is,

$$R^m = \{R_j^m\}_{j=1}^m, \; R = \bigcup_j R_j^m. \tag{3.1}$$

At any scale m, all m aggregates are approximately equal in size. At the finest scale, each aggregate is made up of a single representation level, that is, $m = n$. When passing from scale m to scale $m/2$ (the next coarser scale), the m aggregates are grouped in pairs to form approximately $m/2$ new aggregates. This set of $m/2$ new aggregates is scale $m/2$. The coarsest scale consists of two aggregates.

At each scale, we determine the poorest aggregate, that is, the aggregate that would benefit the most from an additional representation level. In a similar manner, we determine the richest aggregate, that is, the aggregate that would suffer the least from giving up a representation level. Then we transfer a representation level from the richest aggregate to the poorest aggregate.

3.2. Acquiring an initial guess. For the purpose of acquiring an initial guess we exploit our knowledge of the optimal point density function (2.12). Let

$$D(R_j^m) = \bigcup_{\vec{r}_i \in R_j^m} D_i, \tag{3.2}$$

that is, $D(R_j^m)$ is the union of the decision regions of all the representation levels in R_j^m. For each aggregate R_j^m of the m aggregates in R^m, (2.12) suggests the fractional number of representation levels that an optimal quantizer allocates to the region $D(R_j^m)$. Subtracting this amount (multiplied by n) from the actual number of representation levels in $D(R_j^m)$, we arrive at the , , $\mathcal{W}_j^m$, of aggregate R_j^m. More formally,

$$\mathcal{W}_j^m = |R_j^m| - n \left\{ \frac{\int_{D(R_j^m)} p^{\frac{k}{k+2}}(\vec{x}) d\vec{x}}{\int_\Omega p^{\frac{k}{k+2}}(\vec{x})\, d\vec{x}} \right\}. \tag{3.3}$$

According to this definition, the wealth of an aggregate of decision regions is simply the excess of representation levels it currently contains. We call aggregates whose wealth is above zero , aggregates and aggregates whose wealth is below zero , aggregates.

Our algorithm for obtaining an initial guess is as follows. We initialize R^n by randomly choosing n representation levels uniformly in Ω. We use this uniform initial approximation for two reasons. First, it is extremely cheap to compute. Second, and more important, the $Obtain-Guess$ algorithm transfers representation levels to places in Ω that already contain other representation levels. Therefore, a uniform initial approximation gives an equal chance for all areas in Ω to receive representation levels. Next, we call the following recursive function with R^n as input (note that $n = |R^n|$).

ALGORITHM.
$R^m \leftarrow Obtain-Guess(p(\vec{x}), R^m)$.

1. Perform a single Lloyd–Max relaxation.
2. Calculate $\mathcal{W}^m = \{\mathcal{W}_i^m\}_{i=1}^m$.
3. Let $\mathcal{W}_{max}^m$ denote $\max(\mathcal{W}^m)$ and $\mathcal{W}_{min}^m$ denote $\min(\mathcal{W}^m)$.
4. *WHILE* $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$, *DO*:
   (a) $\vec{r}_j \leftarrow Select-Source(R_{max}^m)$.
   (b) $\vec{t} \leftarrow Select-Destination(R_{min}^m)$.
   (c) Transfer $\vec{r}_j$ from R_{max}^m to R_{min}^m at location $\vec{t}$.
   (d) $\mathcal{W}_{min}^m \leftarrow \mathcal{W}_{min}^m + 1$, $\mathcal{W}_{max}^m \leftarrow \mathcal{W}_{max}^m - 1$.
5. $R^{m/2} \leftarrow Match(R^m)$.
6. *IF* $R^{m/2} = R^m$
   (a) *RETURN* R^m.
7. *ELSE*,
   (a) Recursively call $Obtain-Guess(p(\vec{x}), R^{m/2})$.

Throughout the $Obtain-Guess$ algorithm, subscripts *max* and *min* refer to aggregates of maximal and minimal wealth, respectively. Note that the decision regions are not recalculated after the transition of representation levels, but only within the Lloyd–Max relaxation occurring at step 1 of $Obtain-Guess$. This single Lloyd–Max relaxation at each scale is performed to improve the solution. In addition, this relaxation is needed for the calculation of $\mathcal{W}_i^m$, as this calculation requires knowing the decision regions. The output of the $Obtain-Guess$ algorithm is the set of representation levels yielding the lowest distortion achieved.

Note that $Obtain-Guess$ is an adaptive multiscale redistribution of the representation levels. The goal of this redistribution is to have the quantizer's point density function obey (2.12) as best as possible. Since checking (2.12) for all possible sets S is intractable, we do so only for various unions of decision regions. The following lemma shows how each transition of a representation level at step 4(c) of $Obtain-Guess$ makes the point density function of the quantizer more coherent with (2.12).

LEMMA 3.1. [illegible] $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$ [illegible] 4 [illegible] $Obtain-Guess$ [illegible] 4(c) [illegible] $Obtain-Guess$ [illegible] $\sum_i |\mathcal{W}_i^m|$ [illegible]. We shall prove that $|\mathcal{W}_{max}^m - 1| + |\mathcal{W}_{min}^m + 1| < |\mathcal{W}_{max}^m| + |\mathcal{W}_{min}^m|$. Assume that $0 \le \mathcal{W}_{max}^m < 1$ and $-1 < \mathcal{W}_{min}^m \le 0$. When this is not true, the result is trivial. Note that $\mathcal{W}_{max}^m \ge 0$ and $\mathcal{W}_{min}^m \le 0$ always hold. $|\mathcal{W}_{max}^m - 1| = 1 - \mathcal{W}_{max}^m < -\mathcal{W}_{min}^m = |\mathcal{W}_{min}^m|$. The first equality is implied by $\mathcal{W}_{max}^m < 1$ and the inequality by $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$. In a similar manner, $|\mathcal{W}_{min}^m + 1| = 1 + \mathcal{W}_{min}^m < \mathcal{W}_{max}^m = |\mathcal{W}_{max}^m|$, where the first equality is implied by $\mathcal{W}_{min}^m > -1$ and the inequality by $\mathcal{W}_{max}^m - \mathcal{W}_{min}^m > 1$. □

The selection of a representation level in the richest aggregate, at step 4(a) of $Obtain - Guess$, is described next.

ALGORITHM.
$\vec{r}_i \leftarrow Select - Source(R_j^m)$.

1. *IF* $\left|R_j^m\right| > 1$
   (a) *IF* $R_j^m = R_s^{2m}$ for some $1 \le s \le 2m$,
      i. Recursively call $Select - Source(R_s^{2m})$.
   (b) *ELSE*,
      i. $R_j^m = R_u^{2m} \cup R_v^{2m}$ for some $1 \le u, v \le 2m$.
      Let R_ℓ^{2m} be the aggregate that achieves $\max\{\mathcal{W}_u^{2m}, \mathcal{W}_v^{2m}\}$.
      ii. Recursively call $Select - Source(R_\ell^{2m})$.
2. *ELSE*,
   (a) *RETURN* the single $\vec{r}_i \in R_j^m$.

Less formally, we wish to remove the representation level in R_j^m that belongs to the richest aggregate [illegible]. We traverse the scales from the present scale to the finer scales. At each scale m we deal with a single aggregate R_j^m. R_j^m is usually made up of two $2m$-scale aggregates (the actual matching is performed by $Match$). We choose the richer aggregate of the two and continue recursively. When we reach an aggregate containing a single representation level, we choose it as the representation level to remove.

The selection of a destination in the poorest aggregate, at step 4(b) of $Obtain - Guess$, is described next.

ALGORITHM.
$\vec{t} \leftarrow Select - Destination(R_j^m)$.

1. *IF* $\left|R_j^m\right| > 2$
   (a) *IF* $R_j^m = R_s^{2m}$ for some $1 \le s \le 2m$,
      i. Recursively call $Select - Destination(R_s^{2m})$.
   (b) *ELSE*,
      i. $R_j^m = R_u^{2m} \cup R_v^{2m}$ for some $1 \le u, v \le 2m$.
      Let R_ℓ^{2m} be the aggregate that achieves $\min\{\mathcal{W}_u^{2m}, \mathcal{W}_v^{2m}\}$.
      ii. Recursively call $Select - Source(R_\ell^{2m})$.
2. *ELSE*,
   (a) *RETURN* the mean of the set R_j^m.

Less formally, we wish to insert a representation level at a location that belongs to the poorest aggregate [illegible]. In a manner similar to $Select - Source$, we traverse the scales from the present scale to the finer scales. This time, at each scale m we choose the poorest aggregate of the two that make up R_j^m and continue recursively. When we reach an aggregate containing just two representation levels, we choose the destination as the midpoint between the two levels.

The coarser scales are required to take care of the situations in which many small aggregates are moderately poor and many other small aggregates are moderately rich. In this case, as the wealth function (3.3) is linear, the grouping of several somewhat poor aggregates should form a larger aggregate that is poor enough to warrant an additional representation level. In a similar fashion, the grouping of several somewhat rich aggregates should form a larger aggregate that is rich enough to allow a

representation level to be removed. In the appendix, we demonstrate the importance of wrapping the redistribution in a multiscale framework.

In light of the above, the matching between pairs of aggregates that is performed at step 5 of $Obtain - Guess$ and is designed to produce larger aggregates should only match aggregates that are either both rich or both poor. In addition we prefer to match aggregates that are close to each other. This is because adding a new representation level to an aggregate made up of two distant regions only increases the wealth of the region that accepts the representation level. This accepting region may not even be one of the two distant ones, in which case the transition is clearly not the right course of action. If the accepting region is one of the two distant ones and indeed the transfer is beneficial, then it should have been performed at a finer scale, where each of the two regions was a separate aggregate.

We define the distance between two aggregates as the distance between the mean of the representation levels in each aggregate, i.e.,

$$\|R_i^m, R_j^m\| = \left\| \frac{\sum_{\vec{r}_i \in R_i^m} (\vec{r}_i)}{|R_i^m|} - \frac{\sum_{\vec{r}_j \in R_j^m} (\vec{r}_j)}{|R_j^m|} \right\|_2 . \tag{3.4}$$

More formally, the matching algorithm is defined as follows.

ALGORITHM.
$R^{m/2} \leftarrow Match(R^m)$.

1. Initialize $R^{m/2} = \phi$.
2. Sort the aggregates in R^m according to $|\mathcal{W}_i^m|$.
3. Traverse the sorted list, *FOR EACH* aggregate R_j^m, *DO*:
   (a) Denote by Δ the set of unmatched aggregates, R_k^m, that satisfy $\text{sign}(\mathcal{W}_k^m) = \text{sign}(\mathcal{W}_j^m)$.
   (b) *IF* $\Delta = \phi$
      i. Add R_j^m to $R^{m/2}$ and mark R_j^m as matched.
   (c) *ELSE*,
      i. Find $R_i^m = \underset{R_k^m \in \Delta}{\text{argmin}} \|R_k^m, R_j^m\|$.
      ii. Add $R_j^m \cup R_i^m$ to $R^{m/2}$ and mark both R_j^m and R_i^m as matched.
4. *RETURN* $R^{m/2}$.

Using k-d trees [9] to compute $\text{argmin} \|R_k^m, R_j^m\|$ in step 3(c)(i) of $Match$, the expected complexity of the matching algorithm is $O(km \log m)$, where m is the number of aggregates. Note that this is asymptotically much less than the $O(kN \log n)$ required for a Lloyd–Max relaxation (recall that N is the number of points, $\vec{x}$, for which $p(\vec{x}) > 0$).

We apply $Obtain - Guess$ several times until no representation levels are transferred at any scale. Each call to $Obtain - Guess$ applies $O(\log(n))$ Lloyd–Max relaxations that are the bulk of its time complexity. In practice we have applied this method to many examples. We have found that about 2–3 calls to $Obtain - Guess$ generate a "steady" initial guess, i.e., one that is not affected by another call to $Obtain - Guess$.

From our tests, the distortion of the initial guess thus obtained may be quite close to the final distortion achieved by concatenating our three methods. Whether or not the subsequent gain justifies the additional work is both application and user

dependent. For example, video compression may be performed off-line and the additional computational cost may therefore by acceptable in return for an even mild improvement in the resulting compressed file size. On the other hand, real time applications do not have the luxury of long processing time even if the improvement in the resulting distortion is significant. We expect those who need a good solution and require a fast algorithm to be quite satisfied with the initial guess and not apply the following two phases. On the other hand, those who require the best solution possible and/or are willing to accept a somewhat slower process are encouraged to apply the entire three phase process. For $p(\vec{x})$ that are sparse and patchy (e.g., color images), the ratio between the distortion achieved by AMR (either the single phase algorithm or the three phase algorithm) and the distortion achieved by LBG may be arbitrarily small, as shall be demonstrated in section 4.

3.3. Improving an initial guess. Improving a guess requires more leniency than acquiring an initial guess. The harsh demand that $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > 1$ (step 4 of $Obtain-Guess$) in order to perform a transition is relaxed by $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > T$ for some positive threshold T. In all of our tests, we have used $T = 1/2$. In addition we redefine an aggregate's wealth. This time the definition is based on Corollary 2.2,

$$\mathcal{W}_j^m = 1 - \frac{average\ distortion\ at\ aggregate\ j}{average\ distortion\ in\ \Omega}, \tag{3.5}$$

where "*average distortion*" is the average distortion per representation level.

ALGORITHM.
$R^m \leftarrow Improve-Guess(R^m, T, \nu)$.

1. Perform a single Lloyd–Max relaxation.
2. *WHILE* the distortion of R^m, $\mathcal{D}(R^m)$, decreases, *DO:*
   (a) Calculate $\mathcal{W}^m = \{\mathcal{W}_i^m\}_{i=1}^m$.
   (b) *IF* $\max(\mathcal{W}^m) - \min(\mathcal{W}^m) > T$,
      i. Move 1 representation level from the richest aggregate to the poorest aggregate, using $Select-Source$ and $Select-Destination$. Denote by $\tilde{R}^m$ the set of aggregates after the transition.
      ii. $\tilde{R}^m \leftarrow Beat-Target-Distortion(\tilde{R}^m, \mathcal{D}(R^m), \nu)$.
      iii. *IF* the distortion has not improved,
         A. discard the new $\tilde{R}^m$ and keep R^m.
      iv. *ELSE*,
         A. $R^m \leftarrow \tilde{R}^m$.
3. Form a set of new aggregates: $R^{m/2} \leftarrow Match(R^m)$.
4. *IF* $R^{m/2} = R^m$
   (a) *RETURN* R^m.
5. *ELSE*,
   (a) Recursively call $Improve-Guess(R^{m/2}, T, \nu)$.

According to Corollary 2.2, an optimal quantizer obeys $\mathcal{W}_j^m = 0$ for all $1 \le j \le m \le n$. This definition still permits us to call aggregates whose wealth is above zero [illegible] aggregates and call aggregates whose wealth is below zero [illegible] aggregates.

Our improvement method uses the above distortion-based wealth definition (3.5). From our tests, the distortion-based wealth definition (3.5) leads to better results. On the other hand, a meaningful threshold for the initial guess phase (step 4 in

$Obtain-Guess$), such as the one that Lemma 3.1 provides for (3.3), is more difficult to come by for (3.5). This is why we use the point density–based wealth definition (3.3) with threshold $T = 1$ when acquiring an initial guess.

$Beat-Target-Distortion$ applies Lloyd–Max relaxations to $\tilde{R}^m$, attempting to achieve a distortion lower than $\mathcal{D}(R^m)$. After each relaxation, the function tries to assess if ν additional relaxations are likely to reduce the distortion enough. This assessment is made by assuming that each additional Lloyd–Max relaxation will reduce the distortion by the actual reduction achieved via the previous Lloyd–Max relaxation.

The linear extrapolation we use to forecast the distortion reduction has the following property. It is optimistic in the early stages, at which time a more accurate approximation may halt the process prematurely, whereas in the later stages, as the distortion reduction rate reduces, it becomes more accurate. If the target distortion is beaten, then $Beat-Target-Distortion$ applies ν additional Lloyd–Max relaxations and the result is returned. The additional ν relaxations are performed to allow the new solution to realize its potential. More formally, we have the following algorithm.

ALGORITHM.
$R^m \leftarrow Beat-Target-Distortion(R^m, \mathcal{D}, \nu)$.

1. *DO*: {
   (a) Denote the current distortion $\mathcal{D}^{old} = \mathcal{D}(R^m)$.
   (b) Perform a Lloyd–Max relaxation on R^m.
   (c) Denote the new distortion by $\mathcal{D}^{new} = \mathcal{D}(R^m)$.
   (d) *IF* $\mathcal{D}^{new} < \mathcal{D}$
      i. Apply ν additional Lloyd–Max relaxations to R^m and return the result.

   } *WHILE* $\mathcal{D}^{new} - \nu|\mathcal{D}^{new} - \mathcal{D}^{old}| < \mathcal{D}$.
2. *RETURN* R^m.

When a representation level is added or removed (step 2(b)(i) of $Improve-Guess$), a Lloyd–Max iteration (1.5), (1.6) propagates this change only to the neighbors of this representation level. Assuming that the representation levels are spread uniformly in the domain Ω, we see that it would take approximately $n^{1/k}$ Lloyd–Max iterations to propagate this change throughout all of the representation levels. For this reason, we used $\nu = \lceil n^{1/k} \rceil$ throughout our tests. For the threshold T, we used $T = 1/2$.

Coarse scale changes may upset the balance achieved at the finer scales. Therefore, it may be beneficial to call $Improve-Guess$ several times. Before calling $Improve-Guess$ an additional time, ν may be raised and T lowered. In our examples we always called $Improve-Guess$ two to three times, while T and ν were never changed.

The AMR process can be stopped at any time. In our tests, no matter when the process was stopped, the results were better than those achieved by LBG after the same amount of time.

4. Vector quantization results. Many digital signal processing applications involve some form of quantization. As an example, color images may be compressed by representing them with only a fraction of their initial amount of colors. The histograms of color images are usually sparse and patchy, as Figure 4.1 illustrates. Close inspection of Figure 4.1 reveals that the image's histogram is concentrated around seven patches. Six of the patches are fairly close to one another and represent the brighter areas in the image, e.g., sky, lawn, and walls and roof of the house. The seventh patch is quite distant from the rest and represents the darker areas in the

image, e.g., the house's window shutters, the bushes surrounding the house, and the shadows of the car and the utility pole. We demonstrate our algorithms on discrete 2-D data using the discrete distortion definition (1.2) with $t = 2$. Let

$$p_{ij} = \begin{cases} 5 & i, j \in \{1, 1, 2, \ldots, 150\}, \\ 3 & i, j \in \{201, 201, 202, \ldots, 256\}, \\ 0 & otherwise. \end{cases} \tag{4.1}$$

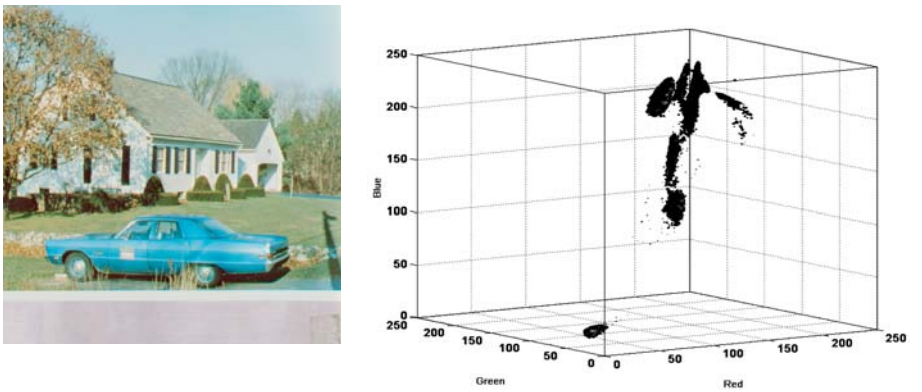


FIG. 4.1. *Left: House and car,* 512×512 *color image. Right: An isosurface of the image's histogram* $p(\vec{x})$.

p_{ij} is a simple probability density function comprised of only two patches (see Figure 4.2). Nevertheless, this p_{ij} is quite similar to probability density functions of real color images, such as the one depicted in Figure 4.1. Therefore, we consider the results exhibited in this section for the above 2-D naive p_{ij} as evidence of our methods' abilities to quantize real multidimensional signals.

From p_{ij}'s structure alone we can deduce that most of the representation levels should be allocated to the larger patch. The fact that the patches are square is completely unimportant; we have defined p_{ij} in this way solely for simplicity.

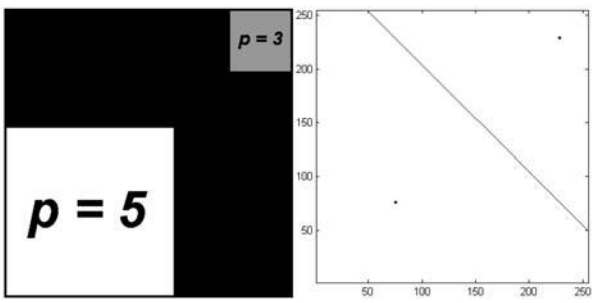


FIG. 4.2. *Left:* p_{ij}. *Right: Globally optimal solution for* $n = 2$ *produced by both LBG and AMR.*

We applied both LBG and AMR to this problem. Each algorithm execution in this section is performed independently of the others, that is, each execution starts from scratch. LBG was used with threshold $1 - 10^{-2}$, that is, Lloyd–Max iterations were

performed at each scale until the ratio of successive distortions ($\mathcal{D}^{new}/\mathcal{D}^{old}$) exceeded $1-10^{-2}$. On the finest scale, we have taken as LBG's initial guess the solution at the iteration in which the ratio of successive distortions exceeded this $1-10^{-2}$ threshold. Thereafter, Lloyd–Max relaxations were performed until the solution reached machine accuracy, that is, until no further improvement could be made. At the stages in which the representation levels are split, LBG adds a new representation level next to each existing one. Our implementation of LBG inserts the new representation levels in a random direction and at a random short distance relative to each of the existing representation levels. An implementation in which the new representation levels are inserted at a constant direction and distance from the existing ones would permit the building of simple probability density functions, for which LBG performs quite poorly.

For $n = 2$ both LBG and our method produce the same solution, namely, one representation level in each patch (see Figure 4.2). Figure 4.3 displays the solutions achieved by AMR and LBG for $n = 4, 8, 16, 32$, and 64. Figure 4.3 implies that LBG cannot always remedy errors made at coarse scales and that coarse scale solutions are not always good initial approximations for the finer scales, an assumption implicitly made by LBG. At the stage when $n = 2$, LBG decided to allocate one representation level to each patch. From that moment on, no representation levels could move from patch to patch since, for any representation level, the center of mass of its decision region is always located within its patch.

TABLE 4.1

LBG and AMR results when applied to the p_{ij} that is defined by (4.1) *and depicted in* 4.2. *From left to right, n = number of representation levels.* ***AMR iter.*** *are the cumulative numbers of Lloyd–Max iterations required to complete the first, second, and third phases of the AMR algorithm, respectively.* ***AMR dist.*** *is the distortion of the initial guess (phase one) and of the final result (phase three).* ***LBG iter.*** *are the numbers of Lloyd–Max iterations required to acquire an initial guess and are the numbers of Lloyd–Max iterations applied throughout the LBG algorithm.* ***LBG dist.*** *is the distortion of the initial guess and of the final result achieved by LBG.* ***LBG:AMR*** *are the ratios of the distortions achieved by LBG to those achieved by AMR (both at the initial guess and final result stages). These ratios may be considered as the factor by which AMR is better than LBG.*

n	**AMR iter.**			**AMR dist.**		**LBG iter.**		**LBG dist.**		**LBG:AMR**	
2	5	11	18	$4.73 \cdot 10^8$	$4.27 \cdot 10^8$	16	17	$4.3 \cdot 10^8$	$4.27 \cdot 10^8$	0.91	1.00
4	8	28	60	$1.73 \cdot 10^8$	$1.72 \cdot 10^8$	21	31	$2.72 \cdot 10^8$	$2.67 \cdot 10^8$	1.57	1.56
8	10	38	85	$6.99 \cdot 10^7$	$6.90 \cdot 10^7$	27	29	$1.07 \cdot 10^8$	$1.07 \cdot 10^8$	1.53	1.56
16	12	41	64	$3.41 \cdot 10^7$	$3.32 \cdot 10^7$	37	79	$5.73 \cdot 10^7$	$5.47 \cdot 10^7$	1.68	1.65
32	14	60	130	$1.68 \cdot 10^7$	$1.63 \cdot 10^7$	43	83	$2.87 \cdot 10^7$	$2.66 \cdot 10^7$	1.71	1.64
64	17	88	190	$8.20 \cdot 10^6$	$8.07 \cdot 10^6$	50	189	$1.43 \cdot 10^7$	$1.33 \cdot 10^7$	1.74	1.64
128	19	122	139	$4.10 \cdot 10^6$	$4.01 \cdot 10^6$	56	115	$7.19 \cdot 10^6$	$6.67 \cdot 10^6$	1.75	1.67
256	21	153	157	$2.03 \cdot 10^6$	$2.00 \cdot 10^6$	62	135	$3.49 \cdot 10^6$	$3.31 \cdot 10^6$	1.72	1.66

Table 4.1 compares AMR to LBG. The table displays the running times as well as the distortion values of the solutions attained by AMR and LBG for various n's. The running time is measured in Lloyd–Max iterations to obtain an implementation-independent comparison. Note that the computational cost of both AMR and LBG is dominated by that of the Lloyd–Max relaxations. The last column of Table 4.1 displays the ratio between the AMR distortion and the LBG distortion. It can be seen that, in this example, the first phase of the AMR algorithm (obtaining a guess) already yields a distortion that is close to the final result achieved by applying all three phases. It can also be seen that the complexity of AMR's first phase is $O(\log n)$

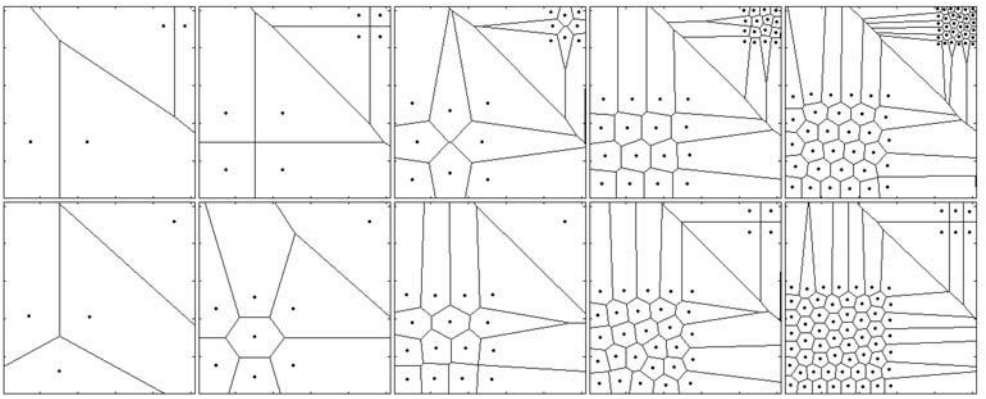


FIG. 4.3. *Quantization results of AMR and LBG applied to the p_{ij} that is defined by* (4.1) *and depicted in Figure* 4.2 *with* $n = 4, 8, 16, 32,$ *and* 64. *Top five plots: LBG. Bottom five plots: AMR.*

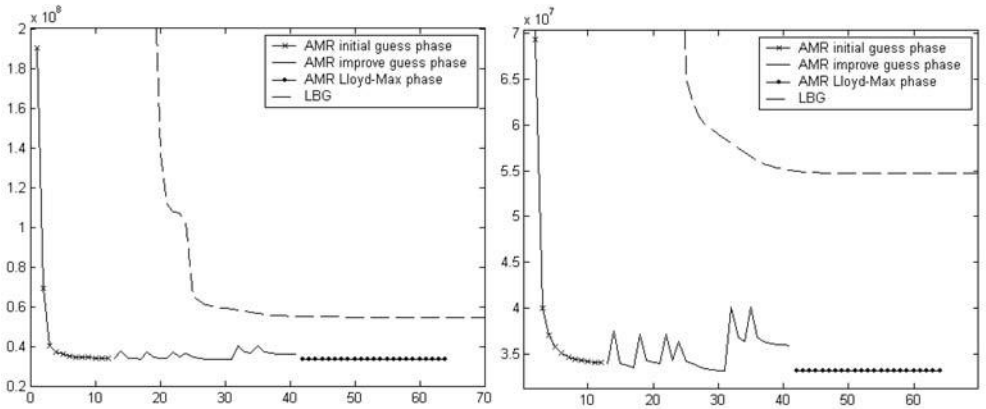


FIG. 4.4. *Distortion versus iterations for the AMR and LBG methods applied to p_{ij} that is defined in* (4.1) *and depicted in Figure* 4.2, *with* $n = 16$. *Note the different scale in each plot. Left: overall look at both processes. LBG's graph is dashed whereas AMR's three phases are solid with x-marks, solid without marks, and solid with point-marks, respectively. Right: close-up plot of the final distortion values.*

Lloyd–Max iterations, as multiplying n by 2 results in only a constant increase in the number of Lloyd–Max iterations.

From Figure 4.3 it is obvious that raising the LBG distortion improvement threshold for any $n > 2$ would not be beneficial. Raising the threshold would yield a solution with lower-valued distortion at each of the coarse scales. LBG implicitly assumes that coarse scale solutions are good initial guesses for the finer scales. Figure 4.3 demonstrates that, for example, when moving from $n = 2$ to $n = 4$, this is not necessarily the case.

In contrast to LBG, where the coarse scales lead us astray, AMR uses the coarse scales effectively to improve the fine scale solution. This is done by correcting errors at the coarse scales that were unrecognizable at the finer scales.

Figure 4.4 displays the distortion versus Lloyd–Max iterations for both LBG and

AMR when $n = 16$. The left-hand plot is an overall view of both processes. At its first stage, AMR begins with a uniform guess and redistributes the representation levels between the patches using the point density–based wealth function (3.3). Note the substantial drop in distortion achieved by the first three iterations. LBG falls short of this result even when run to completion. The phenomenon of 2–3 AMR iterations yielding lower distortion than tens or even hundreds of LBG iterations is quite common when quantizing sparse and patchy data. In general, for this kind of data, AMR may provide better solutions at a fraction of the computation time required by LBG, as LBG's solution at any time before $O(\log(n))$ Lloyd–Max iterations have been performed is quite poor. At its second phase, AMR tries to improve the solution by using (3.5). The purpose of this improvement stage is to traverse other feasible solutions that may have been overlooked at the first phase. Indeed, several improvements are found. Finally, the best solution to date is brought to a local minimum by application of Lloyd–Max iterations (1.5), (1.6). That is, Lloyd–Max iterations are performed until no further improvement is possible. The right-hand plot displays a close-up view of the processes when approaching the local minimum.

Our multiscale improvement method, $Improve-Guess$, was carried out until the first time that no transitions of representation levels were beneficial at any scale.

Table 4.1 demonstrates how the AMR solution may be significantly better than the LBG solution. We now show that the LBG solution may be arbitrarily bad.

OBSERVATION 1. [illegible] ℓ [illegible] $p(\vec{x})$ [illegible] $p(\vec{x})$ [illegible] ℓ [illegible]. Consider a patchy $p(\vec{x})$. Let $p(\vec{x})$ be nonzero at only $\ell+1$ disjoint patches $\{\Omega_i\}_{i=1}^{\ell+1}$. One patch covers a considerable part of the domain Ω and the other ℓ patches cover an arbitrarily small part of Ω. Assume that $p(\vec{x})$ is constant within each patch and that $\int_{\Omega_i} p(\vec{x})dx$ is equal for all $1 \le i \le \ell+1$. That is, all patches have the same mass. Figure 4.5 is an example of such a $p(\vec{x})$ with eight patches that is defined formally in (4.2).

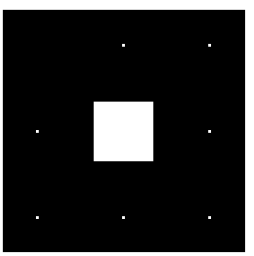


FIG. 4.5. *$p(x, y)$ with eight patches of different sizes but same mass, defined formally in* (4.2).

At the scale $n = \ell+1$, LBG's solution is, at best, comprised of one representation level per patch, which is the optimal solution. For larger n, LBG doubles each representation level. The Lloyd–Max relaxations do not permit representation levels to move from patch to patch, as the center of mass of any decision region always resides within the single patch covered by this region. Therefore, for any $n > \ell$, LBG places approximately $n/(\ell+1)$ representation levels in each patch. Assuming that the ℓ small patches are impulse functions (multiplied by the constant patch mass), the distortion that they contribute is zero once one or more representation levels are allocated to each of them. Therefore, for $n > \ell$, the optimal solution has $n-\ell$ representation levels

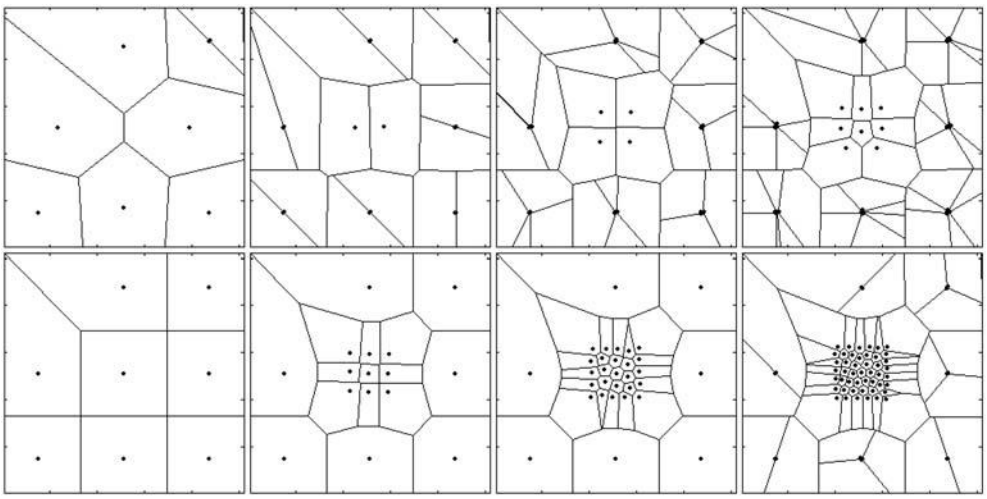


FIG. 4.6. *Quantization results of AMR and LBG on the eight-patch* $p(\vec{x})$ *defined by* (4.2) *and depicted in Figure* 4.5 *with* $n = 8, 16, 32,$ *and* 64. *Top four plots: LBG. Bottom four plots: AMR.*

in the large patch S, whereas LBG has only $n/(\ell+1)$ representation levels there.

It can be shown that, for the large patch, the distortion is roughly proportional to the inverse of the number of representation levels. This implies that the LBG solution is approximately ℓ times worse than the optimal solution.

More generally, for a d-dimensional problem, it can be shown that the distortion of the large patch is proportional to $n^{-2/d}$. Therefore, for a d-dimensional domain, one should choose a $p(\vec{x})$ with $(\ell+1)^{d/2}$ patches ($\ell+1$ in the 2-D case) to realize the stated result. □

Note that even when the ℓ smaller patches are not impulse functions, the optimal solution behaves similarly; that is, the great majority of representation levels are allocated to the larger patch. This is due to the quadratic penalty for input points that are far from their representative (1.1).

To demonstrate that Observation 1 is applicable in practice, consider the probability density function depicted in Figure 4.5. This $p(\vec{x})$ is comprised of one large patch and seven small ones, all of which have the same mass. $p(\vec{x})$ is formally defined as

$$p(x,y) = \begin{cases} 1 & (x,y) \in \{97,98,\ldots,158,159\} \times \{97,98,\ldots,158,159\}, \\ 441 & (x,y) \in \{36,37,38\} \times \{127,128,129\}, \\ 441 & (x,y) \in \{36,37,38\} \times \{218,219,220\}, \\ 441 & (x,y) \in \{127,128,129\} \times \{36,37,38\}, \\ 441 & (x,y) \in \{127,128,129\} \times \{218,219,220\}, \\ 441 & (x,y) \in \{218,219,220\} \times \{36,37,38\}, \\ 441 & (x,y) \in \{218,219,220\} \times \{127,128,129\}, \\ 441 & (x,y) \in \{218,219,220\} \times \{218,219,220\}, \\ 0 & \textit{otherwise}. \end{cases} \tag{4.2}$$

Figure 4.6 presents the quantization results of AMR and LBG on this $p(\vec{x})$ when $n = 8, 16, 32,$ and 64. Note the differences in the number of representation levels per patch for various n in the LBG results. These differences are due to the aforementioned randomized implementation and the independence of each execution. Table 4.2

demonstrates how the AMR solution for this $p(\vec{x})$ is generally between four and six times better than the LBG solution for various n. In general, the larger the number of patches and the smaller the area that each of the smaller patches covers, the worse the LBG solution becomes relative to the optimal solution and the AMR solution. Note that the computation time of the LBG algorithm in this example is quite fast. This is because the Lloyd–Max relaxations mainly affect the representation levels in the large patch. As the Lloyd–Max process's convergence rate is highly dependent on the number of representation levels, and as only a fraction of the original number of representation levels n are active, the LBG algorithm converges faster than for a more general $p(\vec{x})$. The gain of AMR over LBG is not as high as predicted by Observation 1 only because the patches are not impulse functions.

TABLE 4.2

Results of LBG and AMR applied to the eight-patch $p(\vec{x})$ defined by (4.2) *and depicted in Figure* 4.5. *This table's structure is identical to that of Table* 4.1.

n	**AMR iter.**			**AMR dist.**		**LBG iter.**		**LBG dist.**		**LBG:AMR**	
8	10	56	57	$5.00 \cdot 10^7$	$2.66 \cdot 10^6$	11	24	$1.66 \cdot 10^7$	$1.64 \cdot 10^7$	0.33	6.16
16	13	56	94	$4.27 \cdot 10^5$	$3.28 \cdot 10^5$	14	21	$1.69 \cdot 10^6$	$1.66 \cdot 10^6$	3.96	5.06
32	14	32	42	$1.47 \cdot 10^5$	$1.41 \cdot 10^5$	17	32	$6.76 \cdot 10^5$	$6.72 \cdot 10^5$	4.60	4.77
64	16	72	96	$7.78 \cdot 10^4$	$7.60 \cdot 10^4$	22	73	$3.88 \cdot 10^5$	$3.50 \cdot 10^5$	4.99	4.60

5. Conclusions and further discussion. With respect to the multiple local minima problem, we have demonstrated an adaptive multiscale redistribution (AMR) strategy based on the point density function and distortion attributes of an optimal quantizer.[1] This strategy is nonlocal and may determine from a local glance that the solution is not globally optimal, an attribute not shared by the Lloyd–Max method. Our strategy includes a fast method for obtaining an initial guess, as well as a method for improving a guess and bringing it to a local minimum. This strategy may beat LBG by an arbitrary factor, and it may be substantially faster than LBG.

In a previous paper [20] we have presented a multigrid acceleration of the 1-D Lloyd–Max iterative process. That is, given an initial guess, a local minimum near it may be reached at a complexity equivalent to just a few Lloyd–Max iterations. Using the AMR methods for the scalar case may be very powerful. This is because, at the guess improvement stage, we may replace every application of Lloyd–Max iterations (step 2(b)(i) of $Improve-Guess$) with our monotonically convergent multigrid solver [20]. Doing so implies that we need not guess if a transition of a representation level is worthwhile. Instead, we can use our multigrid method to arrive at a local minimum and compare its distortion with the best distortion to date.

An interesting question is whether or not an acceleration scheme similar to that of the scalar quantization problem exists for vector quantization. We believe that a standard multigrid approach, such as the one in [20], cannot work since the problem does not behave "elliptically." A small change in the solution at some point may cause a large change in the solution at some other distant point. For this reason, the Lloyd–Max relaxations do not smooth the distortion/residual, which is a basic demand of multigrid schemes.

We believe that the ideas proposed in this paper may be applied to other interesting representation problems, for example, representing a k-dimensional manifold $S(\vec{x})$ by n points. From these n points, given some interpolation scheme, one can

[1] Implementations are available at http://www.cs.technion.ac.il/~irad/Quantization/index.htm.

compute an approximating manifold $S'(\vec{x})$. The objective is to minimize

$$\int \|S(\vec{x}) - S'(\vec{x})\|^2 d\vec{x}. \tag{5.1}$$

With suitable adaptations, our proposed methods may also be able to deal with other image processing problems such as halftoning, where the average color value of the halftoned image should equal the average color value of the original image at all scales. These problems are currently being researched.

Appendix. The importance of *multiscale* redistribution. Performing redistribution on the finest scale alone yields noteworthy results. Nevertheless, wrapping the redistribution operation within a multiscale framework yields a noticeably better performance, as the following observation implies.

OBSERVATION 2.

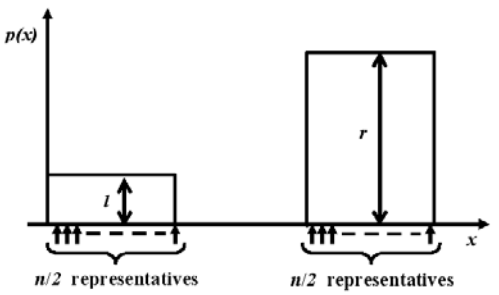


FIG. A.1. *Probability density function $p(x)$ with two patches.*

. Consider the probability density function

$$p(x) = \begin{cases} l : 0 \le x \le 1/3, \\ r : 2/3 \le x \le 1, \\ 0 : otherwise. \end{cases} \tag{A.1}$$

For simplicity, we choose a 1-D $p(x)$. Similar examples exist for any dimension. Assume that $n/2$ representation levels are allocated to each of the two patches of $p(x)$. This configuration is depicted in Figure A.1. For any l and r, uniformly distributing the representation levels in each patch yields a local minimum of the distortion. For any such local minimum, in each patch separately, the wealth of any two decision regions is equal since $p(x)$ is constant. Denote by $\mathcal{W}_l$ and $\mathcal{W}_r$ the wealth of any decision region in the left and right patches, respectively. Evidently, there exist values of l and r such that $\mathcal{W}_l = T/2 - \epsilon$ and $\mathcal{W}_r = -T/2 + \epsilon$, where T is either the chosen threshold in the $Improve - Guess$ algorithm or $T = 1$ for the $Obtain - Guess$ algorithm. Recall that representation levels are redistributed if and only if $\mathcal{W}_l - \mathcal{W}_r > T$. At the finest scale, no redistribution occurs since $\mathcal{W}_l - \mathcal{W}_r = T - 2\epsilon$. At the second finest scale, where Ω is partitioned into, say, $n/2$ aggregates of decision regions, $\mathcal{W}_l = T - 2\epsilon$ and $\mathcal{W}_r = -T + 2\epsilon$, which implies $\mathcal{W}_l - \mathcal{W}_r > T$ for small enough ϵ, and therefore redistribution occurs. In fact, $n/4$ representation levels are transferred from

the left patch of $p(x)$ to the right patch at this scale. For general scale m, where Ω is partitioned into m aggregates of decision regions after approximately $\log_2(m)$ aggregations, the proof is similar. There exist values of l and r such that, at the finest scale, $\mathcal{W}_l = T/(n/m) - \epsilon$ and $\mathcal{W}_r = -T/(n/m) + \epsilon$. All scales finer than m fail to comply with the redistribution criterion $\mathcal{W}_l - \mathcal{W}_r > T$. However, at scale m, for small enough ϵ, the criterion is satisfied and redistribution occurs because the wealth per m-scale aggregate, which consists of n/m representation levels, is equal to n/m times the fine scale wealth. In fact, approximately $m/2$ representation levels are transferred from the left patch of $p(x)$ to the right patch at this scale. □

REFERENCES

[1] R. Balasubramanian, C. A. Bouman, and J. P. Allebach, *Sequential scalar quantization of vectors: An analysis*, IEEE Trans. Image Process., 4 (1995), pp. 1282–1295.

[2] W. R. Bennett, *Spectra of quantized signals*, Bell Systems Tech. J., 27 (1948), pp. 446–472.

[3] A. Brandt, *Multi-level adaptive solutions to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[4] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[5] R. Clark, *An extended dissimilarity selection method for finding diverse representative subsets*, J. Chem. Inform. Comput. Sci., 37 (1997), pp. 1181–1188.

[6] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices and Groups*, Springer-Verlag, New York, 1988.

[7] Q. Du, V. Faber, and M. Gunzburger, *Centroidal Voronoi tesselations: Applications and algorithms*, SIAM Rev., 41 (1999), pp. 637–676.

[8] P. Fleischer, *Sufficient conditions for achieving minimum distortion in a quantizer*, IEEE Internat. Convention Record, (1964), pp. 104–111.

[9] J. H. Freidman, J. L. Bentley, and R. A. Finkel, *An algorithm for finding best matches in logarithmic expected time*, ACM Trans. Math. Software, 3 (1977), pp. 209–226.

[10] A. Gersho, *Asymptotically optimal block quantization*, IEEE Trans. Inform. Theory, 25 (1979), pp. 373–380.

[11] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.

[12] R. M. Gray and E. D. Karnin, *Multiple local optima in vector quantizers*, IEEE Trans. Inform. Theory, 28 (1982), pp. 256–261.

[13] R. M. Gray and D. L. Neuhoff, *Quantization*, IEEE Trans. Inform. Theory, 44 (1998), pp. 1–63.

[14] J. Hartigan, *Clustering Algorithms*, Wiley-Interscience, New York, 1975.

[15] S. Hasegawa, H. Imai, M. Inaba, N. Katoh, and J. Nakano, *Efficient algorithms for variance-based k-clustering*, in Proceedings of the First Pacific Conference on Computer Graphics and Applications, IEEE, Piscataway, NJ, 1993, pp. 75–89.

[16] P. Heckbert, *Color image quantization for frame buffer display*, in Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Images, ACM, New York, 1982, pp. 297–307.

[17] H. Honda, *Description of cellular patterns by Dirichlet domains: The two-dimensional case*, J. Theoret. Biol., 72 (1978), pp. 523–543.

[18] H. Imai, N. Katoh, and M. Inaba, *Applications of weighted voronoi diagrams and randomization to variance-based k-clusteringgeometric k-clustering*, in Proceedings of the 10th ACM Symposium on Computational Geometry, ACM, New York, 1994, pp. 332–339.

[19] A. Jain and R. Dubes, *Algorithms for Clustering Data*, Prentice–Hall, Englewood Cliffs, NJ, 1998.

[20] Y. Koren, I. Yavneh, and A. Spira, *A multigrid approach to the* 1*-D quantization problem*, IEEE Trans. Inform. Theory, to appear.

[21] Y. Linde, A. Buzo, and R. M. Gray, *An algorithm for vector quantizer design*, IEEE Trans. Commun., 28 (1980), pp. 84–95.

[22] S. P. Lloyd, *Least squares quantization in PCM*, IEEE Trans. Inform. Theory, 28 (1982), pp. 129–137.

[23] J. MacQueen, *Some methods for classification and analysis of multivariate observations*, in Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics, and Probability, Vol. 1: Statistics, University of California Press, Berkeley, CA, 1967, pp. 281–296.

[24] J. Max, *Quantizing for minimal distortion*, IEEE Trans. Inform. Theory, 6 (1960), pp. 7–12.
[25] M. T. Orchard and C. A. Bouman, *Color quantization of images*, IEEE Trans. Signal Process., 39 (1991), pp. 2677–2690.
[26] P. F. Panter and W. Dite, *Quantizing distortion in pulse-count modulation with nonuniform spacing of levels*, in Proc. IRE, 39, 1951, pp. 44–48.
[27] L. Torres and J. Huguet, *An improvement on codebook search for vector quantization*, IEEE Trans. Commun., 42 (1994), pp. 208–210.
[28] L. F. Toth, *Sur la representation d'une population infinie par un nombre fini d'elements*, Acta Math. Acad. Sci. Hungar., 25 (1959), pp. 76–81.
[29] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, New York, 2001.
[30] S. J. Wan and S. K. M. Wong, *An algorithm for multidimensional data clustering*, ACM Trans. Math. Software, 14 (1988), pp. 153–162.
[31] X. Wu, *Color quantization by dynamic programming and principal analysis*, ACM Trans. Graphics, 11 (1992), pp. 348–372.
[32] X. Wu and L. Guan, *Acceleration of the LBG algorithm*, IEEE Trans. Commun., 42 (1994), pp. 1518–1523.
[33] P. L. Zador, *Development and Evaluation of Procedures for Quantizing Multivariate Distributions*, Ph.D. thesis, Stanford University, Stanford, CA, 1963. Also Stanford University Department of Statistics Technical Report.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1594–1607
© 2006 Society for Industrial and Applied Mathematics

A MULTILEVEL METHOD FOR IMAGE REGISTRATION*

ELDAD HABER† AND JAN MODERSITZKI†

Abstract. In this paper we introduce a new framework for image registration. Our formulation is based on consistent discretization of the optimization problem coupled with a multigrid solution of the linear system which evolves in a Gauss–Newton iteration. We show that our discretization is h-elliptic independent of parameter choice, and therefore a simple multigrid implementation can be used. To overcome potential large nonlinearities and to further speed up computation, we use a multilevel continuation technique. We demonstrate the efficiency of our method on a realistic highly nonlinear registration problem.

Key words. image registration, multigrid, multilevel, optimization

AMS subject classifications. 92C55, 65M55, 15A23, 65K10

DOI. 10.1137/040608106

1. Introduction and problem setup. Image registration is one of today's challenging image processing problems. Given a so-called reference R and a so-called template image T, the basic idea is to find a transformation such that a transformed version of the template image becomes to the reference image. Image registration has to be applied whenever images resulting from different times, devices, and/or perspectives need to be compared or integrated; see, e.g., [6, 19] and references therein.

The computation of nonrigid image registration has two main building blocks. The first one is a distance measure $\mathcal{D}$ quantifying distance or similarity of images, and the second one is a regularizer $\mathcal{S}$ which rules out nonreasonable solutions. Note that image registration is an ill-posed problem; see, e.g., [19]. Therefore, regularization is inevitable. A common treatment of the registration problem is based on the following variational approach: Find a transformation $\boldsymbol{u}$ minimizing the joint energy

$$\mathcal{J}^{\alpha}[\boldsymbol{u}] := \mathcal{D}(R, T(\boldsymbol{u})) + \alpha \mathcal{S}(\boldsymbol{u}). \tag{1}$$

Here, $\alpha > 0$ is a regularization parameter that compromises similarity and regularity.

The formulation (1) is common to many ill-posed inverse problems where the data fitting term $\mathcal{D}$ is balanced with a regularization term using the regularization parameter α. Many authors have dealt with the optimal choice of α, using a variety of approaches; see, for example, [14, 24]. If the difference between the images is associated with random noise, then standard methods that are based on the statistics of the noise, such as generalized cross validation (GCV) [9] or χ^2 [21], can be used. However, in image registration the difference between two images (at least in most medical applications) is not associated with random noise. In fact, the difference may be highly structured, and only a trained professional (e.g., a radiologist) could determine the exact amount of regularization needed. Therefore, we prefer to provide a sequence of registered image pairs for various α's and leave the final choice of an optimal α to a trained professional.

*Received by the editors May 12, 2004; accepted for publication (in revised form) January 3, 2005; published electronically February 3, 2006. This research was supported by NSF grant CCF-0427094.
http://www.siam.org/journals/sisc/27-5/60810.html
†Department of Mathematics and Computer Science, Emory University, Atlanta GA 30322 (haber@mathcs.emory.edu, modersit@mathcs.emory.edu).

In general, the optimization problem (1) cannot be solved analytically. Thus, numerical schemes and appropriate discretizations are required. Typically, a discretization leads to a finite dimensional nonlinear problem, where the number of unknowns can be large. For example, the discrete registration problem for two $128 \times 128 \times 64$ MRI scans results in a nonlinear system of equations in about $3 \cdot 10^6$ unknowns. Thus, highly efficient algorithms are required in order to perform the registration in a reasonable amount of time. This applies, in particular, if one is interested not only in a solution for a fixed α but in a sequence of solutions for a variety of different α's. Furthermore, since we want our codes to run in a clinical setting, we require only modest computational hardware.

There are two main approaches for the discretization of the registration problem (1). In the first, the [illegible] approach, one forms the objective function and then differentiates to obtain the continuous Euler–Lagrange equations, which are finally discretized and solved numerically; see, e.g., [7, 15, 19]. The second approach is the [illegible] approach. Here, one directly discretizes the problem and then solves a finite (but typically high) dimensional optimization problem; see, e.g., [13]. The advantage of the latter approach is that standard optimization methods can be used. We prefer the discretize-optimize approach; however, in order to take advantage of efficient optimization techniques, all parts of the discrete problem need to be continuously differentiable. We discuss the implications and advantages of this necessity in the paper.

To gain additional computational efficiency and obtain a scalable algorithm, one can either use a full approximation multigrid scheme (FAS) or a multilevel Newton-type scheme with an efficient solver for the linearized systems of equations. In this paper we propose a multilevel inexact Gauss–Newton scheme combined with a multigrid solver for a computation of a numerical solution of (1). Although no proof exists, the latter approach has an efficiency similar to that of FAS [23, p. 154], but it is much more modular. Particularly, the linear solver is decoupled from the optimization strategy. Therefore, the code can be modified easily to deal with any differentiable distance measure and/or regularizer. However, for the sake of simplicity, we focus on the so-called [illegible] (SSD) as a distance measure $\mathcal{D}$ and the elastic potential as a regularizer $\mathcal{S}$:

$$\mathcal{D}[\boldsymbol{u}] = \frac{1}{2}\int_\Omega (T(\boldsymbol{x}+\boldsymbol{u}(\boldsymbol{x})) - R(\boldsymbol{x}))^2 \, d\boldsymbol{x}, \tag{2a}$$

$$\begin{aligned}\mathcal{S}[\boldsymbol{u}] &= \int_\Omega \langle \mathcal{B}\boldsymbol{u}, \mathcal{B}\boldsymbol{u}\rangle \, d\boldsymbol{x} \\ &:= \frac{1}{2}\left\| \begin{pmatrix} \sqrt{\lambda+\mu}\,\nabla\cdot & & \\ \sqrt{\mu}\,\nabla & & \\ & \ddots & \\ & & \sqrt{\mu}\,\nabla \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_d \end{pmatrix} \right\|^2_{L_2(\Omega)} \\ &= \int_\Omega \frac{\lambda+\mu}{2} \| \nabla\cdot \boldsymbol{u}\|^2 + \frac{\mu}{2}\sum_{i=1}^{d} \| \nabla \boldsymbol{u}_i\|^2 \, d\boldsymbol{x},\end{aligned} \tag{2b}$$

where $\Omega = (0,1)^d$ is the registration domain and λ and μ are the Lamé constants.

Our discretization is based on the weak form of the problem (1) which naturally leads to staggered grid discretization. To the best of our knowledge, staggered grid discretization was introduced into imaging in [13]. In this paper, we prove the h-ellipticity of the discretization for the elastic potential, independent of the problem parameters. As has already been shown in [13], this discretization is also stable for pointwise volume preserving constraints.

Our work relates to the work of Henn and Witsch [15], Clarenz, Droske, and Rumpf [7], and Kalmoun and Rüde [18]. In [15], an FAS based on the discretized optimality conditions for the continuous problem is presented. In [7], a diffusive rather than an elastic regularizer is used. A d-linear interpolation scheme is used to compute the deformed template in [7, 15, 18]. Therefore, the objective function is not continuously differentiable, and this may lead to failure of standard optimization techniques. In [18] a multigrid method is developed for the related optical flow problem but with the diffusive regularizer.

This paper is organized as follows. In section 2, we discuss the discretization of the registration problem (1). Our optimization approach, which is based on a Gauss–Newton-type scheme (cf., e.g., [20]), is discussed in section 3. In section 4, we propose a multigrid method for a solution of the linearized problems. Using a local mode analysis we show that our discretization is particularly amenable to multigrid methods and suggest appropriate multigrid treatment. In section 5, we combine our nonlinear iteration with a multilevel continuation method. Finally, in section 6, we present numerical examples that demonstrate the effectiveness of our algorithm.

2. Discretization. Choosing a stable discretization method for a system of partial differential equations with mixed derivatives is a delicate matter. Here, we use staggered grids (cf. Figure 1), which are very common for stable discretizations of fluid flow (see, e.g., [8]) and electromagnetics (see, e.g., [11, 27]), where operators such as the gradient, curl, and divergence are discretized. It is also well known that staggered grids are tightly connected to mixed finite element methods which are commonly used for elasticity [3].

In this section we briefly summarize the discretization we use. Further discussion and details are given in [13].

2.1. Discretizing $\boldsymbol{u}$. We assume that our discrete images have $m_1 \times \cdots \times m_d$ pixels, where $d = 2, 3$ is the image dimensionality. For ease of presentation, we also assume that each pixel is square or cubic, where each side has length h. In our description we allow for half-step indices. As usual in image processing, we identify pixels/voxels with cell centered grid points $x_{j,k,\ell}$, which are therefore labeled with full integer indices. Given a pixel/voxel $x_{j,k,\ell}$, their faces are numbered with a half index, $x_{j\pm\frac{1}{2},k,\ell}$, $x_{j,k\pm\frac{1}{2},\ell}$, and $x_{j,k,\ell\pm\frac{1}{2}}$, and we discretize the ith component u^i of $\boldsymbol{u}$ on the ith face for every pixel/voxel. With some abuse of notation, we denote the discrete analogue of the continuous vector field by $\boldsymbol{u} = (u^1, \ldots, u^d)^\top$, where u^i denotes the grid function which is approximated on the face staggered grid.

If needed, the derivatives $\partial_j u^k$ are approximated by the short (central) differences,

$$\partial_j u^k \approx \partial_j^h u^k := \frac{1}{h}\Big(u^k_{\ldots,i_j+\frac{1}{2},\ldots} - u^k_{\ldots,i_j-\frac{1}{2},\ldots}\Big). \tag{3}$$

Note that no boundary conditions are needed to approximate derivatives in the normal directions ($\partial_j u^j$). For the tangential directions ($\partial_j u^k$, $j \neq k$) we imposed Neumann boundary conditions.

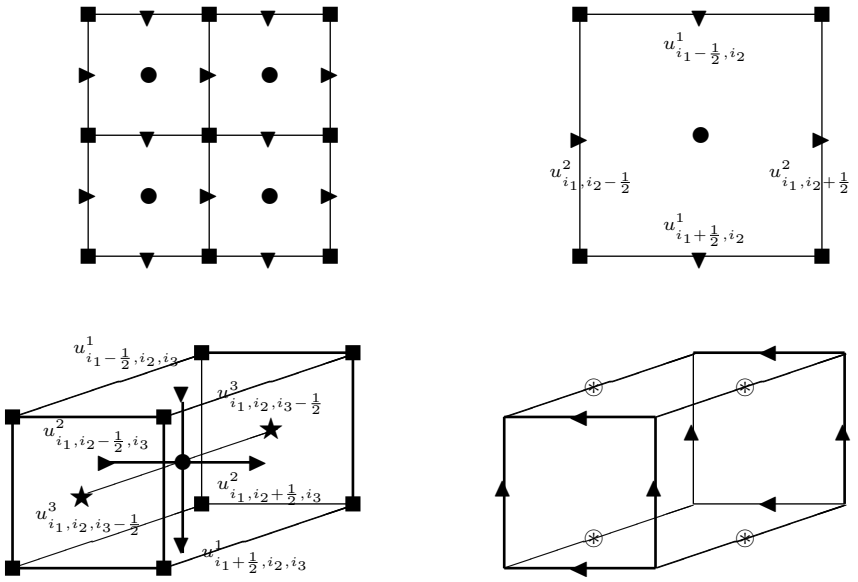


FIG. 1. *Staggered grids for dimensions $d = 2, 3$, nodal ■, cell centered •, face staggered grids (▼ in x_1-, ► in x_2-, and ★ in x_3-direction), and edge staggered grids (▲ in x_1-, ◄ in x_2-, and ⊛ in x_3-direction). Top left: $d = 2$, four pixels. Top right: $d = 2$ pixel (i_1, i_2) with grids. Bottom left: $d = 3$, voxel (i_1, i_2, i_3) with face staggered grids and positions of u^1, u^2, u^3. Bottom right: $d = 3$, edge staggered grids.*

2.2. Discretizing $\mathcal{S}$. Since many regularizers are phrased in terms of the more complex differential operators gradient ∇ and divergence $\nabla\cdot$, we introduce the notation ∇^h_j and $\nabla^h\cdot$ for the discrete analogues,

$$\nabla^h_j\, u^j = (\partial^h_1 u^j, \dots, \partial^h_d u^j)^\top \quad \text{and} \quad \nabla^h \cdot \boldsymbol{u} = \sum \partial^h_j u^j. \tag{4}$$

Using these discrete analogues the elastic potential $\mathcal{S}$ (2b) is discretized as

$$\mathcal{S}^h(\boldsymbol{u}) = \|\boldsymbol{B}\boldsymbol{u}\|_2^2 := \frac{\lambda+\mu}{2}\| \nabla^h \cdot \boldsymbol{u}\|^2 + \frac{\mu}{2}\sum_{i=1}^{d} \| \nabla^h_j \boldsymbol{u}_i\|^2. \tag{5}$$

In the course of the registration process we require derivatives. Upon differentiation of the regularizer we obtain the Navier–Lamé operator

$$\mathcal{S}^h_{\boldsymbol{u}}(\boldsymbol{u}) = (\lambda+\mu)(\nabla^h\cdot)^\top \nabla^h \cdot \boldsymbol{u} - \mu\Delta_h \boldsymbol{u} =: \boldsymbol{A}\boldsymbol{u}, \tag{6}$$

where Δ_h is the usual seven points discrete vector Laplacian. Note that the approach in [7] is diffusion based (i.e., setting $\lambda = -\mu$ in (2b) and (5)), such that coupling terms vanishes. In contrast to [15], where the strong form given by the product $\mathcal{B}^*\mathcal{B}$ is discretized, we discretize $\mathcal{B}$ and build the discrete product $\boldsymbol{A} := \boldsymbol{B}^\top \boldsymbol{B}$. A discretization of the product $\mathcal{S}^h_{\boldsymbol{u}} \approx \mathcal{B}^*\mathcal{B}$ may not be directly linked to a discrete objective function of the form $\int(\mathcal{B}\boldsymbol{u})^2 d\boldsymbol{x}$. Thus our discretization is a mimetic discretization (see [16]).

The advantage of this approach is that the derivative $\mathcal{S}_{\boldsymbol{u}}^h$ is automatically the derivative of the discrete function $\mathcal{S}^h$. Moreover, boundary conditions are naturally related to those of the operator $\mathcal{B}$.

2.3. Discretizing T and R. If we require a continuously differentiable objective function, we need a continuous image model. For noisy images we use a smoothing B-spline to approximate the images where the smoothing parameter is chosen using the GCV [9, 22]. For data interpolation using B-splines see [25]. Since the grid is regular, we can quickly evaluate the spline coefficients using a cosine transform. The continuous smooth approximation is denoted by T^{spline}.

We are looking for a fast and efficient optimization scheme, and therefore differentiability does play a key role. Thus, as previously discussed in [22], although computationally superior, d-linear image approximations cannot be used, since they are not continuously differentiable. In our implementation we use a cubic B-spline approximation. For further discussion on the effects of higher or lower order B-spline interpolation on the quality of the registration we refer to [22].

Given the staggered grid representation of $\boldsymbol{u}$ we use averaging operators P_j for the transfer to the cell centered positions, and we set

$$\begin{aligned}(P_j u^j)_{\dots,i_j,\dots} &= \frac{1}{2}\Big(u^j_{\dots,i_j+\frac{1}{2},\dots} + u^j_{\dots,i_j-\frac{1}{2},\dots}\Big)\\ T(\boldsymbol{u}) &:= T^{\text{spline}}(x^1 + P_1 u^1, \dots, x^d + P_d u^d);\end{aligned}$$

see [13] for details. We denote the Jacobian of T by

$$T_{\boldsymbol{u}} := \frac{\partial T}{\partial \boldsymbol{u}}(\boldsymbol{u}) = (\operatorname{diag}(P_1^\top \partial_1 T), \dots, \operatorname{diag}(P_d^\top \partial_d T)), \tag{7}$$

where the partial derivatives $\partial_j T$ are evaluated at the spatial positions $(x^1 + P_1 u^1, \dots, x^d + P_d u^d)$. Note that, using a spline approximation for T, $T_{\boldsymbol{u}}$ is a sparse matrix with only four nonzero diagonals.

2.4. Discretizing $\mathcal{D}$. Our discretization of SSD (2a) is straightforward:

$$\boldsymbol{D}(\boldsymbol{u}) := \frac{1}{2}\|T(\boldsymbol{u}) - R\|_2^2 \quad \text{and thus} \quad \boldsymbol{D}_{\boldsymbol{u}}(\boldsymbol{u}) = T_{\boldsymbol{u}}(\boldsymbol{u})^\top (T(\boldsymbol{u}) - R).$$

3. Optimization. The discretized analogue of the image registration problem (1) for fixed α reads as follows:

$$\text{find} \quad \boldsymbol{u} \quad \text{such that} \quad \boldsymbol{J}^\alpha(\boldsymbol{u}) := \boldsymbol{D}(\boldsymbol{u}) + \alpha \boldsymbol{S}(\boldsymbol{u}) = \min. \tag{8}$$

To solve this problem numerically we use an inexact Gauss–Newton method; cf., e.g., [20].

Starting with the initial guess $\boldsymbol{u}$, at each step we compute the gradient $\boldsymbol{J}_{\boldsymbol{u}}^\alpha$ of (8)

$$\boldsymbol{J}_{\boldsymbol{u}}^\alpha = T_{\boldsymbol{u}}(\boldsymbol{u})^\top (T(\boldsymbol{x} + \boldsymbol{u}) - R(\boldsymbol{x})) + \alpha \boldsymbol{A}\boldsymbol{u} \tag{9}$$

and approximately solve the system of linear equations

$$\boldsymbol{H}\delta_{\boldsymbol{u}} = -\boldsymbol{J}_{\boldsymbol{u}}^\alpha, \tag{10}$$

using a multigrid technique which is described in section 4. Here $\boldsymbol{H}$ is an approximation to the Hessian,

$$\boldsymbol{H} = \boldsymbol{M} + \alpha \boldsymbol{A}, \tag{11}$$

and $\boldsymbol{M}$ is an approximation to $T_{\boldsymbol{u}}(\boldsymbol{u})^\top T_{\boldsymbol{u}}(\boldsymbol{u})$. For example, for $d = 3$, we have

$$T_{\boldsymbol{u}}(\boldsymbol{u})^\top T_{\boldsymbol{u}}(\boldsymbol{u}) = \begin{pmatrix} (\partial_1 T)^2 & (\partial_1 T)(\partial_2 T) & (\partial_1 T)(\partial_3 T) \\ (\partial_1 T)(\partial_2 T) & (\partial_2 T)^2 & (\partial_2 T)(\partial_3 T) \\ (\partial_1 T)(\partial_3 T) & (\partial_2 T)(\partial_3 T) & (\partial_3 T)^2 \end{pmatrix}.$$

Although from an optimization standpoint it is better to use the matrix $\boldsymbol{M} = T_{\boldsymbol{u}}(\boldsymbol{u})^\top T_{\boldsymbol{u}}(\boldsymbol{u})$, the usage of this matrix results in a more complicated multigrid solver. As discussed in [18] a simple restriction cannot be used effectively, and an operator-induced interpolation is needed. Such interpolation requires more memory and is slower overall than the simple geometric multigrid scheme. In order to have a simple multigrid scheme, which uses less storage, we approximated this matrix by

$$\boldsymbol{M} = \begin{pmatrix} (\partial_1 T)^2 & 0 & 0 \\ 0 & (\partial_2 T)^2 & 0 \\ 0 & 0 & (\partial_3 T)^2 \end{pmatrix}.$$

Such a choice leads to linear convergence of the optimization scheme even if the problem has a low residual. Nevertheless, it is similar to other Newton multigrid methods [23], since each iteration can be done quickly; then our approach tends to be faster than the full Gauss–Newton approach.

The deformation is updated by $\boldsymbol{u} \leftarrow \boldsymbol{u} + \gamma\delta_{\boldsymbol{u}}$, where γ is chosen according to an Armijo line search, and the process is repeated until convergence, i.e., $\|\boldsymbol{u} - \boldsymbol{u}_{\text{old}}\| \leq \text{tol}$. The algorithm is summarized in Algorithm 1.

ALGORITHM 1. Gauss–Newton method for image registration:
$\boldsymbol{u} \leftarrow \text{GNIR}(\alpha, \boldsymbol{u})$;

```
while true do
  compute J_u^α and H;
  approximately solve H δ_u = −J_u^α;
  if ‖δ_u‖ < tol then
    break;
  end if
  using a line search set u ← u + γδ_u
end while
```

3.1. Continuation in α. As discussed in the introduction, the optimal value of the regularization parameter α is in general unknown a priori. In order to estimate a reasonable α, we follow the strategy suggested in [12]. Starting with a large α_0, we compute a sequence of solutions $\boldsymbol{u}^{\alpha_j}$, where $\alpha_{j+1} < \alpha_j$. This sequence can be viewed as a continuation with respect to the regularization parameter α.

For a large regularization parameter the registration problem is almost quadratic, and therefore its solution requires only a few iterations using a Newton-type optimization technique. Let $j = 0$; starting with α_j, we compute $d_j := \boldsymbol{D}(\boldsymbol{u}^{\alpha_j})$ and expect at least the solution $\boldsymbol{u}^{\alpha_0}$ to under-fit the data. We use $\boldsymbol{u}^{\alpha_j}$ as an excellent starting guess for the computation of the solution of the registration problem for α_{j+1}. We stop if a solution $\boldsymbol{u}^{\alpha_j}$ becomes too irregular. Regularity has to be monitored by a trained professional. Technically, the regularity parameter of our code is given by c. The algorithm is summarized in Algorithm 2.

ALGORITHM 2. Continuation in α: $[\boldsymbol{u}, \alpha] \leftarrow \text{continuation}(\boldsymbol{u}, \alpha, c)$

```
set u_0 ← u, choose η < 1;
while true do
  solve (1) using Alg. 1: u ← GNIR(α, u);
  if D(u) < c then
    break;
  end if
  relax the regularization parameter: α ← ηα;
end while
```

4. A multigrid solution of the linear systems. The challenging part within the nonlinear optimization is the solution of the linear systems. Each system can be very large and is strongly coupled, but it has a few characteristics which are particularly amenable to multigrid methods. We now discuss a multigrid method for the solution of the linear system. General comments on smoothing, prolongation, restriction, and coarse grid solution are given in [23].

4.1. *h*-ellipticity. To show that the discrete equations (10) are amenable to multigrid methods, we first analyze them by local Fourier analysis (LFA) and show that the system is h-elliptic. It is well known (see, e.g., [23, p. 121–129]) that h-ellipticity is crucial for any multigrid method to be effective. For simplicity, we perform the analysis in two dimensions, but it can be directly generalized into three dimensions.

Recall that the Hessian for the unconstrained optimization problem reads as

$$\boldsymbol{H} = \alpha \boldsymbol{A} + \boldsymbol{M},$$

where $\boldsymbol{M}$ is a nonnegative diagonal matrix. An LFA with freezing coefficients takes the worst-case scenario into account, i.e., $\boldsymbol{M} = \boldsymbol{0}$. Therefore, we need to study the matrix $\boldsymbol{A}$ alone, which is a discretization of the Navier–Lamé operator

$$\begin{aligned}\mathcal{A} = \mathcal{B}^*\mathcal{B} &:= -\mu\Delta - (\lambda+\mu)\nabla\nabla\cdot \\ &= \begin{pmatrix} (2\mu+\lambda)\partial_{x_1,x_1} + \mu\partial_{x_2,x_2} & (\lambda+\mu)\partial_{x_1,x_2} \\ (\lambda+\mu)\partial_{x_1,x_2} & \mu\partial_{x_1,x_1} + (2\mu+\lambda)\partial_{x_2,x_2} \end{pmatrix},\end{aligned}$$

where the so-called Lamé constants λ and μ reflect material properties. Staggered grid discretization leads to the following four stencils for the four parts of the operator (see [23, p. 302] for notation):

$$\left.\begin{aligned} S^{1,1} &= \begin{pmatrix} 0 & -a & 0 \\ -b & 2(a+b) & -b \\ 0 & -a & 0 \end{pmatrix}, & S^{1,2} &= c\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}, \\ S^{2,1} &= (S^{1,2})^\top, & S^{2,2} &= (S^{1,1})^\top, \end{aligned}\right\} \tag{12}$$

abbreviating $a = (2\mu+\lambda)$, $b = \mu$, and $c = \lambda+\mu$.

Applying the discretized operators $\boldsymbol{A}_h$ to a grid function $\chi(\boldsymbol{x}, \boldsymbol{\theta}) = e^{\frac{i}{h}\boldsymbol{\theta}^\top \boldsymbol{x}}$, we obtain the symbols of our discretized operators:

$$\widetilde{A}_h(\boldsymbol{\theta}) = \begin{pmatrix} 2aw_1 + 2bw_2 & -4c\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} \\ -4c\sin\frac{\theta_1}{2}\sin\frac{\theta_2}{2} & 2bw_1 + 2aw_2 \end{pmatrix},$$

where $w_j := 1 - \cos\theta_j = 2\sin^2\frac{\theta_j}{2}$. Its determinant is given by

$$\begin{aligned} d(\boldsymbol{\theta}) = \widetilde{d}(\boldsymbol{w}) &= 4abw_1^2 + 4abw_2^2 + 4(a^2+b^2)w_1w_2 - 4c^2w_1w_2 \\ &= 4ab(w_1+w_2)^2. \end{aligned} \tag{13}$$

THEOREM 4.1. [illegible] (12)

$$E_h(\boldsymbol{A}_h) = \frac{1}{4},$$

[illegible] independent [illegible] Following [23, section 8.3.2],

$$E_h(\boldsymbol{A}_h) := \frac{\min\{|\det \widetilde{A}_h| :\ \theta \in T^{\text{high}}\}}{\max\{|\det \widetilde{A}_h| :\ -\pi < \theta_j \le \pi\}}.$$

From (13), we see that the min and max are independent of λ and μ:

$$\begin{aligned} \max\{d(\boldsymbol{\theta}) :\ -\pi < \theta_j \le \pi\} &= \max\{\widetilde{d}(\boldsymbol{w}) :\ 0 \le w_j \le 1\} \\ &= \widetilde{d}(1,1) = 16ab = 16\mu(2\mu+\lambda), \\ \min\{d(\boldsymbol{\theta}) :\ \theta \in T^{\text{high}}\} &= \min\left\{\widetilde{d}(\boldsymbol{w}) :\ \frac{1}{2} \le |w_j| \le 1\right\} \\ &= \widetilde{d}\left(\frac{1}{2},\frac{1}{2}\right) = 4ab = 4\mu(2\mu+\lambda). \qquad \square \end{aligned}$$

[illegible] 4.2. The above theorem is remarkable because as the ratio λ/μ becomes larger, the system becomes more ill-conditioned. Nevertheless, the theorem suggests that an appropriate multigrid implementation works well regardless of that ratio. Note also that other iterative techniques (such as conjugate gradient) are very sensitive to this ratio while our multigrid method is not.

4.2. Smoothing. The h-ellipticity of our discretization (independent of choices of the Lamé constants λ and μ) guaranties that, for example, a Kaczmarz relaxation has smoothing properties. Here we have implemented a damped Jacobi smoother with damping parameter of 2/3. Since we want our code to run on modest computational architecture and use only a small amount of memory, our implementation does not generate the matrix and only matrix vector products are calculated.

4.3. Prolongation and restriction. To fulfill the relation between the order of the differential operator and the sum of the orders of prolongation and restriction, we use linear interpolation for prolongation and its transpose for the restriction. This is similar to prolongation and restriction used in computational fluid dynamics. For an elaborate discussion on these operations on a staggered grid see [26, p. 253].

4.4. Coarse grid operator. Although it is possible to use a Galerkin coarse grid operator for building the coarse grid system $\boldsymbol{A}_H$, a straightforward implementation requires either additional memory or additional computations. For problems with uniform and structured grids, one can use rediscretization to get a faster implementation. However, since rediscretization cannot be applied to the diagonal matrix $\boldsymbol{M}_h$, we choose the following approach leading to a diagonal $\boldsymbol{M}_H$. In the first stage we

extract the diagonal entries of $\boldsymbol{M}_h$; this corresponds to a vector with the same dimensions of $\boldsymbol{u}_h$. Second, we generate the diagonal matrix $\boldsymbol{M}_H$ by restricting this vector using the same restriction operator designed for $\boldsymbol{u}_h$, and set the diagonal elements of $\boldsymbol{M}_H$ to the restricted values.

4.5. Multigrid cycles. The above components are linked together to a V- or a W-cycle. In our implementation we use only a single V-cycle for each nonlinear iteration. As we show in the numerical experiments, a $V(3,1)$-cycle is very effective and decreases the residual in one or two orders of magnitude.

5. Multilevel continuation. Image registration can be highly nonlinear, and therefore, one may require many iterations to achieve a solution. An important method to save computational work and to deal with nonlinearities is to use a multilevel continuation. Multilevel continuation is well established for optimization problems and systems of nonlinear equations; see, e.g., [1, 2]. However, in image registration it has an additional advantage. Similar to [10, 4], we use the multilevel approach to efficiently identify the relevant range of α's. Thus, our multilevel approach uses continuation in both the grid and the regularization parameter. Note that even for approaches where the images may not be smooth, the desired solution $\boldsymbol{u}^\alpha$ is smooth. Therefore, multilevel methods provide an effective tool for the continuation problem.

Given an initial upper bound C for the image distance, using a secant method, we compute a parameter α_C, such that

$$\mathcal{D}(\boldsymbol{u}^{\alpha_C}) \approx C. \tag{14}$$

After α_C is computed, we use Algorithm 2 on the fine grid to obtain a sequence of solution $\boldsymbol{u}^\alpha$, where $c \leq \mathcal{D}(\boldsymbol{u}^\alpha) \leq C$. We summarize the multilevel algorithm in Algorithm 3.

ALGORITHM 3. Multilevel image registration: $\boldsymbol{u} \leftarrow \text{MLIR}$

```
choose u on the coarsest grid;
while true do
  solve the registration problem u = GNIR(u, α); cf. Alg. 1;
  if |D(u) − C| ≥ tol then
    adjust α using a secant method;
  end if
  if on finest grid then
    return;
  end if
  prolongate u to the finer grid: u ← prolongate(u);
end while
```

6. Numerical examples. In this section we perform numerical experiments that demonstrate the effectiveness of our algorithm. We present a registration of three-dimensional MRI scans of a human knee (images provided by Thomas Netsch, Philips Medical Systems, Hamburg, Germany). The problem is known to be hard because large nonlinear deformations are needed in order to perform the registration; see also [17].

As is apparent from Figure 2, the reference shows the knee in an almost straight position, and the template shows the knee in a bent position. The deformation is thus

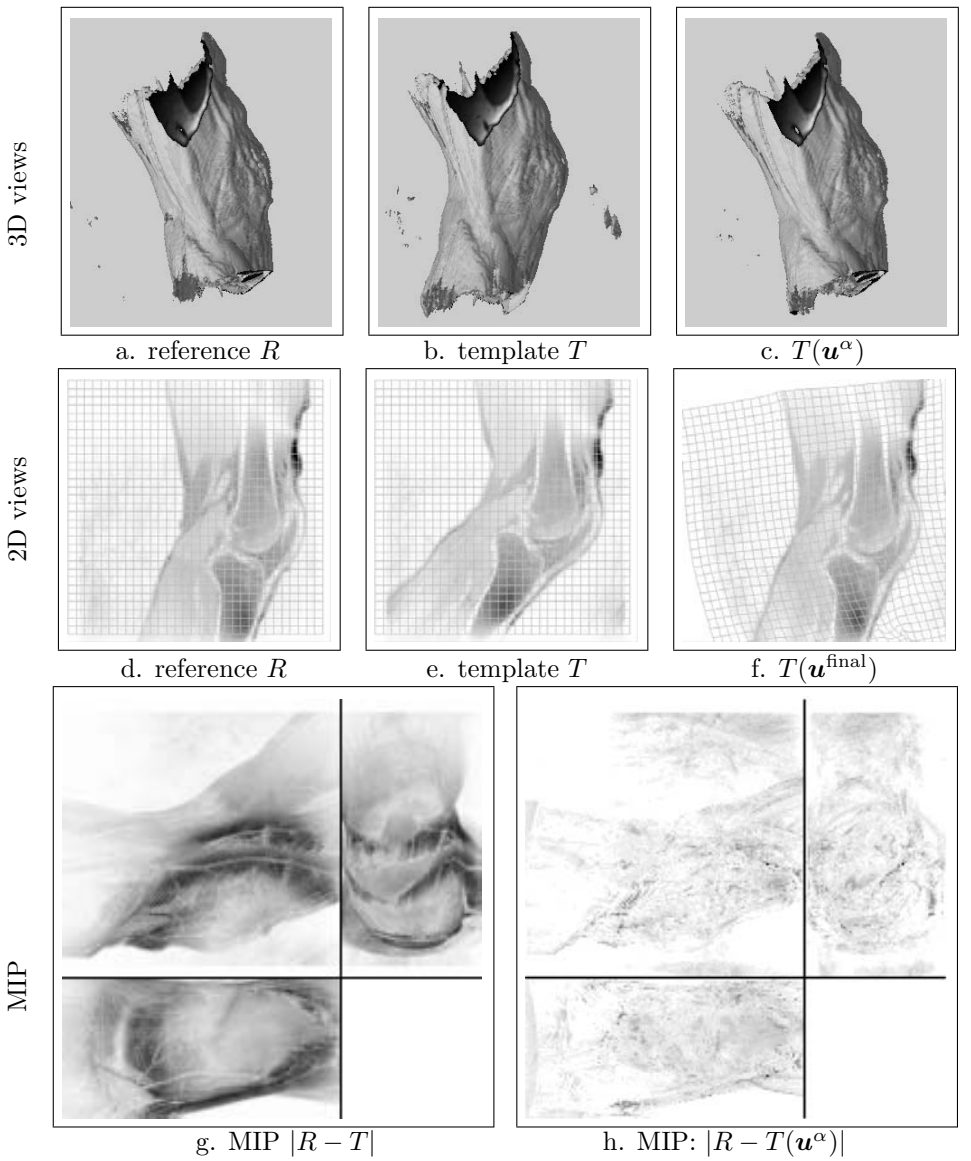


FIG. 2. *Registration results for two* $128\times128\times64$ *MRI scans of a human knee; three-dimensional views of* R (a), T (b), *and* $T(\boldsymbol{u}^{\alpha})$ (c)*; the generic slice* 40 *with grid for* R (d), T (e), *and* $T(\boldsymbol{u}^{\alpha})$ (f)*; maximum intensity projections for* $|R-T|$ (g) *and* $|R-T(\boldsymbol{u}^{\alpha})|$ (h).

expected to be highly nonlinear. To clarify the example, we divide this section into two parts. In the first part we go over the computational details, while in the second we discuss the results from an imaging point of view.

As common to many other registration algorithms we use a two-phase registration (see [19]). In the first phase a multilevel linear preregistration is done, followed by a second-phase multilevel nonlinear registration. The linear registration managed to reduce the image distance to 50.1%, and the rest was done using our nonlinear registration.

TABLE 1
Reductions in the SSD (compared with the initial SSD in %), and iterations (in #) for a decreasing α (image size $128 \times 128 \times 64$).

α	Iter	SSD
$8.5 \cdot 10^{-4}$	1	15.6%
$4.2 \cdot 10^{-4}$	1	12.2%
$2.1 \cdot 10^{-4}$	2	10.9%
$1.0 \cdot 10^{-4}$	2	10.1%

All computations are performed using MATLAB 6.1 on a 2.2GHz laptop with 1GB of RAM. The overall time for a registration ($\alpha = 10^{-3}$) was about 118 minutes, including the computation for the B-spline coefficients ($\approx$ 11 minutes). Since the signal-to-noise ratio in the images is high, for the computation of the B-spline we assume that the images do not contain any noise (thus the B-splines interpolate the images). In our computations, about one third of the time is needed for the linear part, and about two thirds is needed for the nonlinear part. Within the nonlinear part, 66% are needed for the computation of $T(\boldsymbol{u})$ and $T_{\boldsymbol{u}}(\boldsymbol{u})$, and 24% are needed to solve the linear system. An overview with respect to levels is presented in Table 2.

6.1. Computational details. We present three types of experiments. In our first example, we run the algorithm precisely as described in this paper (Algorithm 3); i.e., starting on the coarse grid we compute an initial α and adjust it by using the continuation techniques described in section 5. In order to present comparable multigrid convergence rates, we also present results for fixed values of α in our second example. Finally, in the third set of experiments we test the sensitivity of our multigrid algorithm to the elasticity parameters λ and μ. Although all image registration algorithms known to us fix this ratio at roughly 1, we experiment with a high ratio which challenges our multigrid method.

6.1.1. Continuation in grid and $\boldsymbol{\alpha}$. In our first numerical experiment we set $\mu = \lambda = 1$ and the constant C in Algorithm 3 to 0.2. Using Algorithm 3 we solve the problem, and this yields α which is $1.71 \cdot 10^{-3}$ (performance and results for $\alpha = 1.71 \cdot 10^{-3}$ and $\alpha = 10^{-3}$ are indistinguishable).

We then follow Algorithm 2, gradually reducing the regularization parameter to 10^{-4} by solving the problem for 4 different α's. For each regularization parameter we use the solution for the previous α as described in Algorithm 2. The number of iterations for each α and the SSD is recorded in Table 1. We observe both the images and the distorted grid for each solution. We stop the iteration at $1.0 \cdot 10^{-4}$ because we observe that the grid is highly distorted. The solution converges quickly (1 or 2 steps) for each new regularization parameter when starting from the solution of the previous regularization parameter. This demonstrates the importance of continuation in α for the robustness of the algorithm.

6.1.2. Sensitivity of the algorithm to the regularization parameter. In our second experiment we test the effect of different regularization parameters on the efficiency of our algorithm. We run our code with the (fixed) regularization parameters $\alpha = 10^{-2}, 10^{-3}, 10^{-4}$. The results of these experiment are documented in Table 2.

As common to many inverse problems, convergence of the optimization algorithm slows down as the regularization parameter decreases (see [12]). This implies that starting with an arbitrary guess on the fine grid may result in many more expensive iterations. Nevertheless, using a multilevel continuation approach, we are able to

TABLE 2
Reductions in the SSD (compared with the initial SSD in %), and iterations (in #) for the multilevel elastic registration; image size is $n^3/2$.

Level	n	Iterations			SSD		
		$\alpha = 10^{-4}$	$\alpha = 10^{-3}$	$\alpha = 10^{-2}$	$\alpha = 10^{-4}$	$\alpha = 10^{-3}$	$\alpha = 10^{-2}$
5	8	21	13	10	26.1%	17.2%	11.3%
4	16	18	16	12	25.9%	17.8%	10.9%
3	32	12	9	7	25.5%	17.9%	10.7%
2	64	9	8	6	25.3%	17.2%	10.2%
1	128	7	6	5	25.3%	17.2%	10.1%

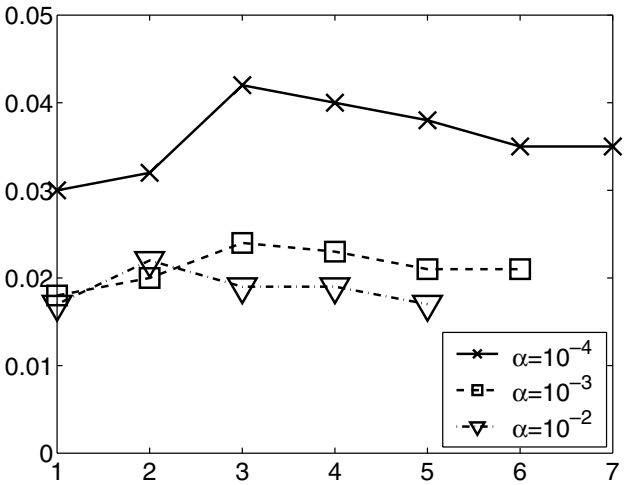


FIG. 3. *Convergence rate for each linear iteration versus iteration for $\alpha = 10^{-2}, 10^{-3}, 10^{-4}$; see also Table 2.*

quickly converge on the finest grid because we have a good starting guess which is obtained on coarser grids. This demonstrates the importance of multilevel continuation.

To observe the effectiveness of our multigrid algorithm solver we record the reduction of the residual for each of the linearized Gauss–Newton iterations on the finest grid. Note that for each iteration we have a different approximation to the Hessian characterized by the different $\boldsymbol{M}$'s. In our experiment, the multigrid solver was very effective, and after only a single $V(3,1)$-cycle our residual is decreased by approximately two orders of magnitudes. The convergence history is plotted in Figure 3.

The convergence properties of the linear iteration was only mildly influenced by the decreasing regularization parameter. This observation is similar to other multigrid methods for inverse problems [5].

6.1.3. Sensitivity of the algorithm to the ratio μ/λ. In the previous experiments, common to many other image registration algorithms [15], we used the ratio $\mu/\lambda = 1$. However, as Theorem 1 shows, the h-ellipticity of our discretized system is 0.25 independent of this choice, which implies in particular that the multigrid iteration should not be affected by this ratio. To put this theory to a test, we vary the ratio μ/λ, starting at 10 to 10^{-3}. Note that for the latter, the regularization operator is close to $\| \nabla \cdot \boldsymbol{u}\|^2$, which is the incompressible limit. However, we emphasize that

TABLE 3
Number of iterations and average residual reduction for different ratios μ/λ.

Ratio	10^1	10^0	10^{-1}	10^{-2}	10^{-3}
Iterations	18	17	16	16	17
Reduction	$2.3 \cdot 10^{-2}$	$2.1 \cdot 10^{-2}$	$2.4 \cdot 10^{-2}$	$2.8 \cdot 10^{-2}$	$2.9 \cdot 10^{-2}$

such a large ratio is meaningless for the purpose of image registration since it results in very irregular grids. From an imaging point of view, only the ratio $\mu/\lambda = 1$ leads to a reasonable displacement (judging by eye inspection). Our experiments are done solely to test our multigrid method.

For these experiments we set $\alpha = 10^{-3}$, and in order to have a long enough iteration history, we run the inversion process on the fine grid alone. For each nonlinear (Gauss–Newton) iteration we perform a single V(3,1)-cycle. For each cycle we record the reduction of the residual. At the end of each registration process we average the reduction of the residuals and record it as the [illegible]. The results of these experiments are summarized in Table 3.

In accordance with the theory, our experiments demonstrate that the multigrid method is in fact not very sensitive the ratio μ/λ. In fact, although the condition number for the system which results from $\mu/\lambda = 10^{-3}$ is roughly 10^3 times the condition number of the system which results from $\mu/\lambda = 1$, the reduction in the residuals (and therefore the convergence) is very minimally influenced. Note that this property is unique to multigrid methods, and a naively preconditioned conjugate gradient method would not have had such properties.

6.2. Qualitative analysis of the images. In Figure 2 we show the results for $\alpha = 10^{-3}$. As is apparent from Figure 2(c), the overall registration is reasonable. We also analyze the results in terms of a pairwise comparison of image slices. A two-dimensional view of the generic slice 40 of a total of 64 is shown in Figure 2(d), (e), (f). We add a regular grid to R and T and the deformed grid to $T(\boldsymbol{u}^\alpha)$. Though highly nonlinear, the deformation appears to be sufficiently smooth, as can be seen from the deformed grid. Finally, Figure 2 also shows maximum intensity projections of the distances $|R - T|$ and $|R - T(\boldsymbol{u}^\alpha)|$, respectively.

Running our scheme with a regularization parameter $\alpha = 10^{-2}$ leads to visually indistinguishable results but to a much higher distance (reduction to 25.3% instead of 17.2% for $\alpha = 10^{-3}$). On the other hand, choosing $\alpha \leq 10^{-4}$ results in nonphysical deformations which are visible on the deformed grid. Therefore, it is easy to derive a lower bound for α. However, choosing an "optimal" regularization parameter α is a delicate matter since it requires a quantification of "physically meaningful." We do not believe in an automatic detection and therefore leave the final choice to an expert.

7. Summary. In this paper we have developed a multilevel registration scheme for image registration based on a variational formulation. The scheme is based on a multilevel inexact Gauss–Newton scheme with a multigrid solver for the linearized systems. We make extensive use of numerical optimization techniques and therefore pay special attention to differentiability issues. We use smoothing B-splines to compute the solution to the forward problem and staggered grids to discretize the transformation. We prove that the discretization is h-elliptic and therefore is amenable to multigrid methods. The numerical experiments support our theoretical prediction and show that we can solve large scale problems using computational tools like MATLAB.

REFERENCES

[1] J. C. Alexander and J. A. Yorke, *The homotopy continuation method: Numerically implementable topological procedures*, Trans. Amer. Math. Soc., 242 (1978), pp. 271–284.

[2] E. Allgower and K. Georg, *Numerical Continuation Methods*, Springer-Verlag, Berlin, 1990.

[3] D. N. Arnold and R. Winther, *Mixed finite elements for elasticity*, Numer. Math., 42 (2002), pp. 401–419.

[4] U. Ascher and E. Haber, *Grid refinement and scaling for distributed parameter estimation problems*, Inverse Problems, 17 (2001), pp. 571–590.

[5] U. Ascher and E. Haber, *A multigrid method for distributed parameter estimation problems*, Electron. Trans. Numer. Anal., 15 (2003), pp. 1–15.

[6] L. G. Brown, *A survey of image registration techniques*, ACM Computing Surveys, 24 (1992), pp. 325–376.

[7] U. Clarenz, M. Droske, and M. Rumpf, *Towards fast non-rigid registration*, in Inverse Problems, Image Analysis and Medical Imaging, Contemp. Math. 313, M. Z. Nashed and O. Scherzes, eds., AMS, Providence, RI, 2002, pp. 67–84.

[8] C. A. J. Fletcher, *Computational Techniques for Fluid Dynamics*, Vol. II, Springer-Verlag, Berlin, 1988.

[9] G. Golub, M. Heath, and G. Wahba, *Generalized cross-validation as a method for choosing a good ridge parameter*, Technometrics, 21 (1979), pp. 215–223.

[10] E. Haber, *A multilevel, level-set method for optimizing eigenvalues in shape design problems*, J. Comput. Phys., 98 (2004), pp. 518–534.

[11] E. Haber and U. M. Ascher, *Fast finite volume simulation of 3D electromagnetic problems with highly discontinuous coefficients*, SIAM J. Sci. Comput., 22 (2001), pp. 1943–1961.

[12] E. Haber, U. Ascher, and D. Oldenburg, *On optimization techniques for solving nonlinear inverse Problems*, Inverse Problems, 16 (2000), pp. 1263–1280.

[13] E. Haber and J. Modersitzki, *Numerical solutions of volume preserving image registration*, Inverse Problems, 20 (2004), pp. 1621–1638.

[14] P. C. Hansen, *Rank-Deficient and Ill-Posed Problems: Numerical Aspects of Linear Inversion*, SIAM, Philadelphia, 1997.

[15] S. Henn and K. Witsch, *Iterative multigrid regularization techniques for image matching*, SIAM J. Sci. Comput., 23 (2001), pp. 1077–1093.

[16] J. M. Hyman and M. Shashkov, *The orthogonal decomposition theorems for mimetic finite difference methods*, SIAM J. Numer. Anal., 36 (1999), pp. 788–818.

[17] S. Kabus, T. Netsch, B. Fischer, and J. Modersitzki, *B-spline registration of 3d images with Levenberg-Marquardt optimization*, in Proceedings of the SPIE 2004, Medical Imaging, San Diego, 2004.

[18] E. M. Kalmoun and U. Rüde, *A Variational Multigrid for Computing the Optical Flow*, Tech. report, Department of Computer Science, Friedrich-Alexander University, Erlangen-Nuremberg, 2003.

[19] J. Modersitzki, *Numerical Methods for Image Registration*, Oxford University Press, New York, 2004.

[20] J. Nocedal and S. Wright, *Numerical Optimization*, Springer-Verlag, New York, 1999.

[21] R. L. Parker, *Geophysical Inverse Theory*, Princeton University Press, Princeton, NJ, 1994.

[22] P. Thevenaz and M. Unser, *Optimization of mutual information for multiresolution image registration*, IEEE Trans. Image Process., 9 (2000), pp. 2083–2089.

[23] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, San Diego, 2001.

[24] C. R. Vogel, *Computational Methods for Inverse Problem*, SIAM, Philadelphia, 2002.

[25] G. Wahba, *Spline Models for Observational Data*, SIAM, Philadelphia, 1990.

[26] P. Wesseling, *An Introduction to Multigrid Methods*, John Wiley & Sons, Chichester, UK, 1992.

[27] K. S. Yee, *Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media*, IEEE Trans. Antennas and Propagation, 14 (1966), pp. 302–307.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1608–1626
© 2006 Society for Industrial and Applied Mathematics

ON IMPROVING LINEAR SOLVER PERFORMANCE: A BLOCK VARIANT OF GMRES*

A. H. BAKER†, J. M. DENNIS‡, AND E. R. JESSUP§

Abstract. The increasing gap between processor performance and memory access time warrants the re-examination of data movement in iterative linear solver algorithms. For this reason, we explore and establish the feasibility of modifying a standard iterative linear solver algorithm in a manner that reduces the movement of data through memory. In particular, we present an alternative to the restarted GMRES algorithm for solving a single right-hand side linear system $Ax = b$ based on solving the block linear system $AX = B$. Algorithm performance, i.e., time to solution, is improved by using the matrix A in operations on groups of vectors. Experimental results demonstrate the importance of implementation choices on data movement as well as the effectiveness of the new method on a variety of problems from different application areas.

Key words. GMRES, block GMRES, iterative methods, Krylov subspace, memory access costs

AMS subject classifications. 65F10, 65Y20

DOI. 10.1137/040608088

1. Introduction.

We consider the solution of a large sparse system of linear equations:

$$Ax = b, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^n$. Iterative methods are often chosen to solve such linear systems, and when A is nonsymmetric, the GMRES (generalized minimum residual) [38] method is a common choice. Because linear systems are ubiquitous in science and engineering applications, improving the robustness of linear solvers continues to be of interest.

The two main costs associated with an iterative linear solver algorithm are the floating-point operations and the cost of moving data through memory, and performance may be improved by reducing either of these costs. Our focus is on achieving a balance between improving the efficiency of an iterative linear solver from a memory-usage standpoint while maintaining favorable numerical properties. In particular, the goal of this work is to demonstrate the feasibility of improving the performance

*Received by the editors May 12, 2004; accepted for publication (in revised form) January 11, 2005; published electronically February 3, 2006. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/27-5/60808.html

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Box 808 L-551, Livermore, CA 94551 (abaker@llnl.gov). The work of this author was primarily supported by the Department of Energy Computational Science Graduate Fellowship Program of the Office of Scientific Computing and Office of Defense Programs in the Department of Energy under contract DE-FG02-97ER25308. Portions of this work were performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

‡Scientific Computing Division, National Center for Atmospheric Research, Boulder, CO 80307-3000 (dennis@ucar.edu). The work of this author was supported by the National Science Foundation.

§Department of Computer Science, University of Colorado, Boulder, CO 80309-0430 (jessup@cs.colorado.edu). The work of this author was supported by National Science Foundation grant ACI-0072119.

of a common iterative linear solver (restarted GMRES), and likewise other iterative solvers, via a combination of a memory-efficient implementation and algorithmic modifications that reduce data movement.

In this paper, we report on our investigation into a memory-efficient iterative linear solver and describe the resulting new block method. We emphasize the implementation of the new method, numerous numerical test results, and evaluation of performance based on the tracking of data movement through parts of the memory hierarchy. In section 2 we discuss related background information. We outline our approach for improving restarted GMRES performance in section 3. Section 4 introduces our new block method. We discuss implications of implementation decisions on data movement and present computational results for a variety of problems in section 5. Finally, we give concluding remarks in section 6.

2. Background. Though the cost of a numerical linear algebra algorithm has traditionally been measured in terms of the floating-point operations required, a more memory-centric approach has long been advocated (e.g., see [19, 16, 23]). It is well known that matrix algorithm performance continues to be limited by the gap between microprocessor performance and memory access time (e.g., see [16, 45, 28, 26, 20, 1]). The discrepancy between DRAM (dynamic random access memory) access time and microprocessor speeds has increased by nearly 50% per year [35]. As a result, the percentage of overall application time spent waiting for data from main memory has increased [35]. The situation is compounded by advances in algorithm development that have decreased the total number of floating-point operations required by many algorithms [20]. Therefore, the number of floating-point operations required by an algorithm is not necessarily an accurate predictor of algorithm performance for a large matrix algebra problem [20]. In fact, performance bounds based on sustainable memory bandwidth (such as the STREAM [27] benchmark) are thought to predict algorithm performance most accurately [20, 1]. In particular, when solving a linear system, efficient data reuse (i.e., reducing data movement) is crucial to reducing an algorithm's memory costs [23, 16, 19].

Krylov subspace methods such as GMRES are based on an iteration loop that accesses the coefficient matrix A once per loop and performs a sparse matrix-vector multiply. Thus, a desirable modification in terms of reducing data movement is to perform more than one matrix-vector product for a single memory access of A. This optimization is natural when solving several linear systems with the same coefficient matrix A but different right-hand sides. Such linear systems can be written as a single block linear system

$$AX = B, \tag{2}$$

where $A \in \mathbb{R}^{n \times n}$, $X, B \in \mathbb{R}^{n \times s}$, and the columns of B are the different right-hand sides. By solving the block linear system (as opposed to solving s systems individually), A now operates on a group of vectors (, , ,) instead of a single vector at each iteration. The matrix A is accessed from memory fewer times than it would be if each system were solved individually.

O'Leary first introduced block iterative solvers for block linear systems with symmetric A with the block conjugate gradient (BCG) and other related algorithms in [34]. For nonsymmetric A, a block version of the GMRES algorithm (BGMRES) is first described in [43], and detailed descriptions can also be found in [39, 40, 37]. BGMRES is essentially identical to GMRES, except that operations are performed with multivectors instead of single vectors. As with GMRES, the resources required by BGMRES

may be impractical since storage and computational requirements increase with each iteration. In this case, a restarted version of BGMRES (BGMRES(m)) is commonly used in practice.

Several variants of the BGMRES method have been developed for both multiple and single right-hand side systems. For multiple right-hand side systems, a hybrid block GMRES method is presented in [39] that often solves the block system faster than solving each single system individually due in part to fewer memory accesses. In addition, block methods that augment the approximation space with eigenvector approximations are given in [22, 29]. These methods are particularly useful for matrices that are sufficiently close to normal and have a few particular eigenvalues that inhibit convergence.

Of primary interest to us is the use of a block method on a block linear system to solve a single right-hand side system. This strategy is briefly noted as a possibility in [34] for BCG algorithms. To our knowledge, the first mention of using block GMRES to solve a single right-hand side system occurs in [9]. In this work, Chapman and Saad suggest that convergence may be improved for $Ax = b$ by using a block method with approximate eigenvectors or random vectors for the additional right-hand side vectors, particularly when convergence is hampered by a few small eigenvalues.

Also of note is a block variant of GMRES for single right-hand sides systems in [25]. This method is mathematically equivalent to standard GMRES and is implemented as an s-step method (matrix A performs s consecutive matrix-vector multiplies at each iteration). The primary drawback is loss of accuracy with increasing block size (s). As with most blocked algorithms, the implementation allows for the use of level 3 BLAS (basic linear algebra subprograms) [14], and codes based on these routines often achieve good performance because they minimize data movement (e.g., see [13, 14, 20]).

3. Approach. Our approach to reducing data movement in restarted GMRES (GMRES(m)) is to modify the algorithm such that more than one matrix-vector product occurs for a single memory access of A. To this end, we investigate an alternative for solving a single right-hand side system as in (1) based on solving a corresponding block linear system as in (2). This algorithmic change requires the selection of additional starting vectors and right-hand sides.

Algorithmic changes that reduce memory access costs should not degrade the numerical properties of the original algorithm. Therefore, our goal in choosing appropriate families of additional right-hand sides and starting vectors was to select vectors that would accelerate convergence to the single right-hand side solution. For this reason, we looked to existing Krylov subspace method acceleration techniques.

It is well known that subspace information is lost in GMRES(m) due to restarting, and a number of acceleration methods attempt to in some sense compensate for this loss (e.g., see [30, 31, 42, 12, 9, 4]). Our block method derives from the LGMRES method [4], an augmented Krylov subspace method with appealing convergence properties. In particular, the extra right-hand sides used in the block method are the same as used to augment the standard GMRES approximation space in LGMRES.

In addition, beyond the algorithmic modifications required for a block formulation of GMRES(m), close attention to implementation techniques is essential for a memory-efficient, sparse linear solver. The compressed storage formats used for sparse matrices generally result in irregular patterns of memory referencing (e.g., see [18, 15]), and many approaches to improving the performance of the sparse matrix-vector multiply have been investigated (e.g., see [41, 36, 44, 19, 8, 23]). In our new

block method, we reduce the cost of the additional matrix-vector operations in each iteration loop via an efficient matrix-multivector multiply routine discussed in [20, 21] that improves the performance of [illegible] sparse matrix-vector multiplies. This approach allows the multiplication of a [illegible] of size four (i.e., a group of four vectors) by a matrix at about 1.5 times the cost of calculating a single matrix-vector product. We discuss this type of matrix-multivector multiply routine and associated routines in section 5.1.

4. A new block algorithm. In this section, we first briefly review the LGMRES method, of which the new algorithm is an extension. We then describe the new algorithm and its implementation and discuss additional considerations.

4.1. The LGMRES method. Consider GMRES(m) when solving (1). Restart parameter m denotes a fixed maximum dimension for the Krylov subspace. If convergence has not occurred at the end of m iterations, the algorithm restarts. We refer to the group of m iterations between successive restarts as a cycle. We denote the restart number with a subscript: x_i is the approximate solution after i cycles or $m \cdot i$ total iterations, and r_i is the corresponding residual ($r_i = b - Ax_i$). The minimum residual property requires that GMRES(m) find $x_{i+1} \in x_i + K_m(A, r_i)$ such that $r_{i+1} \perp AK_m(A, r_i)$ (e.g., see [37]), where $K_i(A, r_i) \equiv span\{r_i, Ar_i, \ldots, A^{i-1}r_i\}$ denotes an i-dimensional Krylov subspace.

Let $\hat{x}$ be the true solution to (1). We denote the true error after the ith restart cycle by e_i, where $e_i \equiv \hat{x} - x_i$. Then the vector

$$z_i \equiv x_i - x_{i-1} \tag{3}$$

is an approximation to that error after i restart cycles, where $z_i \equiv 0$ for $i < 1$.

An error approximation is a natural choice of vector with which to augment the next approximation space $\mathcal{K}_m(A, r_i)$, since augmenting with the true error would solve the problem exactly (e.g., see [17]). Furthermore, because $z_i \in \mathcal{K}_m(A, r_{i-1})$, it in some sense represents the space $\mathcal{K}_m(A, r_{i-1})$ generated and discarded in the previous cycle. Therefore, the LGMRES method accelerates convergence by appending k vectors that approximate the error from previous restart cycles to the standard Krylov approximation space at the end of each restart cycle. In particular, after $i+1$ restart cycles, LGMRES(m, k) finds an approximate solution to (1) in the following way:

$$x_{i+1} = x_i + q_{i+1}^{m-1}(A)r_i + \sum_{j=i-k+1}^{i} \alpha_{ij} z_j,$$

where polynomial q_{i+1}^{m-1} and α_{ij} are chosen such that $\|r_{i+1}\|_2$ is minimized.

4.2. The new algorithm: B-LGMRES. The augmentation scheme used by LGMRES(m, k) is easily extended to a block method by using the k previous error approximations z_j to build additional Krylov subspaces. We refer to this block extension as B-LGMRES(m, k), for "block" LGMRES. Conceptually, we view one restart cycle (i) of B-LGMRES(m, k) as a cycle of block GMRES on the system $AX_{i+1} = B$, where B contains the right-hand side of (1) and the k most recent error approximations z_j, $j = (i-k+1) : i$. In other words, $B = [b, z_i, \ldots, z_{i-k+1}]$. The initial guess at restart cycle i, X_i, is then written as $X_i = [x_i, 0, \ldots, 0]$, where the approximate solution to our single right-hand side system, x_i, is placed in the first column and the remaining columns are set to zero. X and B are size $n \times (k+1)$ or $n \times s$, where $s \equiv k+1$ indicates the block size.

After $i+1$ restart cycles, the B-LGMRES$(m,\ k)$ approximation space consists of the traditional Krylov portion built by repeated application of A to the current residual r_i together with Krylov spaces resulting from the application of A to previous error approximations. Therefore, we write the B-LGMRES$(m,\ k)$ approximation as

$$x_{i+1} = x_i + q_{i+1}^{m-1}(A)r_i + \sum_{j=i-k+1}^{i} \alpha_{ij}^{m-1}(A)z_j, \tag{4}$$

where the degree $m-1$ polynomials α_{ij}^{m-1} and q_{i+1}^{m-1} are chosen such that $\|r_{i+1}\|_2$ is minimized. Recall that $z_i \in \mathcal{K}_m(A, r_{i-1})$ and that $Az_i = r_{i-1} - r_i$ from (3). Therefore, by choosing z_i as an additional right-hand side, the standard approximation space is augmented with part of the information generated in the previous approximation space.

Using (3), we now rewrite B-LGMRES$(m,\ k)$ in (4) in the following way:

$$z_{i+1} = q_{i+1}^{m-1}(A)r_i + \sum_{j=i-k+1}^{i} \alpha_{ij}^{m-1}(A)z_j.$$

In the above form, as shown for the LGMRES method in [4], this block method resembles a truncated polynomial-preconditioned full conjugate gradient method with variable preconditioning. In this analogy, the polynomial preconditioner is q_{i+1}^{m-1}, and it changes at each iteration (i). In addition, the error approximation vectors z_j are analogous to conjugate gradient direction vectors (e.g., see [2]), and now these direction vectors are weighted by polynomials in A (α_{ij}) instead of scalars. Furthermore, as in [4], it is easily shown that these direction vectors z_j are A^TA-orthogonal.

THEOREM 1 (orthogonality of the error approximations). [illegible] $z_j \equiv x_j - x_{j-1}$ [illegible] A^TA [illegible]. We define subspaces $\mathcal{M}_{i+1}$ and $\mathcal{M}_i$ as

$$\mathcal{M}_{i+1} \equiv \mathcal{K}_m(A, r_i) + \sum_{j=i-k+1}^{i} \mathcal{K}_m(A, z_j),$$

$$\mathcal{M}_i \equiv \mathcal{K}_m(A, r_{i-1}) + \sum_{j=i-k}^{i-1} \mathcal{K}_m(A, z_j).$$

By construction, $r_i \perp A\mathcal{M}_i$ and $r_{i+1} \perp A\mathcal{M}_{i+1}$. From (3), $r_i - r_{i+1} = Az_{i+1} \Rightarrow Az_{i+1} \perp A(\mathcal{M}_i \cap \mathcal{M}_{i+1})$. Because $\{z_j\}_{j=(i-k+1):i} \subset \mathcal{M}_i \cap \mathcal{M}_{i+1}$, $z_{i+1} \perp_{A^TA} \{z_j\}_{j=(i-k+1):i}$. ∎

To provide additional insight into the B-LGMRES augmentation scheme, we briefly describe the B-LGMRES method in the framework presented in [9]. In [9], Chapman and Saad consider the addition of an arbitrary subspace, $\mathcal{W}$, to the standard Krylov approximation space of a minimum residual method, such as GMRES. This addition results in the augmented approximation space

$$\mathcal{M} = \mathcal{K} + \mathcal{W}, \tag{5}$$

where $\mathcal{K}$ is the standard Krylov subspace. A minimal residual method finds an approximate solution $\tilde{x} \in x_0 + \mathcal{M}$ such that the residual $\tilde{r}$ satisfies $\tilde{r} \perp A\mathcal{M}$, removing components of the initial residual r_0 in subspace $A\mathcal{M}$ via an orthogonal projection

1. $r_i = b - Ax_i$, $\beta = \|r_i\|_2$, $r_i = r_i/\beta$
2. $R_i = [r_i, z_i, \ldots, z_{i-k+1}]$
3. $R_i = V_1\hat{R}$ (QR factorization)
4. for $j = 1 : m$
5. $\quad U_j = AV_j$
6. $\quad$ for $l = 1 : j$
7. $\quad\quad H_{l,j} = V_l^T U_j$
8. $\quad\quad U_j = U_j - V_l H_{l,j}$
9. $\quad$ end
10. $\quad U_j = V_{j+1} H_{j+1,j}$
11. end
12. $W_m = [V_1, V_2, \ldots, V_m]$, $H_m = \{H_{l,j}\}_{1 \le l \le j+1; 1 \le j \le m}$
13. find y_m s.t. $\|\beta e_1 - H_m y_m\|_2$ is minimized
14. $z_{i+1} = W_m y_m$
15. $x_{i+1} = x_i + z_{i+1}$

FIG. 1. *B-LGMRES(m, k) for restart cycle i.*

process. Following the discussion in [9], the minimization process over the augmented approximation space (5) is

$$\|\tilde{r}\|_2 = \min_{d,w} \|b - Ad - Aw\|_2,$$

where $d \in x_0 + \mathcal{K}$ and $w \in \mathcal{W}$. Chapman and Saad then show that if r_d results from minimizing $\|b - Ad\|_2$, where again $d \in x_0 + \mathcal{K}$, then

$$\|\tilde{r}\|_2 \le \|(I - P_{AW})r_d\|_2, \tag{6}$$

where P_{AW} is an orthogonal projector onto subspace $A\mathcal{W}$. In particular, for a cycle of B-LGMRES(m, k) with $k = 1$, we have that $\mathcal{K} = \mathcal{K}_m(A, r_i)$ and $\mathcal{W} = \mathcal{K}_m(A, z_i) = \mathcal{K}_m(A, e_i - e_{i-1})$. Therefore, from (6) we see that the addition of $\mathcal{W}$ in the B-LGMRES method results in the removal of components of r_d (the residual from the standard Krylov approximation space) from the subspace $A\mathcal{K}_m(A, e_i - e_{i-1})$ or, equivalently, the removal of components of the error from subspace $\mathcal{K}_m(A, e_i - e_{i-1})$.

4.3. Algorithm details. The implementation of B-LGMRES(m, k) is similar to that of BGMRES. For reference, one restart cycle (i) of B-LGMRES(m, k) is given in Figure 1. As with BGMRES, B-LGMRES(m, k) requires the application of s^2 rotations at each iteration to transform H_m into an upper triangular matrix (e.g., see [37]). Because B-LGMRES(m, k) solves $Ax = b$, as opposed to a block system, the least-squares solution step (line 13) varies from that of standard BGMRES; the triangular matrix $\hat{R}$ from the QR decomposition in line 3 does not need to be saved since only $\beta = \|r_i\|_2$ is needed, and only s rotations must be applied to the least-squares problem's right-hand side (βe_1) at each step.

Though we think of the error approximations z_j, $j = (i - k + 1) : i$ as additional right-hand side vectors, we do not form the block approximate solutions X_i. Instead we append the k most recent error approximations to the initial residual to form a block residual R_i, as seen in line 2 of Figure 1. We normalize the error approximations $(z_j/\|z_j\|_2)$ so that each column of the initial residual block R_i is of unit length. The m size $n \times s$ orthogonal block matrices V_j form the orthogonal $n \times m \cdot s$ matrix W_m, where $W_m = [V_1, V_2, \ldots, V_m]$. H_m is a size $(m + 1)s \times m \cdot s$ band-Hessenberg matrix with s subdiagonals, and the following standard relationship holds:

$$AW_m = W_{m+1}H_m.$$

TABLE 1
Algorithm specifications per restart cycle.

Method	Approx. space size	Length n vector storage	Accesses of A
GMRES(m)	m	$m+3$	m
LGMRES(m, k)	$m+k$	$m+3k+3$	m
B-LGMRES(m, k)	$m(k+1)$	$(m+2)k+m+3$	m

As with LGMRES, only i error approximations are available at the beginning of restart cycles with $i < k$. As a result, the creation of the initial residual block in line 2 of the B-LGMRES(m, k) algorithm must be modified for the first k cycles where $i < k$. (The first cycle is $i = 0$.) Recall that $z_j \equiv 0$ for $j < 1$, and when $z_j \equiv 0$, we say that z_j is an error approximation that is not available. In our implementation, we replace any $z_i, \ldots, z_{i-k+1}$ that is not available with a randomly generated vector of length n.

For the B-LGMRES(m, k) implementation given in Figure 1, the multivectors are of size s, where $s = k + 1$. For example, the orthogonal block matrices V_j in lines 3, 5, 8, 10, and 12 of Figure 1 are size s multivectors consisting of s vectors of length n. In addition, R_i in lines 2 and 3 and U_j in lines 5, 7, 8, and 10 are also multivectors. The matrix-multivector multiply occurs in line 5, and the importance of this aspect of the implementation is demonstrated in section 5.1.

One restart cycle, or m iterations, of B-LGMRES(m, k) requires m matrix-[illegible] multiplies, irrespective of the value of k. Therefore, matrix A is accessed from memory m times per cycle. For comparison, we list the approximation space size, the number of vectors of length n stored, and the number of matrix accesses per restart cycle in terms of parameters m and k for GMRES(m), LGMRES(m, k), and B-LGMRES(m, k) in Table 1.

B-LGMRES(m, k) is compatible with both left and right preconditioning. We denote the preconditioner by M^{-1}. For left preconditioning, the initial residual in line 1 of Figure 1 must be preconditioned as usual: $r_i = M^{-1}(b - Ax_i)$. Then we replace A with $M^{-1}A$ in line 5. To incorporate right preconditioning, we replace A with AM^{-1} in line 5. We define $\hat{z}_j \equiv M(x_j - x_{j-1}) = Mz_j$ and replace z with $\hat{z}$ everywhere in lines 2 and 14. While no explicit change is required for line 15 as given in Figure 1, note that, with right preconditioning, line 15 is equivalent to $x_{i+1} = x_i + M^{-1}\hat{z}_{i+1}$.

We note that in our implementation, no reorthogonalization was performed, and residual norms are computed recursively at each step within a restart cycle. Additionally, although deflation is often an important issue for block methods, it is not required for B-LGMRES because we are not solving a block linear system. The possibility of a breakdown due to rank deficiency for the initial residual block does exist, but we have not had any breakdowns in practice (a random vector may be substituted for z_i if a breakdown is detected).

5. Numerical experiments. In this section, we first give the specific details of our B-LGMRES implementation and demonstrate the advantages of some performance programming techniques. We then show the importance of reducing data movement to achieve an efficient implementation. Finally, we present promising experimental results from our efficient implementation of B-LGMRES on a variety of problems.

We implemented the B-LGMRES algorithm in C using a locally modified version

TABLE 2
List of test problems together with the matrix order (n), number of nonzeros (nnz), preconditioner, and a description of the application area (if known).

	Problem	n	nnz	Preconditioner	Application area
1	pesa	11738	79566	none	
2	epb1	14734	95053	none	heat exchanger simulation
3	memplus	17758	126150	none	digital circuit simulation
4	zhao2	33861	166453	none	electromagnetic systems
5	epb2	25288	175027	none	heat exchanger simulation
6	ohsumi	8140	1456140	none	
7	aft01	8202	125567	ILU(0)	acoustic radiation, FEM
8	memplus	17758	126150	ILU(0)	digital circuit simulation
9	arco5	35388	154166	ILU(0)	multiphase flow: oil reservoir
10	arco3	38194	241066	ILU(1)	multiphase flow: oil reservoir
11	bcircuit	68902	375558	ILUTP(.01, 5, 10)	digital circuit simulation
12	garon2	13535	390607	ILUTP(.01, 1, 10)	fluid flow, 2-D FEM
13	ex40	7740	458012	ILU(0)	3-D fluid flow (die swell problem)
14	epb3	84617	463625	ILU(1)	heat exchanger simulation
15	e40r3000	17281	553956	ILU(2)	2-D fluid flow in a driven cavity
16	scircuit	170998	958936	ILUTP(.01, .5, 10)	digital circuit simulation
17	venkat50	62424	1717792	ILU(0)	2-D fluid flow

of PETSc 2.1.5 (Argonne National Laboratory's Portable, Extensible Toolkit for Scientific Computation) [6, 5]. All results provided in this paper were run on a single processor of a 16-processor Sun Enterprise-6500 server with 16 Gbytes RAM. This system consists of 400 Mhz Sun Ultra II processors, each with a 16 Kbyte L1 cache and a 4 Mbyte L2 cache. For reference, Table 2 lists multiple test problems from the University of Florida Sparse Matrix Collection [11], the Matrix Market Collection [33], and the PETSc test collection.

5.1. An efficient implementation. To specifically demonstrate the benefit of an efficient implementation, we implemented B-LGMRES in PETSc 2.1.5 both with and without what we refer to as a [illegible]. For the multivector optimization implementation, the [illegible] implementation, we followed the previously mentioned approach in [20, 21] for improving the performance of a matrix-multivector multiply routine by grouping computations on the same data. Consider the size s multivector V, where $V \equiv [v_1, v_2, \ldots, v_s]$ for vectors $v_1, v_2, \ldots, v_s \in \mathbb{R}^{n\times 1}$. By processing the s individual vectors as a group, the matrix-multivector multiply routine performs a greater number of floating-point operations for each access of a nonzero element of the coefficient matrix A and has the potential to reduce the amount of data moved through parts of the memory subsystem by a factor of s. The approach of grouping computations together affects more than just the matrix multiply routine; it pervades the implementation of the entire B-LGMRES algorithm. The second implementation, referred to as the [illegible] implementation, does not group multivector computations together and represents the best implementation possible with the tools available in PETSc. Both implementations were written so as to eliminate any copying of data from one data structure to another and represent best coding efforts.

Three primary sections of the B-LGMRES code are affected by the multivector optimization: the matrix-vector multiply (MatMult), the modified Gram–Schmidt orthogonalization (MGS), and the application of the preconditioner (Prec), if required. For the MV implementation for our set of test problems, the percentage of time spent in each of the three primary sections varies from 18% to 83% for MatMult, 6% to 55%

for MGS, and 31% to 53% for Prec. Because the Prec section of code shows similar characteristics to the MatMult section, we only discuss the MatMult and MGS sections of code. In addition, note that we concentrate on a multivector of size $s = 2$, which corresponds to B-LGMRES(m, k) with $k = 1$. We show in section 5.3 that size $k = 1$ is generally optimal for the algorithm. However, because our results are applicable to improving the performance of general block linear solvers as well, we also discuss the advantages of a larger block size.

For the remainder of this section, we detail the MV implementation, explain how it differs from the non-MV implementation in terms of data movement, and compare the performance of the two implementations. Because algorithmic changes in a numerical code affect data movement between main memory and cache and between levels of cache, the size of the data structures determines the part of the memory most affected by the multivector optimizations. Therefore, for each we discuss the [illegible], which is the size in Mbytes of data loaded through the memory hierarchy for each operation. Note that for a typical memory hierarchy, L1 caches typically have access times of two to three clock cycles, L2 cache access times are generally 5–15 times slower, and main memory access times are generally at least 100 times slower still (e.g., see [44, 32, 7]).

The MatMult section of the code corresponds to the matrix-vector multiply in line 5 of Figure 1. For the non-MV implementation, successive calls are made to a matrix-vector multiply routine for each individual vector $v_1, v_2, \ldots, v_s$ in multivector V_j. In contrast, for the MV implementation, we store the multivectors in a format referred to as [illegible] in the PETSc 2.1.5 manual [5] and use the associated PETSc matrix-multivector multiply routine. This multicomponent format is a row-wise storage format that stores V as a vector of length $n \cdot s$ in which the components of the s vectors are interlaced. In other words, multivector elements separated by stride s belong to a single vector: $v_1 = [V(0); V(s); V(2s); \ldots; V(ns - s)]$. The working set size for the MatMult section, $WS_{MatMult}$, is approximately equal to the storage required for matrix A. Therefore, for compressed sparse row storage, $WS_{MatMult} = sizeof(double) * nnz + sizeof(int) * (n + nnz)$, where the function $sizeof()$ returns the size of its argument in bytes. If $WS_{MatMult}$ is significantly larger than the L2 cache size, then successive calls to a matrix-vector multiply routine repeatedly read matrix A from main memory through both levels of cache. For $WS_{MatMult}$ smaller than the L2 cache size but larger than L1, portions of matrix A may remain in L2 cache for successive matrix-vector multiplies.

The MGS section (lines 6–10 in Figure 1) often contributes significantly to the overall cost of the B-LGMRES algorithm and is dominated by the PETSc routines [illegible] and [illegible], which are the vector dot product and axpy routines, respectively. For the MV implementation, we wrote multivector versions of these two routines, referred to as [illegible] and [illegible], in a manner that limits movement of data. For example, in the non-MV implementation, determining the dot product of two size s multivectors, which are themselves $n \times s$ matrices, results in an $s \times s$ matrix. Computing that matrix requires s^2 successive calls to VecDot and $2 \cdot n \cdot s^2$ data values to be read from the memory hierarchy. In contrast, one call to VecStrideDot provides the same functionality, but only $2 \cdot n \cdot s$ data values are read because computations on related data are fused together. Therefore, if $WS_{MGS} \equiv sizeof(double) \cdot 2 \cdot n \cdot s$ is greater than either the L1 or L2 cache size, the multivector optimization affects data movement in the MGS section of code.

Both VecStrideAXPY and VecStrideDot were written using loop temporaries and loop unrolling to aid compiler optimization. The use of loop temporaries allows a

TABLE 3
Execution times in μsec for a single call to AXPY for the non-MV and MV implementations as well as for the standard and optimized BLAS 3 DGEMM routines with a block size of $s = 2$.

	Problem	n	non-MV	MV	DGEMM (src)	DGEMM (opt)
13	ex40	7740	2.00	1.25	3.96	2.96
3	memplus	17758	4.55	2.85	9.18	6.73
11	bcircuit	68902	17.43	10.35	35.32	25.73
14	epb3	84617	19.54	12.58	43.49	31.88

TABLE 4
Execution times in μsec for a single call to a dot product routine for the non-MV and MV implementations as well as for the standard and optimized DGEMM routines with a block size of $s = 2$.

	Problem	n	non-MV	MV	DGEMM (src)	DGEMM (opt)
13	ex40	7740	1.20	.78	3.94	2.89
3	memplus	17758	2.60	1.69	9.16	6.66
11	bcircuit	68902	10.91	6.29	35.32	24.98
14	epb3	84617	12.64	7.89	44.01	30.96

compiler to identify data reuse more easily at the register level. Loop unrolling further helps register reuse and allows different iterations of the loop to occur simultaneously. In our MV implementation we unroll the inner loop s times, where s is the block size. We chose to write our own VecStrideAXPY and VecStrideDot routines instead of using the level 3 BLAS DGEMM routine [14] due to the performance advantage of a hand-coded implementation. The timings to perform a multivector AXPY and dot product are given in Tables 3 and 4, respectively, for $s = 2$. The version of DGEMM labeled "src" is compiled from source code and the version labeled "opt" is the vendor supplied optimized library. Timings are given in μsec for a subset of the test problems with a range of matrix orders. For both routines, the MV and non-MV implementations are superior to DGEMM, and the hand-tuned MV implementations are the clear winners. Our MV implementations outperform DGEMM because the loops are unrolled to the specific block sizes. Because the MGS section can consume a large percentage of total execution time, the simple optimizations described can have as significant an impact on overall execution time of the solver as the use of the optimized matrix-multivector multiply routine.

As previously explained, we store the multivectors row-wise in our MV implementation. Alternatively, one might wonder whether column-wise ordering is a viable option as column-wise ordering better facilitates algorithmic needs such as deflation and requires less reorganization of data. Therefore, in Table 5, we give results for the MatMult routine on block sizes $s = 2$ and $s = 4$. MV-row and MV-col indicate row-wise and column-wise variations of the MV implementation. In addition to four problems from our test set, row 5 in Table 5 lists an additional problem, poisson3Db, from the UF collection (n= 85623 and nnz = 2374949) that has a large bandwidth. Our experiments indicate that the use of the column-wise storage format for multivectors on large problems with a large matrix bandwidth causes TLB misses, as suggested in [24]. (The TLB is a cache of translations between the real and virtual addresses of recently referenced pages of memory. A TLB miss incurs two loads to memory: one to load the address translation and a second to load the actual operand.) To illustrate the impact of TLB misses, line 6 of Table 5 lists the results for the poisson3Db problem with reverse Cuthill–McKee (RCM) [10] reordering. The

TABLE 5
Execution times in μsec for the non-MV and MV implementations of the MatMult routine for block sizes $s = 2$ and $s = 4$.

		Block size = 2 (μsec)			Block size = 4 (μsec)		
	Problem	non-MV	MV-row	MV-col	non-MV	MV-row	MV-col
13	ex40	30.7	24.7	25.1	47.1	27.4	27.4
3	memplus	17.8	13.4	13.4	32.4	15.7	16.2
11	bcircuit	65.3	46.2	46.5	116.4	61.9	63.1
14	epb3	84.6	57.8	60.2	134.2	71.0	75.6
–	poisson3Db	424.2	263.5	369.2	831.7	367.6	927.2
–	poisson3Db (RCM)	339.2	152.7	153.6	679.9	184.9	183.5

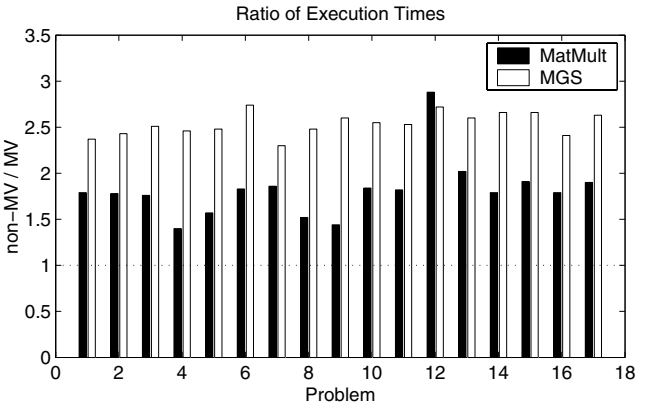


FIG. 2. *A comparison of execution time for the MatMult and MGS sections with the non-MV implementation of B-LGMRES(*15,1*) versus the MV implementation for ten restart cycles for the* 17 *test problems.*

RCM reordering reduces the matrix bandwidth and significantly decreases execution time for the column-wise multivector storage format. A large bandwidth does not impact the row-wise storage as much because consecutive accesses of the multivector elements correspond to data items that are stored contiguously. All further numerical experiments in this manuscript use the row-wise MV implementation.

We now compare the execution times for the non-MV and MV implementations on both the MatMult and MGS sections of code in Figure 2. The x-axis corresponds to the numbered test problems in Table 2, and the y-axis is the execution time for the non-MV implementation divided by the execution time for the MV implementation. A value greater than one indicates that the execution time for MV is less than that for non-MV. For the MatMult section, the MV implementation reduces the execution time by a factor of 1.4 to 2.7 over the non-MV implementation, and it is an open question what impact matrix density (or even nonzero structure) has on the effectiveness of the multivector optimizations. For the MGS section, the execution time shows an even greater improvement on average: the MV implementation of MGS reduces execution time by a factor of 2.3 to 2.7 over the non-MV implementation. As for the execution time for the entire B-LGMRES code, the MV implementation is close to twice as fast on average as the non-MV implementation with multivectors of size $s = 2$, and the performance improvement is independent of problem size (n) and number of nonzeros (nnz) for our test set in Table 2.

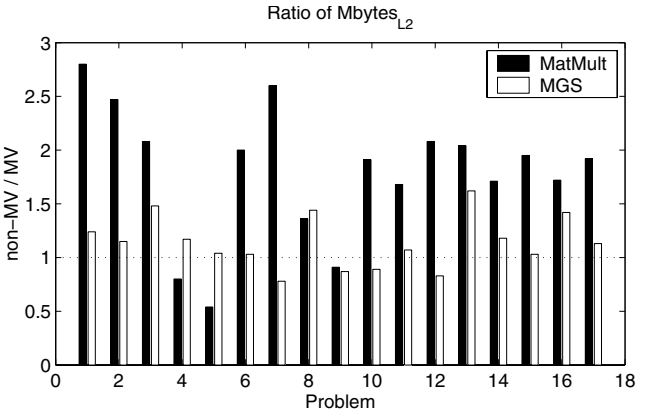


FIG. 3. *A comparison of data movement from main memory to L2 cache in the MatMult and MGS sections for the non-MV implementation of B-LGMRES(*15,1*) versus the MV implementation for ten restart cycles for the* 17 *test problems.*

Additionally, we remark that we modified our local installation of PETSc to include multivector versions of the PETSc routines and . VecMAXPY adds a scaled sum of vectors to a vector, which is used when solving the least-squares problem. MatSolve performs a forward and back solve for use with the ILU preconditioner.

5.2. Impact of data movement on execution time. We now explain reductions in execution time due to the multivector optimization using data from hardware performance counters that monitor data movement. We focus on two counters that measure the megabytes of data moved between main memory and L2 cache, denoted by $Mbytes_{L2}$, and between the L2 and L1 caches, denoted by $Mbytes_{L1}$.

We first examine data movement between main memory and the L2 cache in Figure 3. The y-axis in this figure indicates the ratio of $Mbytes_{L2}$ for non-MV to MV for both the MatMult and MGS sections. Because $s = 2$, we expected a factor of two reduction in $Mbytes_{L2}$ for the MV implementations for test problems with a working set size significantly larger than the L2 cache. For the MatMult section, problems 6 and 11–17 have $WS_{MatMult}$ larger than the 4 Mbytes L2 cache size. These problems all have ratios from 1.75 to 2.0, which correlate well with the factor of two reductions in execution time for those problems seen in Figure 2. For the MGS section, only problem 16 has WS_{MGS} greater than the L2 cache size, and this problem shows a ratio of 1.4, which does not correlate well with the factor of 2.4 improvement in execution time shown in Figure 2. Furthermore, the results in Figure 3 are inconsistent with those in Figure 2 in other ways: several problems appear to show an increase in $Mbytes_{L2}$ for the MGS and MatMult sections for the MV implementation. In fact, for these problems the amounts of data moved are quite small for both implementations and the apparent increases are likely the result of measurement error. Nonetheless, the MV implementation is faster than the non-MV one. These inconsistencies indicate that a reduction in $Mbytes_{L2}$ does not accurately predict a reduction in execution time for MGS for most of the test problems.

We now consider data movement between the L1 and L2 caches. Figure 4 shows the ratio of $Mbytes_{L1}$ for the non-MV to MV implementation. For the MatMult section, we expected test problems with $WS_{MatMult} \gg sizeof(L1\ cache)$ to show a

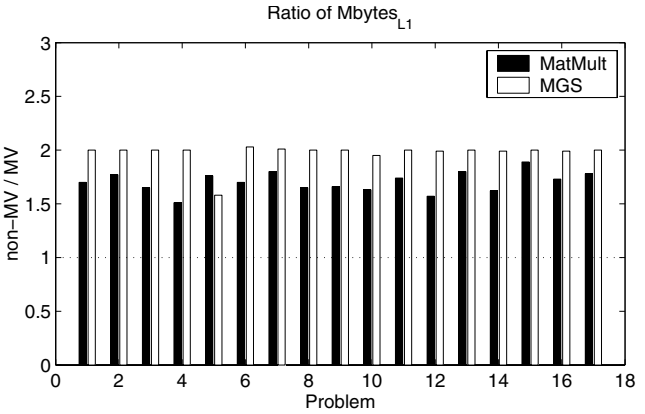


FIG. 4. *A comparison of data movement from L2 cache to L1 cache in the MatMult and MGS sections for the non-MV implementation of B-LGMRES(*15,1*) versus the MV implementation for ten restart cycles for the* 17 *test problems.*

reduction in $Mbytes_{L1}$. Test problem 1 has the smallest $WS_{MatMult}$ at 978 Kbytes, which is significantly larger than the 16 Kbyte L1 cache. Thus, Figure 4 shows that for all of the test problems the ratios of data moved do in fact range from 1.5 to 1.9. Similarly for the MGS section, problem 12 has the smallest WS_{MGS} at 241 Kbytes, which is also significantly larger than the L1 cache. Consequently, all of the test problems have ratios that range from 1.6 to 2.0. The reduction in $Mbytes_{L1}$ is thus consistent with the reduction in execution time seen in Figure 2 for both sections of code.

Therefore, from Figures 3 and 4 one can infer that the reduction in execution time due to the MV implementation correlates to the reduction in $Mbytes_{L1}$ for our test problems (see [3] for more details). The importance of optimizations that reduce movement of data between levels of cache was unexpected in that emphasis is traditionally placed on reducing data movement between cache and main memory, which is of most concern to large problems. For much larger test problems, where both $WS_{MatMult}$ and WS_{MGS} are much larger than the L2 cache size, for example, we expect that the reduction $Mbytes_{L2}$ would more strongly correlate to execution time. However, our results indicate that data movement is not just an issue for very large problems. In fact, the results presented in this section demonstrate that reformulating an iterative solver to use multivectors enables an efficient implementation that can reduce data movement for problems of any size. Furthermore, these multivector optimizations would benefit larger block sizes and other types of block methods, including those that solve systems with multiple right-hand sides.

5.3. Comparison with other methods. In this section, we demonstrate the potential of the B-LGMRES method both with and without preconditioning by presenting experimental results for the test problems in Table 2. If a right-hand side was not provided, we generated a random right-hand side. For all problems, the initial guess was a zero vector. For each problem we report wall clock time for the linear solve only; we do not time any I/O or the setup of the preconditioner. For the preconditioned problems, we use either ILU(p) or ILUTP($droptol$, $permtol$, $lfil$) (e.g., see [37]). The timings reported are averages from five runs and have standard deviations of at most two percent. All tests are run until the relative residual norm is

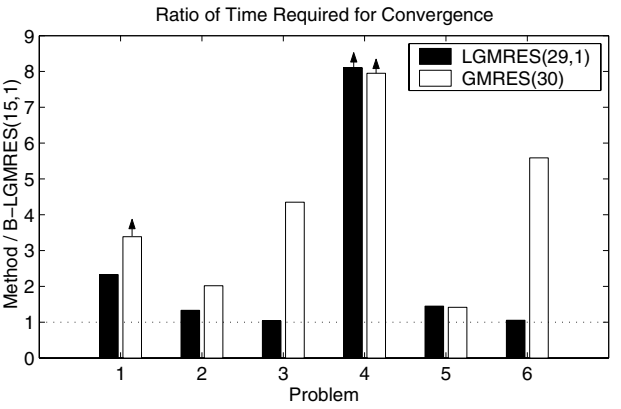


FIG. 5. *A comparison of the time required for convergence for nonpreconditioned test problems* 1–6 *with GMRES(*30*) and LGMRES(*29, 1*) versus B-LGMRES(*15, 1*).*

less than the convergence tolerance $\zeta = 10^{-9}$, i.e., when $\|r_i\|_2/\|r_0\|_2 \leq \zeta$. However, if a method does not converge in 1000 restart cycles, then the execution time reported reflects the time for 1000 cycles and we say that the method does not converge.

We evaluate the performance of B-LGMRES(m, k) for a particular problem by comparing its time to converge to that of GMRES(m) and LGMRES(m, k) with equal-sized approximation spaces. We use the GMRES(m) implementation available in PETSc 2.1.5 and the PETSc implementation of LGMRES(m, k) described in [4]. For GMRES(m), we chose restart parameter $m = 30$ because it is a common choice for GMRES(m) and is the default in PETSc. We required that the approximation spaces for both LGMRES(m, k) and B-LGMRES(m, k) be of size 30 as well. Furthermore, we wanted to evaluate the performance of LGMRES(m, k) and B-LGMRES(m, k) using the same number of error approximation vectors, i.e., the same k, for each. In choosing a value of k, we note that for LGMRES(m, k), $k \leq 3$ is typically optimal, and variations in algorithm performance are small for $k \leq 3$ (see [4]). Note that for a fixed approximation space size, increasing k for B-LGMRES(m, k) decreases the powers of A represented in the approximating subspace. For this set of test problems, preliminary testing showed that using $k > 2$ was typically not beneficial and $k = 1$ was generally optimal. (For an approximation space larger than 30, we expect that using $k > 1$ would be an advantage more often.) As a result, we chose $k = 1$ for the experiments presented here, and this choice together with the constraint of a size 30 approximation space determined parameter m for each method as in Table 1. Note the approximation space for B-LGMRES(15, 1) at restart cycle $i+1$ is given by $\mathcal{K}_{15}(A, r_i)+\mathcal{K}_{15}(A, z_i)$ and that for LGMRES(29, 1) is given by $\mathcal{K}_{29}(A, r_i)+span\{z_i\}$.

We first present results for the nonpreconditioned test problems. Figure 5 compares the time required for convergence for B-LGMRES(15, 1) to both LGMRES(29, 1) and GMRES(30) for the test problems given in Table 2. The y-axis is the time required for convergence for either LGMRES(29, 1) or GMRES(30) divided by the time required for convergence for B-LGMRES(15, 1). Therefore, bars extending above one indicate faster convergence for B-LGMRES(15, 1). For all of these problems, B-LGMRES(15, 1) converges. However, arrows above the bars in Figure 5 indicate that GMRES(30) did not converge in 1000 restart cycles for problem 1 and both LGMRES(29, 1) and GMRES(30) did not converge for problem 4. Again, the time

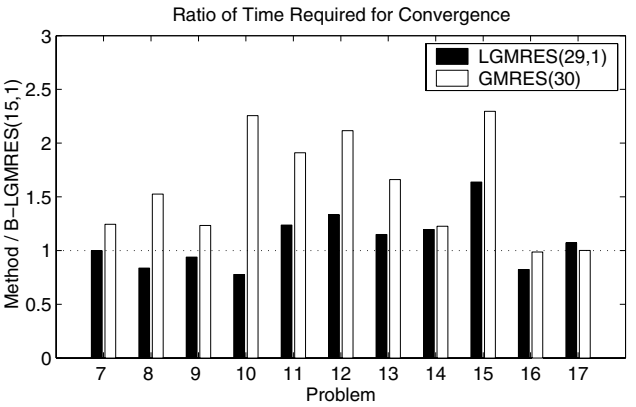


FIG. 6. *A comparison of the time required for convergence for preconditioned test problems* 7–17 *with GMRES(*30*) and LGMRES(*29, 1*) versus B-LGMRES(*15, 1*).*

for 1000 restart cycles is reported for methods that do not converge, resulting in an understated ratio of improvement for B-LGMRES(15, 1) for the bars with arrows. B-LGMRES(15, 1) converges in less time than GMRES(30) for all problems, and improvements over LGMRES(29, 1) are more modest since LGMRES is typically also an improvement over GMRES(m) [4].

We now do the same performance evaluation for B-LGMRES(m, k) on preconditioned problems, problems 7–17, in Figure 6. We use left preconditioning, and determination of convergence is based on the preconditioned residual norm as usual. As in Figure 5, the bars extending above one favor B-LGMRES(15, 1). The time required for convergence for B-LGMRES(15, 1) is less than that for GMRES(30) for problems 7–15 and about the same as GMRES(30) for problems 16 and 17. However, performance gains are not as dramatic as those without preconditioning due to lower iteration counts. For large problems, though, even a small improvement in iteration count translates into a nontrivial time savings. The comparison of B-LGMRES(15, 1) to LGMRES(29, 1) is not as straightforward. The LGMRES method is quite effective for these test problems, and as a result, predicting which algorithm will "win" for a particular test problem is an open question. As an example, we tested problem memplus both without and with preconditioning, problems 3 and 8, respectively. For this problem, B-LGMRES(15, 1) converges in slightly less time than LGMRES(29, 1) when no preconditioner is used, but LGMRES(29, 1) is faster with preconditioning.

For all of the test problems in Table 2, we found that the time to solution of the restarted methods correlates well with the number of accesses of A, as opposed to the number of matrix-vector multiplies. In particular, Figure 7 illustrates that the number of accesses of A largely determines execution time. Therefore, the method that converges in the least amount of time for a particular problem in Figures 5 and 6 is generally that with the smallest number of iterations. However, it is important to acknowledge that the efficient implementation of B-LGMRES(15, 1) makes the preceding statement possible because each iteration of B-LGMRES(15, 1) requires two matrix-vector multiplies; a poor implementation of B-LGMRES(15, 1) could potentially cost twice as much per iteration as GMRES(30).

Because our machine has sufficient resources, full GMRES is a viable option. Therefore, we also compared the time to solution of B-LGMRES(15, 1) to full GMRES for each of the 17 test problems. B-LGMRES converges in significantly less time than

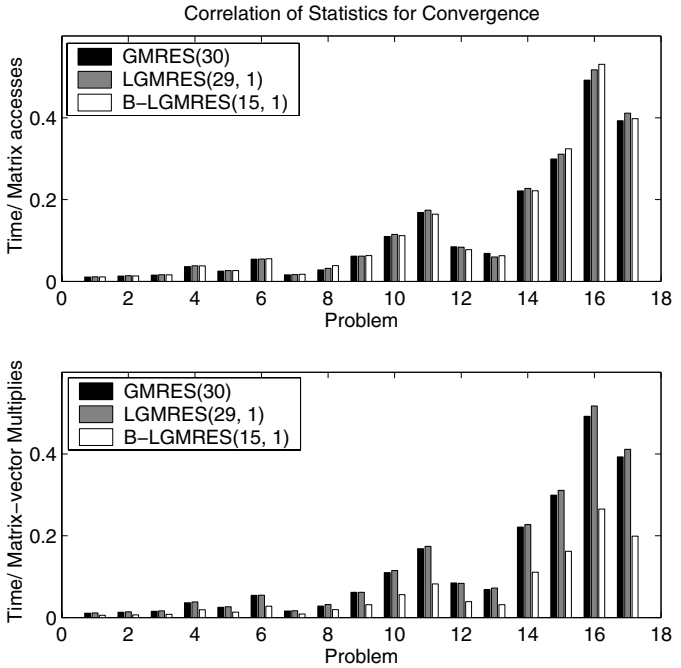


FIG. 7. *The upper panel shows the ratios of time required for convergence to number of accesses of A for GMRES (*30*), LGMRES(*29, 1*), and B-LGMRES(*15, 1*) on the* 17 *test problems. The lower panel shows the ratios of time required for convergence to number of matrix-vector multiplies.*

GMRES on all but four problems: 7, 10, 13, and 15. There was no correlation between problem size and method in terms of time required for convergence for these test problems. See [3] for more details.

For the problems presented here, the time to solution of B-LGMRES(m, k) is typically an improvement over that of GMRES(m), as shown in Figures 5 and 6. The additional right-hand side vector(s) in B-LGMRES inhibit the tendency of restarted GMRES to form similar polynomials at every other restart cycle (see [4]), thus improving convergence. As with the LGMRES method, B-LGMRES acts as an accelerator, but in general does not help with stalling. In other words, B-LGMRES(m, k) has a total approximation space size of $m \cdot (k+1)$ and typically does not help problems that stall for GMRES($m \cdot (k+1)$), though we have found a few exceptions. For problems that stall, full GMRES or Morgan's GMRES-DR method [31] can be good options. Although a thorough understanding of the convergence behavior of B-LGMRES is an open question that warrants further investigation, the experimental results presented in this section demonstrate that solving a single right-hand side system via a block system is a viable means of improving the time to solution of an iterative method such as GMRES(m). In particular, achieving a balance between maintaining or improving an iterative linear solver algorithm's numerical properties and reducing data movement is possible.

6. Concluding remarks. In this paper, we explore the feasibility of modifying restarted GMRES to reduce data movement. We show that for iterative linear solvers, the time to solution is largely dependent on the number of accesses of A from memory as opposed to the total number of matrix-vector multiplies. Therefore, mod-

ifications that reduce data movement are a particularly effective means of improving performance. Furthermore, our results indicate that using available performance programming techniques to drive algorithm development can be effective and that careful attention to implementation details can be quite beneficial. For example, hand-tuned routines can outperform subroutines such as DGEMM for certain matrix sizes, and an efficient implementation of a block version of an iterative method may be only marginally more expensive per access of coefficient matrix A than a nonblock version. A unique aspect of this study is the thorough investigation of data movement through the memory hierarchy. Typically the impact of memory performance on an iterative linear solver is not studied in this detail.

Our investigation led to a block variant of the GMRES method that solves a linear system with a single right-hand side. This new method, B-LGMRES, results from choosing error approximation vectors as additional right-hand side vectors. This choice mimics a truncated polynomial-preconditioned conjugate gradient method. In our experiments, B-LGMRES performed best with a block size of two for an approximation space of dimension 30. For larger approximation space sizes and larger problems, a larger block size may prove more beneficial. We find that predicting the algorithm's performance is nontrivial due to its dependence on many factors: problem size (number of nonzeros and matrix order), restart parameter, block size, matrix properties, preconditioner choices, and machine characteristics. Furthermore, other right-hand side vectors may be more appropriate for particular problem classes. Given the potential performance gains from larger block sizes described in section 5.1, our implementation of B-LGMRES can serve as a template for other block methods.

Nevertheless, the substantial improvement in performance due to reduced data movement is sufficient enticement toward pursuing block methods for single right-hand side problems. Given the increasing gap between processor performance and memory access time, re-examining popular linear solver algorithms is particularly important to achieving respectable performance on modern architectures.

REFERENCES

[1] W. K. Anderson, W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *Achieving high sustained performance in an unstructured mesh CFD application*, in Proceedings of Supercomputing '99, 1999. Also published as Mathematics and Computer Science Division, Argonne National Laboratory, Technical Report ANL/MCS-P776-0899.

[2] S. F. Ashby, T. A. Manteuffel, and P. E. Saylor, *A taxonomy for conjugate gradient methods*, SIAM J. Numer. Anal., 27 (1990), pp. 1542–1568.

[3] A. H. Baker, *On Improving the Performance of the Linear Solver Restarted GMRES*, Ph.D. thesis, University of Colorado, Boulder, CO, 2003.

[4] A. H. Baker, E. R. Jessup, and T. Manteuffel, *A technique for accelerating the convergence of restarted GMRES*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 962–984.

[5] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *PETSc Users Manual*, Tech. Report ANL-95/11—Revision 2.1.5, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, 2003.

[6] S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, L. C. McInnes, and B. F. Smith, *PETSc Home Page*, http://www.mcs.anl.gov/petsc (2001).

[7] S. Behling, R. Bell, P. Farrell, H. Holthoff, F. O'Connell, and W. Weir, *The POWER4 Processor Introduction and Tuning Guide*, IBM Redbooks, 2001.

[8] S. Carr and K. Kennedy, *Blocking linear algebra codes for memory hierarchies*, in Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, J. Dongarra, P. Messina, D. C. Sorensen, and R. G. Voigt, eds., SIAM, 1990, pp. 400–405.

[9] A. Chapman and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[10] E. H. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings 24th Nat. Conf. Assoc. Comp. Mach., ACM Publications, New York, 1969, pp. 157–172.

[11] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices (2002).

[12] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., 36 (1999), pp. 864–889.

[13] J. W. Demmel, N. J. Higham, and R. S. Schreiber, *Block LU factorization*, Numer. Linear Algebra Appl., 2 (1995), pp. 173–190.

[14] J. Dongarra, J. DuCroz, S. Hammarling, and I. Duff, *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.

[15] J. Dongarra and V. Eijkhout, *Self-Adapting Numerical Software for Next Generation Applications*, Tech. Report ICU-UT-02-07, LAPACK Working Note 157, 2002.

[16] J. J. Dongarra, D. C. Sorensen, and S. J. Hammarling, *Block reduction of matrices to condensed forms for eigenvalue computations*, J. Comput. Appl. Math., 27 (1989), pp. 215–227.

[17] M. Eiermann, O. G. Ernst, and O. Schneider, *Analysis of acceleration strategies for restarted minimal residual methods*, J. Comput. Appl. Math., 123 (2000), pp. 261–292.

[18] B. B. Fraguela, R. Doalla, and E. L. Zapata, *Cache misses prediction for high performance sparse algorithms*, in Proceedings of the Fourth International Euro-Par Conference (Euro-Par '98), 1998, pp. 224–233. Also published as University of Malaga, Department of Computer Architecture, Technical Report UNMA-DAC-98/22.

[19] K. Gallivan, W. Jalby, U. Meier, and A. H. Sameh, *Impact of hierarchical memory systems on linear algebra algorithm design*, Internat. J. Supercomput. Appl., 2 (1988), pp. 12–48.

[20] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *Toward realistic performance bounds for implicit CFD codes*, in Proceedings of Parallel CFD'99, D. Keyes, A. Ecer, J. Periaux, N. Satofuka, and P. Fox, eds., Elsevier, New York, 1999, pp. 233–240.

[21] W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith, *High-performance parallel implicit CFD*, Parallel Comput., 27 (2001), pp. 337–362.

[22] G. Gu and Z. Cao, *A block GMRES method augmented with eigenvectors*, Appl. Math. Computation, 121 (2001), pp. 278–289.

[23] M. S. Lam, E. E. Rothberg, and M. E. Wolf, *The cache performance and optimizations of blocked algorithms*, in Proceedings of the Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, 1991.

[24] B. C. Lee, R. W. Vuduc, J. W. Demmel, K. A. Yelick, M. de Lorimier, and L. Zhong, *Performance Optimizations and Bounds for Sparse Symmetric Matrix-Multiple Vector Multiply*, Tech. Report UCB-CSD-03-1297, Computer Science Division, University of California, Berkeley, 2003.

[25] G. Li, *A block variant of the GMRES method on massively parallel processors*, Parallel Comput., 23 (1997), pp. 1005–1019.

[26] J. D. McCalpin, *Memory bandwidth and machine balance in current high performance computers*, IEEE Computer Society Technical Committee on Computer Architecture Newsletter, (1995).

[27] J. D. McCalpin, *STREAM: Sustainable Memory Bandwidth in High Performance Computers*, http://www.cs.virginia.edu/stream (2003).

[28] S. McKee and W. Wulf, *Access order and memory-conscious cache utilization*, in First Symposium on High Performance Computer Architecture (HPCA1), Raleigh, NC, 1995.

[29] R. Morgan, *GMRES with deflated restarting and multiple right-hand sides*, presentation at the Seventh Copper Mountain Conference on Iterative Methods, 2002.

[30] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[31] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[32] S. Naffziger and G. Hammond, *The implementation of the next generation* 64*b Itanium microprocessor*, in Proceedings of the IEEE International Solid-State Circuits Conference, Vol. 2, 2002, pp. 276–504.

[33] National Institute of Standards and Technology, Mathematical and Computational Sciences Division, *Matrix Market*, http://math.nist.gov/MatrixMarket (2002).

[34] D. O'Leary, *The block conjugate gradient algorithm and related methods*, Lin. Algebra Appl., 29 (1980), pp. 293–322.

[35] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick, *A case for intelligent RAM*, IEEE Micro, 17 (1997), pp. 34–44.

[36] A. Pinar and M. T. Heath, *Improving performance of sparse matrix-vector multiplication*, in Proceedings of Supercomputing '99, 1999.
[37] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, Boston, 1996.
[38] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statistical Comput., 7 (1986), pp. 856–869.
[39] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
[40] V. Simoncini and E. Gallopoulos, *Convergence properties of block GMRES and matrix polynomials*, Lin. Algebra Appl., 247 (1996), pp. 97–119.
[41] S. Toledo, *Improving the memory-system performance of sparse-matrix vector multiplication*, IBM J. Research Development, 41 (1997), pp. 711–725.
[42] H. A. van der Vorst and C. Vuik, *GMRESR: A family of nested GMRES methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 369–386.
[43] B. Vital, *Etude de quelques méthodes de résolution de problèmes linéaires de grande taille sur multi-processeur*, Ph.D. thesis, Université de Rennes I, Rennes, France, 1990.
[44] R. Vuduc, J. Demmel, K. A. Yelick, S. Kamil, R. Nishtala, and B. Lee, *Performance optimizations and bounds for sparse matrix-vector multiply*, in Proceedings of Supercomputing '02, Baltimore, MD, 2002.
[45] W. A. Wulf and S. A. McKee, *Hitting the Wall: Implications of the Obvious*, Tech. Report CS-94-48, University of Virginia, Charlottesville, VA, 1994.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1627–1650
© 2006 Society for Industrial and Applied Mathematics

MULTILEVEL PRECONDITIONERS CONSTRUCTED FROM INVERSE-BASED ILUs*

MATTHIAS BOLLHÖFER† AND YOUSEF SAAD‡

Abstract. This paper analyzes dropping strategies in a multilevel incomplete LU decomposition context and presents a few strategies for obtaining related ILUs with enhanced robustness. The analysis shows that the incomplete LU factorization resulting from dropping small entries in Gaussian elimination produces a good preconditioner when the inverses of these factors have norms that are not too large. As a consequence a few strategies are developed whose goal is to achieve this feature. A number of "templates" for enabling implementations of these factorizations are presented. Numerical experiments show that the resulting ILUs offer a good compromise between robustness and efficiency.

Key words. incomplete LU decompositions, ILU, preconditioning, multilevel ILU, approximate inverse, algebraic multilevel method, iterative solver

AMS subject classifications. 65F05, 65F10, 65F50, 65Y05

DOI. 10.1137/040608374

1. Introduction.

In recent years, research on "black-box" techniques for solving sparse linear systems of equations has accelerated due to two factors. First, linear systems are becoming ever more difficult to solve due to their bigger sizes. In spite of recent progress in parallel direct solvers [8, 21, 13], practitioners are looking at cheaper alternatives. The second factor is that preconditioning methods have also made some tangible progress in improving robustness. Iterative methods based on preconditioned Krylov subspace techniques are increasingly viewed as attractive alternatives to direct solvers, both for structured problems, such as those arising from discretized partial differential equations (PDEs), and for systems arising from applications areas which yield highly unstructured systems and for which iterative methods were viewed as ineffective in the past.

One essential improvement for the use of Krylov subspace solvers consists of preconditioning methods [18], in particular methods based on incomplete LU decompositions [17]. One of the first attempts at using Krylov subspace methods as a general purpose black-box solver was by Simon [22], who used standard (symmetric) reordering techniques from direct solution methods to preprocess matrices prior to applying level-of-fill ILU. More recent work which employs nonsymmetric reordering strategies [10, 2] indicates that systems arising from a wide range of applications can be successfully solved by this approach in conjunction with ILUs.

Aside from this progress, there still remains a drawback of incomplete LU decompositions, which is that they are sensitive to parameters such as drop tolerances or certain static reordering strategies. Progress has been made in improving the stability of ILUs by taking into account the inverse triangular factors [5, 3, 14]. Also,

*Received by the editors May 15, 2004; accepted for publication (in revised form) January 10, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-5/60837.html

†Department of Mathematics, MA 4-5, TU Berlin, D-10623 Berlin, Germany (bolle@math.tu-berlin.de, http://www.math.tu-berlin.de/∼bolle/). The work of this author was supported by DFG research center MATHEON "Mathematics for Key Technologies" in Berlin.

‡Department of Computer Science and Engineering, University of Minnesota, 200 Union Street S.E., Minneapolis, MN 55455 (saad@cs.umn.edu, http://www.cs.umn.edu/∼saad/). This author's work was supported by National Science Foundation grants NSF/ACI-0305120 and NSF/INT-0003274 and by the Minnesota Supercomputer Institute.

combining incomplete LU decompositions in a hierarchical fashion with preordering pivoting strategies [19] helps in improving stability of the preconditioner.

In this paper we will discuss the rationale for a new multilevel strategy which mainly focuses on keeping the inverse triangular factors bounded. This is justified by an analysis that indicates that stability can be enhanced by this approach.

2. Impact of dropping on the preconditioned system. This section analyzes how a perturbation introduced by dropping small terms while computing incomplete LU decompositions affects the preconditioned system. This analysis will help design incomplete LU factorization preconditioners with improved robustness.

2.1. The inverse error for a single level. We start with a partial factorization of our matrix that is terminated after k steps. We assume that the initial $n \times n$ nonsingular matrix A is rearranged as

$$P^\top A Q = \begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

where the leading matrix B is nonsingular and of size $k \times k$ and where P and Q are suitably chosen permutation matrices. Suppose that this matrix is approximately factored as

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} \tilde{L}_B & 0 \\ \tilde{L}_E & I \end{pmatrix} \begin{pmatrix} \tilde{D}_B & 0 \\ 0 & \tilde{S} \end{pmatrix} \begin{pmatrix} \tilde{U}_B & \tilde{U}_F \\ 0 & I \end{pmatrix} + \mathcal{E}_k \equiv \tilde{\mathcal{L}}_k \tilde{\mathcal{D}}_k \tilde{\mathcal{U}}_k + \mathcal{E}_k. \tag{2.1}$$

Here, $\tilde{L}_B$ and $\tilde{U}_B^\top$ are unit lower triangular and $\tilde{D}_B$ is diagonal. The matrix $\tilde{S}$ is an approximation to the Schur complement $S = C - EB^{-1}F$. Assume for simplicity that we do not encounter zero pivots. We note that in principle one could generalize the following analysis using pivoting. To keep the analysis simple we assume that no pivoting is necessary for this partial incomplete factorization.

At any given step $l = 1, \ldots, k$, entries of $\tilde{\mathcal{L}}_l$ and $\tilde{\mathcal{U}}_l^\top$ may be dropped in some positions (m, l), where $m > l$. We represent these dropped values by an $n \times n$ matrix

$$\mathcal{V}_l = \left(\begin{array}{ccc|ccc} v_1 & \cdots & v_l & 0 & \cdots & 0 \end{array} \right) = \left(\begin{array}{c|c} \begin{array}{c} 0 \\ \diagdown \\ 0 \end{array} & 0 \\ \hline 0 & 0 \end{array} \right) \tag{2.2}$$

for $\tilde{\mathcal{L}}_l$ and

$$\mathcal{W}_l = \begin{pmatrix} w_1^\top \\ \vdots \\ w_l^\top \\ \hline 0 \\ \vdots \\ 0 \end{pmatrix} = \left(\begin{array}{cc} \begin{array}{c} 0 \\ \diagdown \\ 0 \end{array} & \square \\ \hline 0 & 0 \end{array} \right) \tag{2.3}$$

for $\tilde{\mathcal{U}}_l$. The elimination process adds one row/column at every step l.

Depending on how the approximate Schur complement $\tilde{S}$ is defined, a different error matrix $\mathcal{E}_k$ will be obtained. We will only distinguish between two choices.

S- : This corresponds to the "simple" approximate Schur complement defined by

$$\tilde{S} = C - \tilde{L}_E \tilde{D}_B \tilde{U}_F. \tag{2.4}$$

T- : This corresponds to the more expensive Schur complement proposed in [23] and defined by

$$\tilde{T} = \begin{pmatrix} -\tilde{L}_E \tilde{L}_B^{-1} & I \end{pmatrix} P^\top A Q \begin{pmatrix} -\tilde{U}_B^{-1} \tilde{U}_F \\ I \end{pmatrix}. \tag{2.5}$$

Note that the T-version of the Schur complement results from applying the inverse factors $\tilde{\mathcal{L}}_k^{-1}$ to the left and $\tilde{\mathcal{U}}_k^{-1}$ to the right to $P^\top A Q$ and taking the lower right block. Obtaining $\tilde{T}$ does not really require inverting $\tilde{L}_B$ and $\tilde{U}_B$. It can be computed by updating $\tilde{T}$ by a low-rank matrix at each step $l = 1, \ldots, k$.

The next result shows how the error matrix $\mathcal{E}_k$ can be characterized for the two choices of the approximate Schur complement.

LEMMA 2.1. (2.4)

$$\mathcal{E}_k = \mathcal{V}_k \tilde{\mathcal{D}}_k + \tilde{\mathcal{D}}_k \mathcal{W}_k, \tag{2.6}$$

(2.5)

$$\mathcal{E}_k = \mathcal{V}_k \tilde{\mathcal{D}}_k \tilde{\mathcal{U}}_k + \tilde{\mathcal{L}}_k \tilde{\mathcal{D}}_k \mathcal{W}_k. \tag{2.7}$$

. We will show this by induction. Initially at step $l = 0$ there is no error present and we set $\mathcal{V}_0 = \mathcal{W}_0 = 0$ and $\tilde{\mathcal{L}}_0 = \tilde{\mathcal{U}}_0 = I$.

Now going from step $l-1$ to step l ($l \leqslant k$) we have

$$P^\top A Q = \tilde{\mathcal{L}}_{l-1} \tilde{\mathcal{D}}_{l-1} \tilde{\mathcal{U}}_{l-1} + \mathcal{E}_{l-1}.$$

To distinguish between step $l-1$ and l we add a subscript l to all matrices. We partition the approximate Schur complement $\tilde{S}_{l-1}$ from $\tilde{\mathcal{D}}_{l-1}$ as

$$\tilde{S}_{l-1} = \begin{pmatrix} \beta & f^\top \\ e & \hat{C} \end{pmatrix}.$$

At step l we get

$$\tilde{\mathcal{L}}_l = \tilde{\mathcal{L}}_{l-1} \left(I + [y - v]\, e_l^\top \right),$$

where

$$y = \begin{pmatrix} 0 \\ 0 \\ \beta^{-1} e \end{pmatrix} \begin{matrix} \leftarrow \text{size} = l-1, \\ \leftarrow \text{size} = 1, \\ \leftarrow \text{size} = n-l \end{matrix}$$

is the leading column of the approximate Schur complement divided by the diagonal entry and

$$v = \begin{pmatrix} 0 \\ 0 \\ \beta^{-1} \varepsilon_e \end{pmatrix}$$

denotes the vector of those entries from y that are dropped at step l. For the U-part, we similarly obtain

$$\tilde{\mathcal{U}}_l = (I + e_l(z-w)^\top)\tilde{\mathcal{U}}_{l-1},$$

where

$$z^\top = \begin{pmatrix} 0 & 0 & f^\top/\beta \end{pmatrix}$$

and

$$w^\top = \begin{pmatrix} 0 & 0 & \varepsilon_f^\top/\beta \end{pmatrix}$$

denotes the vector of those entries from z being dropped.

We now consider the S-version and T-version separately, beginning with the S-version. For the S-version the next approximate Schur complement $\tilde{S}_l$ is set as

$$\tilde{S}_l = \hat{C} - (e-\varepsilon_e)\beta^{-1}(f-\varepsilon_f)^\top.$$

It follows that

$$\begin{aligned}
&\tilde{\mathcal{L}}_l\tilde{\mathcal{D}}_l\tilde{\mathcal{U}}_l \\
&= \begin{pmatrix} L_{B,l-1} & 0 \\ L_{E,l-1} & \begin{pmatrix} 1 & 0 \\ \frac{e-\varepsilon_e}{\beta} & I \end{pmatrix} \end{pmatrix} \begin{pmatrix} D_{B,l-1} & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \tilde{S}_l \end{pmatrix} \begin{pmatrix} U_{B,l-1} & U_{F,l-1} \\ 0 & \begin{pmatrix} 1 & \frac{(f-\varepsilon_f)^\top}{\beta} \\ 0 & I \end{pmatrix} \end{pmatrix} \\
&= \begin{pmatrix} L_{B,l-1} & 0 \\ L_{E,l-1} & I \end{pmatrix} \begin{pmatrix} D_{B,l-1} & 0 & 0 \\ 0 & \beta & (f-\varepsilon_f)^\top \\ 0 & e-\varepsilon_e & \hat{C} \end{pmatrix} \begin{pmatrix} U_{B,l-1} & U_{F,l-1} \\ 0 & I \end{pmatrix} \\
&= \tilde{\mathcal{L}}_{l-1}\tilde{\mathcal{D}}_{l-1}\tilde{\mathcal{U}}_{l-1} - v\beta e_l^\top - e_l\beta w^\top.
\end{aligned}$$

By induction we already have

$$\tilde{\mathcal{L}}_{l-1}\tilde{\mathcal{D}}_{l-1}\tilde{\mathcal{U}}_{l-1} = P^\top AQ - \mathcal{E}_{l-1} = P^\top AQ - \mathcal{V}_{l-1}\tilde{\mathcal{D}}_{l-1} - \tilde{\mathcal{D}}_{l-1}\mathcal{W}_{l-1},$$

and by definition we have $v = \mathcal{V}_l e_l$ and $w^\top = e_l^\top \mathcal{W}_l$. The leading $l \times l$ part of $\tilde{\mathcal{D}}_l$ is diagonal and the (l,l) entry is just β. Thus we obtain

$$\tilde{\mathcal{L}}_l\tilde{\mathcal{D}}_l\tilde{\mathcal{U}}_l = P^\top AQ - \mathcal{V}_{l-1}\tilde{\mathcal{D}}_{l-1} - \tilde{\mathcal{D}}_{l-1}\mathcal{W}_{l-1} - v\beta e_l^\top - e_l\beta w^\top = P^\top AQ - \mathcal{V}_l\tilde{\mathcal{D}}_l - \tilde{\mathcal{D}}_l\mathcal{W}_l,$$

which shows (2.6).

We now consider the T-version. In contrast with the S-version, for the T-version we get for $\tilde{T}_l$

$$\tilde{T}_l = \left(-(e-\varepsilon_e)\frac{1}{\beta},\ I\right)\tilde{S}_{l-1}\begin{pmatrix} -(f-\varepsilon_f)^\top\frac{1}{\beta} \\ I \end{pmatrix} = \hat{C} - e\frac{1}{\beta}f^\top + \varepsilon_e\frac{1}{\beta}\varepsilon_f^\top.$$

From this it follows that

$$\begin{aligned}
\tilde{\mathcal{L}}_l^{-1}P^\top AQ\tilde{\mathcal{U}}_l^{-1} &= \tilde{\mathcal{L}}_l^{-1}\left(\tilde{\mathcal{L}}_{l-1}\tilde{\mathcal{D}}_{l-1}\tilde{\mathcal{U}}_{l-1} + \mathcal{E}_{l-1}\right)\tilde{\mathcal{U}}_l^{-1} \\
&= (I-(y-v)e_l^\top)\tilde{\mathcal{D}}_{l-1}(I-e_l(z-w)^\top) \\
&\quad +\tilde{\mathcal{L}}_l^{-1}\mathcal{V}_{l-1}\tilde{\mathcal{D}}_{l-1}(I-e_l(z-w)^\top) \\
&\quad +(I-(y-v)e_l^\top)\tilde{\mathcal{D}}_{l-1}\mathcal{W}_{l-1}\tilde{\mathcal{U}}_l^{-1} \\
&= \tilde{\mathcal{D}}_l + v\beta e_l^\top + e_l\beta w^\top \\
&\quad +\tilde{\mathcal{L}}_l^{-1}\mathcal{V}_{l-1}\tilde{\mathcal{D}}_{l-1} \\
&\quad +\tilde{\mathcal{D}}_{l-1}\mathcal{W}_{l-1}\tilde{\mathcal{U}}_l^{-1}.
\end{aligned}$$

For the last equation we used the definition of $\tilde{\mathcal{D}}_l$ and, in particular, that of $\tilde{T}_l$. Also note that only the leading $l-1$ columns of $\mathcal{V}_{l-1}\tilde{\mathcal{D}}_{l-1}$ are nonzero (analogously, $e_l^\top \tilde{\mathcal{D}}_{l-1}\mathcal{W}_{l-1} = 0$).

It is easy to verify that $v\beta e_l^\top = \tilde{\mathcal{L}}_l^{-1} v\beta e_l^\top$ since the leading l entries of v are zero. A similar argument can be made for $e_l\beta w^\top$. Altogether we have

$$\tilde{\mathcal{L}}_l^{-1} P^\top A Q \tilde{\mathcal{U}}_l^{-1} = \tilde{\mathcal{D}}_l + \tilde{\mathcal{L}}_l^{-1}\mathcal{V}_l\tilde{\mathcal{D}}_l + \tilde{\mathcal{D}}_l\mathcal{W}_l\tilde{\mathcal{U}}_l^{-1},$$

since $v\beta e_l^\top$ and $e_l\beta w^\top$ are the l column/row of $\mathcal{V}_l\tilde{\mathcal{D}}_l$ and $\tilde{\mathcal{D}}_l\mathcal{W}_l$. This completes the proof. ∎

A corollary of the representation of the error is obtained when we consider the inverse error

$$\mathcal{F}_k = \tilde{\mathcal{L}}_k^{-1}\mathcal{E}_k\tilde{\mathcal{U}}_k^{-1}. \tag{2.8}$$

COROLLARY 2.2. [illegible] 2.1

$$\mathcal{F}_k = \tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k\tilde{\mathcal{U}}_k^{-1} + \tilde{\mathcal{L}}_k^{-1}\tilde{\mathcal{D}}_k\mathcal{W}_k\tilde{\mathcal{U}}_k^{-1}, \tag{2.9}$$

[illegible]

$$\mathcal{F}_k = \tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k + \tilde{\mathcal{D}}_k\mathcal{W}_k\tilde{\mathcal{U}}_k^{-1}. \tag{2.10}$$

An important consequence of Corollary 2.2 is that the inverse triangular factors $\tilde{\mathcal{L}}_k^{-1}$ and $\tilde{\mathcal{U}}_k^{-1}$ amplify the size of the entries being dropped during the incomplete LU decomposition. For the S-version this impact is likely to be stronger since both inverse factors contribute to the error at the same time.

An interesting observation can be made when taking a detailed look at the patterns of the inverse errors. To do this, partition $\mathcal{V}_k$ and $\mathcal{W}_k$ as

$$\mathcal{V}_k = \begin{pmatrix} V_B & 0 \\ V_E & 0 \end{pmatrix} \equiv \begin{pmatrix} \overset{0}{\blacktriangle} & 0 \\ \underset{0}{\blacksquare} & 0 \end{pmatrix}, \tag{2.11}$$

$$\mathcal{W}_k = \begin{pmatrix} W_B & W_F \\ 0 & 0 \end{pmatrix} \equiv \begin{pmatrix} \overset{0}{\blacktriangle} & \blacksquare \\ \underset{0}{} & 0 \end{pmatrix}. \tag{2.12}$$

Then the inverse error of the S-version can be sketched with a pattern

$$\mathcal{F}_k = \underbrace{\begin{pmatrix} \overset{0}{\blacktriangle} \\ \underset{0}{\blacksquare} \end{pmatrix}\begin{pmatrix} \blacktriangle & \blacksquare \end{pmatrix}}_{(\tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k)\cdot\tilde{\mathcal{U}}_k^{-1}} + \underbrace{\begin{pmatrix} \blacksquare \\ \blacksquare \end{pmatrix}\begin{pmatrix} \overset{0}{\blacktriangle} & \blacksquare \end{pmatrix}}_{\tilde{\mathcal{L}}_k^{-1}\cdot(\mathcal{V}_k\tilde{\mathcal{D}}_k\tilde{\mathcal{U}}_k^{-1})},$$

which shows that the error produced by dropping in the lower triangular part also contributes to the upper triangular part and vice versa. However, this is different for the T-version. Here we have

$$\mathcal{F}_k = \underbrace{\begin{pmatrix} \overset{0}{\blacktriangle} & 0 \\ \underset{0}{\blacksquare} & 0 \end{pmatrix}}_{\tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k} + \underbrace{\begin{pmatrix} \overset{0}{\blacktriangle} & \blacksquare \\ 0 & 0 \end{pmatrix}}_{\tilde{\mathcal{D}}_k\mathcal{W}_k\tilde{\mathcal{U}}_k^{-1}},$$

showing that dropping in the lower/upper triangular part produces errors in these parts only, and by the definition of the approximate Schur complement there clearly will be no perturbation in the lower right block. Note also that for the T-version, there are no errors in the (2,2) block, which is not surprising since the Schur complement is computed exactly from applying the inverses of the (inexact) factors $\mathcal{L}_k$ and $\mathcal{U}_k$ to $P^\top AQ$.

Let $\tilde{d}_1, \ldots, \tilde{d}_k$ be the leading k diagonal entries of $\tilde{\mathcal{D}}_k$. If we keep in mind that the entries being dropped at step l are the entries in column l of $\mathcal{V}_l$ and $\mathcal{W}_l^\top$, then we can see that for the S-version we have

$$\mathcal{F}_k = \sum_{l \leqslant k} \left(\tilde{\mathcal{L}}_k^{-1} v_l \tilde{d}_l e_l^\top \tilde{\mathcal{U}}_k^{-1} + \tilde{\mathcal{L}}_k^{-1} e_l \tilde{d}_l w_l^\top \tilde{\mathcal{U}}_k^{-1} \right) \tag{2.13}$$

and, analogously for the T-version,

$$\mathcal{F}_k = \sum_{l \leqslant k} \left(\tilde{\mathcal{L}}_k^{-1} v_l \tilde{d}_l e_l^\top + e_l \tilde{d}_l w_l^\top \tilde{\mathcal{U}}_k^{-1} \right). \tag{2.14}$$

Equation (2.13) implies that the perturbation introduced at step l of the incomplete LU decomposition is given by

$$\tilde{\mathcal{L}}_k^{-1} v_l \tilde{d}_l e_l^\top \tilde{\mathcal{U}}_k^{-1}$$

from the L-part and

$$\tilde{\mathcal{L}}_k^{-1} e_l \tilde{d}_l w_l^\top \tilde{\mathcal{U}}_k^{-1}$$

from the U-part. For the T-version one can skip one of the inverse factors, as can be seen from (2.14). This indicates that it is wise to incorporate the norm $\|\tilde{\mathcal{L}}_k^{-1}\|$ into the dropping strategy at step l when entries of $\mathcal{L}_l$ are being dropped. In principle one should also include the norm $\|\tilde{\mathcal{U}}_k^{-1}\|$ in the S-version case when dropping in $\mathcal{L}_k$. Similar arguments apply when dropping in $\mathcal{U}_k$.

Unless we control the growth of $\tilde{\mathcal{L}}_k^{-1}$ and $\tilde{\mathcal{U}}_k^{-1}$ and keep their norm below a constant κ, the error introduced by dropping at step l can be arbitrarily amplified by the inverse factors.

COROLLARY 2.3. [illegible] $\|L_l^{-1}\| \leqslant \kappa$ [illegible] $\|U_l^{-1}\| \leqslant \kappa$ [illegible] $l = 1, \ldots, k$ [illegible] $\kappa > 0$ [illegible] $\tilde{d}_l$ [illegible] l

1. [illegible] $\tilde{s}_{lj}$ [illegible] $\tilde{s}_{il}$ [illegible] $i, j > l$ [illegible] l [illegible] $\tilde{s}_{lj}$ $\tilde{s}_{il}$ [illegible]

$$\kappa^2 |\tilde{s}_{lj}| \leqslant \varepsilon |\tilde{d}_l|, \ \kappa^2 |\tilde{s}_{il}| \leqslant \varepsilon |\tilde{d}_l|;$$

[illegible]

$$\mathcal{F}_k = \sum_{l=1}^{k} \mathcal{F}_{L,l} + \sum_{l=1}^{k} \mathcal{F}_{U,l},$$

[illegible] $F_{L,l}$ [illegible] $F_{U,l}$ [illegible] 1 [illegible] $\varepsilon |\tilde{d}_l|$

2. [illegible] $\tilde{t}_{lj}$ [illegible] $\tilde{t}_{il}$ [illegible] $i,j > l$ [illegible] l [illegible] $i,j > l$ $\tilde{t}_{lj}$ [illegible] $\tilde{t}_{il}$ [illegible]

$$\kappa|\tilde{t}_{lj}| \leqslant \varepsilon|\tilde{d}_l|, \ \kappa|\tilde{t}_{il}| \leqslant \varepsilon|\tilde{d}_l|;$$

[illegible]

$$\mathcal{F}_k = \mathcal{F}_{L,k}\tilde{\mathcal{D}}_k + \tilde{\mathcal{D}}_k\mathcal{F}_{U,k},$$

[illegible]

$$\max_{i,j} e_i^\top |\mathcal{F}_{L,k}| e_j \leqslant \varepsilon, \qquad \max_{i,j} e_i^\top |\mathcal{F}_{U,k}| e_j \leqslant \varepsilon.$$

[illegible]. The results follow immediately from the assumptions and (2.13) for the S-version case and (2.14) for the T-version case. □

2.2. The multilevel framework. Multilevel ILU procedures do not usually proceed until $k = n$; stop at a certain step k and postpone factoring the remaining part to the next level. Thus, one can write at a given level η

$$P^\top A_\eta Q = \begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} \tilde{L}_B & 0 \\ \tilde{L}_E & I \end{pmatrix} \begin{pmatrix} \tilde{D}_B & 0 \\ 0 & \tilde{S} \end{pmatrix} \begin{pmatrix} \tilde{U}_B & \tilde{U}_F \\ 0 & I \end{pmatrix} + \mathcal{E}_k \rightarrow A_{\eta+1} = \tilde{S}.$$

Postponing the factorization of $A_{\eta+1} \equiv S$ may be mandatory because the prescribed bound may otherwise be exceeded. When considering the next level matrix $A_{\eta+1}$, specific techniques, such are nonsymmetric permutations, can be invoked to ensure that the factorization at this level is more accurate and reliable. The simple version of the Schur complement is often preferred since it leads to less fill-in than the T-version. However, the stability analysis seen above has shown that this version is much more sensitive to perturbations. On the other hand, the T-version gains from its higher robustness, but it will suffer from the fact that it is more expensive to compute and it may have significantly more fill-in.

A natural way of combining the S-version and the T-version in a multilevel framework could be to approximate B only using the S-version. This could still give a good approximation as long as B is, say, close to being diagonal dominant. But for the next level which requires the remaining Schur complement, one could switch to the T-version. This leads to the M-version.

M-[illegible]: Define the approximate Schur complement via

$$\tilde{M} = \begin{pmatrix} -\tilde{L}_E\tilde{L}_B^{-1} & I \end{pmatrix} P^\top A Q \begin{pmatrix} -\tilde{U}_B^{-1}\tilde{U}_F \\ I \end{pmatrix} \tag{2.15}$$

at step k and via the S-version prior to step k.

The motivation for this variant is that in a multilevel approach, it sometimes is possible to ensure that the factors of the B block are nicely bounded. In this situation it pays to compute the Schur complement more accurately, and the T-version may be ideal for this.

COROLLARY 2.4. [illegible] LU [illegible] $l = 1, \ldots, k$ [illegible] k [illegible]

$$\Pi = \begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix}.$$

(2.16)
$$\begin{aligned}\mathcal{F}_k = &\left(\tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k\tilde{\mathcal{U}}_k^{-1} + \tilde{\mathcal{L}}_k^{-1}\tilde{\mathcal{D}}_k\mathcal{W}_k\tilde{\mathcal{U}}_k^{-1}\right)\\ &-\Pi\left(\tilde{\mathcal{L}}_k^{-1}\mathcal{V}_k\tilde{\mathcal{D}}_k\tilde{\mathcal{U}}_k^{-1} + \tilde{\mathcal{L}}_k^{-1}\tilde{\mathcal{D}}_k\mathcal{W}_k\tilde{\mathcal{U}}_k^{-1}\right)\Pi.\end{aligned}$$

. Since we compute until step k the incomplete LU decomposition via the S-version, we will have until that step

$$\mathcal{E}_k = \mathcal{V}_k\tilde{\mathcal{D}}_k + \tilde{\mathcal{D}}_k\mathcal{W}_k.$$

As we now switch to the T-version, it means that we precisely have

$$\tilde{M} = \begin{pmatrix} 0 & I \end{pmatrix}\tilde{\mathcal{L}}_k^{-1}(P^\top AQ)\tilde{\mathcal{U}}_k^{-1}\begin{pmatrix} 0 \\ I \end{pmatrix} \Rightarrow \begin{pmatrix} 0 & I \end{pmatrix}\mathcal{F}_k\begin{pmatrix} 0 \\ I \end{pmatrix} = 0.$$

This means that there will be no error in the lower right block. ∎

The inverse error that has been computed so far is based on the assumption that the approximate factorization from (2.1) is used with either the S-version or the T-version as approximate Schur complement. Since we would like to use this ILU in a multilevel context, we do not have to keep the factors $\tilde{L}_E$ and $\tilde{U}_F$. Instead we can substitute them with $E\tilde{U}_B^{-1}\tilde{D}_B^{-1}$ and $\tilde{D}_B^{-1}\tilde{L}_B^{-1}F$. Keeping in mind that we will not compute these factors explicitly but only solve systems with them, this leads to the approximate factorization

(2.17)
$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \underbrace{\begin{pmatrix} \tilde{L}_B & 0 \\ E\tilde{U}_B^{-1}\tilde{D}_B^{-1} & I \end{pmatrix}}_{\tilde{L}_k}\underbrace{\begin{pmatrix} \tilde{D}_B & 0 \\ 0 & \tilde{S} \end{pmatrix}}_{\tilde{D}_k}\underbrace{\begin{pmatrix} \tilde{U}_B & \tilde{D}_B^{-1}\tilde{L}_B^{-1}F \\ 0 & I \end{pmatrix}}_{\tilde{U}_k} + \mathcal{E}_k.$$

In this case one can easily see that the error and the inverse error only show up in the B-part and the C-part.

THEOREM 2.5. *LU* (2.17) $\mathcal{V}_k$ $\mathcal{W}_k$ (2.11)–(2.12)

1.
$$\mathcal{F}_k = \tilde{\mathcal{L}}_k^{-1}\begin{pmatrix} V_B\tilde{D}_B + \tilde{D}_BW_B & 0 \\ 0 & 0 \end{pmatrix}\tilde{\mathcal{U}}_k^{-1}.$$

2.
$$\begin{aligned}\mathcal{F}_k = &\quad \tilde{\mathcal{L}}_k^{-1}\begin{pmatrix} V_B\tilde{D}_B + \tilde{D}_BW_B & 0 \\ 0 & 0 \end{pmatrix}\tilde{\mathcal{U}}_k^{-1}\\ &-\Pi\tilde{\mathcal{L}}_k^{-1}\begin{pmatrix} V_B\tilde{D}_B + \tilde{D}_BW_B & 0 \\ 0 & 0 \end{pmatrix}\tilde{\mathcal{U}}_k^{-1}\Pi,\end{aligned}$$

Π 2.4

. From (2.17) one can immediately see that

$$\mathcal{E}_k = \begin{pmatrix} * & 0 \\ 0 & * \end{pmatrix}.$$

Then the assertion follows immediately from Corollaries 2.2 and 2.4. ∎

In principle, for the T-version the same approach leads to analogous bounds which are more complicated. But one essential property will be lost: Entries dropped in the lower triangular part would only contribute to the perturbation in this part (similar for the upper triangular part).

2.3. The inverse error contribution from lower levels. In the multilevel context the best we can achieve is to find permutation matrices P_C and Q_C for the approximate Schur complement $\tilde{S}$ such that

$$\left.\begin{array}{c} P_C^\top \tilde{S} Q_C \\ P_C^\top \tilde{M} Q_C \\ P_C^\top \tilde{T} Q_C \end{array}\right\} = L_C D_C U_C + \mathcal{E}_C,$$

where D_C is diagonal and $\|L_C^{-1}\|, \|U_C^{-1}\| \leqslant \kappa$ with a reasonable small inverse error $\mathcal{F}_C = L_C^{-1} \mathcal{E}_C U_C^{-1}$. In this case we finally end up with an approximate factorization:

$$\begin{aligned} &\begin{pmatrix} I & 0 \\ 0 & L_C^{-1} P_C^\top \end{pmatrix} \mathcal{L}_k^{-1} P^\top A Q \mathcal{U}_k^{-1} \begin{pmatrix} I & 0 \\ 0 & Q_C U_C^{-1} \end{pmatrix} \\ &= \begin{pmatrix} D_B & 0 \\ 0 & D_C \end{pmatrix} + \begin{pmatrix} I & 0 \\ 0 & L_C^{-1} P_C^\top \end{pmatrix} \mathcal{F}_k \begin{pmatrix} I & 0 \\ 0 & Q_C U_C^{-1} \end{pmatrix} + \begin{pmatrix} I & 0 \\ 0 & \mathcal{F}_C \end{pmatrix}. \end{aligned}$$

This result indicates that the inverse error might be amplified by the additional inverse factors L_C^{-1} and U_C^{-1} from the lower levels. Although this is an extremal case and we expect this to rarely happen in practice, this suggests again that it is strongly advisable to also keep the inverse factors L_C^{-1} and U_C^{-1} bounded.

2.4. Perturbation of the approximate Schur complement. The results discussed so far dealt with perturbations introduced by dropping compared with the diagonal part and the remaining approximate Schur complement

$$\tilde{\mathcal{D}}_k = \begin{pmatrix} \tilde{D}_B & 0 \\ 0 & \tilde{X} \end{pmatrix},$$

where $\tilde{X}$ is one of $\tilde{S}$, $\tilde{M}$, or $\tilde{T}$. Nothing has been said so far about the error between the approximate Schur complements $\tilde{S}$, $\tilde{M}$, and $\tilde{T}$ and the exact Schur complement $S = C - EB^{-1}F$. This information is of great importance since the approximate Schur complement will be used as the input matrix for the next level. The perturbation results deal only with the approximate Schur complement, and as long as the inverse triangular factors are kept bounded it is likely that the errors are small. But in a relative sense even small perturbations may have a serious impact on the preconditioned system if the diagonal entries of $\mathcal{D}_k$ become small in absolute value. This may be a property of the underlying original system. But this may in particular happen if the approximate Schur complement becomes ill conditioned.

For this reason we will investigate the error introduced to the approximate Schur complement by computing the incomplete LU factorization. An incomplete LU decomposition results in an approximate factorization from (2.1) of the type

$$A = \tilde{\mathcal{L}}_k \tilde{\mathcal{D}}_k \tilde{\mathcal{U}}_k + \mathcal{E}_k,$$

where it is assumed that no pivoting is necessary to ensure that $\|\tilde{\mathcal{L}}_k\|, \|\tilde{\mathcal{U}}_k\| \leqslant \kappa$. Suppose that we can also exactly factor A as

$$A = \mathcal{L}_k \mathcal{D}_k \mathcal{U}_k,$$

where

$$\mathcal{L}_k = \begin{pmatrix} L_B & 0 \\ L_E & I \end{pmatrix}, \quad \mathcal{U}_k = \begin{pmatrix} U_B & U_F \\ 0 & I \end{pmatrix}, \quad \mathcal{D}_k = \begin{pmatrix} D_B & 0 \\ 0 & S \end{pmatrix},$$

and $L_B, U_B^\top$ are unit lower triangular; D_B is diagonal, and S is the exact Schur complement.

Comparing the exact Schur complement and the approximate Schur complements we find that

$$\tilde{\mathcal{L}}_k^{-1}\mathcal{L}_k\mathcal{D}_k\mathcal{U}_k\tilde{\mathcal{U}}_k^{-1} = \tilde{\mathcal{D}}_k + \mathcal{F}_k.$$

Looking only at the lower right block yields

$$\left.\begin{matrix}\tilde{S}\\ \tilde{M}\\ \tilde{T}\end{matrix}\right\} = S - \tilde{L}_E\tilde{L}_B^{-1}(L_B - \tilde{L}_B)D_B(U_B - \tilde{U}_B)\tilde{U}_B^{-1}\tilde{U}_F - (0,I)\mathcal{F}_k\begin{pmatrix}0\\ I\end{pmatrix}. \tag{2.18}$$

From this equation we can see that the error between the exact and approximate Schur complements is driven by

1. the previous error $L_B - \tilde{L}_B$ and $U_B - \tilde{U}_B$ between both factorizations,
2. the inverse error $(0,I)\mathcal{F}_k\left(\begin{smallmatrix}0\\ I\end{smallmatrix}\right)$ produced by dropping and the choice of the approximate Schur complement, and
3. the norm of the inverse triangular factors $\tilde{\mathcal{L}}_k^{-1}$ and $\tilde{\mathcal{U}}_k^{-1}$ ($\|\tilde{L}_E\tilde{L}_B^{-1}\|, \|\tilde{U}_B^{-1}\tilde{U}_F\| \leqslant \kappa$).

Suppose that the error from previous steps is bounded by ε. If we ensure that at step k entries are dropped such that $\|(0,I)\mathcal{F}_K\left(\begin{smallmatrix}0\\ I\end{smallmatrix}\right)\| \leqslant \varepsilon$ and that $\varepsilon^2\kappa^2 \leqslant \varepsilon$, then the remaining error in (2.18) will reveal the same order ε. We will summarize this observation in a theorem.

Denote the entries of the approximate Schur complements $\tilde{S}^{(l)}, \tilde{T}^{(l)}$ after $l = 1, \ldots, k$ steps of incomplete Gaussian elimination by $\tilde{s}_{ij}^{(l)}$ ($\tilde{t}_{ij}^{(l)}$, respectively). To simplify the analysis we assume that

$$\Gamma \geqslant |\tilde{s}_{ll}^{(m)}|, |\tilde{t}_{ll}^{(m)}| \geqslant \gamma, \tag{2.19}$$

$m = 0, \ldots, k$, $l = 1, \ldots, k$. For the leading k diagonal entries this means that they are approximately of the same order. Initially this can be achieved by scaling and reordering, which will in fact be done in the section on numerical examples.

THEOREM 2.6. [illegible] (2.19) [illegible] κ [illegible] ε [illegible] $\frac{1}{\kappa^2}$

1. [illegible] m $m = 1, \ldots, k$ [illegible] l_{im} u_{mj} $\tilde{\mathcal{L}}_m$ $\tilde{\mathcal{U}}_m$ [illegible]

$$|\tilde{l}_{im}|, |\tilde{u}_{mj}| \leqslant \frac{\varepsilon}{\kappa^2}.$$

[illegible] K [illegible] i, j [illegible]

$$|s_{ij} - \tilde{s}_{ij}| \leqslant K\varepsilon.$$

2. [illegible] K [illegible] i, j [illegible]

$$|s_{ij} - \tilde{m}_{ij}| \leqslant K(\kappa\varepsilon)^2.$$

3. [illegible] m $m = 1, \ldots, k$ [illegible] l_{im} [illegible] u_{mj} [illegible] $\tilde{\mathcal{L}}_m$ [illegible] $\tilde{\mathcal{U}}_m$ [illegible]

$$|\tilde{l}_{im}|, |\tilde{u}_{mj}| \leqslant \varepsilon.$$

[illegible] K [illegible] i, j [illegible]

$$|s_{ij} - \tilde{t}_{ij}| \leqslant K(\kappa\varepsilon)^2.$$

[illegible] Suppose that at step m we have

$$|s_{ij}^{(m)} - \tilde{s}_{ij}^{(m)}| \leqslant K\varepsilon$$

for the S- and M-versions, respectively, and $|s_{ij}^{(m)} - \tilde{t}_{ij}^{(m)}| \leqslant K\varepsilon$ for the T-version. Because of (2.19) it follows that there exists a further constant C such that

$$|l_{ij} - \tilde{l}_{ij}|, |u_{ji} - \tilde{u}_{ji}| \leqslant C\varepsilon, \qquad j = 1, \ldots, m, \quad i > j.$$

From (2.18) we obtain that after step $m+1$, the error $|s_{ij}^{(m+1)} - \tilde{s}_{ij}^{(m+1)}|$ and $|s_{ij}^{(m+1)} - \tilde{t}_{ij}^{(m+1)}|$ can be bounded by

$$\max |\tilde{L}_E \tilde{L}_B^{-1}(L_B - \tilde{L}_B) D_B (U_B - \tilde{U}_B)\tilde{U}_B^{-1}\tilde{U}_F| + \max \left| (0, I)\mathcal{F}_k \begin{pmatrix} 0 \\ I \end{pmatrix} \right|.$$

For the first part we obtain the bound

$$\max |\tilde{L}_E \tilde{L}_B^{-1}(L_B - \tilde{L}_B) D_B (U_B - \tilde{U}_B)\tilde{U}_B^{-1}\tilde{U}_F| \leqslant \Gamma\varepsilon^2\kappa^2.$$

Since $\varepsilon\kappa^2 < 1$, we see that

$$\max |\tilde{L}_E \tilde{L}_B^{-1}(L_B - \tilde{L}_B) D_B (U_B - \tilde{U}_B)\tilde{U}_B^{-1}\tilde{U}_F| \leqslant \Gamma\varepsilon.$$

Suppose we can show that the entries $|s_{ij}^{(m+1)} - \tilde{s}_{ij}^{(m+1)}|$ and $|s_{ij}^{(m+1)} - \tilde{t}_{ij}^{(m+1)}|$ are bounded by $K\varepsilon$. Then the same has to hold for the $(m+1)$ column of $\mathcal{L}_{m+1} - \tilde{\mathcal{L}}_{m+1}$ and $\mathcal{U}_{m+1}^\top - \tilde{\mathcal{U}}_{m+1}^\top$, since by hypothesis (2.19) the diagonal entries of the approximate Schur complement are uniformly bounded away from zero.

It remains to show that we can bound the error $|s_{ij}^{(m+1)} - \tilde{s}_{ij}^{(m+1)}|$, $|s_{ij}^{(m+1)} - \tilde{t}_{ij}^{(m+1)}|$ by $K\varepsilon$; i.e., we have to show that

$$\max |(0, I)\mathcal{F}_k \begin{pmatrix} 0 \\ I \end{pmatrix}| \leqslant L\varepsilon$$

for some constant L.

1. According to Theorem 2.5, for the S-version we will have

$$\mathcal{F}_{m+1} = \tilde{\mathcal{L}}_{m+1}^{-1} \begin{pmatrix} V_B \tilde{D}_B + \tilde{D}_B W_B & 0 \\ 0 & 0 \end{pmatrix} \tilde{\mathcal{U}}_{m+1}^{-1}.$$

Thus, if the entries in L_B and U_B in every step are dropped at most if they are less than ε/κ^2, then the inverse error is also bounded by a constant L times ε. From this it follows that the error $|s_{ij}^{(m+1)} - \tilde{s}_{ij}^{(m+1)}|$ remains below a constant times ε.

2. For step $1, \ldots, k$, the M-version and the S-version are the same. The difference between the M-version and the S-version is that after the final step k the remaining Schur complement is computed in a different way; i.e., the lower right block of the inverse error $\mathcal{F}_k$ vanishes.
3. For the T-version we note that by Corollary 2.2 the inverse error $\mathcal{F}_m$ is given by

$$\mathcal{F}_m = \tilde{\mathcal{L}}_m^{-1}\mathcal{V}_m\tilde{\mathcal{D}}_m + \tilde{\mathcal{D}}_m\mathcal{W}_m\tilde{\mathcal{U}}_m^{-1}.$$

That is, the entries that are dropped in $\tilde{\mathcal{L}}_m$ and $\tilde{\mathcal{U}}_m$ are only amplified by one inverse triangular factor. But by construction there is no inverse error in those positions, where the entries of the approximate Schur complement $\tilde{T}$ are located. Thus we don't have to divide by κ (or κ^2). Since $\varepsilon\kappa^2 < 1$, the error between the exact and the approximate Schur complements can be estimated by some constant K times ε. ∎

- For Theorem 2.6 it is essential to require that (2.19) be fulfilled. This is needed to ensure that the leading k diagonal entries of the approximate Schur complement are uniformly bounded away from zero, as well as to protect $\tilde{L}_E\tilde{L}_B^{-1}(L_B - \tilde{L}_B)D_B(U_B - \tilde{U}_B)\tilde{U}_B^{-1}\tilde{U}_F$ from being amplified by too large diagonal entries in D_B.
  This certainly indicates a strong need for preordering and scaling algorithms that properly prepare the leading block B. To safeguard this process, small diagonal pivots should be moved to the end. This is a side effect of keeping the inverse triangular factors below a bound κ, since $\kappa|s_{mm}^{(m)}| \geqslant \max_{i,j}\{|s_{im}^{(m)}|, |s_{mj}^{(m)}|\}$ is necessary to keep the inverse triangular factors below κ.
- From Theorem 2.6 we may conclude that the entries of $\tilde{\mathcal{L}}_k$ and $\tilde{\mathcal{U}}_k$ outside the leading $k \times k$ block are not needed at all. This is only true if the approximate Schur complement is not computed from these entries but by using $\tilde{\mathcal{L}}_k$, $\tilde{\mathcal{U}}_k$ from (2.17). Otherwise, using Corollary 2.2 we need to keep

$$\max|\mathcal{F}_{m+1}| = \max|\tilde{\mathcal{L}}_{m+1}^{-1}\mathcal{V}_{m+1}\tilde{\mathcal{D}}_{m+1}\tilde{\mathcal{U}}_{m+1}^{-1} + \tilde{\mathcal{L}}_{m+1}^{-1}\tilde{\mathcal{D}}_{m+1}\mathcal{W}_{m+1}\tilde{\mathcal{U}}_{m+1}^{-1}| \leqslant L\varepsilon,$$

which shows that even the entries in the E and F block need to be less than ε/κ^2.

3. Inverse-based multilevel ILUs. Corollary 2.2 suggests that, whenever possible, incomplete LU factorizations should be computed in such a way that the inverse triangular factors remain bounded. While keeping the entries of L and U small is straightforward, e.g., with the help of pivoting, making the same demand for the inverse matrices is more delicate. We have adopted a strategy to achieve these features, which is based on combining the following three ingredients.

1. A static preordering of the system that puts the original matrix A in the form

$$P^\top AQ = \begin{pmatrix} B & F \\ E & C \end{pmatrix},$$

where the leading block B is likely to have nicely bounded inverse triangular factors.
2. A partial incomplete LU factorization which approximately factors $P^\top AQ$ and uses pivoting to keep the inverse triangular factors below a given bound κ. The factorization is partial in that it only proceeds with the elimination of the unknowns corresponding to the B block. Details are given below.

3. A multilevel setting, possibly a recursive one, which completes the partial incomplete factorization of step 2. Indeed, this amounts to (recursively) repeating the previous two steps on the Schur complement system resulting from step 2.

A few clarifications are now given for steps 2 and 3. Step 2 starts with a new reordering $P^\top AQ \rightarrow \hat{P}^\top A\hat{Q}$ of A with a smaller leading block. This reordering consists of moving to the end rows and columns that are responsible for undesired large inverse factors. In other words, we obtain

$$\hat{P}^\top A\hat{Q} = \tilde{P}^\top (P^\top AQ)\tilde{Q} = \tilde{P}^\top \begin{pmatrix} B & F \\ E & C \end{pmatrix} \tilde{Q} = \left(\begin{array}{c|cc} B_{11} & B_{12} & F_1 \\ \hline B_{21} & B_{22} & F_2 \\ E_1 & E_2 & C \end{array}\right) \equiv \left(\begin{array}{c|c} \hat{B} & \hat{F} \\ \hline \hat{E} & \hat{C} \end{array}\right).$$

The incomplete factorization is then computed for

$$\hat{P}^\top A\hat{Q} = \left(\begin{array}{c|c} \hat{B} & \hat{F} \\ \hline \hat{E} & \hat{C} \end{array}\right) \approx \underbrace{\left(\begin{array}{c|c} L_{\hat{B}} & 0 \\ \hline L_{\hat{E}} & I \end{array}\right)}_{\mathcal{L}} \left(\begin{array}{c|c} D_{\hat{B}} & 0 \\ \hline 0 & S \end{array}\right) \underbrace{\left(\begin{array}{c|c} U_{\hat{B}} & U_{\hat{F}} \\ \hline 0 & I \end{array}\right)}_{\mathcal{U}}, \tag{3.1}$$

where $L_{\hat{B}}, U_{\hat{B}}^\top$ are unit lower triangular factors, $D_{\hat{B}}$ is diagonal, and S is the approximate Schur complement. The partial incomplete factorization is performed ensuring that the inverse triangular factors fulfill $\|\mathcal{L}^{-1}\|, \|\mathcal{U}^{-1}\| \leqslant \kappa$.

Further comments on these three pieces of the factorization are given next.

3.1. Template 1: Preorderings. Standard static ordering techniques can be used to give some desirable properties to the original matrix. Among these are all the graph-based routines to reduce fill-in and more recent approaches to improve diagonal dominance. A few popular examples of fill-reducing methods are the reverse Cuthill–McKee (RCM), the multiple minimum degree ordering (MMD), the nested dissection (ND), and the approximate minimum fill (AMF). See [11] for an overview of some of these reorderings and [16, 1] for the AMF ordering.

The nonsymmetric orderings used in MC64 [10] and in the ddPQ strategy [18] attempt to improve the diagonal dominance of the matrix in different ways. The objective is to rearrange the columns and/or the rows so that the diagonal part gets a maximum weight. In [10] scaling is part of this approach but fill reduction is performed by a static (symmetric) postordering of the resulting matrix. In contrast, ddPQ assumes that some scaling is done a priori and the reordering is performed as a compromise between diagonal dominance and fill. A more detailed description of MC64 is provided in section 4.

3.2. Template 2: Partial incomplete LU decomposition. A partial (I)LU of a partitioned matrix is a factorization of the matrix in the form of the right-hand side of (3.1). Any Gaussian elimination-based procedure can be adapted to yield a partial factorization. However, the most attractive method exploits the Crout version of Gaussian elimination; see [9, p. 50] and [14]. The Crout (sometimes referred to as the Crout–Doolittle) algorithm computes the kth column of L and the kth row of U at step k. A partial factorization consists of performing a number, say p, steps of this algorithm and then computing the Schur complement. There are several advantages of Crout versions of ILU, some of which are discussed in [14].

In the context of the partial ILU factorization needed for a multilevel scheme, diagonal pivoting is necessary. In fact pivoting is an integral part of the reordering

scheme. Diagonal pivoting relies on various estimates for the inverse triangular factors; see, e.g., [6, 12]. When the estimated norm of the inverse triangular factor exceeds the prescribed bounds, then the associated column and row are pushed to the end (see Figure 3.1). From the point of view of implementation, a diagonal pivoting strategy of this type can be easily added to the Crout factorization.

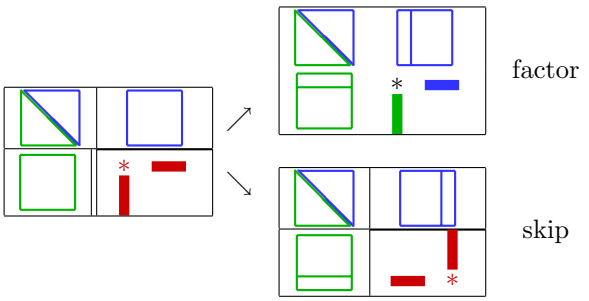


FIG. 3.1. *Diagonal pivoting in the Crout version.*

3.3. Template 3: The multilevel scheme. The next part of the preconditioner involves recursion by repeating the first two steps on the approximate Schur complement resulting from step 2. This leads naturally to a multilevel strategy. To solve a linear system using this multilevel ILU, we could simply proceed as in [20]. Based on the underlying approximate factorization

$$P^\top AQ = \begin{pmatrix} B & F \\ E & C \end{pmatrix} \approx \begin{pmatrix} I & 0 \\ EB^{-1} & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & \tilde{S}_C \end{pmatrix} \begin{pmatrix} I & B^{-1}F \\ 0 & I \end{pmatrix},$$

the associated preconditioner would require the application of

$$(P^\top AQ)^{-1} \approx \begin{pmatrix} \tilde{B}^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -\tilde{B}^{-1}F \\ I \end{pmatrix} \tilde{S}_C^{-1} \begin{pmatrix} -E\tilde{B}^{-1} & I \end{pmatrix}. \tag{3.2}$$

Here $\tilde{B}$ is the approximation to B corresponding to an ILU of B, and a solve with $\tilde{S}_C$ in (3.2) corresponds to a recursive solve invoking the next "coarser" level.

4. Numerical tests. This section presents numerical experiments to test the inverse-based multilevel ILU approach. We begin by pointing out that the algorithms described in this paper are part of a recently released package called ILUPACK [4], which is available online. Along with the package, the ILUPACK website [4] also posts a rather exhaustive set of experiments with various publicly available test matrices. The numerical computations were performed on an IBM RISC 6000 with four Power 3-II processors (375 MHz). Sixty-four bit address length and up to 4 GB memory were used. All codes use optimization.

The discussion of the experiments centers around two main questions:

- How does the reliability depend on the drop tolerance? (sensitivity)
- How many problems (statistically) can be solved with an incomplete LU factorization (with or without multilevel) if we fix the number of nonzeros of L, U relative to the nonzeros of A?

We will compare three approaches:

1. ILUTP [17, 18] using a binary search tree;

2. inverse-based ILUTC [14], which computes a single ILU without pivoting and uses the inverse triangular factors within dropping;
3. inverse-based multilevel ILU as described in section 3, hereafter referred to as ILUPACK. According to the theory presented here we will use the S-version (SIMPLE version) and the M-version (DEFAULT version).

These methods are combined with several reordering strategies:

(a) AMF [1],
(b) MMD [11],
(c) RCM [11], and
(d) ddPQ, a strategy recently presented in [19], which we will also use for the multilevel strategy.

A recent algorithm called MC64 [10] that is designed to improve the diagonal dominance is also included in the numerical results for comparison and in order to see how MC64 can help improve performance. The MC64 algorithm computes a nonsymmetric permutation via a maximum weight matching based on the bipartite graph representation of the input matrix A. This is a well-known assignment problem in operations research and combinatorial analysis. In the solution of the assignment problem, a permutation matrix P and two vectors $u = (u_i)_{i=1,\dots,n}$ and $v = (v_i)_{i=1,\dots,n}$ are generated (so-called dual variables). $D_r = \text{diag}\,(u_1, \dots, u_n)$ and $D_c = \text{diag}\,(v_1, \dots, v_n)$ can be used to rescale the input matrix and P is used to reorder its columns. The matrix $D_r A D_c P$ resulting from these operations has diagonal entries which are one in modulus and off-diagonal entries which do not exceed one in modulus. Note that in our numerical experiments the time for MC64 is almost negligible relative to that of the incomplete LU factorization and the iterative solution process. This confirms earlier observations made in [2].

All codes that are used refer to their versions as they were implemented in ILUPACK [4]. As iterative solver restarted GMRES(30) [18] is used. The iteration is stopped whenever the residual is reduced by $\sqrt{\text{eps}} \approx 1.5 \cdot 10^{-8}$. The iterative process is deemed to have failed if more than 500 steps are required or a breakdown occurs.

For the inverse-based multilevel ILU we prescribe the bound κ for the inverse factors. However, dropping is not performed with respect to some drop tolerance divided by this bound κ, but by the estimated maximum norm $\tilde{\kappa}$ of L_k^{-1} and U_k^{-1} at step k of the algorithm, which may be less than κ. This is done because one cannot predict a priori whether this bound κ is ever attained, so using $\tilde{\kappa} = \max(\|L_k^{-1}\|, \|U_k^{-1}\|)$ is a good compromise. As mentioned earlier the inverse norms are estimated by essentially using the results of [6].

4.1. Sensitivity with respect to the drop tolerance. The first set of numerical experiments investigates how the success of the incomplete LU decomposition is affected by the choice of the drop tolerance. To measure this sensitivity to drop tolerance, we use 81 test problems from PDEs available at the Davis collection [7] and from the Matrix Market [15]. We chose several larger matrices (at least a few thousand unknowns) that were available at the time of the numerical tests.

Figures 4.1, 4.2, and 4.3 show the relative number (percentage) of problems solved using a specific ILU (ILUTP, inverse-based ILUTC, and inverse-based multilevel ILU) and one of the above orderings (AMF/MMD/RCM/PQ). We show the results for the S-version of the inverse-based multilevel ILU (referred to as ILUPACK) and $\kappa = 10$.

Figures 4.1, 4.2, and 4.3 show clearly that the multilevel ILU is the least sensitive to the given drop tolerance. Using MC64 [10] one could even improve the results further. However, we noticed that for this class of problems only slight improvements

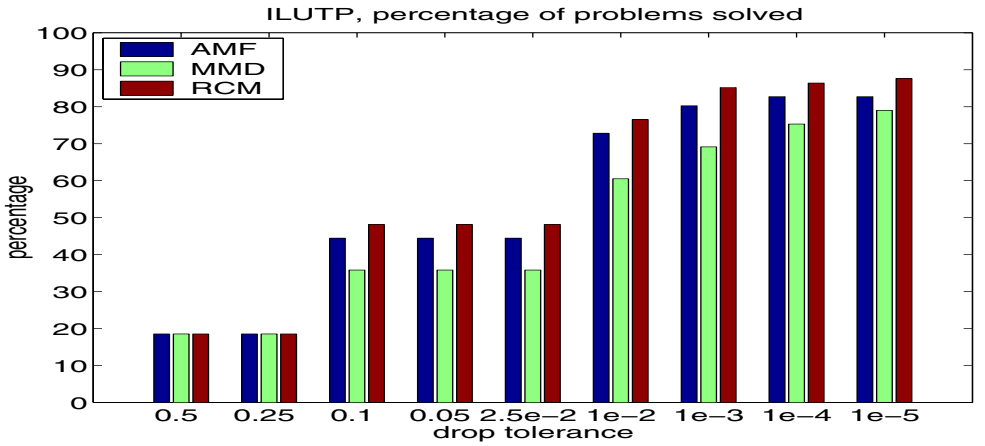


FIG. 4.1. *Numerical sensitivity of ILUTP (PDE problems).*

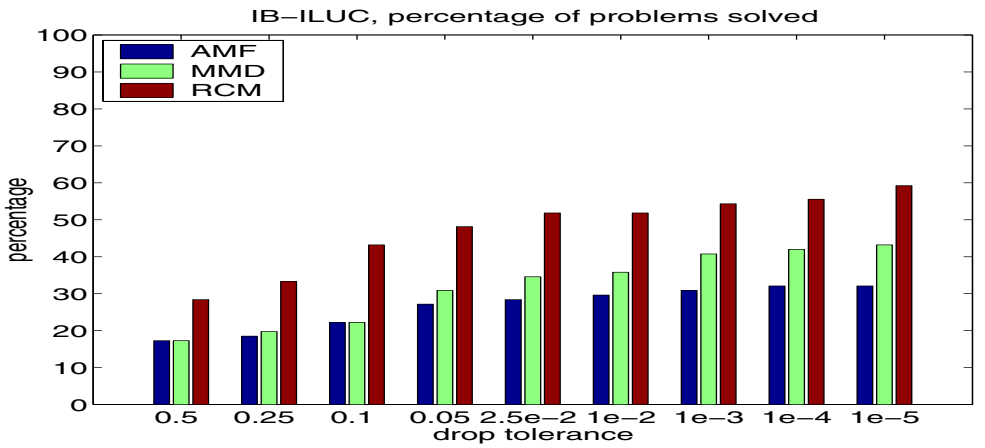


FIG. 4.2. *Numerical sensitivity of inverse-based ILUTC (PDE problems).*

were made. Note that matrices arising from PDEs are typically symmetrically structured and that orderings like AMF, MMD, and RCM, and ILUs such as ILUTC and the inverse-based multilevel ILU preserve symmetry.

4.2. Efficiency via fill-in. The above statistics regarding the dependence of the drop tolerance give only a partial idea of the efficiency of the methods. To get a more detailed view on the fill-in of the triangular factors resulting from the incomplete factorizations, we now study how the number of successfully solved systems is related to the fill-in resulting from the factorization. In a typical situation, an ILU which allows more fill-in is more accurate and should therefore yield faster convergence.

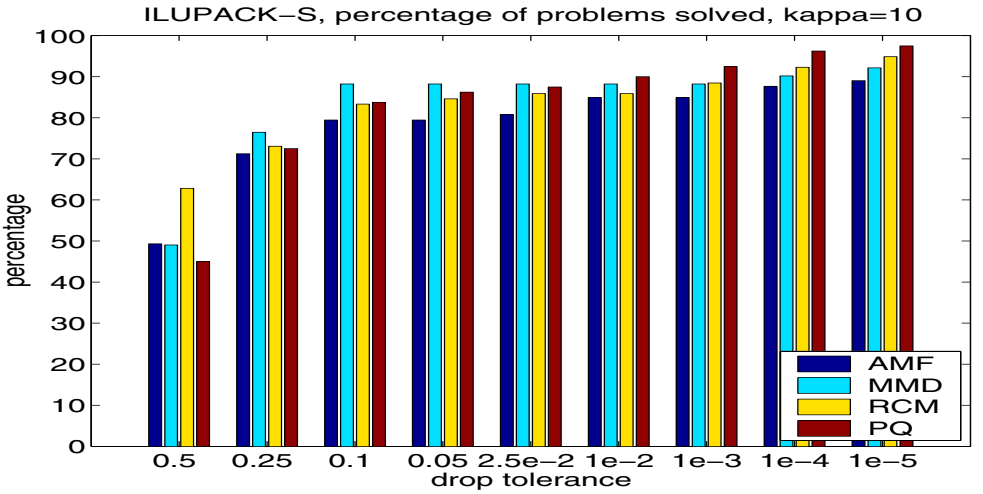


FIG. 4.3. *Numerical sensitivity of ILUPACK, S-version,* $\kappa = 10$ *(PDE problems).*

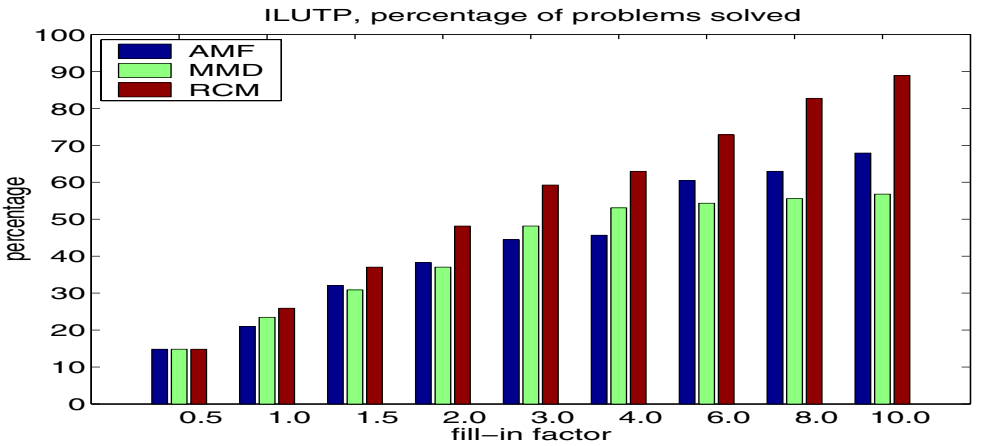


FIG. 4.4. *ILUTP, successful computation versus fill-in (PDE problems).*

The relevant results are in Figures 4.4, 4.5, and 4.6.

It turns out again that the fill-in of the triangular factor of the inverse-based multilevel ILU is the least related to the number of problems that could be solved. Better results can again be obtained by using MC64. Compared with the upcoming section 4.3 on unstructured problems, the improvements by MC64 here are not that impressive and show only a slight improvement.

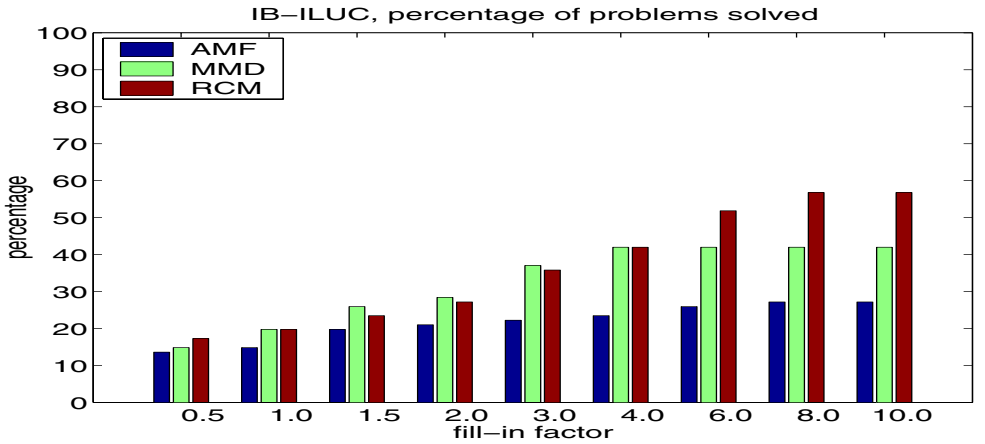


FIG. 4.5. *Inverse-based ILUTC, successful computation versus fill-in (PDE problems).*

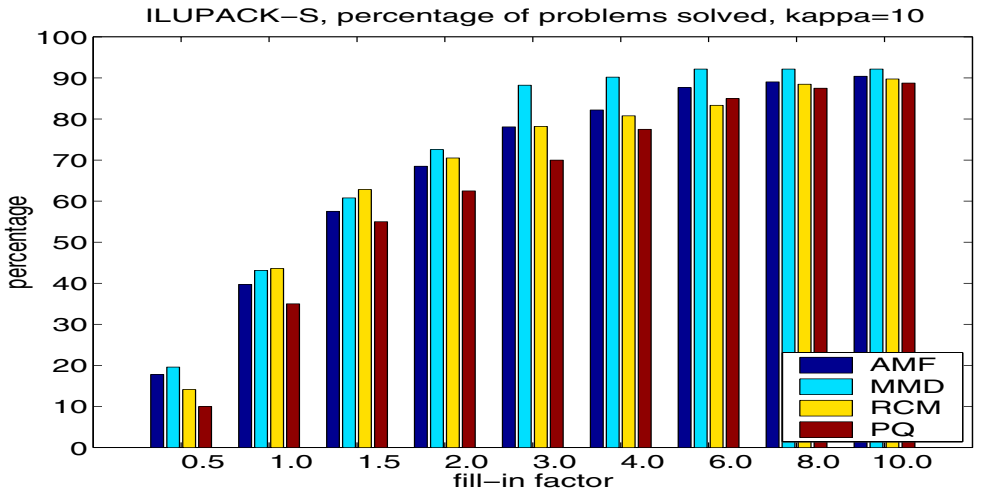


FIG. 4.6. *ILUPACK, S-version,* $\kappa = 10$*, successful computation versus fill-in (PDE problems).*

4.3. Unstructured problems. To give an idea of how the inverse-based multilevel ILU performs on highly unstructured problems we consider a set of 33 test problems from chemical engineering, available at [7]. These matrices are typically highly indefinite and pivoting, or an a priori nonsymmetric reordering, is often helpful. To show this we first present in Figures 4.7, 4.8, and 4.9 the statistics on the sensitivity with respect to the drop tolerance (analogous to the set of PDE problems) when the original matrices are considered.

It is important to mention that ILUPACK does not implement pure versions

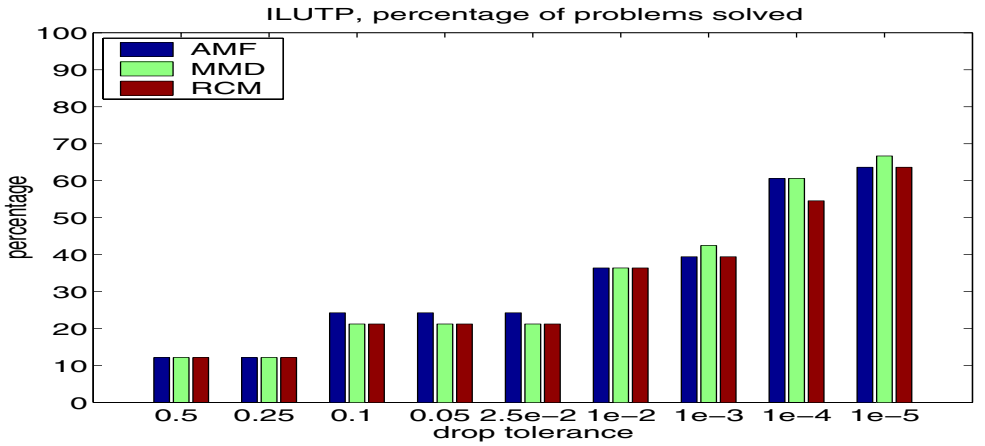


FIG. 4.7. *Numerical sensitivity of ILUTP (chemical engineering).*

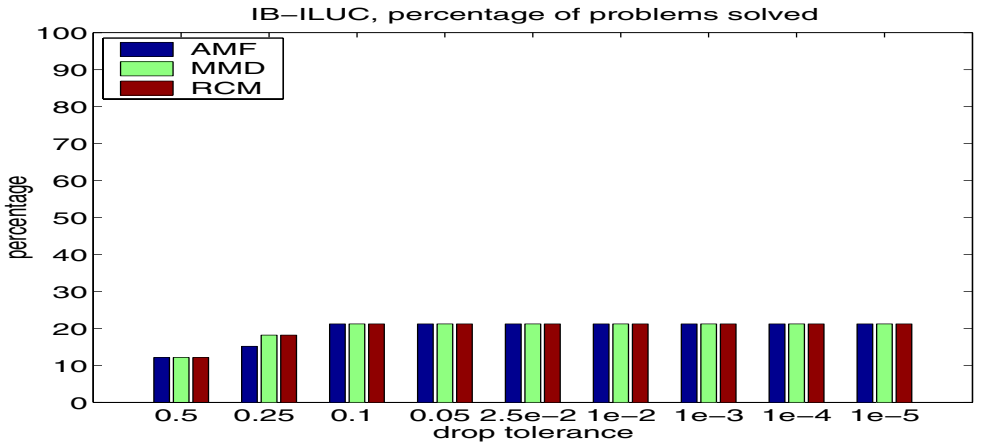


FIG. 4.8. *Numerical sensitivity of inverse-based ILUTC (chemical engineering).*

of the standard symmetric orderings like AMF, MMD, or RCM. Since these do not include any pivoting they are likely to fail on highly unstructured problems. The multilevel ILU in ILUPACK uses only diagonal pivoting, so it was configured so as to automatically switch to the ddPQ ordering when the other simple orderings (AMF, MMD, and RCM) fail to produce a sufficiently large leading diagonal block B. This explains why in Figures 4.7, 4.8, and 4.9 the orderings like AMF, MMD, and RCM perform quite well despite the high indefiniteness of the test problem class.

Due to the high indefiniteness of this problem class, MC64 dramatically improves the numerical performance. On the other hand, relative to ILUTP and ILUTC, the inverse-based multilevel ILU is only mildly affected by this reordering (cf. Figures

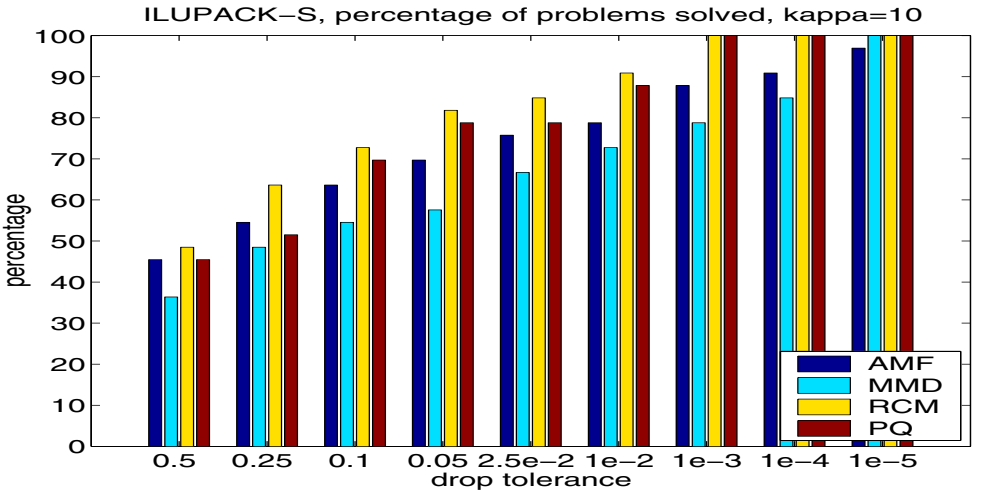


FIG. 4.9. *Numerical sensitivity of ILUPACK, S-version, $\kappa = 10$ (chemical engineering).*

4.10, 4.11, and 4.12).

We should point out again that ILUTC does not include any form of pivoting. This explains why it fails without MC64 for most problems but works very well after MC64 has been applied.

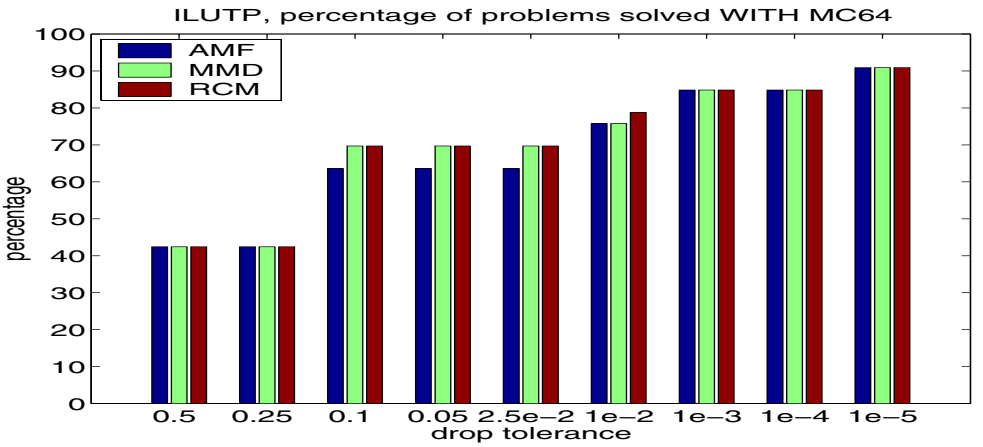


FIG. 4.10. *Numerical sensitivity of $ILUTP + MC64$ (chemical engineering).*

4.4. Variants of the inverse-based multilevel ILU. The previous results compare the new inverse-based multilevel ILU with existing ILU approaches in general terms. In this section we briefly comment on the impact of parameters such as the

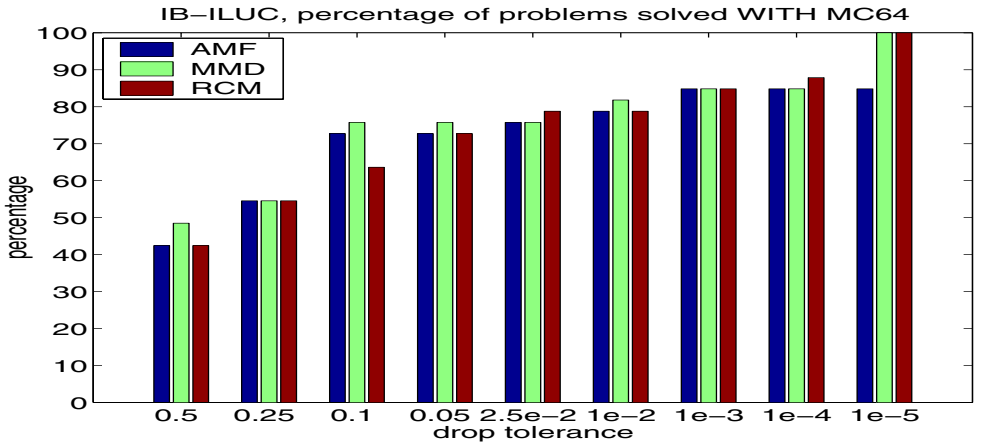


FIG. 4.11. *Numerical sensitivity of inverse-based ILUTC + MC64 (chemical engineering).*

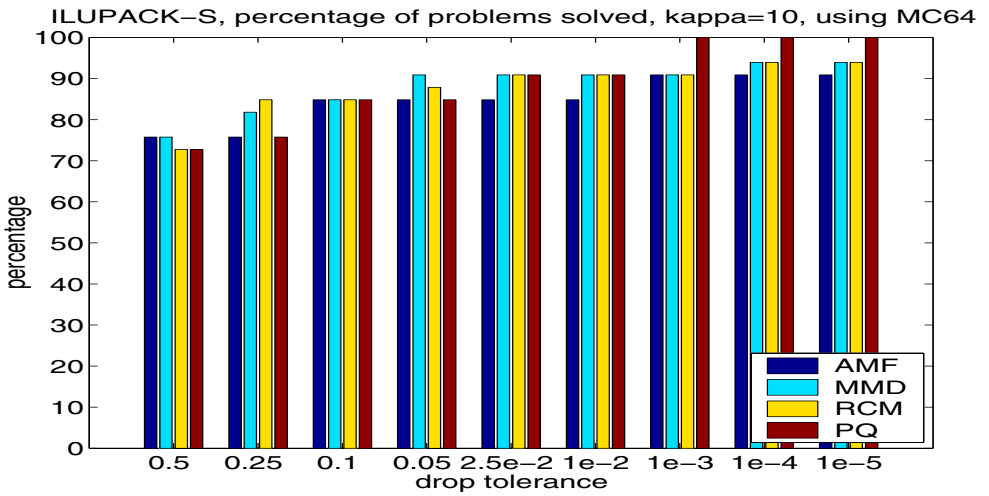


FIG. 4.12. *Numerical sensitivity of ILUPACK + MC64, S-version, $\kappa = 10$ (chemical engineering).*

bound κ for the inverse triangular factors, as well as the choice between the S-version and the M-version.

Recall that the norms of the inverse factors are kept below a threshold κ, which is a parameter of the algorithm. In our numerical tests we found that the influence of the prescribed bound κ on performance is relatively mild. This is caused by the use of the maximal norm of the estimates $\tilde{\kappa}$ for $\|L_k^{-1}\|$ and $\|U_k^{-1}\|$ as a substitute for κ. Clearly κ remains an upper bound, but since it is not clear whether this bound will

ever be reached by the inverse triangular factor, the estimate $\tilde{\kappa}$ seems to be a good compromise. Only $\tilde{\kappa}$ is used for dropping in conjunction with the drop tolerance.

The theory developed in earlier sections suggests that the M-version will be less sensitive to the drop tolerance since the corresponding approximate Schur complement is more accurate. However, this is likely to comes at a higher expense in terms of fill-in. By using a coarser approximation of the Schur complement, the S-version is likely to require smaller drop tolerances. This was confirmed in the numerical examples.

We now show, in a few tables, test examples where fixed parameters are used. For ILUTP and ILUTC a drop tolerance of $\varepsilon = 10^{-3}$ is used. For the inverse-based multilevel ILU, a significantly coarser drop tolerance $\varepsilon = 10^{-1}$ and $\kappa = 10$ are used. We examine some sample problems which arise in the numerical treatment of PDEs. Therefore, we use RCM as a fixed ordering. The results are summarized in Tables 1, 2, 3, and 4. The results show that ILUTP with RCM is fairly robust, but this robustness comes at an exceedingly high memory cost. On the other hand, the multilevel inverse-based ILU from ILUPACK consumes very little memory, though at least the S-version fails to solve two of the problems.

TABLE 1
ILUTP+RCM with $\varepsilon = 10^{-3}$ for selected PDE problems.

Name	$\frac{nnz(L+U)}{nnz(A)}$	GMRES steps	ILU time	GMRES time	Total time
kim1	1.7	7	3.9e+0	1.4e+0	5.30e+0
rma10	4.0	17	3.5e+1	5.4e+0	4.04e+1
garon2	7.9	12	1.1e+1	1.1e+0	1.21e+1
rim	3.2	49	7.6e+0	6.2e+0	1.38e+1
raefsky3	3.5	7	2.7e+1	1.1e+0	2.81e+1
venkat01	4.2	5	1.6e+1	1.5e+0	1.75e+1
wang4	6.1	12	1.4e+0	7.4e-1	2.14e+0
e40r0000	8.4	11	1.9e+1	1.4e+0	2.04e+1
e40r5000	10.2	7	2.7e+1	9.9e-1	2.80e+1

TABLE 2
Inverse-based ILUC+RCM with $\varepsilon = 10^{-3}$ for selected PDE problems.

Name	$\frac{nnz(L+U)}{nnz(A)}$	GMRES steps	ILU time	GMRES time	Total time
kim1	1.8	6	1.4e+1	1.2e+0	1.52e+1
rma10	5.3	6	3.4e+1	2.6e+0	3.66e+1
garon2	8.1	12	7.7e+0	1.3e+0	9.00e+0
rim	5.4	—	1.2e+1	—	—
raefsky3	4.4	6	2.7e+1	1.1e+0	2.81e+1
venkat01	4.6	5	1.6e+1	1.7e+0	1.77e+1
wang4	7.3	10	2.1e+0	7.1e-1	2.81e+0
e40r0000	10.0	6	1.9e+1	9.8e-1	2.00e+1
e40r5000	13.0	—	2.6e+1	—	—

5. Conclusion. While it is clear that iterative solvers are unlikely to ever compete with direct solution methods in terms of robustness and generality, one can

TABLE 3
Inverse-based multilevel ILU+RCM, S-version with $\varepsilon = 10^{-1}$ and $\kappa = 10$ for selected PDE problems.

Name	Levels	$\frac{nnz(L+U)}{nnz(A)}$	GMRES steps	ILU time	GMRES time	Total time
kim1	1	0.8	78	6.6e+0	1.3e+1	1.96e+1
rma10	7	1.7	135	2.9e+1	3.4e+1	6.30e+1
garon2	4	1.6	49	3.3e+0	2.3e+0	5.60e+0
rim	13	1.6	—	1.2e+1	—	—
raefsky3	5	0.5	—	4.8e+0	—	—
venkat01	4	1.4	10	7.6e+0	3.0e+0	1.06e+1
wang4	3	1.6	54	9.7e-1	2.1e+0	3.07e+0
e40r0000	3	1.3	101	3.0e+0	7.6e+0	1.06e+1
e40r5000	5	3.2	28	1.8e+1	2.6e+0	2.06e+1

TABLE 4
Inverse-based multilevel ILU+RCM, M-version with $\varepsilon = 10^{-1}$ and $\kappa = 10$ for selected PDE problems.

Name	Levels	$\frac{nnz(L+U)}{nnz(A)}$	GMRES steps	ILU time	GMRES time	Total time
kim1	1	0.8	78	6.7e+0	1.2e+1	1.87e+1
rma10	6	2.4	31	1.0e+2	1.0e+1	1.10e+2
garon2	3	1.9	27	1.4e+1	1.2e+0	1.52e+1
rim	10	2.2	—	3.1e+1	—	—
raefsky3	3	0.5	81	1.3e+1	7.8e+0	2.08e+1
venkat01	4	1.4	10	2.2e+1	2.9e+0	2.49e+1
wang4	4	1.6	45	2.8e+0	1.6e+0	4.40e+0
e40r0000	3	1.4	37	9.2e+0	2.1e+0	1.13e+1
e40r5000	4	4.4	22	8.4e+1	2.4e+0	8.64e+1

certainly argue that recent progress in the field has considerably shortened the gap between the two classes of methods. The ingredients in this progress came in large part from a better understanding of the impact of dropping on the inverse factors. As long as ILU techniques were only applied to easy problems arising from elliptic-type PDEs, little attention was paid to these inverses because their condition numbers are often within reasonable range. When the success of ILUs grew and their range of applicability expanded, it was realized that the classical ideas that were behind the development of ILU basically failed. Recent work [3] showed that it becomes critical in this situation to analyze the effect of dropping on the inverse factors, and a further analysis of this was given in section 2. In particular, it becomes critical to design the preconditioners in such a way that the perturbations of the inverse factors, caused by dropping, remain small. From a practical implementation viewpoint, this calls for a multilevel strategy which strives to yield bounded inverse factors of the B block corresponding to the "fine" level. Then, recursion can be invoked to deal with the Schur complement resulting from the fine-coarse partitioning.

A package based on three "templates" for implementing the basic ideas has been implemented and thoroughly tested. Numerical experiments indicate that this multilevel ILU can be a good alternative to direct solvers in most cases. They also show

that nonsymmetric reorderings, diagonal pivoting, and efficient Crout implementations can contribute to a better robustness and efficiency of the preconditioner. The code and a large set of experiments are available online; see [4].

Still lacking is a parallel implemention of the algorithms. While the algorithms have not been designed with parallelism in mind, many of the ideas are extensible to parallel environments, at a minimum via a domain decomposition viewpoint.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] M. Benzi, J. C. Haws, and M. Tůma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[3] M. Bollhöfer, *A robust and efficient ILU that incorporates the growth of the inverse triangular factors*, SIAM J. Sci. Comput., 25 (2003), pp. 86–103.

[4] M. Bollhöfer and Y. Saad, *ILUPACK—Preconditioning Software Package*, http://www.math.tu-berlin.de/ilupack/.

[5] M. Bollhöfer and Y. Saad, *On the relations between ILUs and factored approximate inverses*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 219–237.

[6] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson, *An estimate for the condition number of a matrix*, SIAM J. Numer. Anal., 16 (1979), pp. 368–375.

[7] T. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices.

[8] T. A. Davis, *A column pre-ordering strategy for the unsymmetric-pattern multifrontal method*, ACM Trans. Math. Software, 30 (2004), pp. 165–195.

[9] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Clarendon Press, Oxford, UK, 1986.

[10] I. S. Duff and J. Koster, *The design and use of algorithms for permuting large entries to the diagonal of sparse matrices*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 889–901.

[11] A. George and J. W. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[13] A. Gupta, *Improved symbolic and numerical factorization algorithms for unsymmetric sparse matrices*, SIAM J. Matrix Anal. Appl., 24 (2002), pp. 529–552.

[14] N. Li, Y. Saad, and E. Chow, *Crout versions of ILU for general sparse matrices*, SIAM J. Sci. Comput., 25 (2003), pp. 716–728.

[15] National Institute of Standards, *Matrix Market*, http://math.nist.gov/MatrixMarket.

[16] E. Rothberg and S. C. Eisenstat, *Node selection strategies for bottom-up sparse matrix ordering*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 682–695.

[17] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[18] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, PA, 2003.

[19] Y. Saad, *Multilevel ILU with reorderings for diagonal dominance*, SIAM J. Sci. Comput., 27 (2005), pp. 1032–1057.

[20] Y. Saad and B. J. Suchomel, *ARMS: An Algebraic Recursive Multilevel Solver for General Sparse Linear Systems*, Technical Report umsi–1999-107, University of Minnesota at Minneapolis, Dept. of Computer Science and Engineering, 1999.

[21] O. Schenk and K. Gärtner, *Solving unsymmetric sparse systems of linear equations with PARDISO*, J. Future Generation Comput. Syst., 20 (2004), pp. 475–487.

[22] H. D. Simon, *Incomplete LU preconditioners for conjugate gradient type iterative methods*, in Proceedings of the SPE 1985 Reservoir Simulation Symposium, Dallas, TX, Society of Petroleum Engineers of AIME, 1988, paper number 13533, pp. 302–306.

[23] M. Tismenetsky, *A new preconditioning technique for solving large sparse linear systems*, Linear Algebra Appl., 154–156 (1991), pp. 331–353.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1651–1668
© 2006 Society for Industrial and Applied Mathematics

BLOCK PRECONDITIONERS BASED ON APPROXIMATE COMMUTATORS*

HOWARD ELMAN†, VICTORIA E. HOWLE‡, JOHN SHADID§, ROBERT SHUTTLEWORTH¶, AND RAY TUMINARO‡

Abstract. This paper introduces a strategy for automatically generating a block preconditioner for solving the incompressible Navier–Stokes equations. We consider the "pressure convection–diffusion preconditioners" proposed by Kay, Loghin, and Wathen [*SIAM J. Sci. Comput.*, 24 (2002), pp. 237–256] and Silvester, Elman, Kay, and Wathen [*J. Comput. Appl. Math.*, 128 (2001), pp. 261–279]. Numerous theoretical and numerical studies have demonstrated mesh independent convergence on several problems and the overall efficacy of this methodology. A drawback, however, is that it requires the construction of a convection–diffusion operator (denoted F_p) projected onto the discrete pressure space. This means that integration of this idea into a code that models incompressible flow requires a sophisticated understanding of the discretization and other implementation issues, something often held only by the developers of the model. As an alternative, we consider automatic ways of computing F_p based on purely algebraic considerations. The new methods are closely related to the "*BFBt* preconditioner" of Elman [*SIAM J. Sci. Comput.*, 20 (1999), pp. 1299–1316]. We use the fact that the preconditioner is derived from considerations of commutativity between the gradient and convection–diffusion operators, together with methods for computing sparse approximate inverses, to generate the required matrix F_p automatically. We demonstrate that with this strategy the favorable convergence properties of the preconditioning methodology are retained.

Key words. preconditioning, Navier–Stokes, iterative algorithms

AMS subject classifications. Primary, 65F10, 65N30; Secondary, 15A06

DOI. 10.1137/040608817

1. Introduction.

Recently, the development of efficient iterative methods for the fully implicit solution of the Navier–Stokes equations has seen considerable activity. In this paper, we consider a promising class of methods (denoted , , ,

solve linear systems associated with the incompressible Navier–Stokes equations

$$\begin{aligned} \alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \operatorname{grad}) \mathbf{u} + \operatorname{grad} p &= \mathbf{f} \\ -\operatorname{div} \mathbf{u} &= 0 \end{aligned} \quad \text{in } \Omega \subset \mathbb{R}^3, \tag{1.1}$$

where $\mathbf{u}$ are the velocities, p are the pressures, and $\mathbf{u}$ satisfies suitable boundary conditions on $\partial\Omega$. The value $\alpha = 0$ corresponds to the steady-state problem, and $\alpha = 1$ to the transient case. Linearization and stable discretization (discussed in section 2) yield systems of equations of the form

$$\begin{pmatrix} F & G \\ D & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix}, \tag{1.2}$$

where G and D are discrete gradient and (negative) divergence operators, respectively, and F contains all other terms coming from the linearized momentum equation and time derivatives. These systems, which are the focus of this paper, must be solved at each step of a nonlinear (Picard or Newton) iteration or at each time step.

The pressure convection–diffusion preconditioner is based on an approximate block factorization of (1.2). The key to attaining mesh independent convergence lies with the effective approximation of the inverse of the following operator:

$$S = DF^{-1}G. \tag{1.3}$$

S is the Schur complement of (1.2) and is obtained by algebraically eliminating the velocity equations. Most effective block preconditioners for (1.2) must approximate the inverse of the Schur complement in some fashion. The presence of the term F^{-1} between the rectangular operators D and G makes S prohibitively expensive to compute and almost completely dense. The pressure convection–diffusion preconditioner can be viewed as a way to approximate (1.3) by an expression in which the inverse term is moved (or commuted) so that it no longer appears between the two rectangular operators. This approximation takes the form

$$S \approx \hat{S} = DGF_p^{-1}, \tag{1.4}$$

where F_p is a discrete convection–diffusion operator defined on the discrete pressure space. This operator will be discussed in detail in section 2.

This idea requires the explicit construction of the matrix F_p. In all previous work on these schemes, this operator has been constructed using a discretization consistent with that used for the pressure term of the Navier–Stokes equations. For example, if linear finite elements are used to discretize the pressures in (1.1), then F_p is defined using the same basis. Although most codes used to model incompressible flow have the primary kernels needed for such a construction, this places a significant burden on a potential user of this methodology, something not found in standard algebraic preconditioners.

Our aim in this study is to rectify this situation, and to enable the preconditioning methodology derived from (1.4) to be available to users in "black-box" fashion requiring nothing more than a statement of the problem (1.2). This is achieved using the property of commutativity alluded to above. Specifically, we derive an approximate solution to the algebraic problem

$$GF_p \approx FG, \tag{1.5}$$

where F_p must be determined given G and F. This is accomplished by considering a least-squares approximation to (1.5). We show how this leads to a natural interpretation of the related "[illegible] algorithm" [6]. We also illustrate how the above commutator equation can be modified to incorporate diagonal scaling and how proper scaling is critical to obtaining satisfactory convergence rates. In one algebraic variant, sparse approximate inverse ideas are adapted to the above commutator strategy. We present this approach in section 4, where we give a specific algorithm based on the methods of Grote and Huckle [9]. In section 5, numerical results are given to illustrate the competitiveness of the automatically computed pressure convection–diffusion preconditioner with the original version of this approach [11, 16] (which requires the user to provide the matrix F_p) in terms of convergence speed.

2. The pressure convection–diffusion preconditioner. We are concerned with the incompressible form of the Navier–Stokes equations given by (1.1). Our focus is on solution algorithms for the systems of equations that arise after linearization of the system (1.1). We will restrict our attention to a Picard iteration for the nonlinear system, derived by lagging the convection coefficient in the quadratic term $(\mathbf{u} \cdot \text{grad})\,\mathbf{u}$. For the steady-state problem, this lagging procedure starts with some initial guess $\mathbf{u}^{(0)}$ (satisfying the discrete incompressibility constraint) for the velocities and then computes the kth Picard iterate for velocity and pressure by solving the

$$\begin{aligned} -\nu\nabla^2\mathbf{u}^{(k)} + (\mathbf{u}^{(k-1)} \cdot \text{grad})\,\mathbf{u}^{(k)} + \text{grad}\, p^{(k)} &= \mathbf{f}, \\ -\text{div}\,\mathbf{u}^{(k)} &= 0. \end{aligned} \tag{2.1}$$

For transient problems, a strategy of this type can be combined with an implicit time discretization; see [17, 18].

A stable finite difference, finite volume, or finite element discretization of (2.1) leads to a linear system of equations of the form (1.2) which must be solved at each step of the Picard iteration. For the steady problem, the matrix F has block diagonal form in which each individual diagonal block consists of a discretization of a convection–diffusion operator

$$-\nu\nabla^2 + (\mathbf{w} \cdot \text{grad})\,, \tag{2.2}$$

where $\mathbf{w} = \mathbf{u}^{(k-1)}$. For the transient problem, the blocks of F represent discretizations of an operator essentially of the form

$$\frac{1}{\Delta t} I - \nu\nabla^2 + (\mathbf{w} \cdot \text{grad})\,, \tag{2.3}$$

which arises from an implicit time discretization of the time-dependent convection–diffusion equation.

The strategy we employ for solving (1.1) is derived from the block factorization

$$\begin{pmatrix} F & G \\ D & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ DF^{-1} & I \end{pmatrix} \begin{pmatrix} F & G \\ 0 & -S \end{pmatrix}, \tag{2.4}$$

where $S = DF^{-1}G$ is the Schur complement. This implies that

$$\begin{pmatrix} F & G \\ D & 0 \end{pmatrix} \begin{pmatrix} F & G \\ 0 & -S \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ DF^{-1} & I \end{pmatrix}, \tag{2.5}$$

which, in turn, suggests a preconditioning strategy for (1.1). If it were possible to use the matrix

$$\mathcal{Q} = \begin{pmatrix} F & G \\ 0 & -S \end{pmatrix} \tag{2.6}$$

as a right-oriented preconditioner, then the preconditioned operator would be the one given in (2.5). All the eigenvalues have the value 1, and it can be shown that this operator contains Jordan blocks of dimension at most 2, and consequently that at most two iterations of a preconditioned GMRES iteration would be needed to solve the system [12].

The key bottleneck in applying $\mathcal{Q}^{-1}$ is the action of the inverse Schur complement operator $(DF^{-1}G)^{-1}$. In practice, this must be replaced with an inexpensive approximation. If it were possible to find a matrix F_p that makes (1.5) an equality, then this would imply that

$$F^{-1}G = GF_p^{-1}, \tag{2.7}$$

which in turn would mean $S = \tilde{S}$ in (1.4). In general, (1.5) is an overdetermined system (with range(G) a proper subset of range(F)), so it is not possible to find such an F_p. Consequently, we have an approximation in (1.4). Using $\tilde{S} = (DG)F_p^{-1}$ within a preconditioning operation entails applying the action of the inverse, given by

$$\tilde{S}^{-1} = F_p(DG)^{-1}. \tag{2.8}$$

Note that DG is a scaled discrete Laplacian operator, so that working with (2.8) will require a Poisson solve followed by a matrix-vector product with F_p. For purposes of efficiency, the action of $(DG)^{-1}$ is often approximated by an inexact Poisson solver without any significant degradation in convergence. Thus, application of (2.8) is inexpensive.

A good approximation can be expected in (1.5) because of the nature of the differential operators associated with the matrices. In particular, suppose the convection–diffusion operator

$$(-\nu\nabla^2 + (\mathbf{w}\cdot\operatorname{grad}))_p \tag{2.9}$$

can be defined on the pressure space (where the subscript p indicates that the operator is acting on the pressure space). Consider the commutator with the gradient operator, as follows:

$$(-\nu\nabla^2 + (\mathbf{w}\cdot\operatorname{grad}))\operatorname{grad} - \operatorname{grad}(-\nu\nabla^2 + (\mathbf{w}\cdot\operatorname{grad}))_p\,. \tag{2.10}$$

If $\mathbf{w}$ is constant, then this expression would be zero on the interior of Ω, and it is also reasonable to expect it to be small for smooth $\mathbf{w}$. This suggests that if F_p is a discretization of (2.9), then the discrete commutator

$$FG - GF_p \tag{2.11}$$

will also be small. A similar statement would apply if time derivatives are present: as in (2.3), discretization in time would add terms essentially of the form $\frac{1}{\Delta t}I$ to both convection–diffusion operators in (2.10).

In [11, 16], F_p is obtained from discretization of the convection–diffusion operator of (2.9). In this paper, we develop an automated construction of F_p designed explicitly

to make the commutator (1.5) small. This removes the process of discretization from the design of the iterative solution algorithm and enables the preconditioning to adapt automatically to different discretizations without any intervention on the part of the designer of a Navier–Stokes code. One of the ideas for approximating (1.5) is to adapt methods for computing sparse approximate inverses to define sparse approximate commutators. This will be considered in section 4.

Before exploring sparse approximations, it is worthwhile to consider a classical normal equations solution to (1.5). In particular, suppose f_j, the jth column of F_p, is computed to solve the least-squares problem with respect to the Euclidean norm

$$\min \|Gf_j - [FG]_j\|_2 ,$$

where $[FG]_j$ represents the jth column of the matrix FG. This leads to the normal equations

$$(G^T G) f_j = G^T [FG]_j .$$

An equivalent formulation is that F_p minimizes the Frobenius norm of the error in the complete system

$$\min \|GF_p - FG\|_F . \tag{2.12}$$

Thus, the solution is

$$F_p = (G^T G)^{-1} G^T FG. \tag{2.13}$$

When $D = G^T$ (as is normally the case), the resulting Schur complement preconditioner becomes

$$(G^T G)^{-1} G^T FG (G^T G)^{-1}. \tag{2.14}$$

This corresponds exactly to the [illegible] preconditioner proposed in [6] and also in [15]. This method can be considered as an alternative to the pressure convection–diffusion preconditioner; it is of particular interest for Newton's method, for which (in contrast to Picard iteration) it is difficult to derive a discretization of (2.9) that will make (2.11) small. We should also point out that the original derivation of the [illegible] preconditioner is considerably different from the viewpoint given here. The notion of the normal equations solution to (1.5) is in our opinion more intuitive than the description given in [6] and provides additional insight into this preconditioner. From this perspective, however, the approximate solution of the commutator equation is certainly not a new idea, as it has already been done with the [illegible] preconditioner. In effect, we are proposing alternate solutions and modifications to (1.5) that may have better properties. These modifications will include the use of diagonal scaling in (2.12) as well as boundary condition adjustments. It will be shown how these simple improvements can significantly accelerate convergence. Note also that use of (2.14) will entail two Poisson solves in each invocation of the preconditioner; it will become apparent that sparse approximate inverses can be used to eliminate one of these two solves.

3. Diagonally scaled commutators. The construction of an algebraic commutator need not explicitly rely on the PDE analogue used to derive the pressure

convection–diffusion preconditioning. This implies that alternative relationships derived from commutators can be explored when developing Schur complement approximations. Consider, for example,

$$M_2^{-1} G F_p \approx M_2^{-1} F M_1^{-1} G, \tag{3.1}$$

where M_2 and M_1 are diagonal matrices. The operator M_2^{-1} can be thought of as a weight matrix that transforms (2.12) into a weighted least-squares problem. The introduction of M_1^{-1} can be viewed as a way to precondition F and G so that they are more amenable to commuting. After premultiplying (3.1) by $DF^{-1}M_2$, it is easy to show that the inverse Schur complement is now approximated by

$$(DF^{-1}G)^{-1} \approx F_p (DM_1^{-1}G)^{-1}, \tag{3.2}$$

where

$$F_p = (G^T M_2^{-2} G)^{-1} G^T M_2^{-2} F M_1^{-1} G \tag{3.3}$$

when the normal equations are used to produce an approximate solution to (3.1). The net effect of "preconditioning" the commutator equation in this way is that a new definition of F_p is given and the inverse discrete Poisson operator in (2.8) is now replaced by a discrete variable-coefficient diffusion operator (when M_1 is a diagonal matrix), which can still be effectively handled with solution methods such as multigrid. In this paper, we will primarily consider choices for M_1 and M_2 that are based on the diagonal of the velocity mass matrix. However, before exploring this, it is worthwhile to consider another choice for M_1 and M_2 that highlights connections between F_p solvers and the well-known SIMPLE method [13].

3.1. Diagonal scaling and SIMPLE. One interesting aspect of the diagonally scaled commutator equation is that it bridges the gap between pressure convection–diffusion solvers and traditional pressure-correction methods such as SIMPLE [13]. In particular, the original SIMPLE method is a numerical time marching procedure for incompressible flow in which the inverse Schur complement is approximated by $(D\,\mathrm{diag}(F)^{-1}G)^{-1}$, where $\mathrm{diag}(F)$ is the diagonal matrix whose entries are taken from the main diagonal of F. When $M_1 = \mathrm{diag}(F)$ and F_p is the identity matrix, the right side of (3.2) is identical to SIMPLE's approximation of the inverse Schur complement. In general, F_p will be close to the identity when $F\,\mathrm{diag}(F)^{-1}$ is close to the identity. This, in turn, occurs when the entries on the diagonal of F are much larger than the off-diagonal entries, as is often the case for time-dependent problems with small time steps. Thus, we can conclude that the Schur complement approximations for the diagonally scaled commutator and SIMPLE are basically the same in the small time-step scenario.

The true advantage of the F_p solvers occurs for larger time steps or steady-state problems. Let A denote the coefficient matrix of (1.2). Consider the new block preconditioner based on the factorization

$$\tilde{A} \equiv \begin{pmatrix} F & 0 \\ D & -D\,\mathrm{diag}(F)^{-1}G \end{pmatrix} \begin{pmatrix} I & \mathrm{diag}(F)^{-1}G \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & F_p^{-1} \end{pmatrix}. \tag{3.4}$$

This new preconditioner is similar to standard F_p methods with the exception that the underlying block decomposition is built on SIMPLE's approximate block factorization

as opposed to (2.4). It is well known that when F_p is the identity, the LU decomposition given by (3.4) corresponds to SIMPLE's underlying matrix factorization [14]. By multiplying out matrices, it is easy to show that

$$\tilde{A} = \begin{pmatrix} F & F\mathrm{diag}(F)^{-1}GF_p^{-1} \\ D & 0 \end{pmatrix}$$

and that the difference between $\tilde{A}$ and the discrete incompressible Navier–Stokes operator, A, of (1.2) is given by

$$A - \tilde{A} = \begin{pmatrix} 0 & (GF_p - F\mathrm{diag}(F)^{-1}G)F_p^{-1} \\ 0 & 0 \end{pmatrix}. \tag{3.5}$$

It follows once again that the SIMPLE method (where F_p is the identity) will perform well when $G - F\ \mathrm{diag}(F)^{-1}G$ is small, i.e., when F has relatively large diagonal entries. On the other hand, comparison of the structure of the nonzero block in the $(1,2)$ entry of (3.5) with (3.1), the "preconditioned commutator relation" using $M_1 = \mathrm{diag}(F)$ and $M_2 = I$, indicates that the error matrix $A - \tilde{A}$ will be small if the commutator is small, irrespective of the size of the diagonal of F.

3.2. Diagonal scaling and the mass matrix. Replacement of a differential commuting relationship with an algebraic commuting relationship assumes certain additional properties of the discrete operator. If, for example, a finite difference approach such as the marker-and-cell method [10] is used to discretize the Navier–Stokes equations, then the relation (2.11) looks very much like its continuous analogue (2.10), and heuristic justification for the validity of (2.11) comes from Taylor series arguments. On the other hand, for mixed finite element discretizations with different order elements, such as the stable Q_2–Q_1 (biquadratic velocity, bilinear pressure) element, the discrete commutator (2.11) will typically not be small. This does not present a problem when the required matrix F_p is obtained, as in [11, 16], by constructing a discrete convection–diffusion operator using the given pressure discretization (bilinear elements in the example noted above). However, it complicates the construction of F_p when a purely algebraic approach is used.

To gain insight into this matter, consider the differential operators

$$\mathcal{L}_1 u \equiv -u_{xx} \qquad \text{and} \qquad \mathcal{L}_2 u \equiv u_x$$

on the one-dimensional domain $0 \le x \le 1$, with periodic boundary conditions. Clearly, the differential operators formally commute. That is, $\mathcal{L}_1\mathcal{L}_2$ is equivalent to $\mathcal{L}_2\mathcal{L}_1$. Now let L_1 and L_2 be the discrete approximations to $\mathcal{L}_1$ and $\mathcal{L}_2$ obtained from piecewise linear nodal finite elements

$$\phi_i(x) = \begin{cases} \alpha_i \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x_{i-1} \le x \le x_i, \\ \alpha_i \frac{x - x_{i+1}}{x_i - x_{i+1}}, & x_i \quad \le x \le x_{i+1}, \\ 0 & \text{otherwise} \end{cases}$$

and a Galerkin discretization. Here x_i denotes the location of the ith mesh point and α_i effectively scales each element in an arbitrary way. Standard nodal elements have $\alpha_i = 1$ for all i, and it can be shown that on uniform meshes

$$L_1^s L_2^s = L_2^s L_1^s,$$

where the superscript s indicates standard (or unscaled) nodal element discretization. The use of arbitrary scaling, however, effectively scales the corresponding discrete operators:

$$L_1 = D_\alpha L_1^s D_\alpha \qquad \text{and} \qquad L_2 = D_\alpha L_2^s D_\alpha,$$

where L_1 and L_2 denote discretizations generated with arbitrary scaling and D_α is the diagonal matrix with α_i in the (i,i)th entry. Although L_1 and L_2 do not commute for arbitrary scaling, we do have

$$L_1 D_\alpha^{-2} L_2 = L_2 D_\alpha^{-2} L_1, \tag{3.6}$$

that is, an appropriate scaling enables commutativity.

The example above is artificial, but an analogous strategy can be used to enhance performance in the setting we are concerned with. In particular, it is possible to choose a diagonal matrix M_1 so that a version of the commutator is small on some components of the pressure space. For the Q_2–Q_1 discretization, if M_1 is chosen to be $\mathrm{diag}(M)$, the diagonal matrix whose entries are those on the diagonal of the velocity mass matrix M, then $M_1^{-1}Gq$ is constant on grid points at the interior of Ω whenever the nodal vector q comes from a linear function. Moreover, for such q and this choice of M_1, if F_p is a discrete bilinear approximation to (2.9) with constant $\mathbf{w}$, then

$$(GF_p - FM_1^{-1}G)q = 0 \quad \text{in the interior of } \Omega.$$

Thus, we use this strategy for choosing M_1. Note that in the example leading to (3.6), the diagonal of the mass matrix is simply D_α^2, and thus this choice of M_1 is consistent with the motivating example.

We will derive an automated version of the pressure convection–diffusion preconditioner in the following section. We conclude here by presenting a variant of the [illegible] method based on this approach. Insertion of the normal equations solution to (3.1) into (3.2) yields the following approximation to the inverse Schur complement (assuming $D = G^T$):

$$(DF^{-1}G)^{-1} \approx (G^T M_2^{-2} G)^{-1} (G^T M_2^{-2} F M_1^{-1} G)\, (G^T M_1^{-1} G)^{-1}.$$

As noted above, M_1 is the diagonal of the velocity mass matrix. We take $M_2 = sqrt(M_1)$ to make the two variable-coefficient Poisson operators identical. The resulting preconditioner, with these choices for M_1 and M_2, will be referred to as the [illegible] method. It can also be shown that the matrix F_p obtained from the normal equations for (3.1) corresponds to a convection–diffusion operator in (2.9) for which the L_2-norm of the commutator (2.10), viewed as an operator on the (finite element) pressure space, is approximately minimized; see [8, Chapter 8] for details. As will be seen in section 5, the introduction of this diagonal scaling can have a dramatic impact on the overall convergence of the method.

4. Sparse approximate commutator (*SPAC*). One disadvantage of both the [illegible] and scaled [illegible] methods is that they require two Poisson-like solves during each invocation, as opposed to the single solve needed by the pressure convection–diffusion preconditioners. This implies that the application of the approximate inverse Schur complement is more costly for [illegible] methods than for pressure convection–diffusion methods.

To alleviate the burden of the additional solve required by [illegible] methods, we consider approximate solutions to (1.5) built on concepts from sparse approximate

inverses. The resulting method will still require only one invocation of the Poisson solver per iteration, although there is an additional setup cost associated with building the sparse approximate inverse. Since sparse approximate inverses are typically used to approximate the matrix inverse of a square matrix, an adaptation that is suitable for the rectangular matrix associated with the approximate commutator equation is considered. We begin by discussing standard sparse approximate inverses.

Sparse approximate inverses were originally developed as preconditioning operators for sparse matrices; they were designed to be easier to compute and use on parallel architectures than other preconditioners such as incomplete LU factorizations. Given a large, sparse matrix A, these methods produce a sparse matrix Q that approximates A^{-1}. The sparse approximate inverse is inherently parallel and intended to retain the convergence properties of an incomplete LU factorization. In addition, sparse approximate inverses are important for situations in which incomplete LU factorizations have difficulties, such as when the factorization encounters a zero pivot or produces unstable factors [5]. Let A represent a generic matrix of order n, and let I denote the identity matrix of order n. Typically, sparse approximate inverse techniques are based on producing a sparse matrix Q that minimizes the Frobenius norm of the residual matrix

$$\min_{Q} \|AQ - I\|_F, \tag{4.1}$$

which was first suggested by Benson and Frederickson [1, 2]. The residual matrix minimization reduces to n independent least-squares problems

$$\min_{q_j} \|Aq_j - e_j\|_2, \quad j = 1, \ldots, n, \tag{4.2}$$

where q_j is the jth column of Q and e_j is the jth unit vector, i.e., the jth column of I.

Note that our task is less difficult than that required for construction of sparse approximate inverses. The key obstacle in the latter setting is picking the proper sparsity pattern of Q to capture enough of A^{-1}, which is usually dense even when A is sparse. In our case, it is reasonable to expect that the matrix F_p needed in the discrete commutator equation is sparse (as we know this to be the case in standard pressure convection–diffusion preconditioning). Thus we have the simpler task of computing a sparse approximation to a sparse matrix.

There are several strategies for computing sparse approximate inverses (see, e.g., [3, section 5]). Many sparse approximate inverse methods (e.g., Chow [4]) require iterated applications of matrix-vector products of the form $Ar, A(Ar), \ldots$. The SPAI algorithm of Grote and Huckle [9] avoids this construction and is natural to apply in the present setting where a commutator is required for a rectangular matrix. We call this adaptation , (for sparse approximate commutator). Before discussing the adaptation of SPAI to commutators, it is important to note that approximate inverses based on square matrices can also be used by instead considering the normal equations

$$(G^T M_2^{-2} G) F_p = G^T M_2^{-2} F M_1^{-1} G.$$

In this case, one replaces (4.1) with

$$\min_{F_p} \|G^T M_2^{-2} G F_p - G^T M_2^{-2} F M_1^{-1} G\|_F^2,$$

which involves square matrices. We do not pursue this further but instead consider ideas for computing sparse matrices that directly address the commutator relationship.

[illegible] is based on replacing (4.1) with

$$\min_{F_p \text{ sparse}} \|\tilde{G}F_p - \tilde{F}\tilde{G}\|_F^2, \tag{4.3}$$

where $\tilde{G} = M_2^{-1}G$ and $\tilde{F} = M_2^{-1}FM_1^{-1}M_2$. This reduces to a set of n constrained least-squares problems as follows:

$$\min_{f_j \text{ sparse}} \|\tilde{G}f_j - b_j\|_2^2, \tag{4.4}$$

where b_j is the jth column of $\tilde{F}\tilde{G}$ and f_j is the jth column of F_p.

Our [illegible] algorithm follows the technique of Grote and Huckle [9]. We will use the following notation: given index sets $\mathcal{I}$, $\mathcal{J}$, let $\tilde{G}(\mathcal{I}, \mathcal{J})$ denote the submatrix of $\tilde{G}$ with entries $\tilde{g}_{ij}$, where $i \in \mathcal{I}$, $j \in \mathcal{J}$; by analogy with MATLAB, a colon (:) represents the complete index set $\{1, \ldots, n\}$. Similarly, for a vector v, $v(\mathcal{I})$ or $v(\mathcal{J})$ denotes the subvector of v with entries v_i or v_j, where $i \in \mathcal{I}$ and $j \in \mathcal{J}$. As in [9], we describe the algorithm for one column of F_p. We begin by getting an initial sparsity pattern. Let $\mathcal{B}$ be the set of indices of the nonzero rows in b_j, let $\mathcal{J}$ be the set of indices of nonzero columns of $\tilde{G}(\mathcal{B}, :)$, and let $\mathcal{I}$ be the set of indices of the nonzero rows of $\tilde{G}(:, \mathcal{J})$. The algorithm starts by solving the $n_1 \times n_2$ least-squares problem $\tilde{G}(\mathcal{I}, \mathcal{J})f_j(\mathcal{J}) = b_j(\mathcal{I})$, where $n_1 = |\mathcal{I}|$ and $n_2 = |\mathcal{J}|$. The vector f_j obtained by inserting $f_j(\mathcal{J})$ into the zero vector is the initial approximation to the jth column of F_p. We improve F_p by augmenting the sparsity structure to obtain a more effective commutator.

As in [9], we accomplish this by reducing the current error $\|\tilde{G}F_p - \tilde{F}\tilde{G}\|_F^2$, i.e., reducing $\|\tilde{G}f_j - b_j\|_2^2$ for each column $j = 1, \ldots, n$ of F_p. Let the residual $r = \tilde{G}f_j - b_j$. If $r = 0$, then f_j is exactly the jth column of the commutator and no improvement is possible. If $r \neq 0$, we augment the sets of indices $\mathcal{I}$ and $\mathcal{J}$ to reduce $\|r\|_2$. Let $\mathcal{R}$ be the set of indices of the nonzero elements of r, and let $\tilde{\mathcal{J}}$ be the set of new column indices that appear in $\mathcal{R}$ but not in $\mathcal{J}$. To determine the most profitable reduction in $\|r\|_2$, for each $k \in \tilde{\mathcal{J}}$ we solve the one-dimensional minimization problem

$$\min_{\mu_k} \|r + \mu_k \tilde{G}(:, \tilde{\mathcal{J}}(k))\|_2, \tag{4.5}$$

whose solution is given by

$$\mu_k = -\frac{(r^T \tilde{G}(:, \tilde{\mathcal{J}}(k)))^2}{\|\tilde{G}(:, \tilde{\mathcal{J}}(k))\|_2^2}, \tag{4.6}$$

where $\tilde{\mathcal{J}}(k)$ is the kth element of the index set $\tilde{\mathcal{J}}$. For each k, we compute $\rho_k = \|r + \mu_k \tilde{G}(:, \tilde{\mathcal{J}}(k))\|_2^2$, the square of the Euclidean norm of the new residual, using

$$\rho_k = \|r\|_2^2 - \frac{(r^T \tilde{G}(:, \tilde{\mathcal{J}}(k)))^2}{\|\tilde{G}(:, \tilde{\mathcal{J}}(k))\|_2^2}. \tag{4.7}$$

We reduce $\tilde{\mathcal{J}}$ to the set of indices with the lowest ρ_k. In particular, we delete from $\tilde{\mathcal{J}}$ the indices whose ρ values are greater than the median of the ρ_k. We improve $\mathcal{J}$

by augmenting it with $\tilde{\mathcal{J}}$, $\mathcal{J} = \mathcal{J} \cup \tilde{\mathcal{J}}$. $\mathcal{I}$ is updated to contain the indices of the nonzero rows of $\tilde{G}(:, \mathcal{J})$. We then repeat the solution of the least-squares problem $\tilde{G}(\mathcal{I}, \mathcal{J}) f_j(\mathcal{J}) = b_j(\mathcal{I})$, updating the elements of f_j indexed by $\mathcal{J}$ to produce an improved column of F_p. To reduce fill, we drop entries of f_j that are less than a given tolerance.

We stop improving $\mathcal{J}$ and the current column of the commutator F_p when we have achieved sufficient reduction in the error $\|\tilde{G}F_p - \tilde{F}\tilde{G}\|_F^2$. The entire algorithm is given below.

ALGORITHM 1.

- [illegible] b_j [illegible] $\tilde{F}\tilde{G}$
- [illegible] [illegible] $\mathcal{I}, \mathcal{J}$ [illegible] $\tilde{G}$
  - [illegible] $\mathcal{B} =$ [illegible] b_j
  - [illegible] $\mathcal{J} =$ [illegible] $\tilde{G}(\mathcal{B}, :)$
  - [illegible] $\mathcal{I} =$ [illegible] $\tilde{G}(:, \mathcal{J})$
- [illegible] $r_{norm} = 1$
- [illegible] $reduction = 1$
- [illegible] $reduction / r_{norm} > tol_1$ & $r_{norm} > tol_2$
  - [illegible] $\mathcal{J}_{prev} = \mathcal{J}$
  - [illegible] $\min \|\tilde{G}(\mathcal{I}, \mathcal{J}) f_j(\mathcal{J}) - b_j(\mathcal{I})\|_2^2$ [illegible] $f_j(\mathcal{J})$ [illegible] f_j [illegible] $\mathcal{J}$
  - [illegible] $r = \tilde{G} f_j - b_j$
  - [illegible] $r_{norm} = \|r\|_2$
  - [illegible] $\mathcal{R} =$ [illegible] r
  - [illegible] $\tilde{\mathcal{J}} =$ [illegible] $\tilde{G}$ [illegible] $\mathcal{R}$ [illegible] $\mathcal{J}$
  - [illegible] $k = 1, \ldots, |\tilde{\mathcal{J}}|$
    - [illegible] $\mu_k = (r^T \tilde{G}(:, \tilde{\mathcal{J}}(k)))^2 / (\tilde{G}(:, \tilde{\mathcal{J}}(k))^T \tilde{G}(:, \tilde{\mathcal{J}}(k)))$
    - [illegible] $\rho_k = \|r\|_2^2 - \mu_k$
  - [illegible] $reduction = \max_k \mu_k$
  - [illegible] $\mathcal{J}$ [illegible] $\tilde{\mathcal{J}}$ [illegible] ρ [illegible] $\tilde{\mathcal{J}}$ [illegible] ρ [illegible] ρ_k [illegible] $\mathcal{J}$ $\mathcal{J} \cup \tilde{\mathcal{J}}$
  - [illegible] $\mathcal{I}$ $\mathcal{I} =$ [illegible] $\tilde{G}(:, \mathcal{J})$
- [illegible] $f_j < drop_tol$ [illegible] $ind =$ [illegible]
- [illegible] $F_p(\mathcal{J}_{prev}(ind), j) = f_j(ind)$

The resulting [illegible] algorithm is almost identical to the method presented by Grote and Huckle. Therefore, its cost should be similar to this sparse approximate inverse algorithm. It is also important to keep in mind that the dimension of the Schur complement system is only a small fraction of the entire system (often about one quarter for three-dimensional problems) and that fairly sparse approximations can be used (as F_p in pressure convection–diffusion preconditioning is sparse). Thus, it is natural to expect that the cost to construct the [illegible] operator (relative to the cost of solving the entire saddle point system) is considerably less than the setup costs associated with standard sparse approximate inverses.

5. Results. Numerical experiments are presented for two benchmark problems in two dimensions: models of flow over a backward facing step and in a lid driven cavity application. The linear problems were generated by applying a Picard iteration to

the nonlinear system and stopping this iteration when the nonlinear residual satisfied

$$\left\| \begin{pmatrix} \mathbf{f} - \left(\mathbf{F}(\mathbf{u}^{(m)})\mathbf{u}^{(m)} + Gp^{(m)}\right) \\ g - D\mathbf{u}^{(m)} \end{pmatrix} \right\|_2 \leq 10^{-5} \left\| \begin{pmatrix} \mathbf{f} \\ g \end{pmatrix} \right\|_2, \tag{5.1}$$

where $\mathbf{F}(\mathbf{u})\mathbf{u}$ is the discretization of the nonlinear convection–diffusion operator, $-\nu\nabla^2\mathbf{u} + (\mathbf{u} \cdot \text{grad})\mathbf{u}$, and g comes from boundary data. At each Picard step m, the Oseen equations (1.2) are solved for a correction to the current iterate; the right-hand side of the linear system is the nonlinear residual given on the left of (5.1). The numbers reported correspond to the total number of GMRES iterations (without restarting) required to solve the saddle point system associated with the last Picard iteration, using various preconditioners. The linear iteration was stopped when the residual of the linear system (1.2) satisfied

$$\|\mathbf{r}^{(k)}\|_2 \leq 10^{-6}\|\mathbf{f}^{(m)}\|_2 \tag{5.2}$$

starting with a zero initial iterate, where $\mathbf{f}^{(m)}$ is the current nonlinear residual vector and $\mathbf{r}^{(k)}$ is the residual at the kth linear iteration. The performance of the iterative solvers was not sensitive to the particular Picard step, and these results are representative.

All the preconditioners are of the form given by (2.6) where an exact solver is used for all needed Poisson solves and convection–diffusion solves. The inverse Schur complement is the only aspect that is varied; preconditioners tested include an application-provided pressure convection–diffusion preconditioner (i.e., the method of [11, 16], denoted "F_p" in the tables), [illegible], scaled [illegible] (denoted [illegible]), [illegible] F_p without diagonal scaling, and [illegible] $-M\ F_p$ ([illegible] using the same diagonal scaling as scaled [illegible] to define $\tilde{F}$ and $\tilde{G}$). For the application-provided pressure convection–diffusion preconditioner, the matrix F_p is obtained from the convection–diffusion operator that determines F, discretized on the pressure space. Two discretizations were considered: the Q_2–Q_1 (biquadratic velocity/bilinear pressure) mixed finite element discretization [8] and the marker-and-cell (MAC) finite difference discretization [10]. The experiments were done in MATLAB. The Q_2–Q_1 discretization and all tests of iterative methods were performed using the Incompressible Flow Iterative Solution Software package IFISS [7] developed in conjunction with [8].

First, we compare methods on the backward facing step. In this example, a Poiseuille flow profile (steady horizontal flow in a channel driven by pressure difference between the two ends) is imposed on the inflow boundary ($x = -1; 0 \leq y \leq 1$), and a no-flow (zero velocity) condition is imposed on the walls. At the outflow boundary ($x = 5; -1 < y < 1$), the Neumann condition

$$\begin{aligned} \nu\tfrac{\partial u_x}{\partial x} - p &= 0, \\ \tfrac{\partial u_y}{\partial x} &= 0 \end{aligned} \tag{5.3}$$

is satisfied and automatically sets the mean outflow pressure to zero [8]. (Here the velocity vector is $\mathbf{u} = (u_x, u_y)^T$.) The Q_2–Q_1 discretization is performed on a uniform finite element grid, where the grid is set up so that the rectangle enclosing the step would contain a uniform $n \times 3n$ grid of velocity nodes of width $h = 2/n$. Figure 5.1 shows the velocity field and pressure field for a representative solution to a two-dimensional backward facing step problem with $n = 64$.

In Table 5.1 and Table 5.2, the number of GMRES iterations needed to satisfy the stopping criterion (5.2) are shown for a variety of Reynolds numbers $Re = 2/\nu$, where

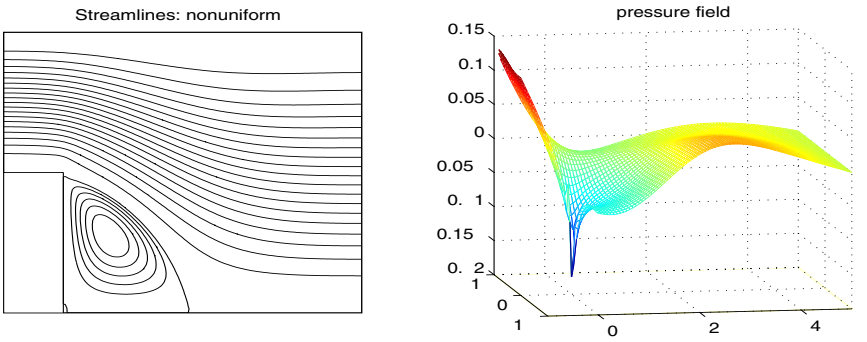


FIG. 5.1. *Streamline contours (left) and pressure plot (right) of a Q_2–Q_1 approximation to a backward facing step problem with parabolic inflow boundary conditions and natural outflow boundary conditions with $Re = 100$ on an underlying 64×192 grid.*

the viscosity ν is as in (1.1). For the [illegible]-generated F_p (both diagonally scaled and unscaled), we use the following drop tolerances in Algorithm 1: $drop_tol = 0.1, tol_1 = 0.01,$ and $tol_2 = 0.1$.

TABLE 5.1
Preconditioned GMRES iterations for backward facing step problem with an underlying 64×192 grid, Q_2–Q_1 discretization.

Re	F_p	*BFBt*	Sc-*BFBt*	*SPAC*	*SPAC–M*
10	30	54	19	55	23
100	42	64	21	76	30
200	47	65	22	86	41

TABLE 5.2
Preconditioned GMRES iterations for backward facing step problem with an underlying 128×384 grid, Q_2–Q_1 discretization.

Re	F_p	*BFBt*	Sc-*BFBt*	*SPAC*	*SPAC–M*
10	33	82	23	83	32
100	58	104	29	124	39
200	63	106	29	142	60

As the tables illustrate, the standard [illegible] method requires noticeably more iterations than the application-provided pressure convection–diffusion method. This type of behavior has been observed by others and has led to the perception that the [illegible] method is largely inferior to the pressure convection–diffusion method. However, inclusion of diagonal scaling improves the method so that it now significantly outperforms the pressure convection–diffusion method (often achieving a factor of two improvement in the number of iterations for higher Reynolds numbers). The tables further illustrate how the [illegible] method somewhat mirrors the behavior of [illegible]. In particular, the nonscaled version does not perform well while the scaled version is quite competitive with the application-provided pressure convection–diffusion preconditioner. It is also worth noting that these [illegible] iterations correspond to relatively large drop tolerances. We have performed additional experiments with smaller tolerances and have noticed only modest reductions in iteration counts. (See also some comments below on use of [illegible] tolerances.) It should be kept in mind that in the

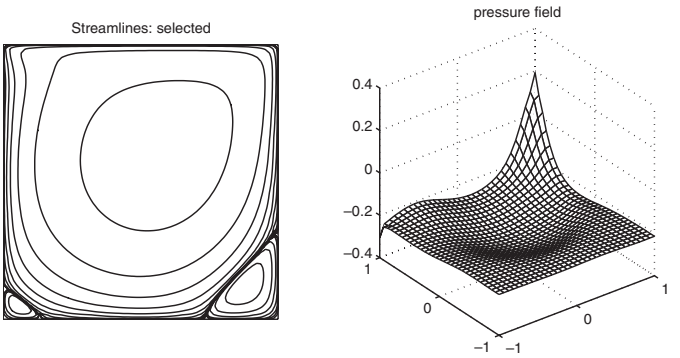


FIG. 5.2. *Exponentially spaced streamline contours (left) and pressure plot (right) of a Q_2–Q_1 approximation to a two-dimensional lid driven cavity problem with $Re = 500$ on a 64×64 grid.*

limit as the drop tolerance goes to zero, the [illegible] methods become identical to the [illegible] methods. Of course, in this limit the computation of the sparse approximate commutator would be quite costly compared to one using a large tolerance.

Next, we compare preconditioners on the lid driven cavity in two dimensions using a Q_2–Q_1 discretization. Specifically, we consider a square region with unit length sides. Velocities are zero on all edges except the top (lid), which has a driving velocity of one. The two-dimensional lid driven cavity is a well-known benchmark for fluids problems and contains many features of harder flows. Figure 5.2 shows a representative solution.

In Table 5.3 and Table 5.4, the numbers of iterations needed to reduce the initial residual by 10^6 are shown for a variety of Reynolds numbers $Re = 2/\nu$; the [illegible] drop tolerances are identical to those used for the backward facing step.

TABLE 5.3
GMRES iterations for lid driven cavity problem on a 64×64 grid, Q_2–Q_1 discretization.

Re	F_p	*BFBt*	Sc-*BFBt*	*SPAC*	*SPAC–M*
100	27	46	21	49	21
500	47	77	34	81	38
1000	70	93	55	98	63
2000	130	131	110	132	119

TABLE 5.4
GMRES iterations for lid driven cavity problem on a 128×128 grid, Q_2–Q_1 discretization.

Re	F_p	*BFBt*	Sc-*BFBt*	*SPAC*	*SPAC–M*
100	27	68	27	73	27
500	43	106	37	127	42
1000	56	125	45	149	65
2000	106	154	85	192	105

In this case, the [illegible] results are better relative to the pressure convection–diffusion method than in the case of the backward facing step. Once again the use of diagonal scaling improves the [illegible] method so that it outperforms the pressure convection–diffusion method (though not as dramatically as in the backward facing step problem). Similar to the results for the backward facing step, [illegible] and [illegible]–*M* roughly mirror their [illegible] counterparts, and the scaled algebraic methods compare

quite well with results for the application-provided pressure convection–diffusion preconditioners. We believe that in the large Reynolds number case, lack of accuracy of the discrete convection–diffusion operators F and F_p hinders the effectiveness of the pressure convection–diffusion preconditioner; comparison of Tables 5.3 and 5.4 shows that performance of the pressure convection–diffusion preconditioner and of the scaled algebraic methods improves as the mesh is refined. This is in contrast to the backward facing step problem where all the preconditioners exhibit some iteration growth. We suspect that iteration growth on the backward facing step problem (for all the preconditioners) is associated with the boundary conditions of the F_p system. This will be explored in our final example.

Next, we verify that similar convergence behavior is exhibited for different discretizations. Specifically, iterations for a lid driven cavity problem with a MAC discretization are given in Table 5.5 and Table 5.6. (Here the problem is posed on $[0,1] \times [0,1]$ and $Re = 1/\nu$.) In this case, scaled versions of [illegible] and [illegible] are not considered, as all the velocity basis functions effectively have the same scaling. That is, the equivalent of the mass matrix for the MAC scheme has a constant diagonal and so the introduction of M_1 and M_2 does not change the method. In the last column of these tables ([illegible]), we consider a further modification to the matrix F_p produced by [illegible] so that the boundary conditions correspond to those in the application-provided version of F_p. In particular, for problems such as the lid driven cavity with enclosed flow, it is standard for the application-provided method to enforce Neumann boundary conditions when discretizing the convection–diffusion operator on the pressure space. For problems such as the backward facing step where both inflow and outflow conditions are present the situation is somewhat more complicated [8]. Unfortunately, the [illegible] preconditioner does not automatically reproduce the proper boundary conditions. This should not be surprising as the notion of commuting does not apply at the boundaries. This implies that the algebraic commuting equation is not satisfied near boundaries and therefore the computed F_p stencil at the boundaries may not be best. To mimic Neumann boundary conditions, the diagonal of the [illegible] F_p is modified so that the row sums are identically zero.

TABLE 5.5
Preconditioned GMRES iterations for lid driven cavity problem on a 64×64 *grid, MAC discretization.*

Re	F_p	*BFBt*	*SPAC*	*SPACbc*
10	15	18	21	15
100	26	25	30	23
1000	54	42	51	50

TABLE 5.6
Preconditioned GMRES iterations for lid driven cavity problem on a 128×128 *grid, MAC discretization.*

Re	F_p	*BFBt*	*SPAC*	*SPACbc*
10	16	22	27	15
100	26	32	39	24
1000	57	53	65	54

Once again the results indicate that both the [illegible] and the [illegible] preconditioners are competitive with the application-provided pressure convection–diffusion preconditioner. Further, the inclusion of the boundary modification to [illegible]'s F_p

yields an additional reduction in the number of required iterations, with boundary-modified , performing better than the application-provided version of the pressure convection–diffusion method. More importantly, the , *bc* method appears to exhibit mesh independent convergence while mesh independence is less clear for the other two algebraic algorithms. Given this impact on convergence, further study of boundary conditions within these preconditioners needs to be explored.

We conclude this section with some observations on the costs, in CPU time, of the , preconditioning. It has been observed [4] that the setup cost may be significant when computing a sparse approximate inverse with a dynamically determined sparsity pattern. We have observed a similar phenomenon in computing , . As with standard sparse approximate inverses, the cost can be significantly reduced for the , algorithm if a predetermined sparsity pattern is employed. As we recall from section 4, the basic computational task involves the solution of the least-squares problem $\min \|\tilde{G}(\mathcal{I},\mathcal{J})f_j(\mathcal{J}) - b_j(\mathcal{I})\|_2^2$, where b_j is the jth column of $\tilde{F}\tilde{G}$ and f_j is the jth column of F_p to be computed. Instead of dynamically determining the index sets, $\mathcal{I}$ and $\mathcal{J}$, as in section 4, we can simply define $\mathcal{I}$ to be the set of nonzero row indices in b_j and $\mathcal{J}$ to be the set of nonzero column indices in $G(\mathcal{I},:)$. The least-squares problem is then solved , any additional computation to reduce the size of the residual commutator. This algorithm may not lead to as accurate a construction of the commutator, but it tends to perform almost as well and is significantly cheaper than the construction given in Algorithm 1.

Table 5.7 shows the iteration counts and timings using the simplified version of the , *bc* preconditioner for the driven cavity problem using a MAC discretization on a 128×128 grid. For the sake of comparison, timings for the application-provided pressure convection–diffusion preconditioner are also provided. These data were obtained in serial on a 3.20GHz Intel Xeon processor using the GMRES algorithm provided by MATLAB; the factors of F and G^TG used by the preconditioners were computed once and used throughout the iterations. The , *bc* iteration counts should be compared with those of Table 5.6. It is evident that this variant of the , preconditioner is competitive with that defined by Algorithm 1 (in terms of iteration counts), and the overhead associated with computing the preconditioner is small.

TABLE 5.7

Comparison of CPU times and iterations for simplified SPAC preconditioning, on a 128×128 *grid, MAC discretization.*

	F_p		*SPACbc*		
Re	Iters	Time	Build time	Iters	Time
10	16	37.40	7.17	16	37.46
100	26	60.56	7.45	25	58.53
1000	57	131.61	6.90	55	127.42

6. Conclusions. Variants of the pressure convection–diffusion preconditioning have been considered for the solution of the linear systems associated with the incompressible Navier–Stokes equations. Standard versions of this method require users to provide a discretization of a convection–diffusion equation over the pressure space. Although kernels needed for this discretization are available within most application codes, the additional burden of supplying this somewhat unnatural operator, denoted F_p, can be quite cumbersome. As an alternative, we have considered the use of an algebraic commuting relationship to determine the F_p operator automatically. We have shown that automatic techniques for generating F_p compete quite well

with application-provided pressure convection–diffusion methods in terms of convergence rates. These automatic methods are based on solving an algebraic commuting equation in a least-squares sense.

Two solution strategies to the commuting equation have been considered. The first centers on the normal equations and actually corresponds to a new interpretation of the [illegible] method. This new interpretation is somewhat more natural than that given in the original paper proposing the [illegible] method. Further, this new perspective leads to a diagonally scaled variant of the [illegible] preconditioner. This diagonal scaling is based on the diagonal of the mass matrix associated with the velocity equations and yields significantly improved convergence rates for the method. Overall, the diagonally scaled [illegible] method converges in a similar fashion to the application-provided pressure convection–diffusion method and actually outperforms it many times. It should, of course, be kept in mind that applying a [illegible] method is more costly than an application-provided pressure convection–diffusion method, because two Poisson solves and one convection–diffusion solve are required for each preconditioner invocation as opposed to only one Poisson solve and one convection–diffusion solve for the pressure convection–diffusion preconditioning. The significance of this extra Poisson solve is often quite modest, however, as the convection–diffusion system is much larger than the Poisson system and frequently dominates the run time.

The second solution strategy for the commuting equation builds on sparse approximate inverse techniques. Each column of F_p is built independent of every other column. The basic idea is to identify a sparsity pattern within each column and solve a local least-squares problem to minimize the error in the residual of the commuting equation. To some extent, this method can be viewed as an approximation of the [illegible] methods. The main advantage of the sparse approximate commutator, [illegible], computed in this way is that it requires one Poisson solve per preconditioner invocation similar to the application-provided pressure convection–diffusion methods. Thus, the cost per iteration should be nearly identical for the [illegible] and the application-provided pressure convection–diffusion methods. Numerical results have been given to illustrate how this method also competes quite effectively with application-provided pressure convection–diffusion preconditioners.

REFERENCES

[1] M. W. Benson, *Iterative Solution of Large Scale Linear Systems*, M.Sc. thesis, Lakehead University, Thunder Bay, ON, 1973.

[2] M. W. Benson and P. O. Frederickson, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Util. Math., 22 (1982), pp. 127–140.

[3] M. Benzi, *Preconditioning techniques for large linear systems: A survey*, J. Comput. Phys., 182 (2002), pp. 418–477.

[4] E. Chow, *A priori sparsity patterns for parallel sparse approximate inverse preconditioners*, SIAM J. Sci. Comput., 21 (2000), pp. 1804–1822.

[5] E. Chow and Y. Saad, *Approximate Inverse Preconditioners for General Sparse Matrices*, Tech. Report UMSI 94/101, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1994.

[6] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM J. Sci. Comput., 20 (1999), pp. 1299–1316.

[7] H. C. Elman, A. R. Ramage, D. J. Silvester, and A. J. Wathen, *Incompressible Flow Iterative Solution Software Package*, http://www.cs.umd.edu/∼elman/ifiss.html.

[8] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, 2005, to appear.

[9] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[10] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[11] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[12] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.

[13] S. V. Patankar and D. A. Spalding, *A calculation procedure for heat, mass and momentum transfer in three dimensional parabolic flows*, Int. J. Heat and Mass Trans., 15 (1972), pp. 1787–1806.

[14] J. B. Perot, *An analysis of the fractional step method*, J. Comput. Phys., 108 (1993), pp. 51–58.

[15] F. Saleri and A. Veneziani, *Pressure-Correction Algebraic Splitting Methods for the Incompressible Navier–Stokes-Equations*, Tech. report MOX-26, Politecnico Milano, 2003.

[16] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, J. Comput. Appl. Math., 128 (2001), pp. 261–279.

[17] A. Smith and D. Silvester, *Implicit algorithms and their linearisation for the transient incompressible Navier–Stokes equations*, IMA J. Numer. Anal., 17 (1997), pp. 527–545.

[18] S. Turek, *A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: From fully implicit non-linear schemes to semi-implicit projection methods*, Int. J. Numer. Meth. Fluids, 22 (1996), pp. 987–1011.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1669–1688
© 2006 Society for Industrial and Applied Mathematics

TOWARD AN h-INDEPENDENT ALGEBRAIC MULTIGRID METHOD FOR MAXWELL'S EQUATIONS*

JONATHAN J. HU†, RAYMOND S. TUMINARO†, PAVEL B. BOCHEV‡, CHRISTOPHER J. GARASI§, AND ALLEN C. ROBINSON¶

Abstract. We propose a new algebraic multigrid (AMG) method for solving the eddy current approximations to Maxwell's equations. This AMG method has its roots in an algorithm proposed by Reitzinger and Schöberl. The main focus in the Reitzinger and Schöberl method is to maintain null-space properties of the weak $\nabla \times \nabla \times$ operator on coarse grids. While these null-space properties are critical, they are not enough to guarantee h-independent convergence rates of the overall multigrid scheme. We present a new strategy for choosing intergrid transfers that not only maintains the important null-space properties on coarse grids but also yields significantly improved multigrid convergence rates. This improvement is related to those we explored in a previous paper, but is fundamentally simpler, easier to compute, and performs better with respect to both multigrid operator complexity and convergence rates. The new strategy builds on ideas in smoothed aggregation to improve the approximation property of an existing interpolation operator. By carefully choosing the smoothing operators, we show how it is sometimes possible to achieve h-independent convergence rates with a modest increase in multigrid operator complexity. Though this ideal case is not always possible, the overall algorithm performs significantly better than the original scheme in both iterations and run time. Finally, the Reitzinger and Schöberl method, as well as our previous smoothed method, are shown to be special cases of this new algorithm.

Key words. Maxwell's equations, eddy currents, AMG, multigrid, smoothed aggregation

AMS subject classifications. 76D05, 76D07, 65F10, 65F30

DOI. 10.1137/040608118

1. Introduction.

We consider the three-dimensional (3D) eddy current formulation of Maxwell's equations

$$\nabla \times \nabla \times \vec{E} + \sigma \vec{E} = \vec{f}, \tag{1.1}$$

with Neumann, Dirichlet, or periodic boundary conditions, where $\vec{E}$ is the unknown electric field to be computed, σ is the spatially varying electrical conductivity, and $\vec{f}$ is the known right-hand side. We assume that (1.1) is discretized with first-order edge elements on arbitrary unstructured meshes so that the following vector identity is preserved in a discrete sense:

$$\nabla \times (\nabla \phi) = 0. \tag{1.2}$$

*Received by the editors May 12, 2004; accepted for publication (in revised form) January 20, 2005; published electronically February 3, 2006. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94-AL85000. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/27-5/60811.html

†Sandia National Laboratories, Computational Math/Algorithms, P.O. Box 969, MS 9159, Livermore, CA 94551 (jhu@sandia.gov, rstumin@sandia.gov).

‡Sandia National Laboratories, Computational Math/Algorithms, P.O. Box 5800, MS 1110, Albuquerque, NM 87185 (pbboche@sandia.gov).

§Sandia National Laboratories, HEDP Theory & ICF Target Design, P.O. Box 5800, MS 1186, Albuquerque, NM 87185 (cjgaras@sandia.gov).

¶Sandia National Laboratories, Computational Physics R&D, P.O. Box 5800, MS 0378, Albuquerque, NM 87185 (acrobin@sandia.gov).

This identity states that gradients of scalar functions lie within the kernel of the curl operator. The discrete version of (1.1) is denoted by

$$K_1^{(e)} e_1 = f_1, \tag{1.3}$$

where $K_1^{(e)} = S_1 + M_1$. S_1 is the discrete approximation to the weak form of the first term in (1.1), and M_1 is the discrete approximation to the weak form of the second term in (1.1). The discrete null-space analogue of (1.2) is represented by the rectangular matrix T_1. This matrix is a discrete gradient operator and is trivial to construct. In particular, each row of T_1 contains at most two nonzeros (± 1) and corresponds to an edge between two nodes in the associated nodal graph [5]. The following relationship holds:

$$S_1 T_1 = \Theta,$$

where Θ denotes the zero matrix. In this paper, we use the notation (*curl*, *curl*) to refer to S_1 as it emphasizes the character of the operator. Edge element discretizations of the eddy current formulation with the discrete null-space property have been investigated for both tetrahedral and hexahedral meshes. While the multigrid techniques described here are not restricted to hexahedral meshes, all of our experiments have been with hexahedral meshes, where it is assumed that the domain of interest can in fact be covered by such elements [4, 16].

The focus of this paper is the solution of the discrete linear system given in (1.3) and the special treatment required by the multigrid smoother and the multigrid grid transfers to address the large (*curl*, *curl*) kernel. Several groups have considered this problem. Hiptmair developed a geometric multigrid method with h-independent (or mesh-independent) convergence rates in [10]. Reitzinger and Schöberl [11] proposed an initial algebraic multigrid (AMG) method for Maxwell's equations that maintains a coarse grid notion of (1.2) but does not have mesh-independent convergence rates. We considered modifications to Reitzinger and Schöberl's strategy for choosing a prolongator in a previous paper [3]. The current paper proposes an improvement to the intergrid transfers that is related to those explored in [3]. The resulting method performs noticeably better than the previous method in terms of both multigrid cost per iteration and convergence. In addition, the new prolongator is much simpler to understand and implement and leads to nearly h-independent convergence of a multigrid W-cycle on some model problems.

2. Background. In this section, we give an overview of multigrid, as well as previous work in applying multigrid to Maxwell's equations.

2.1. Multigrid overview. Multigrid methods (e.g., [9, 12, 8]) are among the most efficient iterative algorithms for solving the linear system, $Ax = b$, associated with elliptic partial differential equations. The basic idea is to damp errors by utilizing multiple resolutions in the iterative scheme. High-energy (or oscillatory) components are effectively reduced through a simple smoothing procedure, while the low-energy (or smooth) components are tackled using an auxiliary lower resolution version of the problem (coarse grid). The idea is applied recursively on the next coarser level. An example multilevel iteration is given in Figure 2.1 to solve

$$A_1 u_1 = b_1. \tag{2.1}$$

The two operators needed to fully specify the multigrid method are the relaxation procedures, R_k, $k = 1, \ldots, N_{levels}$, and the grid transfers, P_k, $k = 2, \ldots, N_{levels}$.

```
// Solve A_k u_k = b_k
procedure multilevel(A_k, b_k, u_k, k)
    u_k = R_k(A_k, b_k, u_k);
    if (k ≠ N_levels)
        r_k = b_k − A_k u_k;
        A_{k+1} = P_{k+1}^T A_k P_{k+1};
        u_{k+1} = 0;
        multilevel(A_{k+1}, P_{k+1}^T r_k, u_{k+1}, k + 1);
        u_k = u_k + P_{k+1} u_{k+1};
        u_k = R_k(A_k, b_k, u_k);
```

FIG. 2.1. *Multigrid* V*-cycle consisting of* N_{levels} *grids to solve* $A_1 u_1 = b_1$.

Note that P_k is an interpolation operator that transfers grid information from level k to level $k-1$. The coarse grid discretization operator A_{k+1} $(k \geq 1)$ is specified by the Galerkin product

$$A_{k+1} = P_{k+1}^T A_k P_{k+1}. \tag{2.2}$$

The key to fast convergence is the complementary nature of these two operators. That is, errors not reduced by R_k must be well interpolated by P_k.

When grid transfers are defined using mesh and/or finite element information, this is generally termed geometric multigrid. While geometric multigrid methods often work well, they can be somewhat difficult to implement for highly unstructured meshes with irregular boundaries (especially on parallel machines). Further, they often rely on details of the specific simulation program. That is, applications are forced to develop their own version of geometric multigrid where they supply a mesh hierarchy and a means to move between grids. While potentially efficient, software development can be time consuming and the resulting multigrid code may not be well suited for other applications. In algebraic multigrid, however, only A_1 and b_1 are given, hence R_k and P_k must be deduced from purely algebraic principles. While constructing multigrid methods via algebraic concepts presents certain challenges, AMG can be used for several problem classes without requiring a major effort for each application. In this paper, we focus on a strategy to determine the P_k's based on algebraic principles. It is assumed that A_1 and b_1 are given.

2.2. Geometric multigrid for Maxwell's equations. A unique difficulty in solving Maxwell's equations arises from the kernel of the $(curl, curl)$ operator. This kernel includes the gradients of all differentiable scalar functions. Since the gradient of a smooth function is smooth and the gradient of an oscillatory function is oscillatory, it is clear that both smooth and oscillatory functions lie within the kernel. We denote the range space of operator A by $\mathcal{R}an(A)$. In the standard multigrid context, relaxation must damp oscillatory errors in $\mathcal{R}an(T_1)$ and $\mathcal{R}an(T_1^{\perp})$, and the coarse grid correction must approximate smooth errors in $\mathcal{R}an(T_1)$ and $\mathcal{R}an(T_1^{\perp})$. While the canonical coarse grid transfers of a geometric multigrid method sufficiently approximate smooth errors in $\mathcal{R}an(T_1)$ and $\mathcal{R}an(T_1^{\perp})$,[1] a special relaxation procedure must be used so that oscillatory errors in $\mathcal{R}an(T_1)$ are damped. This is because methods like Jacobi and Gauss–Seidel rely on oscillatory error components being well represented in the residual. When M_1 is small compared to S_1, however, errors in the kernel of the

[1] Using the edge element basis functions.

procedure distributed_smoother $(K_1^{(e)}, b, u)$
Smooth on $K_1^{(e)} u = b$
$r = b_1 - K_1^{(e)} u$
$\tilde{K}_1 = T_1^T K_1^{(e)} T_1 \qquad // = T_1^T M_1 T_1$
$\tilde{b} = T_1^T r,\ \tilde{u} = 0$
Smooth on $\tilde{K}_1 \tilde{u} = \tilde{b}$
$u = u + T_1 \tilde{u}$

FIG. 2.2. *Distributed smoother for system* (1.3).

(*curl*, *curl*) operator are not well represented. This can be seen by considering the following error decomposition:

$$
\begin{aligned}
&\epsilon = \epsilon_T + \epsilon_{T^\perp}, \qquad \epsilon_T \in \mathcal{R}an(T_1),\ \epsilon_{T^\perp} \in \mathcal{R}an(T_1^\perp),\ ||\epsilon_T|| \approx ||\epsilon_{T^\perp}||, \\
&r = (S_1 + M_1)(\epsilon_T + \epsilon_{T^\perp}) \\
&\quad = S_1 \epsilon_{T^\perp} + M_1(\epsilon_T + \epsilon_{T^\perp}) \\
&\quad \approx S_1 \epsilon_{T^\perp} \text{ when } ||S_1|| \gg ||M_1||.
\end{aligned} \tag{2.3}
$$

Hiptmair was essentially the first to propose a suitable relaxation procedure for (1.3), and this led to a geometric multigrid method with h-independent convergence rates [10]. This smoother is a specific instance of a more general idea termed distributed relaxation [7]. Specifically, a standard smoother is first applied to $K_1^{(e)}$. Then, a correction equation is formed and projected onto $\mathcal{R}an(T_1)$. Standard smoothing is applied to the projected equation, and the correction is added into the previous solution estimate. A summary of this smoother is given in Figure 2.2. It is important to note that a basis for the kernel of the (*curl*, *curl*) operator must be explicitly constructed on all levels within a multigrid scheme to apply this relaxation procedure. A related smoother was considered in [15] for mixed finite element problems, and alternative smoothers have been proposed in [2].

2.3. AMG for Maxwell's equations. While several good geometric multigrid options now exist, the situation is not nearly so clear for AMG methods. The first practical AMG method for solving Maxwell's equations was developed by Reitzinger and Schöberl [11]. The principal idea of the method is to maintain the representation of a discrete gradient and its relationship to the discrete (*curl*, *curl*) operator on coarse meshes. Reitzinger and Schöberl achieve this by creating two multigrid hierarchies. The primary hierarchy is a coarsening of the space where the problem (1.3) lies. The interpolation operator is given by $P_k^{(e)}$, and the coarse grid operator is given by the Galerkin product

$$K_{k+1}^{(e)} = (P_k^{(e)})^T K_k^{(e)} P_k^{(e)}.$$

This is the hierarchy that is used during the multigrid iterations. The other hierarchy serves an auxiliary role and is used only in the setup phase to construct the intergrid transfers $P_k^{(e)}$ for the primary hierarchy.

To construct the auxiliary hierarchy, Reitzinger and Schöberl discretize the partial differential equation

$$\int_\Omega \sigma u \cdot v + \int_\Omega \nabla u \cdot \nabla v \tag{2.4}$$

using nodal finite elements on the same mesh as (1.1). This yields a matrix $K_1^{(n)}$. The matrix T_1 is also needed for this AMG method. T_1 can be created directly from the mesh. This is done by adding one matrix row for each edge in $K_1^{(e)}$ with the two nonzero entries corresponding to nodal degrees of freedom associated with the mesh. It is also possible, however, to construct T_1 using the matrix graph of $K_1^{(n)}$ [5]. Specifically, each off-diagonal nonzero (i, j) in the upper triangular portion of $K_1^{(n)}$ corresponds to a row in T_1 containing a 1 and a -1 in columns i and j. This implies that one way to build a coarse grid discrete "gradient," T_2, is to apply an AMG procedure to $K_1^{(n)}$. This yields a prolongation operator, $P_k^{(n)}$, and a coarse grid discretization matrix, $K_2^{(n)} = (P_1^{(n)})^T K_1^{(n)} P_1^{(n)}$. The graph of $K_2^{(n)}$ can then be used to define a coarse grid discrete gradient.

The auxiliary nodal hierarchy can be viewed as an explicit coarsening of the (*curl*, *curl*) null space. The key question is now how to choose the interpolation, $P_1^{(e)}$, so that T_2 is precisely the null space of the coarse grid discrete (*curl*, *curl*) operator,

$$S_2 = (P_1^{(e)})^T S_1 P_1^{(e)}. \tag{2.5}$$

Reitzinger and Schöberl showed that the following commuting relationship is necessary and sufficient to maintain the null-space property on coarse grids:

$$P_k^{(e)} T_{k+1} = T_k P_k^{(n)}, \tag{2.6}$$

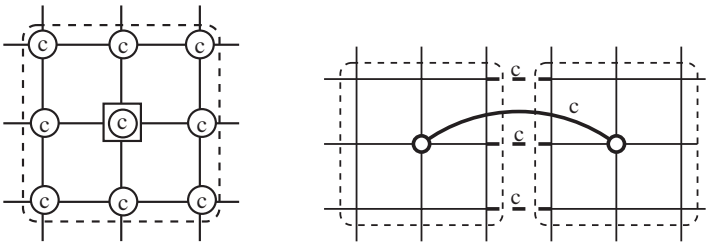
where T_k, $P_k^{(n)}$, and $P_k^{(e)}$ are the discrete gradient, nodal prolongator, and edge prolongator on the kth level. Specifically, assume that $S_k T_k = \Theta$ and that (2.6) holds. Then,

$$\begin{aligned} S_{k+1} T_{k+1} &= (P_k^{(e)})^T S_k P_k^{(e)} T_{k+1} \\ &= (P_k^{(e)})^T S_k T_k P_k^{(n)} \qquad \text{by (2.6)} \\ &= \Theta. \end{aligned}$$

Further discussions of the commuting property (2.6) are given in [11]. Thus, our computational task is reduced to producing a $P_k^{(e)}$ given T_k, T_{k+1}, and $P_k^{(n)}$ such that (2.6) holds. This will preserve the null space on coarse meshes and allow the distributed relaxation procedure to be applied. To do this effectively, however, a prolongation operator that maintains good approximation properties and leads to low multigrid operator complexities (cost per multigrid iteration) is necessary.

Reitzinger and Schöberl proposed a strategy for computing a $P_k^{(e)}$ that satisfies (2.6) and gives low multigrid operator complexities. Specifically, coarse nodes are first constructed by aggregating (or grouping) fine nodes together. $P_k^{(n)}$ is now defined as piecewise-constant interpolation from each coarse node to all of the fine nodes within its aggregate. A coarse nodal discretization is then defined using $P_k^{(n)}$, $K_k^{(n)}$, and the Galerkin product, (2.2). Coarse edges in the coarse nodal matrix simply link coarse nodes from adjacent aggregates. $P_k^{(e)}$ is then defined by injecting the value from each coarse edge to all the fine edges connecting the two adjacent aggregates associated with the coarse edge. We omit the details of the prolongator and refer the reader to [11, 3]. Figure 2.3 depicts a simple example for a single nodal basis function and a single edge basis function in the respective prolongators.

Reitzinger and Schöberl's paper properly identified the importance of (2.6). Their AMG method is easy to compute, maintains low multigrid operator complexity, and



(a) Node interpolation. (b) Edge interpolation.

FIG. 2.3. (a) *Reitzinger and Schöberl piecewise-constant nodal interpolation. The coarse grid value (in square) is interpolated only to nodes that form the aggregate (in circles).* (b) *Interpolation on a uniform mesh within a single edge of* $P_k^{(e)}$. *The coarse grid edge (heavy solid) with value c is interpolated only to fine edges (heavy dashed) passing between nodal aggregates (in dashed boxes).*

TABLE 1
Number of iterations to reduce initial residual by 10^{10} *with the Reitzinger and Schöberl AMG method applied to a model problem on the unit square with* $\sigma = 10^{-3}$.

Grid size	CG/AMG iterations
15×15	16
25×25	24
50×50	42
100×100	76
150×150	93

is relatively insensitive to the electric conductivity. The main drawback, however, is that the prolongator has poor approximation properties, and so multigrid convergence rates suffer. In particular, h-independent AMG convergence rates typically rely on accurately approximating functions with smaller energy norm (i.e., algebraically smooth functions) on coarse meshes. Within smoothed aggregation, for example, the theory relies on bounding the energy of all the coarser grid basis functions independent of mesh spacing. This is not possible with piecewise-constant basis functions. The Reitzinger and Schöberl edge prolongation operators are built using piecewise-constant interpolation for the nodal space and actually are less accurate than piecewise-constant operators. In fact, when interpolating coarse grid edges, there are no contributions to fine grid edges that are contained completely within an aggregate. Contributions are made only to fine grid edges that connect the two aggregates defining the coarse grid edge. Thus, it is not possible to bound the energy of the coarse grid basis functions and so it is not surprising that the resulting method has convergence rates that are far from h-independent. Table 1 illustrates the growth in iterations as the mesh size increases for a model two-dimensional (2D) problem using a preconditioned conjugate gradient (CG) algorithm where a Reitzinger and Schöberl AMG method is used in a V(1,1) cycle.[2] The smoother consists of three steps: one symmetric Gauss–Seidel iteration on the entire system, one symmetric Gauss–Seidel iteration on the null-space projected equations, and a final symmetric Gauss–Seidel iteration on the entire system. It is clear for very large problems that this lack of scalability will be prohibitive.

[2]V(1,1) indicates that a V-cycle multigrid algorithm is used with 1 presmoothing sweep and 1 postsmoothing sweep on each level.

Therefore, the remainder of this paper describes a new prolongation operator that improves scalability.

3. An improved prolongation operator. Consider the new prolongation operator

$$\mathcal{P}_k^{(e)} = (I - \alpha D_{S,k}^{-1} S_k + \beta T_k D_{T,k}^{-1} T_k^T M_k) P_k^{(e)}, \tag{3.1}$$

where $P_k^{(e)}$ is the Reitzinger and Schöberl edge prolongator, S_k is the (*curl*, *curl*) term on the kth level, M_k is the mass term on the kth level, $D_{S,k}$ is the diagonal of S_k, $D_{T,k}$ is the diagonal of $T_k^T M_k T_k$, and α and β are damping parameters. It is easy to show that $\mathcal{P}_k^{(e)}$ still satisfies the commuting properties. Namely,

$$\begin{aligned}
\mathcal{P}_k^{(e)} T_{k+1} &= (I - \alpha D_{S,k}^{-1} S_k + \beta T_k D_{T,k}^{-1} T_k^T M_k) P_k^{(e)} T_{k+1} \\
&= (I - \alpha D_{S,k}^{-1} S_k + \beta T_k D_{T,k}^{-1} T_k^T M_k) T_k P_k^{(n)} \qquad \text{by (2.6)} \\
&= (I + \beta T_k D_{T,k}^{-1} T_k^T M_k) T_k P_k^{(n)} \qquad \text{using } S_k T_k = \Theta \\
&= T_k (I + \beta D_{T,k}^{-1} T_k^T M_k T_k) P_k^{(n)} \\
&= T_k \mathcal{P}_k^{(n)},
\end{aligned}$$

where

$$\mathcal{P}_k^{(n)} = (I + \beta D_{T,k}^{-1} T_k^T M_k T_k) P_k^{(n)}. \tag{3.2}$$

Thus, the commuting property is satisfied when the nodal prolongator is given by $\mathcal{P}_k^{(n)}$.

The new nodal prolongator (3.2) can be viewed as a *smoothed* version of the original piecewise-constant nodal interpolation operator. This terminology comes from the smoothed aggregation multigrid method [14, 13]. In smoothed aggregation multigrid, an original (or tentative) prolongator is developed. Then, a new prolongator, Q, is produced by lowering the energy of the tentative prolongator via the following damped Jacobi iteration:

$$Q = (I - \gamma D^{-1} A) Q_T, \tag{3.3}$$

where Q_T is a tentative prolongator, A is some fine grid discretization matrix, D is the diagonal of A, and γ is a damping parameter. When applied to a Poisson operator, the tentative prolongator corresponds to piecewise-constant interpolation. For this paper, it is important to understand that the smoothed prolongator has significantly better approximation properties than piecewise-constant interpolation and that the smoothing step is critical to obtaining h-independent multigrid convergence rates for the nodal problem [6, 13, 14].

In our context, it is reasonable to expect that a smoothed nodal prolongator *might* give rise to an improved edge prolongator because low-energy modes are better approximated. If the nodal problem is given by

$$K_1^{(n)} = T_1^T K_1^{(e)} T_1 = T_1^T M_1 T_1 \tag{3.4}$$

instead of discretizing (2.4), then (3.2) and (3.3) are now identical with $\gamma = \beta$ and $A = K_1^{(n)}$. That is, the new nodal prolongator corresponds to a *smoothed* version of the original piecewise-constant prolongator. Using (3.4) has the distinct advantage

that the user no longer needs to provide a separate nodal discretization matrix. Within the multigrid method, matrix $K_1^{(n)}$ can be constructed automatically as described in (3.4) by operators that are available. In fact, $K_1^{(n)}$ also appears in the distributed relaxation method in Figure 2.2 and so is already computed and available within the method. Additionally, the operator $T_k^T M_k T_k$ still incorporates material properties, σ, which are advantageous to utilize in the nodal aggregation phase.

It is now clear that the $\beta T_k D_{T,k}^{-1} T_k^T M_k$ term in (3.1) corresponds to smoothing the nodal prolongator. To complete our understanding of this term, however, it is necessary to look at the important role the nodal prolongator plays in the AMG iterations (even though it does not appear explicitly). To this end, we consider the Galerkin product and the coarse grid nodal matrix obtained by projecting the coarse edge operator into the null space. It is important to keep in mind that this matrix (which is used in relaxation) governs convergence in the null space. Specifically, we have

$$\begin{aligned} K_{k+1}^{(n)} &= (\mathcal{P}_k^{(n)})^T K_k^{(n)} \mathcal{P}_k^{(n)} \\ &= (\mathcal{P}_k^{(n)})^T T_k^T K_k^{(e)} T_k \mathcal{P}_k^{(n)} && \text{by (3.4)} \\ &= T_{k+1}^T (\mathcal{P}_k^{(e)})^T K_k^{(e)} \mathcal{P}_k^{(e)} T_{k+1} && \text{by (2.6)} \\ &= T_{k+1}^T K_{k+1}^{(e)} T_{k+1}. \end{aligned}$$

This implies that it is, in fact, the nodal interpolation operator that completely determines the null-space properties of the discretization matrix on the coarse meshes. Thus, an improved nodal prolongator improves the coarse discretization within the null space.

Before concluding this discussion of nodal prolongators, it is important to make one modification to (3.1). In particular, the M_k term can be simplified to significantly reduce the nonzeros in both $\mathcal{P}^{(e)}$ and the resulting coarse grid discretization matrices. To do this, we replace M_k by $\tilde{M}_k$, where $\tilde{M}_k$ is obtained by lumping the off-diagonal elements in M_k to the diagonal. This is a well-known technique in approximating mass matrices. In our context, it can be helpful in reducing the number of nonzeros and does not affect the commuting property. As a result, we now have

$$\mathcal{P}_k^{(e)} = (I - \alpha D_{S,k}^{-1} S_k + \beta T_k D_{T,k}^{-1} T_k^T \tilde{M}_k) P_k^{(e)}. \tag{3.5}$$

The $\alpha D_{S,k}^{-1} S_k$ term in (3.5) does not appear in the new nodal prolongator. In fact, for a given nodal prolongator there are several possible edge prolongators that satisfy the commuting property. To better understand the $\alpha D_{S,k}^{-1} S_k$ term, we rewrite (3.5) as

$$\begin{aligned} \mathcal{P}_k^{(e)} &= (I - \alpha D_{S,k}^{-1} S_k)(I + \beta T_k D_{T,k}^{-1} T_k^T \tilde{M}_k) P_k^{(e)} \\ &= (I - \alpha D_{S,k}^{-1} S_k) \hat{\mathcal{P}}_k^{(e)}. \end{aligned} \tag{3.6}$$

It can now be seen that $\hat{\mathcal{P}}_k^{(e)}$ is essentially a tentative edge prolongator obtained using the smoothed nodal prolongator. To improve $\hat{\mathcal{P}}_k^{(e)}$, a further smoothing step in the edge space is performed to arrive at $\mathcal{P}_k^{(e)}$. This additional smoothing step is needed because the edge prolongator may not be as accurate (or as smooth) as the corresponding nodal prolongator.

To better understand the relationship between nodal and smoothed prolongators, consider an operator, $Q^{(n)}$, that corresponds to linear nodal interpolation on a uniform mesh. Further, let us assume that we have obtained the operator $Q^{(e)}$ satisfying the commuting equation

$$Q^{(e)}T_{k+1} = T_k Q^{(n)}. \tag{3.7}$$

If the right-hand side of (3.7) is applied to a coarse grid linear function v, then the resulting function is essentially constant. This follows from the fact that $Q^{(n)}v$ produces a linear function; a discrete gradient T_k applied to $Q^{(n)}v$ yields a constant function. Combining this with (3.7) implies that

$$Q^{(e)}z = w, \tag{3.8}$$

where w is a constant and $z = T_{k+1}v$ (implying that z is a constant function). Clearly, there are several $Q^{(e)}$ that satisfy (3.8), including inaccurate piecewise-constant edge interpolation! The important conclusion to draw from this fact is that the commuting relationship does not imply that the edge prolongator is as accurate as the corresponding nodal prolongator. In our context, this has been observed experimentally and it has motivated the application of an additional edge smoothing step (3.6) to improve the properties of the edge prolongator.

To complete the prolongator description (3.5), we must choose the damping parameters α and β. These parameters are chosen according to standard smoothed aggregation multigrid criteria, based on energy minimization [13]. Specifically,

$$\alpha = \frac{4}{3}\rho(D_{S,k}^{-1}S_k)^{-1} \tag{3.9}$$

and

$$\beta = \frac{4}{3}\rho(D_{T,k}^{-1}T_k^T M_k T_k)^{-1}, \tag{3.10}$$

where $\rho(\cdot)$ is the spectral radius. This spectral radius is estimated using a few (e.g., 5–10) iterations of a symmetric eigenvalue solver (as in standard smoothed aggregation).

4. AMG operator complexity. The main AMG difficulty in solving (1.1) is that the grid transfers must satisfy the commuting relationship and, at the same time, be sufficiently accurate in both the null space and its complement. The remarkable aspect of (3.5) is that it improves the accuracy of the Reitzinger and Schöberl prolongator in both the null space and its complement *without* adversely affecting the overall cost per iteration of the AMG process. To see this, we first define multigrid operator complexity as follows:

$$\text{AMG operator complexity} = \frac{\sum_{k=1}^{N_{levels}} nnz(K_k^{(e)})}{nnz(K_1^{(e)})}, \tag{4.1}$$

where $nnz(A)$ is the number of nonzeros in the matrix A. Ideally, $nnz(K_k^{(e)}) \ll nnz(K_1^{(e)})$ for $k > 1$. This occurs when coarse grid matrices have significantly fewer rows and only a modest increase in the number of nonzeros per row relative to $K_1^{(e)}$.

To analyze nonzero behavior, we look more closely at the coarse grid discretization matrix given by the Galerkin formula

$$K_{k+1}^{(e)} = (\mathcal{P}_k^{(e)})^T K_k^{(e)} \mathcal{P}_k^{(e)}.$$

Substituting a simplified version of (3.5), we define

$$\hat{K}_{k+1}^{(e)} = (P_k^{(e)})^T (I - S_k + Z_k) K_k^{(e)} (I - S_k + Z_k) P_k^{(e)},$$

where $Z_k = T_k T_k^T$. $\hat{K}_{k+1}^{(e)}$ is identical to $K_{k+1}^{(e)}$ when the damping parameters are one, and the diagonal matrices are the identity in (3.5). These simplifications obviously do not affect multigrid operator complexity. If we multiply the three inner terms and keep only the higher order terms, we have

$$\tilde{K}_{k+1}^{(e)} = (P_k^{(e)})^T (S_k K_k^{(e)} S_k - S_k K_k^{(e)} Z_k - Z_k K_k^{(e)} S_k + Z_k K_k^{(e)} Z_k) P_k^{(e)}. \tag{4.2}$$

Once again, the nonzero pattern of $\tilde{K}_{k+1}^{(e)}$ is identical to that of $K_{k+1}^{(e)}$ as the higher order terms govern this pattern. The multigrid operator complexity of the AMG iteration is connected to understanding how the number of nonzeros per row grows in the above expression. For general unstructured matrices this is quite cumbersome. However, for simple restricted cases it is possible to show how the number of nonzeros behave. Specifically, consider a regular uniform mesh on a square. Let us further assume that each aggregate is a perfect 3×3 brick (as depicted in Figure 2.3). Typically, aggregates are defined by some greedy algorithm. A center node is picked, preferably one not adjacent to existing aggregates, and then this node along with its immediate neighbors defines a new aggregate. In the ideal case, perfect 3×3 bricks will be formed when the discretization of (2.4) on a uniform grid yields a 9-point stencil or when $K^{(n)}$ has the form (3.4) on a uniform grid. Though this case is idealized, it gives us insight into the multigrid operator complexity growth we might expect in more realistic settings. Clearly, if the number of nonzeros grows drastically in the ideal case, one should expect high multigrid operator complexity in the nonideal situation.

For the ideal case, it is possible to illustrate the support of a single basis function (or column) in $P_k^{(e)}$ and show how this support grows with the four terms in (4.2). Specifically, consider a vector, v, with a set of nonzero edge entries denoted E_v. These edges are incident to a set of nodes, denoted N_v, and a set of grid cells, denoted C_v.

The application of S_k or $K_k^{(e)}$ to v results in a new vector with nonzero elements in any edge contained in the cells C_v. The application of Z_k to v results in nonzero elements in any edge incident to N_v. Figures 4.1–4.4 show the multigrid operator complexity details. The three vertical lines in the center represent the jth column of $P_k^{(e)}$ and the additional lines illustrate the support growth. The key point to notice in the figures is that the support never touches bordering aggregates (aggregates that are only partially illustrated). This implies that the jth column of $\tilde{K}_{k+1}^{(e)}$ is zero for all coarse grid edges incident to these bordering aggregates. Thus, there are only seven nonzeros in the jth column corresponding to the seven coarse edges that connect the six central aggregates depicted in the figures. All of this means that the nonzeros per row do not grow in this simple case as the original fine grid stencil contains seven nonzeros. In three dimensions, the situation is identical when each aggregate corresponds to a $3 \times 3 \times 3$ brick. Here, there is once again no growth in the number of nonzeros per row. Of course, ideal aggregates will not generally occur. However, we have some confidence that the multigrid operator complexity will not grow drastically. Computed complexities will be given with the numerical results to support this claim.

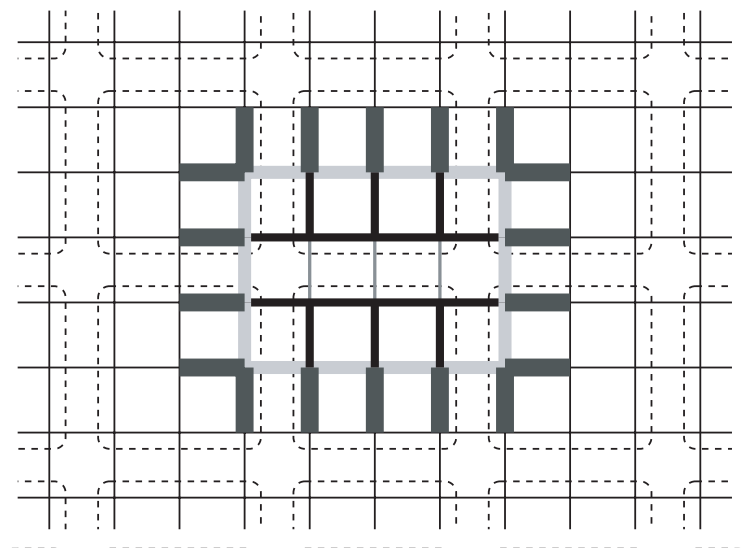


FIG. 4.1. *Nonzero edges due to* $Z_k K_k^{(e)} Z_k P_k^{(e)}$ *term.* ——— *denotes the three nonzero edges in* $P_k^{(e)} e_j$ *where* e_j *has only one nonzero in the* j*th position.* ▬▬ *denotes additional nonzero edges in* $Z_k P_k^{(e)} e_j$. ▬▬ *denotes the extra nonzero edges in* $K_k^{(e)} Z_k P_k^{(e)} e_j$. *Finally* ▬▬ *denotes the remaining nonzero edges induced by* $Z_k K_k^{(e)} Z_k P_k^{(e)} e_j$.

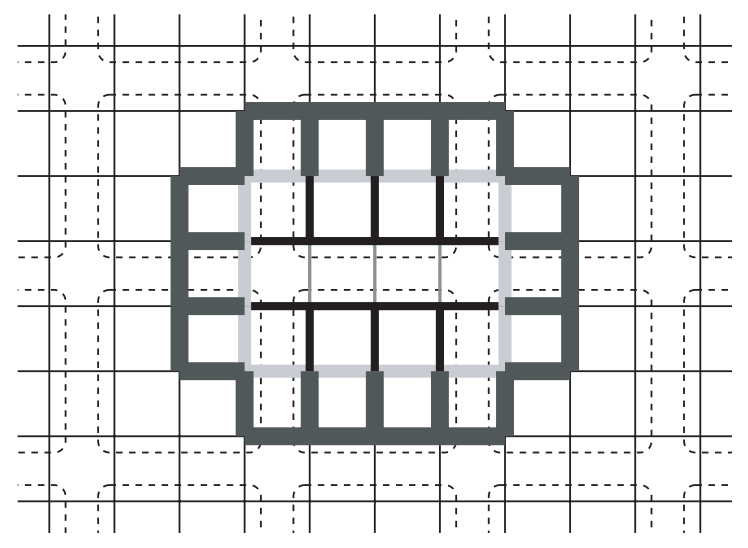


FIG. 4.2. *Nonzero edges due to* $S_k K_k^{(e)} Z_k P_k^{(e)}$ *term.* ——— *denotes the three nonzero edges in* $P_k^{(e)} e_j$ *where* e_j *has only one nonzero in the* j*th position.* ▬▬ *denotes additional nonzero edges in* $Z_k P_k^{(e)} e_j$. ▬▬ *denotes the extra nonzero edges in* $K_k^{(e)} Z_k P_k^{(e)} e_j$. *Finally* ▬▬ *denotes the remaining nonzero edges induced by* $S_k K_k^{(e)} Z_k P_k^{(e)} e_j$.

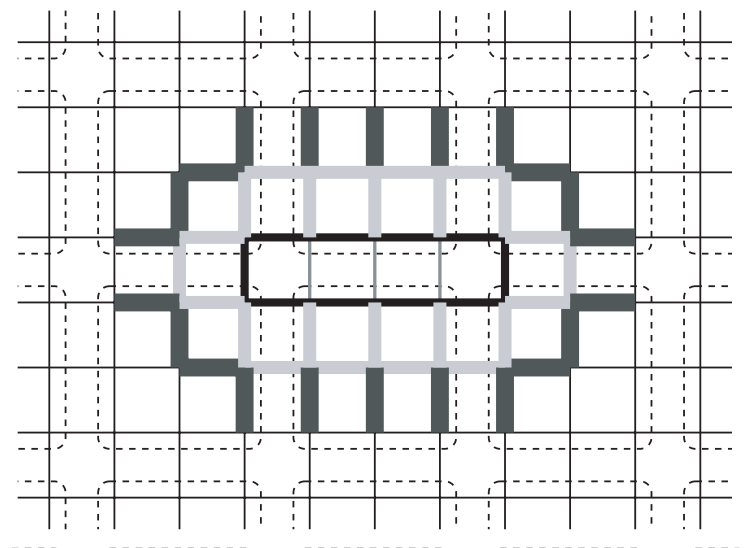


FIG. 4.3. *Nonzero edges due to* $Z_k K_k^{(e)} S_k P_k^{(e)}$ *term.* ——— *denotes the three nonzero edges in* $P_k^{(e)} e_j$ *where* e_j *has only one nonzero in the* j*th position.* ▬ *denotes additional nonzero edges in* $S_k P_k^{(e)} e_j$. ▬ *denotes the extra nonzero edges in* $K_k^{(e)} S_k P_k^{(e)} e_j$. *Finally* ▬ *denotes the remaining nonzero edges induced by* $Z_k K_k^{(e)} S_k P_k^{(e)} e_j$.

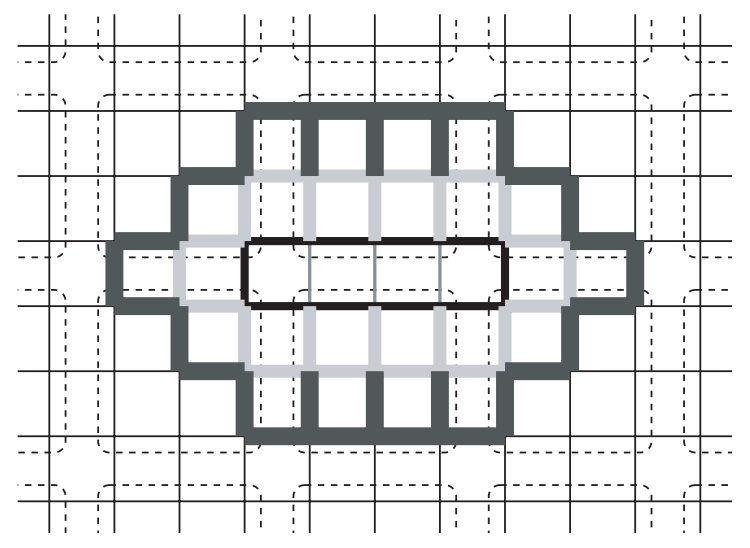


FIG. 4.4. *Nonzero edges due to* $S_k K_k^{(e)} S_k P_k^{(e)}$ *term.* ——— *denotes the three nonzero edges in* $P_k^{(e)} e_j$ *where* e_j *has only one nonzero in the* j*th position.* ▬ *denotes additional nonzero edges in* $S_k P_k^{(e)} e_j$. ▬ *denotes the extra nonzero edges in* $K_k^{(e)} S_k P_k^{(e)} e_j$. *Finally* ▬ *denotes the remaining nonzero edges induced by* $S_k K_k^{(e)} S_k P_k^{(e)} e_j$.

TABLE 2
Interpolation used in experiments. Parameters α and β are chosen by (3.9) and (3.10), respectively.

$\mathcal{P}^{(e)}(\cdot,\cdot)$	Description of interpolants used in experiments
$\mathcal{P}^{(e)}(0,0)$	Reitzinger and Schöberl interpolation
$\mathcal{P}^{(e)}(\alpha,0)$	edge smoothed interpolation
$\mathcal{P}^{(e)}(\alpha,\beta)$	edge and nodal smoothed interpolation

5. Numerical experiments. To assess the interpolation strategy proposed in this paper, we compare three AMG methods. Each method uses $\mathcal{P}^{(e)}$ with different choices of α and β. See Table 2 for the choices of α and β. In each case a multigrid W-cycle is used as a preconditioner for a conjugate gradient iteration. The one exception is the first experiment, where we use both V- and W-cycles to demonstrate the efficacy of $\mathcal{P}^{(e)}(\alpha,\beta)$. The presmoother consists of two steps.

1. A damped Jacobi iteration. For an initial guess, $u^{(0)}$, and right-hand side, f, this gives $u^{(\frac{1}{2})} = u^{(0)} + \mu_1 D^{-1}(f - K_k^{(e)} u^{(0)})$, where D is the diagonal of $K_k^{(e)}$.
2. A damped Jacobi iteration for the projected equations. Defining $\tilde{D}$ as the diagonal of $T_k^T K_k^{(e)} T_k$ gives $u^{(1)} = u^{(\frac{1}{2})} + \mu_2 T_k \tilde{D}^{-1} T_k^T (f - K_k^{(e)} u^{(\frac{1}{2})})$.

These parameters are given by

$$\mu_1 = \frac{60}{31\rho(K_k^{(e)})} \qquad \text{and} \qquad \mu_2 = \frac{60}{31\rho(T_k^T K_k^{(e)} T_k)}.$$

The damping parameters correspond to a first-degree Chebyshev polynomial defined over the intervals $[\rho(K_k^{(e)})/30, \rho(K_k^{(e)})]$ and $[\rho(T_k^T K_k^{(e)} T_k)/30, \rho(T_k^T K_k^{(e)} T_k)]$, respectively. The intervals effectively define the range of "high frequencies" that must be damped by the smoother. The lower end point is given by dividing the upper end point by the approximate coarsening rate (ratio of unknowns between the fine and coarse meshes) of the multigrid algorithm. Details of this smoother can be found in [1]. It is oriented toward damping high frequency errors and uses only an estimate of the largest eigenvalue of the matrix. The postsmoother performs *step* 2 first and then *step* 1, thus guaranteeing the symmetry of the preconditioner. For the uniform-grid examples, a right-hand side is generated by taking a random vector and multiplying by the discretization matrix. A zero initial guess is used for the uniform-grid examples.

We use the following T-*norm* to evaluate convergence:

$$||r||_T = ||r||_2 + \gamma ||T_1^T r||_2. \tag{5.1}$$

The idea of this norm is to assure that the residual is reduced in both the range space and the null space of the (*curl*, *curl*) operator. This is needed to compensate for the poor representation of the error when M_1 is small and the two norm of the residual is used. This can be seen from (2.3). The factor γ in (5.1) is chosen so that the two terms are initially on the same scale. For example, $\gamma = h^2/\sigma$ for (1.1) discretized on a uniform quadrilateral mesh with constant conductivity. In general, one can choose

$$\gamma = \frac{||K_1^{(e)} v||_2}{||T_1^T K_1^{(e)} v||_2},$$

where $K_1^{(e)} = S_1 + M_1$ and v is a normalized random vector. In this way, we can equilibrate the two terms by the relative sizes of S_1 and M_1. In our experiments the

iterations terminate when $||r_{\text{final}}||_T/||r_0||_T < 10^{-8}$, where r_{final} is the final residual and r_0 is the initial residual.

Before proceeding with the experimental results, there are three practical implementation considerations that should be discussed. First, while in our analysis we assumed that the mass term M_1 is lumped, in practice we do not always lump. As is shown in the experiments, not lumping can result in slightly higher, but still acceptable, multigrid operator complexities. The second is that the mass and stiffness matrices must be projected separately (via the Galerkin product) due to the presence of S_k and $\tilde{M}_k$ in (3.5). This requires additional computation when compared to directly projecting $K_k^{(e)}$ (though this can be significantly reduced if M_k is lumped before projection). There is, however, an important advantage in maintaining the M_k's with respect to numerical cancellation (i.e., rounding errors). Cancellation can be significant when S_k is much larger than M_k and the nodal discretization (needed in the distributed relaxation) is computed via $T_k^T K_k^{(e)} T_k$. This cancellation can be avoided by using $T_k^T M_k T_k$ to compute the nodal discretization. If it is absolutely imperative that S_k and M_k not be stored explicitly (e.g., they are not available on the finest mesh) and it is known that the conductivity is relatively constant, (3.5) can be approximated. In particular, M_k can be ignored (equivalent to approximating it by a multiple of the identity) and $K_k^{(e)}$ can be used instead of S_k without significant convergence degradation.

The third implementation issue concerns the choice of β. We have found in our experimentation that convergence can be enhanced by using a larger β than that given by standard smooth aggregation arguments, (3.10). The reason for this can be seen in (3.6). Specifically, β determines not only the nodal prolongator, (3.2), but also the tentative edge prolongator. While (3.10) is given to optimize the nodal prolongator, it does not necessarily provide for the best tentative edge prolongator. That is, higher values of β can yield a better starting tentative edge prolongators while sacrificing a bit the nodal prolongator. We emphasize that having a good tentative edge prolongator is crucial; for example, just smoothing the original Reitzinger and Schöberl edge prolongator does not yield h-independent multigrid convergence rates. When combined with an appropriate edge smoothing parameter, α, an appropriate choice of β can yield an improved smoothed edge prolongator, $\mathcal{P}^{(e)}$. This difficulty of finding a single β that yields both suitable tentative edge prolongators and smoothed nodal prolongators leads us to suspect that the convergence of the resulting method is not always mesh independent. We are still exploring alternatives to (3.10) that may produce the best compromise.

5.1. AMG with constant conductivity. We first consider (1.1) discretized with rectangular elements on the unit square with periodic boundary conditions and a constant value of σ. Table 3 gives iteration counts and run times for CG preconditioned with a V-cycle. Table 4 gives iteration counts and run times for CG preconditioned with a W-cycle. Table 5 gives AMG multigrid operator complexity results for the 810×810 mesh.

There are several things to notice about the data in Tables 3 and 4. The main observation is that AMG using the full $\mathcal{P}^{(e)}(\alpha, \beta)$ operator exhibits near h-independence, as can be seen in Table 4. Neither $\mathcal{P}^{(e)}(0,0)$ (the original Reitzinger and Schöberl operator) nor $\mathcal{P}^{(e)}(\alpha, 0)$ has an h-independent convergence rate, even with W-cycles. Smoothing alone (i.e., $\alpha \neq 0$, $\beta = 0$) is beneficial but not enough for grid independence. These improved convergence rates translate directly into significantly improved run times (one example is more than five times faster). In addition, for a fixed problem

TABLE 3
Iteration counts/run times (in seconds) for the CG/AMG V(1,1) *schemes on a* 2*D model problem with constant conductivity* σ.

σ	Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
10^1	30×30	23/0.1	18/0.1	13/0.1
	90×90	40/1.6	26/1.2	16/0.9
	270×270	77/50.5	36/23.4	20/17.1
	810×810	167/1061.9	49/370.6	23/204.3
10^{-1}	30×30	27/0.1	21/0.1	14/0.1
	90×90	47/1.9	33/1.3	18/0.9
	270×270	80/52.7	50/35.1	21/17.2
	810×810	175/1113.5	74/555.7	25/184.4
10^{-3}	30×30	28/0.1	23/0.1	14/0.04
	90×90	48/2.0	35/1.5	18/0.9
	270×270	107/70.1	61/42.6	21/17.7
	810×810	175/1115.0	77/577.7	25/184.7

TABLE 4
Iteration counts/run times (in seconds) for the CG/AMG W(1,1) *schemes on a* 2*D model problem with constant conductivity* σ.

σ	Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
10^1	30×30	20/0.1	17/0.1	13/0.1
	90×90	32/1.5	21/1.0	15/0.8
	270×270	58/39.6	26/19.5	15/13.3
	810×810	99/698.3	29/243.0	16/134.0
10^{-1}	30×30	23/0.1	20/0.1	13/0.1
	90×90	34/1.4	26/1.2	15/0.9
	270×270	60/41.1	32/24.8	15/14.4
	810×810	105/741.9	35/291.1	16/134.2
10^{-3}	30×30	25/0.1	21/0.1	13/0.1
	90×90	34/1.4	27/1.2	15/0.8
	270×270	71/49.7	46/34.9	15/13.7
	810×810	105/739.7	37/309.9	16/134.0

size, the iteration counts of the new prolongator is fairly insensitive to the size of the conductivity for this model problem. This is highly desirable for difficult problems with low conductivity. The operator complexity of the multigrid hierarchy is given in Table 5, where the first line corresponds to the multigrid operator complexity for the original Reitzinger and Schöberl algorithm. The complexity for AMG based on $\mathcal{P}^{(e)}(\alpha,\beta)$ requires some explanation. For larger σ, the mass in the third term of (3.1) results in nonzero growth of the coarser level operators. The contribution of the mass

TABLE 5
AMG multigrid operator complexities for 2D 810×810 *grid,* $\sigma \in \{10^1, 10^{-1}, 10^{-3}\}$.

Method	AMG complexity
$\mathcal{P}^{(e)}(0,0)$	1.12
$\mathcal{P}^{(e)}(\alpha,0)$	1.12
$\mathcal{P}^{(e)}(\alpha,\beta)$	1.12 to 1.51

TABLE 6
Iteration counts/run times (in seconds) for the CG/AMG W(1,1) *schemes on a 3D model problem with constant conductivity* σ.

σ	Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
10^1	$15 \times 15 \times 15$	19/0.9	19/1.1	19/1.3
	$45 \times 45 \times 45$	30/79.7	20/55.5	20/69.9
	$135 \times 135 \times 135$	57/5532.7	26/2601.4	19/1968.9†
10^{-1}	$15 \times 15 \times 15$	20/1.0	21/1.1	21/1.5
	$45 \times 45 \times 45$	34/89.3	22/60.9	21/64.3
	$135 \times 135 \times 135$	65/6313.9	29/2913.9	20/2084.2†
10^{-3}	$15 \times 15 \times 15$	20/1.0	22/1.1	22/1.5
	$45 \times 45 \times 45$	34/85.9	24/64.0	21/61.9
	$135 \times 135 \times 135$	78/7526.7	27/2729.5	19/1970.8†

† *The mass, M, in* (3.5) *is omitted when defining the edge prolongator.*

TABLE 7
AMG multigrid operator complexities for 3D $135 \times 135 \times 135$ *grid,* $\sigma \in \{10^1, 10^{-1}, 10^{-3}\}$.

Method	AMG complexity
$\mathcal{P}^{(e)}(0,0)$	1.04
$\mathcal{P}^{(e)}(\alpha,0)$	1.04
$\mathcal{P}^{(e)}(\alpha,\beta)$	1.04†

† *Note that we omitted the mass M in the third term when defining the edge prolongator.*

term drops with σ. For small σ, these entries can be dropped, and we recover the nonzero pattern of $\mathcal{P}^{(e)}(0,0)$ and $\mathcal{P}^{(e)}(\alpha,0)$. As mentioned in section 4, an alternative to using the true mass term is approximating it with a lumped version. Although we did not use a mass approximation in this experiment, doing so has the advantage that there is no growth in the multigrid operator complexities due to the mass term. (See the final slotted-square problem for results using an approximation to the mass matrix.)

Similar results are given in Table 6 for a 3D cube. In this case, convergence is independent of the mesh size for the W-cycles, the overall multgrid operator complexity is quite low, and the new prolongator $\mathcal{P}^{(e)}(\alpha,\beta)$ results in significant run time improvements compared to the original prolongator $\mathcal{P}^{(e)}(0,0)$. See Table 7 for multigrid operator complexities. We note that in the case of $\mathcal{P}^{(e)}(\alpha,\beta)$ we omitted the

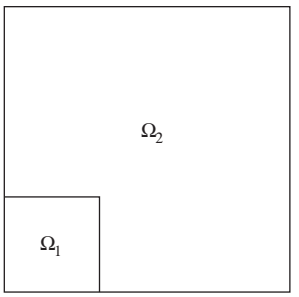


FIG. 5.1. *Domain with two regions of differing conductivity and periodic boundary conditions. In Ω_2 the conductivity is $\sigma_2 = 1.0$. In Ω_1, the conductivity is a constant value, either $\sigma_1 = 10^{-1}$ or 10^{-3}, depending on the experiment.*

TABLE 8
Iteration counts/run times (in seconds) for CG/AMG W(1,1) *schemes for model problem* (1.1) *with two regions of different conductivities. The domain is defined in Figure* 5.1.

σ_1, σ_2	Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
$10^{-1}, 1$	30×30	24/0.1	21/0.1	13/0.1
	90×90	36/1.8	26/1.3	15/1.1
	270×270	62/41.5	31/24.0	15/15.1
	810×810	107/829.8	34/294.9	16/161.1
$10^{-3}, 1$	30×30	24/0.1	21/0.1	14/0.1
	90×90	36/1.8	27/1.4	15/1.1
	270×270	62/41.8	31/23.6	15/16.2
	810×810	107/823.9	35/302.1	16/159.0

mass term, M, from the third term of (3.1); i.e., we used only $\beta T D^{-1} T^T$ (due to difficulty in extracting just the mass term within our software). Therefore, we cannot comment on the effect on multigrid operator complexity if the original M were used. If M is lumped, however, clearly there should be no effect on the multigrid operator complexities. We observe h-independent convergence rates for the model problem even without M.

5.2. AMG with conductivity jumps in a 2D domain. For the second experiment, we consider (1.1) discretized with rectangular elements on the unit square, Ω, with periodic boundary conditions. The domain is split into two regions, Ω_1 and Ω_2. Region Ω_1 is given by $[0, 1/3]^2$, and $\Omega_2 = \Omega \setminus \Omega_1$. In Ω_1, the conductivity is either $\sigma_1 = 10^{-3}$ or $\sigma_1 = 10^{-1}$, depending on the experiment. In Ω_2, the conductivity is $\sigma_2 = 1.0$. The domain is depicted in Figure 5.1.

Results are given for a W-cycle in Table 8. As with the previous experiments, only the $\mathcal{P}^{(e)}(\alpha,\beta)$ prolongator displays h-independence. There is iteration growth in both $\mathcal{P}^{(e)}(0,0)$ and $\mathcal{P}^{(e)}(\alpha,0)$ as the mesh size increases.

5.3. AMG on a nonorthogonal mesh. As our final computational example, we consider a slotted unit square with a mixture of Neumann and Dirichlet boundary conditions. The exact boundary conditions are given in Figure 5.2. Figure 5.3 illustrates the computational domain for a mesh with 1,994 edges. Each sphere represents

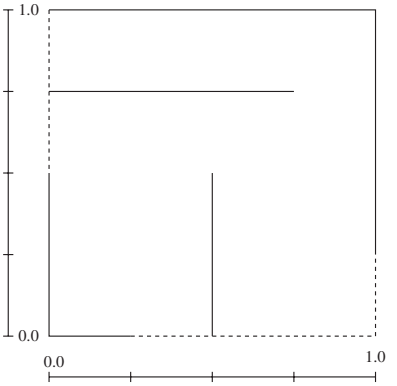


FIG. 5.2. *Boundary conditions for slotted unit square problem. The dashed lines are imposed Neumann conditions, and the solid lines are Dirichlet conditions.*

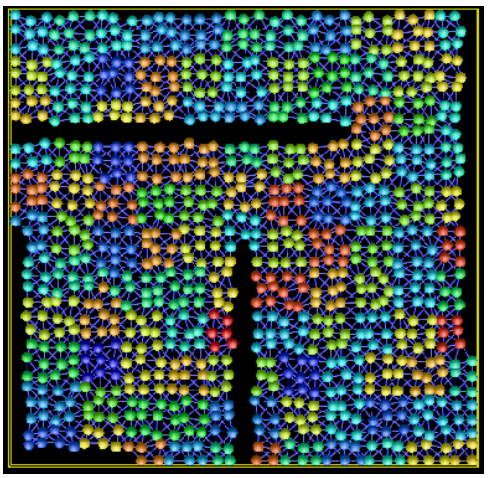


FIG. 5.3. *Visualization of aggregates for a* 33×33 *mesh.*

a nodal grid point. The colors of the spheres give an indication of the nodal aggregates used on the finest level. Due to color limitations in the visualization package, it is difficult to entirely distinguish all the aggregates (especially when two aggregates of similar color are adjacent to each other). However, it is clear that the aggregates are certainly not uniform or regular. For these experiments, the following function is used for the conductivity:

$$\sigma = \frac{\eta}{4000}(5 - \sin(2\pi xy + 3y)e^{(xy-2x)}), \tag{5.2}$$

where η is a scalar and the domain corresponds to $0 \le x \le 1$ and $0 \le y \le 1$. Figure 5.4 shows a plot of (5.2) for $\eta = 10$ on a 33×33 mesh on the unit square. The results given in Table 9 illustrate the convergence for different values of η. These results essentially mirror the other experiments given in this paper. In particular, the only convergence rates that are nearly mesh independent correspond to $\mathcal{P}^{(e)}(\alpha, \beta)$.

In these experiments, the mass matrix has many nonzeros (almost as many as the stiffness matrix). This is due to the nonorthogonality of the mesh. To avoid

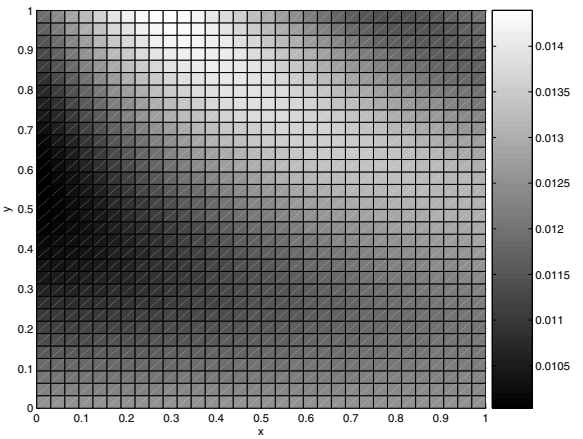


FIG. 5.4. *Variation in σ, as given by* (5.2)*, for $\eta = 10$ on a 33×33 mesh on the unit square.*

TABLE 9
Iteration counts/run times (in seconds) for CG/AMG W(1,1) *schemes for slotted problem. The conductivity is given by* (5.2).

η	Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
10^1	1,994	24/0.09	16/0.07	15/0.06
	19,826	45/1.1	21/0.6	16/0.5
	196,643	86/20.0	27/6.6	17/4.4
	714,162	139/125.6	32/30.6	17/17.0
10^{-1}	1,994	24/0.09	16/0.06	15/0.06
	19,826	45/1.2	21/0.6	16/0.4
	196,643	86/20.0	27/6.6	17/4.5
	714,162	139/125.1	32/30.4	17/16.9
10^{-3}	1,994	24/0.09	16/0.06	16/0.07
	19,826	43/1.1	21/0.6	15/0.4
	196,643	116/26.7	37/8.9	18/4.6
	714,162	133/120.2	34/32.0	17/17.0

complexity growth, a simple diagonal approximation to the mass matrix is used in (3.5). This diagonal is obtained by simply summing the absolute values of the entries within each row. While this is relatively crude, it seems to be sufficient for these problems. In particular, the diagonal approximation ensures that the multigrid complexity is independent of the mass term, and therefore σ, as discussed in section 4. Multigrid complexities are given in Table 10. These complexities are very similar to those observed in the previous experiments.

6. Conclusions. We have proposed a new AMG grid transfer for Maxwell's equations. This operator depends on two parameters that are easy to calculate. Both the Reitzinger and Schöberl intergrid transfers and the smoothed intergrid transfers in [3] are special instances of this new operator. Numerical experiments on model

TABLE 10
Multigrid complexities for slotted problem. Complexities are independent of conductivity.

Grid size	$\mathcal{P}^{(e)}(0,0)$	$\mathcal{P}^{(e)}(\alpha,0)$	$\mathcal{P}^{(e)}(\alpha,\beta)$
1,994	1.12	1.17	1.18
19,826	1.13	1.16	1.16
196,643	1.13	1.13	1.14
714,162	1.13	1.13	1.13

problems show that an AMG method based on this new operator has superior iteration counts and run times to previous operators. In certain cases, the AMG method has h-independent convergence rates. In general, the new AMG/CG method demonstrates improved convergence behavior and has only a slight growth in the number of iterations as the mesh size increases. Future research includes analyzing the behavior of this new method on systems with mass matrices that have coefficient jumps or that have large variation in conductivity.

REFERENCES

[1] M. F. ADAMS, M. BREZINA, J. J. HU, AND R. S. TUMINARO, *Parallel multigrid smoothing: Polynomial versus Gauss-Seidel*, J. Comput. Phys., 188 (2003), pp. 593–610.

[2] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in H(div) and H(curl)*, Numer. Math., 85 (2000), pp. 197–217.

[3] P. B. BOCHEV, C. J. GARASI, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[4] P. B. BOCHEV, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *Towards robust 3D Z-pinch simulations: Discretization and fast solvers for magnetic diffusion in heterogeneous conductors*, Electron. Trans. Numer. Anal., 15 (2003), pp. 186–210.

[5] A. BOSSAVIT, *A rationale for "edge-elements" in 3-D fields computations*, IEEE Trans. Magnetics, 24 (1988), pp. 74–79.

[6] J. BRAMBLE, J. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.

[7] A. BRANDT, *Multigrid Techniques:* 1984 *Guide with Applications to Fluid Dynamics*, Technical report 85, GMD-Studie, Sankt Augustin, Germany, 1984.

[8] W. L. BRIGGS, V. E. HENSON, AND S. MCCORMICK, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.

[9] W. HACKBUSCH, *Multigrid Methods and Applications*, Springer Ser. Comput. Math. 4, Springer-Verlag, Berlin, 1985.

[10] R. HIPTMAIR, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[11] S. REITZINGER AND J. SCHÖBERL, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223–238.

[12] U. TROTTENBERG, C. OOSTERLEE, AND A. SCHÜLLER, *Multigrid*, Academic Press, London, 2001.

[13] P. VANĚK, M. BREZINA, AND J. MANDEL, *Convergence of algebraic multigrid based on smoothed aggregation*, Numer. Math., 88 (2001), pp. 559–579.

[14] P. VANĚK, J. MANDEL, AND M. BREZINA, *Algebraic multigrid based on smoothed aggregation for second and fourth order problems*, Computing, 56 (1996), pp. 179–196.

[15] P. S. VASSILEVSKI AND J. WANG, *Multilevel iterative methods for mixed finite element discretizations of elliptic problems*, Numer. Math., 63 (1992), pp. 503–520.

[16] J. S. VAN WELIJ, *Calculation of eddy currents in terms of H on hexahedra*, IEEE Trans. Magnetics, MAG-21 (1985), pp. 2239–2241.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1689–1708
© 2006 Society for Industrial and Applied Mathematics

A MULTIGRID METHOD FOR VARIABLE COEFFICIENT MAXWELL'S EQUATIONS*

J. JONES† AND B. LEE‡

Abstract. This paper presents a multigrid method for solving variable coefficient Maxwell's equations. The novelty in this method is the use of interpolation operators that do not produce multilevel commutativity complexes that lead to multilevel exactness. Rather, the effects of multilevel exactness are built into the level equations themselves—on the finest level using a discrete $T-V$ formulation and on the coarser grids through the Galerkin coarsening procedure of a $T-V$ formulation. These built-in structures permit the levelwise use of an effective hybrid smoother on the curl-free near-nullspace components and permit the development of interpolation operators for handling the curl-free and divergence-free error components separately. The resulting block-diagonal interpolation operator does not satisfy multilevel commutativity but has good approximation properties for both of these error components. Applying operator-dependent interpolation for each of these error components leads to an effective multigrid scheme for variable coefficient Maxwell's equations, where multilevel commutativity-based methods can degrade. Numerical results are presented to verify the effectiveness of this new scheme.

Key words. Maxwell's equations, multigrid method, edge finite elements

AMS subject classifications. 65N55, 65N30

DOI. 10.1137/040608283

1. Introduction.

In recent years, there has been substantial interest in solving the curl-curl formulation of Maxwell's equations for the electric or magnetic field. This formulation commonly arises in the time-domain and frequency-domain approaches for solving time-dependent electromagnetic problems. For both approaches, the challenges in developing an efficient multigrid algorithm are formidable: the time-domain approach leads to variable coefficient equations with large near-nullspaces, as with the frequency-domain approach but with the added challenge of being indefinite. In this paper, we consider only the definite curl-curl formulation.

There exist several successful geometric [16], [2] and algebraic [20], [5] multigrid methods for solving the definite curl-curl equations. But the performance of these methods degrades as the variation of the material coefficients increases. The reason for this degradation can be traced back to the interpolation operator. For geometric schemes, even with nested finite element spaces, the natural finite element interpolation fails when the coefficients strongly vary much the same way the natural finite element interpolation fails for scalar diffusion equations with rapidly varying coefficients; i.e., finite element interpolation leads to simple arithmetic averaging of the coefficients on the coarser grids. For algebraic schemes, since the interpolation operator is constructed with the constraint that a multilevel commutativity complex is

*Received by the editors May 14, 2004; accepted for publication (in revised form) February 7, 2005; published electronically February 3, 2006. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.
http://www.siam.org/journals/sisc/27-5/60828.html
†Department of Mathematical Sciences, Florida Institute of Technology, 150 West University Boulevard, Melbourne, FL 32901-6975 (jim@fit.edu).

‡Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA (lee123@llnl.gov). This author's work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract W-7405-Eng-48.

formed (see (3.4)), divergence-free error components can be untouched or even amplified by this type of interpolation. In particular, interpolation operators constructed in this manner can handle at most only the curl-free error components, and thus, only half of the Helmholtz decomposition of the error. Thus, current geometric and algebraic multigrid methods do not satisfy the approximation property when the material coefficients vary substantially.

Satisfying the approximation property is not the only difficulty. For Maxwell's equations, the smoothing property is hard to obtain. For this system PDE, near-nullspace components occur on all grid scales, and thus, relaxation must be designed to handle error components of these form on all grid scales. This is in fact the purpose of having a multilevel commutativity complex. Such complex permits easy movement between two-term exact sequences on different grid levels. With a discretization that preserves this two-term exact sequence, nullspace components of the curl operator are explicitly known, and hence, hybrid smoothers [16], [2] can be constructed to effectively handle curl-free error components. Having this two-term sequence on each level then guarantees level smoothers that eliminate curl-free error of any grid scale, and having a multilevel commutativity complex guarantees that curl-free coarse grid corrections are indeed curl-free on the finer grid. But ideally, as previously explained, one would like a multilevel commutativity complex that spans over an exact sequence that also includes the divergence operator. This would involve developing separate interpolation operators for the curl-free and divergence-free components, which implies the existence of an exact discrete Helmholtz decomposition. Or, if an exact discrete Helmholtz decomposition does not exist, one needs an interpolation operator that acts appropriately (i.e., has good approximation) on both the curl-free and divergence-free components. Such an interpolation operator is unfortunately extremely difficult to construct.

The challenge in developing a multigrid scheme for variable coefficient curl-curl equations is then constructing interpolation operators that have good approximation property for both curl-free and divergence-free errors and such that the coarse grid problems constructed using these operators have the relevant effects of two-term exact sequences in order for a hybrid smoother or overlapping Schwarz smoother to be effective on coarser levels. Rather than constructing interpolation operators that satisfy these constraints, in this paper we develop a multigrid scheme that relaxes on the multilevel commutativity constraint and hence has more freedom in the construction of interpolation operators to handle the curl-free and divergence-free error components. This method involves a discrete $T - V$ formulation [19] on the finest level. The construction of this $T - V$ formulation can be obtained from the given curl-curl formulation matrix and a discrete gradient on the finest level.

The goal of this formulation is to introduce the nullspace of the curl operator, i.e., gradients, explicitly into the set of equations. This leads to a discrete system for the curl-free and approximate divergence-free components of the solution, which corresponds to a system PDE for these solution components. Separate operator-based interpolation operators are then constructed, using the original curl-curl equations and the Laplace operator derived from the introduction of the gradients. These block-diagonal interpolation operators are used in the Galerkin coarsening procedure to generate coarse grid problems for the curl-free and divergence-free components of the error. Thus, at all levels, the systems are blocked 2×2, with the diagonal blocks describing the coupling within the curl-free or divergence-free components and with the off-diagonal blocks describing their coupling. Now equations for the curl-free or near-nullspace gradients are explicitly available at all levels, so a hybrid smoother can be applied at each level.

It must be noted that the $T-V$ formulation is introduced discretely only to obtain an efficient multigrid solver. The target continuous curl-curl formulation of Maxwell's equations is not explicitly replaced by a continuous semidefinite $T-V$ formulation. Also, it must be noted that the above scheme resembles Griebel's subspace correction method [14]. However, unlike Griebel's scheme, which creates linearly dependent generating functions by collecting the finite element basis functions on all levels, the above scheme considers the "generating functions" from a Helmholtz decomposition perspective at each level. Indeed, the Helmholtz decomposition perspective gives guidance on developing effective interpolation operators and relaxation schemes, and is much more insightful for $H(\mathrm{div})$ and $H(\mathrm{curl})$ system PDEs than a general subspace correction perspective. In fact, the $T-V$ formulation exposes the extent to which the components of the Helmholtz decomposition can be coarsened separately.

This paper is organized as follows. In section 2 we introduce the curl-curl formulation of the definite Maxwell's equations, the functional setting for the variational problem, and the finite element spaces for the discretization. In section 3, because of their importance in understanding solution methods for the curl-curl formulation, we review the de Rham complex for the curl-curl formulation and the multilevel commutativity diagrams connecting the complexes on different levels. In particular, because the multigrid scheme developed in this paper can be viewed as a generalization of Hiptmair's approach [16] that does not satisfy multilevel commutativity, we need to examine what multilevel commutativity achieves and how much of it is needed in a multigrid solver. In fact, one purpose of this paper is to "isolate" de Rham sequences and commutativity from multigrid. To achieve this, in section 3 we will distinguish between multilevel commutativity and multilevel de Rham complexes and show that multigrid schemes with commutativity-based interpolation operators generally enforce only exactness, whereas multilevel de Rham complexes are really needed for multigrid effectiveness. In section 4 we describe our $T-V$ formulation and the operator-based interpolation operators to handle variable coefficients. An intuitive justification for our choice of operator-based interpolation operators is presented. In particular, since the $T-V$ system is used in the multigrid procedure, we examine the near-nullspace components of this system to show that operator collapsing [10], [11], [12] can be used to handle the curl-free error components and that AMGe element agglomeration [9] can be used to handle the remaining error components. Finally, in section 5 we present some preliminary numerical results illustrating the effectiveness of this multigrid scheme and, equally important, illustrating no need for multilevel commutativity complexes.

2. Curl-curl formulation in the electric field. Let $\Omega \times T$ be the Cartesian product of a bounded simply connected domain $\Omega \in \Re^n$, $n = 2, 3$, and a nonnegative time interval T. To guarantee appropriate regularity of the problem, let Ω have a smooth or polygonal boundary Γ. Electromagnetic phenomena in $\Omega \times T$ can be described by the differential equations

$$\begin{aligned} \nabla \times \mathbf{E} &= -\tfrac{\partial \mathbf{B}}{\partial t} \quad \text{in } \Omega \times T, \\ \nabla \times \mathbf{H} &= \tfrac{\partial \mathbf{D}}{\partial t} + \mathbf{j} \quad \text{in } \Omega \times T \end{aligned} \tag{2.1}$$

with the linear material constitutive relations [19]

$$\begin{aligned} \mathbf{D} &= \epsilon \mathbf{E} \quad \text{in } \Omega \times T, \\ \mathbf{B} &= \mu \mathbf{H} \quad \text{in } \Omega \times T, \\ \mathbf{j} &= \sigma \mathbf{E} \quad \text{in } \Omega \times T. \end{aligned} \tag{2.2}$$

The equations in (2.1) are Faraday and Ampere's laws of Maxwell's equations, and the third equation in (2.2) is a simple Ohm's law. Here, $\mathbf{E}$, $\mathbf{D}$, $\mathbf{H}$, $\mathbf{B}$, and $\mathbf{j}$ are, respectively, the electric field, electric flux, magnetic field, magnetic flux, and electric current; and ϵ, μ, and σ are, respectively, the electric permittivity, magnetic permeability, and electric conductivity, which we assume all to be spatially dependent but independent of t. Using (2.1) and (2.2), we have

$$\nabla \times \frac{1}{\mu} \nabla \times \mathbf{E} + \epsilon \frac{\partial^2 \mathbf{E}}{\partial^2 t} + \sigma \frac{\partial \mathbf{E}}{\partial t} = \mathbf{0}.$$

Differencing the time derivatives and, for simplicity, assuming that the boundary surface is perfectly conducting, we obtain a boundary-value problem of the form

$$\begin{aligned} \nabla \times \alpha \nabla \times \mathbf{E} + \beta \mathbf{E} &= \mathbf{f} \quad \text{in } \Omega, \\ \mathbf{n} \times \mathbf{E} &= \mathbf{0} \quad \text{on } \Gamma, \end{aligned} \tag{2.3}$$

where α and β are positive functions (in $L_\infty(\Omega)$). The parameter β is related to the time step size ($\beta = \frac{\alpha}{\Delta t}$) and may be quite small if large time steps are taken. We will call (2.3) the *curl-curl formulation* for the electric field. At each time step of the solution process for the time-dependent problem, an equation of the form (2.3) must be solved.

To describe the variational formulation of (2.3), several functional spaces are needed. Thus, we will denote the usual kth-order Sobolev spaces by $H^k(\Omega)$ and their homogeneous trace subspaces by $H_0^k(\Omega)$. We also need spaces

$$\begin{aligned} H(\mathrm{div}; \Omega) &= \left\{ \mathbf{v} \in [L^2(\Omega)]^n : \nabla \cdot \mathbf{v} \in L^2(\Omega) \right\}, \\ H(\mathrm{curl}; \Omega) &= \left\{ \mathbf{v} \in [L^2(\Omega)]^n : \nabla \times \mathbf{v} \in [L^2(\Omega)]^{2n-3} \right\}. \end{aligned}$$

We denote their homogeneous trace subspaces by $H_0(\mathrm{div}; \Omega)$ and $H_0(\mathrm{curl}; \Omega)$, respectively. Now, the variational formulation of (2.3) is to find $\mathbf{E} \in H_0(\mathrm{curl}; \Omega)$ such that

$$(\alpha \nabla \times \mathbf{E}, \nabla \times \mathbf{v}) + (\beta \mathbf{E}, \mathbf{v}) = (\mathbf{f}, \mathbf{v}) \qquad \forall \mathbf{v} \in H_0(\mathrm{curl}; \Omega). \tag{2.4}$$

To discretize the variational problem (2.4), the lowest-order Nedelec finite elements are used. Let $\mathcal{T}_h := \{T_i\}$ be a quasi-uniform and shape-regular triangulation of Ω with mesh size h [8]. The lowest-order local Nedelec finite element space is

$$\mathcal{ND}(T_i) := \{ \mathbf{v} = \mathbf{a} + \mathbf{b} \times \mathbf{x};\ \mathbf{x} \in T_i,\ \mathbf{a}, \mathbf{b} \in \Re^n \}.$$

The degrees of freedom are the tangential components along the edges of T_i; along the lth edge e_l,

$$\int_{e_l} \mathbf{t} \cdot \mathbf{v}\, ds.$$

The global finite element space is

$$\mathcal{ND}(\mathcal{T}_h) := \left\{ \mathbf{v}_h \in H_0(\mathrm{curl}; \Omega) : \mathbf{v}_{h|T_i} \in \mathcal{ND}(T_i)\ \ \forall T_i \in \mathcal{T}_h \right\}.$$

This choice of finite element space guarantees continuity of the tangential component of $\mathbf{v}_h$. With $\mathbf{E}_h$ denoting the discrete approximation to the solution of (2.4), the discrete variational problem is to find $\mathbf{E}_h \in \mathcal{ND}(\mathcal{T}_h)$ such that

$$(\alpha \nabla \times \mathbf{E}_h, \nabla \times \mathbf{v}_h) + (\beta \mathbf{E}_h, \mathbf{v}_h) = (\mathbf{f}, \mathbf{v}_h) \qquad \forall \mathbf{v}_h \in \mathcal{ND}(\mathcal{T}_h). \tag{2.5}$$

We also will need the standard first-order scalar Lagrange finite element space

$$\mathcal{H}_h(\mathcal{T}_h) := \left\{ v \in H_0^1(\Omega) : \, v_{|T_i} \in \mathcal{P}_1(T_i) \;\; \forall T_i \in \mathcal{T}_h \right\}$$

with the usual nodal degrees of freedom. Note that the gradient of an element of $\mathcal{H}_h(\mathcal{T}_h)$ is not only in $\mathcal{ND}(\mathcal{T}_h)$ but also in the nullspace of the curl operator. In fact, the assumption of simple-connectedness of Ω guarantees that the nullspace of the curl operator consists only of gradients. By cohomology theory, this is also true in the triangulated domain [7]. (The algorithm of this paper, as with other multigrid algorithms for solving (2.5), requires simple-connectedness of Ω. For domains such as tori, an additional procedure is needed to handle special troublesome subspaces of low dimension. These subspaces can be treated on a coarse grid [17].)

3. De Rham and commutativity complexes and multigrid. The multigrid scheme we will develop is a generalization of the algorithm described in [16]. But a major difference between these two schemes is that the method described in this paper does not produce multilevel commutativity complexes even when the levels are nested, whereas the method described in [16] does. Because of this, we explore de Rham and commutativity complexes and multigrid in this section. We examine whether multigrid efficiency is really a result of these multilevel complexes. In particular, our goal in this section is to isolate commutativity from the multigrid process and to show that effective multilevel complexes are too difficult to produce for variable coefficient Maxwell's equations. This will then lead to a simpler scheme, described in section 4.

Before reviewing de Rham and commutativity complexes, we outline the algorithm of [16] in order to have a base method to illustrate multilevel complexes. This algorithm has the standard multigrid structure:

Given an initial guess $\mathbf{E}^L$ and a right-hand side $\mathbf{f}^L$ on the finest level L

MG(l, $\mathbf{E}^l$, $\mathbf{f}^l$) Cycle:

1. if $l = 0$, the coarsest level, solve exactly, $\mathbf{E}^0 \leftarrow [A_{ee}^0]^{-1}\mathbf{f}^0$
2. else,
   a. presmooth $\mathbf{E}^l \leftarrow S_l(\mathbf{E}^l, \mathbf{f}^l)$
   b. $\mathbf{E}^{l-1} \leftarrow \mathbf{0}$
   c. MG($l-1$, $\mathbf{E}^{l-1}$, $[P_{l-1}^l]^t(\mathbf{f}^l - A_{ee}^l\mathbf{E}^l)$)
   d. coarse grid correction $\mathbf{E}^l \leftarrow \mathbf{E}^l + P_{l-1}^l\mathbf{E}^{l-1}$
   e. postsmooth $\mathbf{E}^l \leftarrow S_l(\mathbf{E}^l, \mathbf{f}^l)$,

where A_{ee}^l is the lth level discretized edge operator and S_l and P_{l-1}^l are, respectively, the lth level smoothing and interpolation operators. However, the smoother is nonstandard:

$S_l(\mathbf{E}^l, \mathbf{f}^l)$ Sweep:

1. Gauss–Seidel sweep on edge equations $A_{ee}^l\mathbf{E}^l = \mathbf{f}^l$
2. edge residual computation, $\mathbf{r}^l \leftarrow \mathbf{f}^l - A_{ee}^l\mathbf{E}^l$
3. transfer to scalar "potential" space (or set up of distributive relaxation), $\mathbf{w}^l \leftarrow [G_{en}^l]^t\mathbf{r}^l$
4. Gauss–Seidel sweep on scalar potential equation $A_{nn}^l\phi^l = \mathbf{w}^l$
5. potential solution correction $\mathbf{E}^l$, $\mathbf{E}^l \leftarrow \mathbf{E}^l + G_{en}^l\phi^l$,

where G_{en}^l is the level l discrete gradient operator mapping a node-based scalar potential into the Nedelec element space and A_{nn}^l is a discrete node-based diffusion operator. When the level edges are nested and the P_{l-1}^l's are the natural Nedelec finite element interpolation operators, then the multilevel de Rham and commutativity complexes are hidden in the interpolation and discrete gradient operators. When the level edges are nonnested, as generally is the case for unstructured grids, the P_{l-1}^l's

can be constructed so that a multilevel commutativity complex is formed (see [20] and [5]).

So, what are de Rham complexes? De Rham complexes describe a relationship between the gradient, curl, and divergence operators over appropriate spaces. They are an amazing tool for systematically describing stable discretizations of PDE systems [1]. Given a scalar or system of PDEs, subtle geometric properties of the PDEs can be exposed by de Rham sequences. These properties should be retained in the discretization for the sake of numerical stability. For the Maxwell's equations, which have intrinsic topological properties reflective of de Rham complexes [3], these complexes have been used to explain the stability of Nedelec finite elements for the curl-curl formulation [6], [1].

In the continuum, the de Rham complex of interest for Maxwell's equations is

$$H^1(\Omega) \xrightarrow{\nabla} H(\mathrm{curl};\Omega) \xrightarrow{\nabla\times} H(\mathrm{div};\Omega). \tag{3.1}$$

Here, we have a sequence of vector spaces defined on our domain Ω and a sequence of operators interrelating them. These vector spaces are *exact* in the sense that the range of an operator defined on the space to its left is in the kernel of the operator and domain space to its right—e.g., the kernel of $\nabla\times$ defined on $H(\mathrm{curl};\Omega)$ is $\nabla H^1(\Omega)$. Let $\{H_h^1,\, H_h(\mathrm{curl}),\, H_h(\mathrm{div})\}$ and $\{S_{2h},\, W_{2h},\, V_{2h}\}$ be two sets of finite elements that need not be nested and of which the first set forms a discrete analogue to (3.1):

$$H_h^1(\Omega) \xrightarrow{\nabla^h} H_h(\mathrm{curl};\Omega) \xrightarrow{\nabla\times^h} H_h(\mathrm{div};\Omega). \tag{3.2}$$

Of course, exactness means that the finite element spaces in the first set cannot be chosen separately. In our case, choosing the lowest-order Nedelec elements ($\mathcal{ND}_h$) for $H_h(\mathrm{curl};\Omega)$, the other finite element spaces are the first-order Lagrangian elements ($\mathcal{H}_h$) and lowest-order Raviart–Thomas elements ($\mathcal{RT}_h$) for $H_h^1(\Omega)$ and $H_h(\mathrm{div};\Omega)$. This consistent choice of finite element spaces is in fact a geometric property (e.g., for the above multigrid algorithm, $\mathbf{E}^l$ is defined on edges ($\mathcal{ND}_h$) and ϕ^l is defined on nodes ($\mathcal{H}_h$)). This shows that a discrete de Rham sequence is not simply a discretization of the functional spaces involved but a "discretization" of a calculus defined over a smooth manifold and its submanifolds [4], [13], [15]. In particular, a discrete de Rham sequence satisfies three properties. It is (i) a discretization of a smooth n-dimensional manifold $\mathcal{M}$ with so-called p-cubes or Euclidean simplices; (ii) a discretization of the *p-forms* defined on the manifold and its submanifolds; and (iii) a discretization of a so-called exterior differential operator so that a generalized Stokes theorem holds over the manifold and its submanifolds (this gives the exactness property). These discretizations are constructed so that the p-forms are simple, unisolvent, conforming, etc. [15]. In our case, $\mathcal{M}$ is a domain in $\Re^3$, the p-cubes are points, edges, and faces, and the p-forms are the above finite element spaces. The location of the degrees of freedom for these finite element spaces indeed reflects a *consistent* choice of simplices to guarantee exactness.

Now, denoting the induced finite element interpolation operators of $\mathcal{H}_h$, $\mathcal{ND}_h$, and $\mathcal{RT}_h$ by $\Pi^h_{\mathcal{H}}$, $\Pi^h_{\mathcal{ND}}$, and $\Pi^h_{\mathcal{RT}}$, the following commutativity complex holds:

$$\begin{array}{ccccc} H^1(\Omega) & \xrightarrow{\nabla} & H(\mathrm{curl};\Omega) & \xrightarrow{\nabla\times} & H(\mathrm{div};\Omega) \\ \downarrow \Pi^h_{\mathcal{H}} & & \downarrow \Pi^h_{\mathcal{ND}} & & \downarrow \Pi^h_{\mathcal{RT}} \\ \mathcal{H}_h & \xrightarrow{\nabla^h} & \mathcal{ND}_h & \xrightarrow{\nabla\times^h} & \mathcal{RT}_h. \end{array} \tag{3.3}$$

This complex relates a set of continuous spaces to a set of discretizations under a collection of commuting differential and interpolation operators. For the above multigrid algorithm and several others [20], [5], additional hierarchies of commutativity relating the level spaces are needed. These methods require a two-level commutativity complex of the form

$$
\begin{array}{lll}
H^1(\Omega) & \xrightarrow{\nabla} H(\mathrm{curl};\Omega) & \xrightarrow{\nabla\times} H(\mathrm{div};\Omega) \\
\downarrow \Pi^h_{\mathcal{H}} & \downarrow \Pi^h_{\mathcal{ND}} & \downarrow \Pi^h_{\mathcal{RT}} \\
\mathcal{H}_h & \xrightarrow{\nabla^h} \mathcal{ND}_h & \xrightarrow{\nabla\times^h} \mathcal{RT}_h \\
\uparrow P^h_{H^1} & \uparrow P^h_{H(\mathrm{curl})} & \uparrow P^h_{H(\mathrm{div})} \\
S_{2h} & \xrightarrow{\nabla^{2h}} W_{2h} & \xrightarrow{\nabla\times^{2h}} V_{2h}(\mathrm{div}),
\end{array}
\tag{3.4}
$$

where $P^h_{(\cdot)}$ are multilevel interpolation operators that are constructed to satisfy commutativity. Note that the second layer of (3.4) is a discrete de Rham sequence since it satisfies the above three properties. However, depending on the choice of $\{S_{2h}, W_{2h}, V_{2h}\}$, the third layer *only* forms an exact sequence between these sets since it may not satisfy condition (ii). Although multilevel exactness requires a consistent choice of coarse simplices, a multilevel de Rham structure also requires that $\{S_{2h}, W_{2h}, V_{2h}\}$ consist of proper p-forms. To stress this, consider the case when the coarse edges do not nest the fine edges, as is the case for the algorithms of [20] and [5]. Because of this nonnesting, $\{S_{2h}, W_{2h}\}$ consists of *weighted* averages generated by the $P^h_{(\cdot)}$, which are not the proper p-forms for the coarse simplices.

Let us return to the above multigrid algorithm and reflect on these complexes. For the curl-curl formulation of Maxwell's equation, some of the near-nullspace components are gradients, and these occur on all grid scales. Thus, not only must coarse scale gradients be in the range of the interpolation operator, but also relaxation on a level must effectively eliminate gradients of scale equal to the level's mesh size. This latter condition leads to the nonstandard smoother, using the level gradient operator G^l_{en}, in the algorithm at the beginning of this section. Of course, effective elimination of these level gradients by level relaxation assumes that each level operator has a curl-curl form. But all of this is reflective of a multigrid solver built on a multilevel commutativity complex: level exactness guarantees the above smoother to be effective on the level gradients.

A multilevel commutativity complex also guarantees good approximation to coarse scale near-nullspace gradient components. Consider the case where $\{S_{2h}, W_{2h}\}$ is a nested subspace of $\{\mathcal{H}_h, \mathcal{ND}_h\}$ and the $P_{(\cdot)}$'s are the natural finite element interpolation operators. Because of our assumption of quasi-uniformity, finite element approximation theory implies multilevel approximation property for the algebraically smooth eigenfunctions that are gradients. For algebraic multigrid on nonnested finite elements, a multilevel approximation property for the algebraically smooth gradient eigenfunctions also holds. Physically, this seems strange since the coarse degrees of freedom in algebraic multigrid are located on coarse edges that are generally not a subset of fine edges; i.e., physically, electric fields that have zero circulation over a given closed cycle (e.g., coarse cycle) generally do not have zero circulation over closed cycles not contained in that given cycle (e.g., fine cycles). But this "strangeness" can be explained away through the commutativity relation

$$
\nabla^h P^h_{H^1} = P^h_{H(\mathrm{curl})} \nabla^{2h}; \tag{3.5}
$$

i.e., a field with zero circulation on a coarse cycle corresponds to a coarse gradient, whose generating potential can be interpolated to a fine scalar function and then operated on with the fine gradient operator to produce a zero circulation field over the fine cycles. Hence, nonnesting of the coarse edges *does not pose a problem* for approximating gradient near-nullspace components.

Now, given an arbitrary coarse edge function that has a curl-free and a divergence-free component, commutativity interpolation interpolates the curl-free component correctly and achieves this without explicitly using $P^h_{H^1}$. But the fact that the divergence-free component may not be interpolated correctly begs the question of overall multigrid efficiency. We contend that multilevel de Rham sequences are needed to give multigrid efficiency. Suppose that the $\{S_{2h}, W_{2h}\}$ generated by $P^h_{H^1}$ and $P^h_{H(\mathrm{curl})}$ contains the proper discrete p-forms that approximate the continuous p-forms of $\{H^1(\Omega), H(\mathrm{curl};\Omega)\}$; i.e., $P^h_{H^1}$ and P^h_{curl} are the natural finite element interpolation operators. Then the set of coarse finite elements is stable, and by the finite element approximation property for each of these spaces, we have good multigrid approximation property everywhere. For example, let $\mathbf{v}_h$ be any type of near-nullspace component and let $\mathbf{v}$ be the continuous function it approximates. On the fine and coarse grids, by the finite element approximation property we have

$$\|\mathbf{v}-\mathbf{v}_h\|_{\mathrm{curl}} \le Ch \quad \text{and} \quad \|\mathbf{v}-P^h_{\mathrm{curl}}\mathbf{v}_{2h}\|_{\mathrm{curl}} = \|\mathbf{v}-\mathbf{v}_{2h}\|_{\mathrm{curl}} \le Ch,$$

where we used the induced finite element interpolation property of P^h_{curl} in the second inequality. Thus, we have good coarse grid approximation to $\mathbf{v}_h$:

$$\|\mathbf{v}_h - P^h_{\mathrm{curl}}\mathbf{v}_{2h}\|_{\mathrm{curl}} \le Ch.$$

When a multilevel de Rham structure does not exist, $P^h_{H(\mathrm{curl})}$ is generally not Nedelec interpolation, and so

$$\|\mathbf{v} - P^h_{H(\mathrm{curl})}\mathbf{v}_{2h}\|_{\mathrm{curl}}$$

can be large for divergence-free $\mathbf{v}$. Hence,

$$\|\mathbf{v}_h - P^h_{H(\mathrm{curl})}\mathbf{v}_{2h}\|_{\mathrm{curl}}$$

can be large, and multigrid performance degrades.

Except for geometric multigrid, constructing multilevel de Rham complexes is very difficult. Moreover, even if we have multilevel de Rham complexes, when the material coefficients vary substantially, $P^h_{H(\mathrm{curl})}$ should depend on these coefficients, which then may break multilevel commutativity. When multilevel de Rham complexes do not exist, to obtain multilevel efficiency, an interpolation operator that acts like $P^h_{H(\mathrm{curl})}$ on the curl-free components and $P^h_{H(\mathrm{div})}$ on the divergence-free components is needed. Construction of such an operator is also difficult. The fundamental misunderstanding is that multilevel communativity considers only a subspace of the whole space, whereas good multigrid performance requires good handling of the whole space (i.e., good handling of the whole multilevel nodal decomposition [16]). For an efficient multigrid scheme, one needs some explicit type of Helmholtz decomposition on each level, and one needs interpolation operators that give good coarse grid approximation to each part of this decomposition. To achieve both of these, explicitly having nodal (to represent the curl-free vector) and edge (to represent the divergence-free vector) degrees of freedom and separate interpolation operators for each part of the Helmholtz decomposition is desirable. This will require more computer memory and additional computational cost per multigrid cycle, but will give robustness and simplicity. We consider such an algorithm in the next section.

4. $T-V$ formulation. As one can derive from the last section, commutativity interpolation is difficult to construct and gives poor approximation to divergence-free error components. There is some computational reduction in using these operators since nodal interpolation is never performed in the solve phase. In some algebraic multigrid algorithms [20], [5], though, this reduction is only minor since the formation of auxiliary nodal interpolation and coarse grid operators is explicitly done in the setup phase but then discarded afterwards. In this section, we will develop an algorithm that recycles this discarded nodal information in the solve phase. Of course, this will lead to actual nodal structures in the solve phase. But, by keeping these structures, interpolation operators can be easily constructed without the commutativity constraints and can be constructed such that the coarse grid approximation property holds for all required bad components. The motivation behind such an approach is to develop an algorithm that can be easily fitted into the framework of existing multigrid codes.

We note that we could present this algorithm as a subspace iteration, like the one presented in [14]. However, we opt to present it in terms of a so-called $T-V$ formulation [19] since this formulation gives guidance on developing effective interpolation operators. We also note that the $T-V$ formulation is used only in the multigrid solution phase—we do not replace continuous problem (2.3) with a continuous semidefinite $T-V$ formulation. Nevertheless, this multigrid solver iterates over the PDE system of the $T-V$ formulation. Hence, interpolation operators must be based on the nontrivial near-nullspace components of this PDE system. Thus, in this section, we characterize these components. We consider the quantitative form of these near-nullspace components in order to justify our choice of interpolation operators.

Therefore, consider curl-curl equation (2.4) with the solution decomposition

$$\mathbf{E} = \mathbf{E}_1 + \nabla\phi, \qquad \mathbf{E}_1 \in H_0(\mathrm{curl};\Omega),\ \phi \in H_0^1(\Omega),$$

and the following variational problem: find $(\mathbf{E}_1, \phi) \in H_0(\mathrm{curl};\Omega) \times H_0^1(\Omega)$ such that

$$(\alpha\nabla\times(\mathbf{E}_1+\nabla\phi), \nabla\times(\mathbf{v}+\nabla\psi)) + (\beta(\mathbf{E}_1+\nabla\phi),(\mathbf{v}+\nabla\psi)) = (\mathbf{f},(\mathbf{v}+\nabla\psi)) \tag{4.1}$$

for all $\mathbf{v} \in H_0(\mathrm{curl};\Omega)$, $\psi \in H_0^1(\Omega)$. Respectively denoting the basis functions of $H_0(\mathrm{curl};\Omega)$ and $H_0^1(\Omega)$ as $\{\mathbf{v}_l\}$ and $\{\psi_i\}$, (4.1) can be written as a system variational problem

$$\begin{bmatrix} A_{ee} & A_{en} \\ A_{ne} & A_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{E}_1 \\ \phi \end{bmatrix} = \begin{bmatrix} \mathbf{f}_e \\ \mathbf{f}_n \end{bmatrix}, \tag{4.2}$$

where

$$A_{ee} = [(\alpha\nabla\times\mathbf{v}_l, \nabla\times\mathbf{v}_m) + (\beta\mathbf{v}_l,\mathbf{v}_m)], \quad A_{en} = [(\beta\mathbf{v}_l, \nabla\psi_j)],$$
$$A_{ne} = [(\beta\nabla\psi_i,\mathbf{v}_m)], \quad A_{nn} = [(\beta\nabla\psi_i,\nabla\psi_j)],$$

and

$$\begin{bmatrix} \mathbf{f}_e \\ \mathbf{f}_n \end{bmatrix} = \begin{bmatrix} (\mathbf{f},\mathbf{v}_l) \\ (\mathbf{f},\nabla\psi_i) \end{bmatrix}$$

and for all $(\mathbf{v}_l,0)^t, (\mathbf{0},\psi_i)^t \in H_0(\mathrm{curl};\Omega)\times H_0^1(\Omega)$. To examine the near-nullspace component $(\tilde{\mathbf{E}},\tilde{\phi})^t$ of (4.2), let $\tilde{\mathbf{E}}$ have the Helmholtz decomposition

$$\tilde{\mathbf{E}} = \tilde{\mathbf{E}}_d + \tilde{\mathbf{E}}_c,$$

where $\tilde{\mathbf{E}}_d$ is divergence-free and $\tilde{\mathbf{E}}_c$ is curl-free. We consider first the quantitative form of two special near-nullspace components and then the quantitative form of a general near-nullspace component. The two special forms are $(\tilde{\mathbf{E}}, \tilde{\phi})^t$ where $\tilde{\mathbf{E}}$ is curl-free and $\tilde{\mathbf{E}}$ is divergence-free.

For near-nullspace component with curl-free $\tilde{\mathbf{E}}$, $\tilde{\mathbf{E}} = \tilde{\mathbf{E}}_c = \nabla\eta$, let us determine the corresponding nodal component $\tilde{\phi}$. From the second equation of (4.2), the nodal component must satisfy

$$(\beta\nabla\tilde{\phi}, \nabla\psi_i) \approx -(\beta\nabla\psi_i, \tilde{\mathbf{E}}_c) = -(\beta\nabla\psi_i, \nabla\eta) \qquad \forall i.$$

Thus $\tilde{\phi}$ is approximately $-\eta$. Substituting this into the first equation of (4.2), we have

$$(\alpha\nabla\times\tilde{\mathbf{E}}_c, \nabla\times\mathbf{v}_l) + (\beta\tilde{\mathbf{E}}_c, \mathbf{v}_l) - (\beta\mathbf{v}_l, \nabla\eta) = (\beta\nabla\eta, \mathbf{v}_l) - (\beta\mathbf{v}_l, \nabla\eta) = 0$$

so that $(\tilde{\mathbf{E}}, \tilde{\phi})^t = (\nabla\eta, -\eta)^t$ is indeed a near-nullspace component.

Next, consider the near-nullspace component with divergence-free $\tilde{\mathbf{E}}$, $\tilde{\mathbf{E}} = \tilde{\mathbf{E}}_d$. We will assume that α, β are sufficiently smooth and consider only interior estimates so that boundary terms produced by integration by parts can be omitted (near-nullspace components are locally interior). From the second equation of (4.2) and integration by parts,

$$\begin{aligned} -(\nabla\cdot\beta\tilde{\mathbf{E}}_d, \psi_i) + (\beta\nabla\tilde{\phi}, \nabla\psi_i) &= -(\nabla\beta\cdot\tilde{\mathbf{E}}_d + \beta\nabla\cdot\tilde{\mathbf{E}}_d, \psi_i) + (\beta\nabla\tilde{\phi}, \nabla\psi_i) \\ &= -(\nabla\beta\cdot\tilde{\mathbf{E}}_d, \psi_i) + (\beta\nabla\tilde{\phi}, \nabla\psi_i) \approx 0 \qquad \forall i \end{aligned} \tag{4.3}$$

or

$$(\beta\nabla\tilde{\phi}, \nabla\psi_i) \approx (\nabla\beta\cdot\tilde{\mathbf{E}}_d, \psi_i) \qquad \forall i.$$

If we further assume that β is smooth in the sense that $\|\nabla\beta\| \le \epsilon$, then using elliptic regularity and the Cauchy–Schwarz inequality,

$$\|\tilde{\phi}\|_1 \le C\|\nabla\beta\cdot\tilde{\mathbf{E}}_d\| \le C\|\nabla\beta\|\|\tilde{\mathbf{E}}_d\| \le C\epsilon\|\tilde{\mathbf{E}}_d\|.$$

That is, $\tilde{\phi}$ has small $H1$ energy or, equivalently, is a smooth eigenfunction of the diffusion operator $[\nabla\cdot\beta\nabla]$. Now for the first equation of (4.2), we have

$$(\alpha\nabla\times\tilde{\mathbf{E}}_d, \nabla\times\mathbf{v}_l) + (\beta\tilde{\mathbf{E}}_d, \mathbf{v}_l) \approx -(\beta\nabla\tilde{\phi}, \mathbf{v}_l) \qquad \forall l, \tag{4.4}$$

and because $\tilde{\phi}$ has small $H1$ energy, $\tilde{\mathbf{E}}_d$ has small scaled $H(\mathrm{curl})$ energy in the order of the size of the eigenvalue corresponding to $\tilde{\phi}$; i.e., coercivity of the curl-curl bilinear form implies

$$\|\tilde{\mathbf{E}}_d\|_{H(\mathrm{curl})} \le C\|\beta\nabla\tilde{\phi}\|,$$

and $\tilde{\phi}$ being a smooth eigenfunction implies

$$\|\tilde{\phi}\|_1 \le \|\tilde{\phi}\|_2 \le C\lambda.$$

Therefore, $\tilde{\mathbf{E}}_d$ is a smooth eigenfunction of the operator $[\nabla\times\alpha\nabla\times + \beta I]$. Moreover, using the identity

$$\begin{aligned} \nabla\times\alpha\nabla\times\mathbf{v} &= \nabla\alpha\times(\nabla\times\mathbf{v}) + \alpha\nabla\times\nabla\times\mathbf{v} \\ &= \nabla\alpha\times(\nabla\times\mathbf{v}) - \alpha\boldsymbol{\Delta}\mathbf{v} + \nabla\nabla\cdot\mathbf{v}, \end{aligned}$$

$\tilde{\mathbf{E}}_d$ is a smooth eigenfunction of

$$[-\alpha\boldsymbol{\Delta} + \nabla\alpha \times \nabla \times + \beta I]. \tag{4.5}$$

Thus, quantitative forms of two types of near-nullspace components of (4.2) are

$$\begin{pmatrix} \tilde{\mathbf{E}} \\ \tilde{\phi} \end{pmatrix} = \begin{pmatrix} \nabla\eta \\ -\eta \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} \tilde{\mathbf{E}} \\ \tilde{\phi} \end{pmatrix} = \begin{pmatrix} \text{small scaled } H(\text{curl}) \text{ norm} \\ \text{small scaled } H1 \text{ norm} \end{pmatrix}. \tag{4.6}$$

Multigrid interpolation operators must be designed to approximate at least these components well. In fact, interpolation operators designed to handle these components will also handle other near-nullspace components. Consider the general case,

$$\tilde{\mathbf{E}} = \tilde{\mathbf{E}}_d + \tilde{\mathbf{E}}_c = \tilde{\mathbf{E}}_d + \nabla\eta.$$

Again using the second equation of (4.2) first, integration by parts leads to

$$\begin{aligned} (\tilde{\mathbf{E}}_d + \tilde{\mathbf{E}}_c, \beta\nabla\psi_i) + (\beta\nabla\tilde{\phi}, \nabla\psi_i) &= (\tilde{\mathbf{E}}_d, \beta\nabla\psi_i) + (\nabla\eta, \beta\nabla\psi_i) + (\beta\nabla\tilde{\phi}, \nabla\psi_i) \\ &= -(\nabla\beta \cdot \tilde{\mathbf{E}}_d, \psi_i) + (\beta\nabla(\eta + \tilde{\phi}), \nabla\psi_i) \\ &\approx 0 \end{aligned}$$

or

$$(\beta\nabla(\eta + \tilde{\phi}), \nabla\psi_i) \approx (\nabla\beta \cdot \tilde{\mathbf{E}}_d, \psi_i).$$

Elliptic regularity and the above smoothness assumption on β lead to

$$\|\eta + \tilde{\phi}\|_1 \leq C\epsilon\|\tilde{\mathbf{E}}_d\|.$$

Thus, $(\eta + \tilde{\phi})$ has small scaled $H1$ norm, from which we conclude $\tilde{\phi} \approx -\eta$. Using this result in the first equation of (4.2) then gives

$$(\alpha\nabla \times \tilde{\mathbf{E}}_d, \nabla \times \mathbf{v}_l) + (\beta\tilde{\mathbf{E}}_d, \mathbf{v}_l) \approx 0.$$

Hence, the general near-nullspace components of (4.2) have the form

$$\begin{pmatrix} \tilde{\mathbf{E}} \\ \tilde{\phi} \end{pmatrix} = \begin{pmatrix} \text{small scaled } H(\text{curl}) \text{ norm} + \nabla\eta \\ -\eta \end{pmatrix}.$$

But the components that must be handled well by coarse grid correction are the components that are left after relaxation. Applying a block relaxation on (4.2) for the nodes and edges, η will be smooth after relaxation on the nodes. After relaxation, η will have small scaled $H1$ norm and $\tilde{\mathbf{E}}$ will have small scaled $H(\text{curl})$ norm. Therefore, interpolation will be constructed based on (4.6).

Note that the $\tilde{\mathbf{E}}$ component of the system near-nullspace is the near-nullspace of (2.4). These must be well approximated by $P^h_{H(\text{curl})}$. Note also that when α, β are constants, the divergence-free near-nullspace components are smooth eigenfunctions of $[-\alpha\boldsymbol{\Delta} + \beta I]$. Hence, to obtain good approximation to these near-nullspace components, $P^h_{H(\text{curl})}$ should act like linear interpolation on the edge degrees of freedom. Constructing $P^h_{H(\text{curl})}$ under the commutativity constraints using $P^h_{H^1}$ generally will not attain this property. A special case when this is attained occurs when the

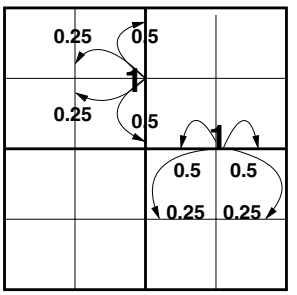


FIG. 1. *Nedelec linear interpolation of horizontal and vertical edges.*

edges are nested and $P^h_{H(\mathrm{curl})}$ is the Nedelec finite element interpolation operator (see Figure 1).

Having determined the quantitative form of the near-nullspace of (4.2), we now develop a multigrid algorithm. First, system (4.2) is discretized over $\mathcal{ND}(\mathcal{T}_h) \times \mathcal{H}_h(\mathcal{T}_h)$, which approximates the continuous near-nullspace well [6]:

$$\begin{bmatrix} A^h_{ee} & A^h_{en} \\ A^h_{ne} & A^h_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{E}^h_1 \\ \psi^h \end{bmatrix} = \begin{bmatrix} \mathbf{f}^h_e \\ \mathbf{f}^h_n \end{bmatrix}. \tag{4.7}$$

From the above discussion, for the curl-free near-nullspace, since $\eta = -\tilde{\phi}$, we need only to approximate the diffusion operator $[-\nabla \cdot \beta \nabla]$ on the coarser grids. Good approximation to this same operator is also required in handling the nodal $\tilde{\phi}$ component of the divergence-free near-nullspace. For the edge component of this divergence-free vector, good approximation to vector "convection"-diffusion operator (4.5) or the curl-curl operator itself is needed on the coarser levels. To accomplish both of these approximation requirements, separate interpolation operators are used for the nodal and edge degrees of freedom. The nodal interpolation P_n will be based on the diffusion operator and the edge interpolation P_e will be based on the curl-curl operator. P_n can be constructed using AMG or BoxMG techniques [10], and P_e can be constructed using AMGe [9] or AMG techniques, since the convection-diffusion operator is being implicitly coarsened. The overall interpolation operator for (4.7) is

$$\begin{bmatrix} P_e & 0 \\ 0 & P_n \end{bmatrix}.$$

The validity of this block-diagonal interpolation operator was based under the assumption that $\|\nabla\beta\|$ is small. When $\|\nabla\beta\|$ is large, a full 2×2 interpolation operator may be needed (see [11], [12] for a numerical comparison of block-diagonal and full block interpolation operators on system PDEs).

With block-diagonal interpolation, the coarse grid operator is formed using Galerkin coarsening:

$$\begin{aligned} \begin{bmatrix} A^{2h}_{ee} & A^{2h}_{en} \\ A^{2h}_{ne} & A^{2h}_{nn} \end{bmatrix} &= \begin{bmatrix} P_e & 0 \\ 0 & P_n \end{bmatrix}^t \begin{bmatrix} A^h_{ee} & A^h_{en} \\ A^h_{ne} & A^h_{nn} \end{bmatrix} \begin{bmatrix} P_e & 0 \\ 0 & P_n \end{bmatrix} \\ &= \begin{bmatrix} P^t_e A^h_{ee} P_e & P^t_e A^h_{en} P_n \\ P^t_n A^h_{ne} P_e & P^t_n A^h_{nn} P_n \end{bmatrix}. \end{aligned} \tag{4.8}$$

Applying this recursively, the coarse grid operator on level i can be constructed.

This algorithm requires the additional computation and storage of the off-diagonal blocks $A_{en}^{2^ih} = (A_{ne}^{2^ih})^t$. However, as the above discussion suggests, the coarse grid problems capture the divergence-free and curl-free components separately, i.e., respectively using $A_{ee}^{2^ih}$ and $A_{nn}^{2^ih}$ only. Hence, omitting the off-diagonal blocks can also lead to an effective scheme. With the off-diagonal blocks, curl-free error components reintroduced when solving the edge correction can be suppressed.

This algorithm also appears to require more user data than just the fine grid stiffness matrix A_{ee}^h and a fine grid discrete gradient operator G_{en}^h : nodes $\rightarrow$ edges as required by commutativity-based multigrid methods. But, as in the commutativity-based methods, A_{nn}^h can be obtained using sparse matrix products and $A_{en}^h = (A_{ne}^h)^t$ can be obtained freely from this same matrix product:

$$A_{nn}^h = (G_{en}^h)^t \left[A_{ee}^h G_{en}^h\right],$$
$$A_{en}^h = A_{ee}^h G_{en}^h.$$

Note that once these operators are formed, discrete gradient operators are not needed on the coarser levels. This apparent minor detail actually implicates some structural differences between this multigrid algorithm and commutativity-based ones. Because these operators are not needed on the coarser grids, there need not be a connection between the coarse nodal and edge grids, although the nodal and edge degrees of freedom are connected through the off-diagonal blocks of (4.8). This implies that the coarse simplices do not have to satisfy a consistency condition; i.e., nodes and edges can be coarsened separately. In turn, this means that a two-term exactness property does not hold on the coarser levels. Even if coarse simplices were consistently chosen, an exactness property may still not hold because the weights generated by the separate interpolation operators lead to discrete differential operators that may not be consistent. All of this should not be surprising because we independently constructed the edge interpolation to capture mainly the divergence-free error and the nodal interpolation to capture the curl-free error, rather than having one edge interpolation constructed to produce exactness to capture the curl-free error. There also should be no concern that the exactness property does not hold on coarser levels because the nodal coarse grid problem defines the equation for the curl-free error; i.e., we have a relaxation procedure for these errors on the coarser levels. With no exactness property on the coarser levels, the multilevel complex for this algorithm is

$$\begin{array}{ccc}
H^1(\Omega) & \xrightarrow{\nabla} & H(\mathrm{curl};\Omega) \\
\downarrow \Pi_{\mathcal{H}}^h & & \downarrow \Pi_{\mathcal{ND}}^h \\
\mathcal{H}_h & \xrightarrow{\nabla^h} & \mathcal{ND}_h \\
\uparrow P_n^h & & \uparrow P_e^h \\
H_{2h}^1 & \overset{A_{en}^{2h}}{\cdots} & H_{2h}(\mathrm{curl}) \\
\vdots & & \vdots \\
H_{2^ih}^1 & \overset{A_{en}^{2^ih}}{\cdots} & H_{2^ih}(\mathrm{curl}),
\end{array}$$

where $H_{2^ih}^1$ and $H_{2^ih}(\mathrm{curl})$ are subspaces of $\mathcal{H}_h$ and $\mathcal{ND}_h$, and $\overset{A_{en}^{2^ih}}{\cdots}$ denotes only a connection between the node and edge degrees of freedom, not an exactness relation.

4.1. Interpolation schemes. There are many well-developed interpolation schemes for variable coefficient problems. In this paper, we will consider only struc-

tured grids and hence employ a BoxMG scheme [10] for P_n and an element agglomeration scheme [9] for P_e.

The BoxMG technique is well known. Let the 9-point stencil of A^h_{nn} be

$$\begin{bmatrix} a^{nw}_{nn} & a^{n}_{nn} & a^{ne}_{nn} \\ a^{w}_{nn} & a^{c}_{nn} & a^{e}_{nn} \\ a^{sw}_{nn} & a^{s}_{nn} & a^{se}_{nn} \end{bmatrix}_{IJ}$$

at coarse node IJ. This technique constructs P_n to satisfy

$$A^h_{nn} P_n \phi^{2h} = 0. \tag{4.9}$$

For fine nodes that underlie a coarse node, the interpolation weight is one; for fine nodes that are horizontally or vertically adjacent to a coarse node, stencil collapsing is used to determine the weights—e.g., for fine nodes that are directly east or west of node IJ, the interpolation weights are, respectively,

$$P^e_n = -\frac{(a^{ne}_{nn} + a^{e}_{nn} + a^{se}_{nn})_{IJ}}{(a^{n}_{nn} + a^{c}_{nn} + a^{s}_{nn})_{IJ}}, \qquad P^w_n = -\frac{(a^{nw}_{nn} + a^{w}_{nn} + a^{sw}_{nn})_{IJ}}{(a^{n}_{nn} + a^{c}_{nn} + a^{s}_{nn})_{IJ}}.$$

The weights for all other fine nodes are obtained using these computed weights and relation (4.9).

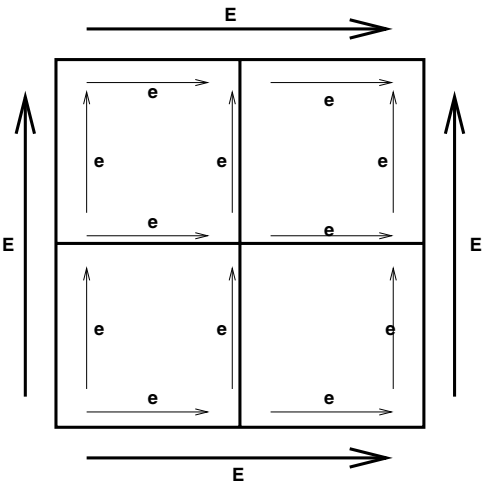


FIG. 2. *Coarse cell agglomerate. Coarse edges are labeled with E and fine edges are labeled with e.*

To describe the AMGe technique to construct P_e, we will refer to the two-dimensional uniform agglomerate shown in Figure 2. We divide the fine edges of this agglomerate into boundary (b) and interior (i) edges, and order them so that locally A^h_{ee} and P_e can be written as

$$A^h_{ee} = \begin{bmatrix} A^{bb}_{ee} & A^{bi}_{ee} \\ A^{ib}_{ee} & A^{ii}_{ee} \end{bmatrix}, \qquad P_e = \begin{bmatrix} P^b_e \\ P^i_e \end{bmatrix}.$$

The boundary edges are the fine edges e that nest a coarse edge E, and the interior edges are the remaining fine edges of the agglomerate. For the boundary edges, the interpolation weights can be defined physically. Since the edge degrees of freedom are voltage drops along the edges, the boundary weights are simply the ratio of the length of the boundary edge to the length of coarse edge that it nests. With these boundary weights, the interior weights are given by the local inversion

$$P_e^i = -[A_{ee}^{ii}]^{-1} A_{ee}^{ib} P_e^b.$$

4.2. Relaxation schemes. Since the equations for both the nodes and edges are Poisson-like, pointwise Gauss–Seidel can be used for each degree of freedom. But as in system PDEs, the sweeping order of relaxation produces different smoothers. One can relax all the nodes of the nodal grid and then all the edges of the edge grid to give the hybrid block Gauss–Seidel scheme of [16]. One can also apply a multiplicative Schwarz smoother similar to that of [2], but where now a node and all the edges branching out of it are solved simultanuously. This is more costly than the pointwise smoother, but it gives robustness. Yet an intermediate ordering is to update a node and then pointwise update all the edges branching out of this node in turn. Since an edge connects two nodes, a given edge is updated more than once in a sweep. Hence, this smoother is more costly than the pointwise hybrid smoother.

5. Numerical experiments. This multigrid algorithm has currently been tested on two-dimensional uniform grids. Three-dimensional problems and algebraic multigrid variants of this algorithm will be presented in a forthcoming paper. The problems solved in this paper are (2.3) on a unit square. Problems with constant and variable α and β are examined, both standard Nedelec interpolation and the operator-based interpolations described in the previous section are used, and the smoothers described in the previous section are employed. (Note that operator-based interpolation produces multilevel structures that do not satisfy exactness.) $V(1,1)$ cycles are used, and all experiments are performed with a zero right-hand side, a random initial guess, and a stopping criterion of twelve-order magnitude reduction of the $l2$ norm of the initial residual. The schemes used are as follows.

- Method 1: Hybrid Gauss–Seidel smoother—all nodes updated, then all edges.
- Method 2: Intermediate Gauss–Seidel smoother—pointwise update one node and then all edges branching out of it in turn.
- Method 3: Node-edge decoupling on coarse grids—i.e., only the block-diagonal terms of (4.8) are used on the coarse grids (see the discussion after (4.8)).
- Method 4: Multiplicative Schwarz smoother—simultaneously solve all edges connected to a node. The nodal degrees of freedom are not involved; this method is used for comparison.

Problem 1: $\alpha = 1$, $\beta =$ constant. This experiment was conducted to illustrate the robustness of the scheme as $\beta \to 0$, so that the operator becomes singular. Table 1 contains the $V(1,1)$ iteration counts. For each iteration column, the left count is for Nedelec interpolation, and the right count is for operator-based interpolation. From these results, all methods are robust with respect to the size of β. In particular, not preserving exactness on coarse grids does not affect convergence. Also, the number of iterations decreases as the smoother becomes "stronger" (Method 2), and Method 3 performs just as well as Method 1 because $\nabla\beta = 0$ so that the edge and nodal components of the divergence-free near-nullspace of (4.2) decouple.

Problem 2: Variable α, β. We consider the performance of this multigrid method on variable coefficient problems. Problem 2a will involve coefficients

TABLE 1
$V(1,1)$-cycle results for Problem 1, $\alpha = 1$, $\beta =$ constant. The iteration counts on the left and right are for Nedelec interpolation and operator-based interpolation, respectively.

β	h	Method 1	Method 2	Method 3	Method 4
10^3	1/16	12/11	8/8	12/13	7/7
	1/32	19/19	12/12	20/19	10/11
	1/64	24/24	14/15	24/24	12/13
	1/128	26/26	15/15	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	26/26	15/15	26/26	13/13
10^2	1/16	17/17	11/12	18/19	10/10
	1/32	23/23	15/15	23/23	12/13
	1/64	26/26	15/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	27/27	15/15	27/27	13/13
10^1	1/16	18/17	12/13	19/19	11/11
	1/32	23/24	15/15	23/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	26/26	15/15	26/26	13/13
10^0	1/16	17/18	13/12	19/18	12/11
	1/32	23/24	15/15	23/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	26/26	15/15	26/26	13/13
10^{-1}	1/16	17/18	12/13	17/17	12/11
	1/32	23/23	15/15	23/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	15/15	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	27/27	15/15	27/27	13/13
10^{-2}	1/16	18/17	13/13	16/17	11/12
	1/32	23/23	15/15	23/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	26/26	15/15	26/26	13/13
10^{-3}	1/16	17/17	13/13	17/18	11/11
	1/32	23/23	15/15	23/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	27/27	15/15	27/27	13/13
10^{-4}	1/16	17/17	13/12	18/17	11/11
	1/32	23/23	15/15	23/23	13/13
	1/64	26/26	15/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	26/26	15/15	26/26	13/13
10^{-5}	1/16	17/16	13/13	17/17	11/11
	1/32	23/23	15/15	24/23	13/13
	1/64	26/26	16/16	26/26	13/13
	1/128	26/26	16/16	26/26	13/13
	1/256	26/26	15/15	26/26	13/13
	1/512	27/27	15/15	27/27	13/13

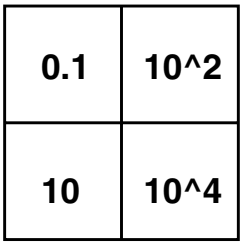


FIG. 3. *Jump C of Problem* 2b.

TABLE 2
$V(1,1)$-cycle results for variable coefficients, Problems 2a *and* 2b.

Problem	h	Method 1	Method 2	Method 3	Method 4
2a	1/16	54/15	48/11	54/17	48/18
	1/32	64/18	58/13	64/20	54/20
	1/64	71/19	59/12	71/20	53/19
	1/128	57/23	53/14	57/23	50/16
	1/256	53/25	49/15	53/25	47/13
	1/512	49/26	45/15	49/26	44/13
2b	1/16	69/13	62/11	94/15	59/44
	1/32	127/15	96/12	126/18	85/26
	1/64	108/18	85/12	110/23	77/45
	1/128	141/23	102/13	140/23	87/43
	1/256	154/25	106/15	153/25	89/31
	1/512	127/26	92/15	127/26	78/21

$$\alpha = (2 + \sin 40\pi x)^2(2 + \cos 40\pi y)^2,$$
$$\beta = (2 + \cos 40\pi x)^2(2 + \sin 40\pi y)^2,$$

and Problem 2b will involve grid-aligned jump coefficients

$$\alpha = C(2 + \sin 40\pi x)^2(2 + \cos 40\pi y)^2,$$
$$\beta = C(2 + \cos 40\pi x)^2(2 + \sin 40\pi y)^2,$$

where jump C is described in Figure 3. The results are tabulated in Table 2. For both problems, we see poor performance when Nedelec interpolation is used, and a dramatic improvement when operator-based interpolation is used. Note that Method 3 performs just as Method 1, which is a bit surprising. Also note that Method 2 performs better than Method 4. Both of these methods do not satisfy exactness on the coarse grids, but only Method 2 has systems for the curl-free component on the coarse grids. Last, note that the performance of Methods 1–3 with operator-based interpolation are unaffected by jumps in the coefficients, whereas coefficient jumps noticeably affect the convergence when Nedelec interpolation is used in these methods.

We also consider an example that demonstrates the importance of using operator-based interpolation in the Galerkin coarsening procedure. In Table 3, we consider Problem 2a again but this time with operator-based interpolation used in the solve phase and Nedelec interpolation used in the Galerkin coarsening procedure. As this table shows, there is almost no improvement in using operator-based interpolation in the solve phase only (compare with the left columns of Table 2, Problem 2a). The coarse grid operators must be formed with these operators also.

Problem 3: Constant coefficient jumps. Our final experiment further examines jump coefficients, but now with jumps that are not grid-aligned on the coarse grids.

TABLE 3
$V(1,1)$-cycle with operator-based interpolation in the solve phase and Nedelec interpolation in Galerkin coarsening, Problem 2a.

h	Method 1	Method 2	Method 3	Method 4
1/16	47	47	47	46
1/32	61	56	61	54
1/64	62	58	62	53
1/128	51	48	51	46
1/256	47	45	47	43
1/512	43	41	43	39

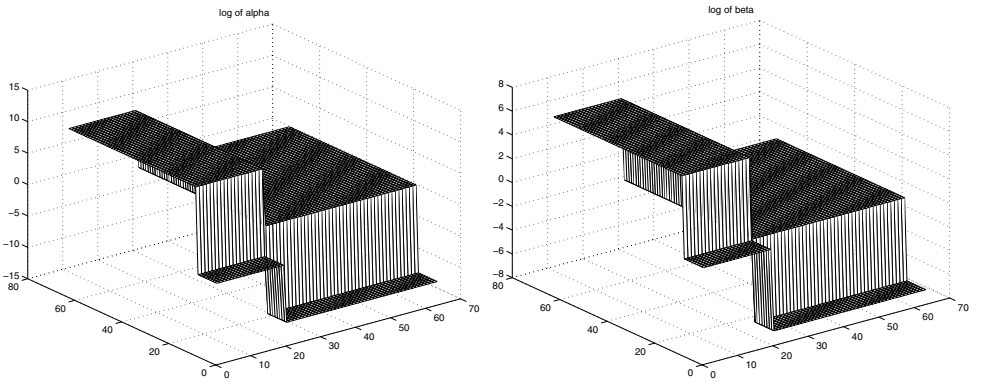


FIG. 4. *Log scale of jumps in α and β for Problem* 3a.

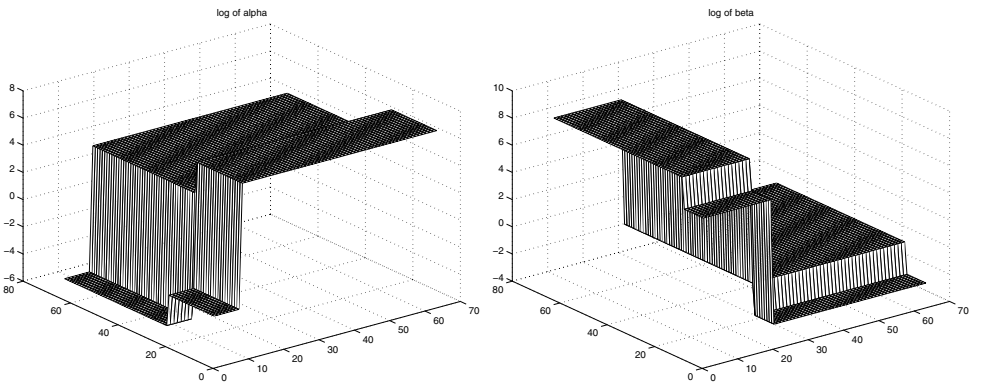


FIG. 5. *Log scale of jumps in α and β for Problem* 3b.

The log scale of the coefficients is shown in Figure 4 (Problem 3a) and Figure 5 (Problem 3b), which has a more dramatic jump pattern for α and β. The order of magnitude of the jumps are around 10. Results are given in Table 4. Again operator-based interpolation performs better than Nedelec interpolation but not as dramatically this time, especially for Problem 3a. This can be accredited to the localization of the coefficient variations. Also, Problem 3b shows Method 1 performing better than Method 3.

6. Conclusions. We presented a multigrid method for solving variable coefficient Maxwell's equations. This method does not construct interpolation operators

TABLE 4
$V(1,1)$-cycle results for "nonaligned" jump coefficients.

Problem	h	Method 1	Method 2	Method 3	Method 4
3a	1/16	86/25	65/26	86/25	48/26
	1/32	37/18	35/11	37/18	29/12
	1/64	31/20	29/13	31/20	24/12
	1/128	31/25	30/15	31/25	26/13
	1/256	27/26	26/15	27/26	23/13
	1/512	28/26	28/15	28/26	25/13
3b	1/16	144/44	148/43	161/60	124/37
	1/32	120/38	122/38	134/46	106/26
	1/64	86/42	87/42	92/46	72/25
	1/128	80/31	81/30	78/36	68/20
	1/256	75/29	75/28	77/33	66/18
	1/512	69/27	68/25	68/30	58/17

based on multilevel commutativity constraints, and hence, does not satisfy exactness on the coarse levels. Rather, interpolation operators are separately developed for the nodes and edges, and operator-based techniques are used to capture the coefficient variations. Moreover, the relevant effects of exactness are obtained by having equations for the curl-free gradients on the coarse levels. Numerical results demonstrate the effectiveness of this algorithm.

REFERENCES

[1] D. N. ARNOLD, *Differential complexes and numerical stability*, in Proceedings of the International Congress of Mathematicians, Vol. 1, Plenary Lectures, Beijing, 2002.

[2] D. N. ARNOLD, R. S. FALK, AND R. WINTHER, *Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$*, Numer. Math., 85 (2000), pp. 197–218.

[3] D. BALDOMIR, *Geometry of Electromagnetic Systems*, Oxford University Press, New York, 1996.

[4] R. L. BISHOP AND S. I. GOLDBERG, *Tensor Analysis on Manifolds*, Dover, New York, 1980.

[5] P. B. BOCHEV, C. J. GARASI, J. J. HU, A. C. ROBINSON, AND R. S. TUMINARO, *An improved algebraic multigrid method for solving Maxwell's equations*, SIAM J. Sci. Comput., 25 (2003), pp. 623–642.

[6] D. BOFFI, *Fortin operator and discrete compactness for edge elements*, Numer. Math., 87 (2000), pp. 229–246.

[7] A. BOSSAVIT, *Computational Electromagnetism: Variational Formulations, Complementarity, Edge Elements*, Academic Press, San Diego, 1998.

[8] S. C. BRENNER AND L. R. SCOTT, *The Mathematical Theory of Finite Element Methods*, Springer, New York, 1994.

[9] M. BREZINA, A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[10] J. E. DENDY, JR., *Black box multigrid*, J. Comput. Phys., 48 (1982), pp. 366–386.

[11] J. E. DENDY, JR., *Black box multigrid for systems*, Appl. Math. Comput., 19 (1986), pp. 57–74.

[12] J. E. DENDY, JR., *Semicoarsening multigrid for systems*, Electron. Trans. Numer. Anal., 6 (1997), pp. 97–105.

[13] H. FLANDERS, *Differential Forms with Applications to the Physical Sciences*, Dover, New York, 1989.

[14] M. GRIEBEL, *Multilevel algorithms considered as iterative methods on semidefinite systems*, SIAM J. Sci. Statist. Comput., 15 (1994), pp. 547–565.

[15] R. HIPTMAIR, *Canonical construction of finite elements*, Math. Comp., 68 (1999), pp. 1325–1346.

[16] R. HIPTMAIR, *Multigrid method for Maxwell's equations*, SIAM J. Numer. Anal., 36 (1998), pp. 204–225.

[17] R. HIPTMAIR, *Multilevel gauging for edge elements*, Computing, 64 (2000), pp. 97–122.

[18] R. HIPTMAIR, *Finite elements in computational electromagnetism*, Acta Numer., 11 (2002),

pp. 237–340.
[19] J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley, New York, 2002.
[20] S. Reitzinger and J. Schoberl, *An algebraic multigrid method for finite element discretizations with edge elements*, Numer. Linear Algebra Appl., 9 (2002), pp. 223–238.
[21] P. S. Vassilevski and J. Wang, *Multilevel iterative methods for mixed finite element discretizations of elliptic problems*, Numer. Math., 63 (1992), pp. 503–520.
[22] H. Whitney, *Geometric Integration Theory*, Princeton University Press, Princeton, NJ, 1957.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1709–1726
© 2006 Society for Industrial and Applied Mathematics

SOLUTION OF EQUILIBRIUM RADIATION DIFFUSION PROBLEMS USING IMPLICIT ADAPTIVE MESH REFINEMENT*

MICHAEL PERNICE† AND BOBBY PHILIP†

Abstract. Diffusion approximations to radiation transport feature a nonlinear conduction coefficient that leads to formation of a sharp front, or Marshak wave, under suitable initial and boundary conditions. The front can vary several orders of magnitude over a very short distance. Resolving the shape of the Marshak wave is essential, but using a global fine mesh can be prohibitively expensive. In such circumstances it is natural to consider using adaptive mesh refinement (AMR) to place a fine mesh only in the vicinity of the propagating front. In addition, to avoid any loss of accuracy due to linearization, implicit time integration should be used to solve the equilibrium radiation diffusion equation. Implicit time integration on AMR grids introduces a new challenge, as algorithmic complexity must be controlled to fully realize the performance benefits of AMR. A Newton–Krylov method together with a multigrid preconditioner addresses this latter issue on a uniform grid. A straightforward generalization is to use a multilevel preconditioner that is tuned to the structure of the AMR grid, such as the fast adaptive composite grid (FAC) method. We describe the resulting Newton–Krylov-FAC method and demonstrate its performance on simple equilibrium radiation diffusion problems.

Key words. adaptive mesh refinement, implicit time integration, Newton–Krylov methods, fast adaptive composite grid, multilevel preconditioning, radiation diffusion

AMS subject classifications. 65H10, 65F10, 65N50, 65N55

DOI. 10.1137/040609069

1. Introduction.

Radiation transport plays an important role in numerous fields of study, including astrophysics, laser fusion, combustion applications, atmospheric dynamics, and medical imaging. When photon mean free paths are much shorter than characteristic length scales, a diffusion approximation provides a reasonably accurate description of radiation penetrating from a hot source to a cold medium. This approximation features a nonlinear conduction coefficient that leads to formation of a sharp front, in which the solution can vary several orders of magnitude over a very short distance. The shape of the front can be very complex as it interacts with different materials having different conduction properties. Resolving these localized features with a global fine mesh can be prohibitively expensive. It is natural to consider reducing the cost of accurately resolving these fronts by using adaptive mesh refinement (AMR), which concentrates computational effort by increasing spatial resolution only locally.

Classical solution techniques for equilibrium radiation diffusion use a linearized conduction coefficient to avoid the expense of solving a system of nonlinear equations at each time step. This introduces a first-order error in time that precludes effective use of higher-order time integration methods, and requires small time steps to maintain time accuracy. Analytic and computational results that demonstrate degradation in time accuracy associated with linearization in the presence of strong nonlinear coefficients can be found in [24, 15]. Such effects can be avoided by using im-

*Received by the editors May 26, 2004; accepted for publication (in revised form) March 29, 2005; published electronically February 3, 2006. This work was supported under the auspices of the U.S. Department of Energy under DOE contract W-7405-ENG-36 at Los Alamos National Laboratory.
http://www.siam.org/journals/sisc/27-5/60906.html
†Computer and Computational Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545 (pernice@lanl.gov, bphilip@lanl.gov).

plicit time integration, which generally requires efficient nonlinear solution techniques to be competitive. Newton–Krylov methods, usually preconditioned by a multigrid method, have been instrumental in demonstrating that this approach is practical for equilibrium radiation diffusion problems [26].

We seek to combine the benefits of improved temporal accuracy of implicit time integration with the improved spatial accuracy that is made possible by AMR. Two important factors must be addressed in order to fully realize the potential efficiency gains. The first is the use of finer grids only in those regions where high resolution is required. This can be accomplished by determining regions where the spatial error is large or where localized solution features warrant enhanced resolution, and immediately translates into lower overall storage costs. The second is the use of algorithms whose arithmetic complexity scales linearly with problem size. This presents a significant challenge when using implicit time integration methods on AMR grids.

Prior efforts in implicit AMR have not adequately addressed the algorithmic scalability issue. Early work by Winkler, Norman, and Mihalas [31] uses r-refinement to solve coupled radiation-hydrodynamics problems. They use a fully coupled formulation, solved by a classical Newton's method, that includes a functional to determine new gridpoint locations for following features of the flow. Their approach is impractical for two- or three-dimensional problems, and even in one dimension difficulties with gridpoint movement are encountered. We use h-refinement, in particular structured AMR (SAMR), to avoid these problems. Both Trompert and Verwer [29, 30] and Li, Petzold, and Hyman [17] use incomplete factorization to precondition their linear solvers. Incomplete factorization can be expensive to set up, can have high storage overhead, scales poorly with increasing resolution, and is difficult to implement in parallel. In addition, this approach requires formation of the full Jacobian, through either analytic or automatic means. By using a Jacobian-free Newton–Krylov method, we eliminate the burden of supplying and storing the Jacobian.

We address the issue of algorithmic scalability by using the fast adaptive composite grid (FAC) method of McCormick [19] and McCormick and Thomas [20] to precondition the systems of linear equations that must be solved in every nonlinear iteration at every time step. FAC has low setup costs, low storage overhead, and converges at a rate independent of the number of refinement levels [20]. Similar approaches for elasto-plasticity [5] and equilibrium radiation diffusion [28] use multilevel methods on unstructured grids to solve the linearized equations but still require formation of the Jacobian. More recently, Howell and Greenough [14] solve a *linearized* radiation diffusion problem on a SAMR grid as part of a semi-implicit radiation-hydrodynamics algorithm. However, performance of the multilevel linear solver is not their primary focus, and no details about its performance are provided. With minor modifications, our approach could be used to implement a fully nonlinear version of their reflux step.

We report on efforts to solve equilibrium radiation diffusion problems using structured AMR and the Newton–Krylov-FAC method. While structured AMR facilitates reuse of existing software written for logically rectangular grids, discretization at locations near changes in resolution must be treated carefully in order to avoid the creation of artificial sources and to produce solutions with accuracy commensurate with the finest resolution on the AMR grid.

This paper is organized as follows. The next section discusses the equilibrium radiation diffusion model. A discussion of SAMR follows in section 3. Special considerations for spatial discretization on SAMR grids appear in section 4. We describe

our algorithmic components in section 5. Results of computations appear in section 6 and conclusions are summarized in section 7.

2. Equilibrium radiation diffusion. We provide a brief description of the equations that govern equilibrium radiation diffusion. See [21] for a more detailed discussion.

Propagation of a radiation field and its interaction with matter can be modeled by an integro-differential equation that accounts for transport, emission, absorption, and scattering of photons. When the radiation field is isotropic, detailed treatment of transport in angle is not needed. The dependence on angle can be averaged out to obtain a description based on spectral energy density and flux. In a static medium at local thermal equilibrium, absorption is independent of frequency, and thus dependence on frequency can also be averaged out. In this case, the radiative flux can be shown to be proportional to the gradient of the energy density, with the radiative conductivity proportional to the inverse of the opacity of the medium. Also, when the medium is in thermal equilibrium, emission equals absorption, and the total radiation energy density E is proportional to T^4, where T is the temperature of the medium. These considerations lead to a parabolic partial differential equation

$$\frac{\partial E}{\partial t} = \nabla \cdot D(E)\nabla E \ \text{ in } \Omega \subset \mathbb{R}^d,\ d \in \{1,2,3\},\ t > 0, \tag{2.1}$$

with initial conditions $E(t = 0) = E_0$. We use a flux-limited model of radiative conductivity

$$D(E) = \left(\frac{1}{Z^\alpha E^\beta} + \frac{\|\nabla E\|}{|E|} \right)^{-1}, \tag{2.2}$$

where $\alpha < 0$, $\beta \in [0,1]$, and Z is the atomic number of the medium. The term $\|\nabla E\|/|E|$ is Wilson's form for flux limiting [21, 32], which is an ad hoc adjustment introduced to prevent nonphysical behavior in the solution, and which reproduces the correct asymptotic behavior. The parameters α and β can be determined experimentally or through simulation [22]. In this work $\alpha = -3$ and $\beta = 0.75$.

Finally, boundary conditions for (2.1) must be prescribed. Penetration of heat into a cold medium at rest from a hot source can be modeled by assuming a constant imposed radiation field on a portion of the physical boundary $\partial\Omega_{\mathcal{R}} \subset \partial\Omega$. We also assume that the remainder of the physical boundary $\partial\Omega_{\mathcal{N}} = \partial\Omega - \partial\Omega_{\mathcal{R}}$ is perfectly insulating. Under the same assumptions that led to (2.1), the boundary conditions may be expressed as

$$\begin{aligned} \tfrac{1}{2}\mathsf{n} \cdot D(E)\nabla E + \tfrac{1}{4}E = R \text{ on } \partial\Omega_{\mathcal{R}},\ t > 0, \\ \mathsf{n} \cdot D(E)\nabla E = 0 \text{ on } \partial\Omega_{\mathcal{N}},\ t > 0, \end{aligned} \tag{2.3}$$

where n is the unit outward normal to $\partial\Omega$. These conditions lead to solutions of (2.1) known as Marshak waves, which were first described in [18].

3. Structured adaptive mesh refinement. Localized sharp propagation fronts and material discontinuities make the radiation diffusion problem an excellent candidate for AMR. h-refinement AMR techniques provide local mesh resolution by refining the computational mesh locally. SAMR is h-refinement with local fine grid patches placed over coarser grid patches providing increased local resolution. SAMR techniques enjoy several advantages. Regular array access patterns improve

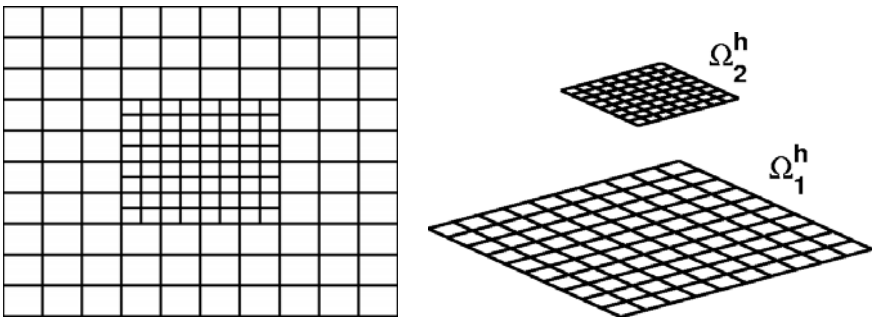


FIG. 3.1. *A composite grid Ω^c of two levels and its component grids Ω_1^h and Ω_2^h.*

cache performance, simple data structures for bookkeeping minimize the overhead for computations, uniform stencil discretizations provide increased accuracy, and reuse of software developed for structured grids is possible.

A SAMR grid consists of a collection of grid patches at different grid resolutions, which together cover the computational domain. Patches with the same mesh resolution are grouped together and form a refinement level. Patches at a refinement level are disjoint or touch along cell boundaries, but do not overlap. A fine grid patch overlying a coarse grid patch is called a child patch, with the underlying coarse patches being the parents. A child patch can have several parent patches and vice versa. The boundaries of child and parent patches are not assumed to align except possibly at physical domain boundaries. This leads to a natural hierarchical structure for SAMR grids that is exploited while designing multilevel algorithms. Operations on the composite grid are decomposed into operations on individual refinement levels which in turn further decompose into operations on individual patches. Valid degrees of freedom at level ℓ are defined in cells not covered by cells belonging to level $\ell+1$. Cells covered by grids at finer levels are "slave" cells with their values being derived from fine cells at the next finer level.

Notation is now introduced for a SAMR grid. Ω_J^c represents a composite SAMR grid with J refinement levels. The subscript is dropped when there is no ambiguity. Refinement level ℓ, $\ell = 1, 2, \ldots, J$, denoted by $\Omega_\ell^{h_\ell}$ or more simply as Ω_ℓ^h, consists of a collection $\{\mathcal{P}\}$ of grid patches at the same grid resolution h_ℓ. The refinement levels are ordered by increasing grid resolution with $h_\ell = r h_{\ell+1}$ $\ell = 1, 2, \ldots, J-1$, where $r \in \mathbb{N}^d$ is the refinement ratio. Each component of r is fixed at 2 in this application, typical values being 2, 3, and 4. The subdomain(s) covered by $\Omega_{\ell+1}^h$ fully nest within the subdomain(s) covered by Ω_ℓ^h. Figure 3.1 illustrates this multilevel structure; Figures 6.4 and 6.6 in section 6 provide more examples of multilevel grid configurations.

4. Discretization. We begin by first discretizing (2.1) in time. Let $t_{n+1} = t_n + \Delta t_n$, $n = 0, 1, \ldots$, where $t_0 = 0$ and Δt_n is a variable time step determined adaptively during the course of the simulation. At $t = t_{n+1}$ an equation of the form

$$E^{n+1} - \gamma \Delta t_n \nabla \cdot D(E^{n+1}) \nabla E^{n+1} - g_n = 0 \tag{4.1}$$

is solved, where E^{n+1} denotes an approximation to $E(t_{n+1})$, $\gamma \in \mathbb{R}$, and g_n is a real-valued function of past information that is determined by the specific choice of the discretization scheme. For backward Euler (BE), $\gamma = 1$ and $g_n(E^n) = E^n$, and for the second-order backward differentiation formula (BDF2), $\gamma = 2/3$ and $g_n(E^n, E^{n-1}) = \frac{4}{3}E^n - \frac{1}{3}E^{n-1}$ for constant Δt_n.

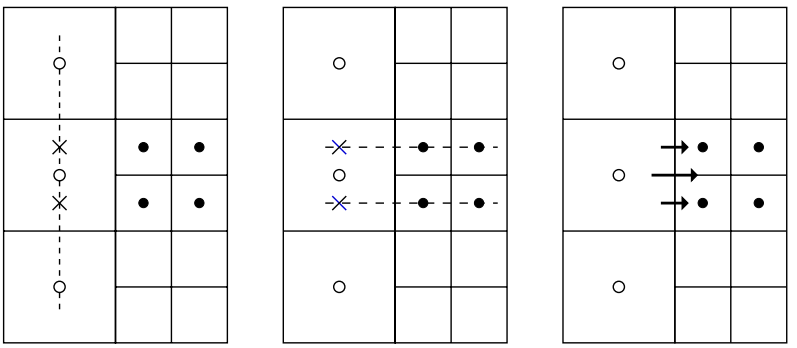


FIG. 4.1. *Schematic of finite volume discretization at interfaces between coarse and fine regions. Open circles represent coarse data; closed circles represent fine data; crosses represent interpolated coarse data; and arrows represent fluxes. The leftmost figure shows the first step, which is to interpolate coarse data to align it with fine data. The second step, shown in the middle figure, can use fine data to either center the aligned data in a ghost cell at the fine resolution, or proceed directly to calculate a flux on the fine face. Finally, fluxes computed at fine resolution must be synchronized with the flux on the underlying coarse face; this is depicted in the rightmost figure.*

Spatial discretization of (4.1) on uniform meshes is done by subdividing Ω into rectangular control volumes and employing standard finite volume discretization techniques. Cell centers are indexed with integer pairs (i, j), and fluxes are computed at faces $(i - \frac{1}{2}, j)$ and $(i, j - \frac{1}{2})$ via

$$\begin{aligned} (D(E)E_x)_{i-\frac{1}{2},j} &\approx F_{i-\frac{1}{2},j} \equiv D(E)_{i-\frac{1}{2},j} \frac{E_{i,j} - E_{i-1,j}}{\Delta x} \Delta y \,, \\ (D(E)E_y)_{i,j-\frac{1}{2}} &\approx F_{i,j-\frac{1}{2}} \equiv D(E)_{i,j-\frac{1}{2}} \frac{E_{i,j} - E_{i,j-1}}{\Delta y} \Delta x \,. \end{aligned} \tag{4.2}$$

A simple scheme is employed for specifying different materials with Z in (2.2) specified at cell centers. With this convention, it is natural to also define $D(E)$ at cell centers, so face-centered values need to be defined to compute these fluxes. For this, harmonic averages

$$\begin{aligned} D(E)_{i-\frac{1}{2},j} &= 2\left(\frac{1}{D(E)_{i-1,j}} + \frac{1}{D(E)_{i,j}}\right)^{-1}, \\ D(E)_{i,j-\frac{1}{2}} &= 2\left(\frac{1}{D(E)_{i,j-1}} + \frac{1}{D(E)_{i,j}}\right)^{-1} \end{aligned} \tag{4.3}$$

are used to define face-centered conductivities. Finally, the discrete spatial operator is obtained by differencing the fluxes on opposite faces and summing the result:

$$\nabla \cdot D(E)\nabla E \approx \left(F_{i+\frac{1}{2},j} - F_{i-\frac{1}{2},j}\right) + \left(F_{i,j+\frac{1}{2}} - F_{i,j-\frac{1}{2}}\right) \,. \tag{4.4}$$

On a SAMR grid, the same spatial discretization is used in patch interiors. There are numerous approaches to handling discretization near changes in resolution. A typical situation for finite volume discretization is depicted in Figure 4.1. In general, data at cell centers is not properly aligned across the change in resolution. The necessary alignment can be achieved by interpolating data on the coarse side of the coarse/fine interface. Fluxes at fine resolution can then be calculated directly from

the aligned data. Alternatively, the aligned data on the coarse side of the interface can be centered in a ghost cell at fine resolution by interpolating in a direction normal to the coarse/fine interface. Finally, the fluxes that reside on fine faces that coincide with a coarse face are summed to obtain a flux on the coarse face. Piecewise linear interpolation with adjustments leads to the symmetric scheme in [11]; piecewise quadratic interpolation is also commonly used [1]. The results presented in this paper use piecewise quadratic interpolation, though similar performance results have been obtained with piecewise linear interpolation.

For ghost points at coarse/fine interfaces, $D(E)$ is interpolated piecewise constant from the coarser level. Because $D(E)$ can exhibit large jumps across material interfaces, refinement regions must be placed carefully to ensure the accuracy of the spatial interpolation schemes.

5. Algorithmic components. The discretization described in section 4 leads to a large-scale system of nonlinear equations defined on a SAMR grid. In particular, note that *all* levels in the grid hierarchy are advanced simultaneously with the same time step, similar to [17] but in contrast to [29, 30]. This means that, on convergence, our solution conserves energy at coarse/fine interfaces, eliminating the need for reflux operations when using local time stepping as in [14]. The system of nonlinear equations at each time step is solved with an inexact Newton method. The linearized problems that must be solved at each iteration of the inexact Newton method are preconditioned by the FAC method. These algorithmic components are described in more detail in the following sections.

5.1. Jacobian-free Newton–Krylov methods. Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a nonlinear function and consider calculating the solution $x^\star \in \mathbb{R}^n$ of the system of nonlinear equations

$$F(x) = 0. \tag{5.1}$$

Starting with an initial approximation x_0 to $x^\star$, the classical Newton's method for solving (5.1) generates a sequence $\{x_k\}$ of approximations to $x^\star$ according to

$$\begin{aligned} F'(x_k)s_k &= -F(x_k) \\ x_{k+1} &= x_k + \theta s_k, \end{aligned} \tag{5.2}$$

where F' is the Jacobian of F evaluated at x_k and $\theta \in (0, 1]$ is a damping parameter. Newton's method is attractive because of its fast local convergence properties. However, for large-scale problems, it is impractical to determine the Newton step s_k in (5.2) with a direct method. Furthermore, when x_k is far from $x^\star$, the linearization that leads to (5.2) may be a poor approximation to $F(x)$. It may be more efficient to require only that s_k satisfy

$$\|F(x_k) + F'(x_k)s_k\| \le \eta \|F(x_k)\| \tag{5.3}$$

for some $\eta \in (0, 1)$ [8]. Appropriate selection of the forcing term η can lead to superlinear and even quadratic convergence [10]. While any iterative method can be used to find an s_k that satisfies (5.3), methods such as transpose-free Krylov subspace methods require only matrix-vector products, which can be approximated by finite differences

$$F'(x_k)v \approx \frac{F(x_k + \varepsilon v) - F(x_k)}{\varepsilon}.$$

In particular, we use GMRES because of its robustness in the presence of inexact matrix-vector products [7]. Such a *Jacobian-free Newton–Krylov* (JFNK) method is especially advantageous when F' is difficult to compute or expensive to store, and has proven to be effective on a wide variety of problems [16].

The damping parameter θ is determined by a combination of traditional linesearch backtracking [9] and problem-specific constraints. Physically, the radiation energy density E must be nonnegative. However, there is nothing to prevent the calculation of an inexact Newton step s_k that leads to $E < 0$ in some locations. We preclude this possibility by first scaling s_k to ensure that the updated solution satisfies the physical constraints. Linesearch backtracking is then invoked on the scaled step in the usual way.

JFNK methods facilitate use of an inexact Newton method, since an application only needs to provide methods to evaluate F, set up a preconditioner, and apply the preconditioner. On a SAMR grid, these operations should ideally exploit the structure of the grid. Considerations for evaluating F are described next; corresponding considerations for operations involving the preconditioner are described in section 5.2.

On a regular grid, evaluation of the discrete form of (4.1) is straightforward, and there are many ways to organize this calculation. One approach is the following.

ALGORITHM 1: NONLINEAR FUNCTION EVALUATION
Evaluate the radiation conductivity (2.2).
Compute the harmonic averages (4.3).
Compute radiative fluxes (4.2).
Difference fluxes to obtain (4.4).
Assemble the nonlinear residual (4.1).

On a SAMR grid, computation is performed in each patch in the grid hierarchy. Since each patch is logically rectangular, software that implements the steps in Algorithm 1 can be used on each of the patches. However, data on each patch must be supplemented to satisfy interpatch data dependencies. The considerations are similar to those needed for a parallel implementation of Algorithm 1, except that in the SAMR case the required data can come from three sources: other patches at the same resolution; data from a coarser level; or physical boundaries. As with parallel computation, these data dependencies are satisfied by supplementing the storage in each patch with a layer of ghost cells. Data is placed in these locations via copies from neighboring patches at the same refinement level, interpolating data from coarser levels as discussed in section 4, and extrapolating data at physical boundaries. In addition, fluxes on coarse faces must be properly synchronized with fluxes on fine faces (cf. the discussion in section 4), and the final result of Algorithm 1 must be synchronized in slave cells via interpolation. These considerations are summarized in Algorithm 2.

Thus, function evaluation on a SAMR grid is a straightforward generalization of function evaluation on a regular grid. Moreover, JFNK has an added advantage for problems on SAMR grids, since the burden of determining the structure and entries of the Jacobian at changes in resolution is eliminated. The additional bookkeeping needed for a SAMR grid can be provided by a SAMR grid management package [25]. However, the potential performance gains made possible by the use of AMR will be lost without effective preconditioning.

Algorithm 2: Nonlinear function evaluation
for $\ell = J$ **to** 1
  Fill ghost cells on Ω_ℓ^h.
  foreach $\mathcal{P} \in \Omega_\ell^h$
    Evaluate the radiation conductivity (2.2).
    Compute the harmonic averages (4.3).
    Compute radiative fluxes (4.2).
  if $\ell < J$
    Coarsen fluxes from $\Omega_{\ell+1}^h$ to Ω_ℓ^h.
for $\ell = 1$ **to** J
  foreach $\mathcal{P} \in \Omega_\ell^h$
    Difference fluxes to obtain (4.4).
    Assemble the nonlinear residual (4.1).
    if $\ell > 1$
      Interpolate the nonlinear residual from Ω_ℓ^h to $\Omega_{\ell-1}^h$.

5.2. Preconditioning. Preconditioning is essential to make the JFNK method competitive. Yet, in the absence of an explicit Jacobian, selection of a preconditioning matrix can be problematic. A simple linearization of (4.1) leads to the choice

$$M = \mathbb{I} - \gamma \nabla \cdot D(E_k^{n+1}) \nabla \tag{5.4}$$

as a preconditioner, where $D(E_k^{n+1})$ is the kth inexact Newton approximation to the time-advanced solution E^{n+1}. This choice was shown to be effective in [26] for problems on uniform grids. The task here is to solve systems of the form $Mz = r$ robustly, accurately, and efficiently when M is discretized on a SAMR grid. For this the FAC method [19] is employed.

FAC extends techniques from multigrid on uniform grids to AMR grids. FAC is a multiplicative Schwarz method. A V-cycle implementation of the method is optimal requiring $\mathcal{O}(n)$ operations where n is the number of degrees of freedom. As a preconditioner for NK methods, FAC employs smoothing on refinement levels with a coarse grid solve using an approximate solver like multigrid. Below we describe the FAC algorithm for completeness after introducing necessary notation.

- I_c^ℓ and I_ℓ^c denote interlevel data transfer operators (restriction and interpolation, respectively) between the composite fine grid Ω^c and refinement level Ω_ℓ^h. For example, in this paper I_ℓ^c is based on bilinear interpolation and I_c^ℓ is based on a simple averaging of fine cell data to coarse cells.
- $I_{\ell+1}^\ell$ and $I_\ell^{\ell+1}$ denote interlevel data transfer operators (restriction and interpolation, respectively) between adjacent refinement levels Ω_ℓ^h and $\Omega_{\ell+1}^h$. Operators I_ℓ^c and I_c^ℓ may be considered as compositions of these operators.
- M^c is the composite fine grid discrete operator obtained by discretizing (5.4) on Ω^c, and M^ℓ approximates M^c on level ℓ.

Expressed in the notation above, one iteration of a V-cycle FAC algorithm for linear problems with smoothing on refinement levels and an approximate solve on the coarsest grid is as follows.

ALGORITHM 3: FAST ADAPTIVE COMPOSITE GRID (FAC) METHOD

Initialize: $r^c = f^c - M^c x^c$; $f^\ell = I_c^\ell r^c$
foreach Ω_ℓ^h, $l = J, \ldots, 2$
  Smooth: $M^\ell e^\ell = f^\ell$
  Correct: $x^c = x^c + I_\ell^c e^\ell$
  Update: $r^c = f^c - M^c x^c$
  Set : $f^{\ell-1} = I_c^{\ell-1} r^c$
Solve : $M^1 e^1 = f^1$
Correct: $x^c = x^c + I_1^c e^1$
foreach Ω_ℓ^h, $l = 2, \ldots, J$
  Update: $r^c = f^c - M^c x^c$
  Set : $f^\ell = I_c^\ell r^c$
  Smooth: $M^\ell e^\ell = f^\ell$
  Correct: $x^c = x^c + I_\ell^c e^\ell$.

6. Numerical results. This section illustrates the performance of Newton–Krylov-FAC for equilibrium radiation diffusion problems on locally refined grids with a comparison to uniform grid calculations. Before describing the test cases, we first specify a few remaining details of the implementation.

Solvers and software. As described earlier, a JFNK solver preconditioned by FAC is used to solve (4.1). This is accomplished using the infrastructure detailed in [25]. This infrastructure provides access to the capabilities of SAMRAI [13] to implement SAMR-specific aspects of Algorithms 2 and 3, and access to an implementation of JFNK from PETSc's Scalable Nonlinear Equation Solver (SNES) package [2]. SNES iterations are terminated when an absolute tolerance $\|F(x_k)\| \leq \epsilon_{abs}$, a relative tolerance $\|F(x_k)\| \leq \epsilon_{rel}\|F(x_0)\|$, or a step tolerance $\|s_k\| \leq \epsilon_{step}$ is satisfied; we use $\epsilon_{abs} = \epsilon_{rel} = 10^{-8}$ and $\epsilon_{step} = 10^{-10}$. For the linear systems that must be solved at each inexact Newton step, GMRES is used with a maximum Krylov subspace dimension of 20, and $\eta = 0.05$ is used in (5.3) to terminate the linear iterations. Note that it was never necessary to restart the GMRES iterations with these choices of maximum Krylov subspace dimension and stopping criteria. The preconditioner is one V-cycle of FAC with two pre- and postsmoothing sweeps of red-black Gauss–Seidel on refinement levels. One V-cycle of semicoarsening multigrid (SMG) [27] implemented in *hypre* [12] is used on the coarsest level. All calculations were performed on Linux workstations with Intel Xeon 2.4 GHz processors in double precision arithmetic.

Time step control. Equilibrium radiation diffusion problems (2.1) are rarely solved in isolation, and are more often encountered as part of a more sophisticated approximation to radiation transport or as one component in a larger multiphysics application. Thus, time step selection for solving (2.1) must account for other time scale restrictions that may be present. In such circumstances, a commonly encountered strategy for choosing Δt is based on restricting the maximum fractional change of E to some prescribed percentage $\sigma \in (0, 1)$ [3, 26]:

$$\max_{(i,j)} \left(\frac{|E_{i,j}^{n+1} - E_{i,j}^n|}{|E_{i,j}^{n+1}|} \right) \leq \sigma \ .$$

For the time discretization schemes used (BE and BDF2), this leads to an effective time step control algorithm. For example, for BE this leads to

$$\Delta t \leq \sigma \min_{(i,j)} \left(\frac{|E_{i,j}^{n+1}|}{|(\nabla \cdot D(E^{n+1}) \nabla E^{n+1})_{i,j}|} \right) . \tag{6.1}$$

In practice, for a stand-alone solution of (2.1), this has several shortcomings. Initially, Δt based on (6.1) is too large, so instead the calculations begin with a smaller time step $\Delta t = 10^{-6}$, which is increased by 10% until (6.1) yields the smaller value for Δt. As the solution approaches steady state, (6.1) produces time steps that quickly become too large; consequently, the rate of growth of Δt is limited by 5% for $\Delta t \in (0.5, 10.0)$ and 3% for $\Delta t \geq 10.0$.

Selection of refinement regions. At regular intervals (specifically, every tenth time step[1]), regions are identified where enhanced resolution is desired. In this work, simple criteria are employed to identify features[2] of the solution that would benefit from enhanced resolution. A gradient detector is used to identify regions where the solution changes rapidly. We also use a curvature-based criterion [6]. It is found that, at least for this problem, the curvature-based criterion leads to slightly larger refinement regions, making it easier to ensure that the base of the thermal front is entirely contained within the finest refinement level without resorting to ad hoc procedures. However, the curvature-based criterion alone can miss relatively flat regions where the solution nevertheless changes rapidly. Finally, if a cell is marked for refinement, its eight nearest neighbors are also marked.

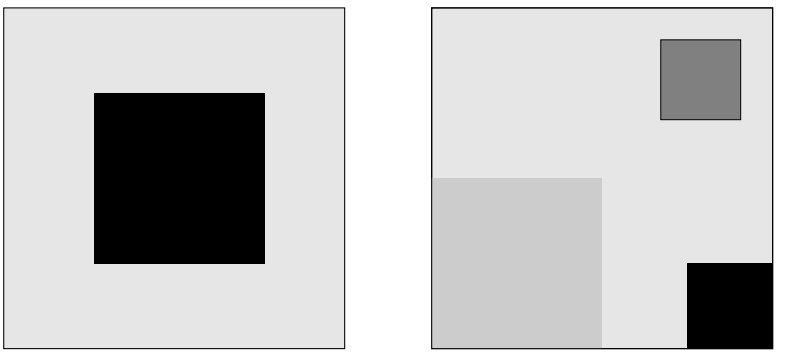


FIG. 6.1. *Material configuration for test cases. Case* 1 *is on the left and Case* 2 *is on the right.*

Test cases. Two test cases are used to illustrate the performance of Newton–Krylov-FAC for solving (2.1), which are depicted in Figure 6.1. In both test cases, $\Omega = [0, 1]^2$ and (2.3) is imposed at both the left $x = 0$ boundary and the right $x = 1$ boundary. Both cases are integrated to a final time of $t = 1500$. Case 1 is taken from [23], where it is used to illustrate the performance of a preconditioner for a

[1]This choice is based on trial and error, and represents a trade-off between the size of the refinement regions and ensuring that the base of the Marshak wave is always contained within the finest refinement level until the next regridding event occurs.

[2]Efforts to develop an *error*-based refinement criterion for this problem are deferred to later work.

nonequilibrium diffusion model. The material properties are defined by

$$Z(x,y) = \begin{cases} 100, & (x,y) \in [\frac{1}{4}, \frac{3}{4}]^2, \\ 10 & \text{otherwise.} \end{cases}$$

With $\alpha = -3$, this leads to a jump of three orders of magnitude in D at the material interface.

Case 2 is adapted from the problem solved in [23]. In this, the material properties are given by

$$Z(x,y) = \begin{cases} 100, & (x,y) \in [\frac{3}{4}, 1] \times [0, \frac{1}{4}], \\ 50, & (x,y) \in [\frac{5}{8}, \frac{5}{8}] \times [\frac{7}{8}, \frac{7}{8}], \\ 20, & (x,y) \in [0, \frac{1}{2}]^2, \\ 10 & \text{otherwise.} \end{cases}$$

6.1. Accuracy. Before presenting performance results for Newton–Krylov-FAC methods on unsteady equilibrium radiation diffusion problems, accuracy measurements are presented. For these measurements, the pattern of local refinement in the dynamic grid calculations is not controlled but allowed to evolve according to the strategies outlined above. Thus, this serves as a verification exercise for both the discretization scheme outlined in section 4 as well as the regridding operations, which collectively identify a new computational grid and transfer the solution from the old grid to the new grid.

The accuracy studies are conducted for Case 2 with BDF2 as the time integrator and $\sigma = 0.05$ in the time step control algorithm. For boundary conditions (2.3), $R = 25.0$ at the $x = 0$ boundary and $R = 0.25$ at the $x = 1$ boundary. Since an analytic solution is not available for comparison, a reference solution on a 512×512 uniform grid is first computed, and results obtained on locally refined grids are compared against this reference solution. In regions where the locally refined grid is not at the same resolution as the uniform grid for the reference solution, the comparison is based on piecewise linear interpolation of the four nearest uniform grid neighbors.

The first set of accuracy results uses a fixed base grid and fixed number of refinement levels, and examines errors over the course of the entire time integration. This appears in Figure 6.2. This shows the relative error over time in the solution computed on locally refined grids against the reference solution. The top curve is for a 32×32 grid with no refinement; each successive curve uses the same base grid but adds a level of refinement. As refinement levels are added, accuracy improvements that appear to be second order are observed. Once four refinement levels are added to the base grid, the finest level of the locally refined calculation is at the same resolution as the reference solution. There, at least three digits of agreement between the reference solution and solution obtained on the locally refined grid are observed.

The next set of accuracy results varies the base grid resolution as well as the number of refinement levels at fixed times. These results are given in Figure 6.3. Here we see excellent agreement among the solutions computed on different locally refined grids, at least until the finest grid level matches the resolution of the reference solution. Note that the finer base grids lead to better agreement with the reference solution.

6.2. Performance. Performance results are now presented for the two test cases described previously. In order to stress the performance of the solvers, we set $R = 2500$

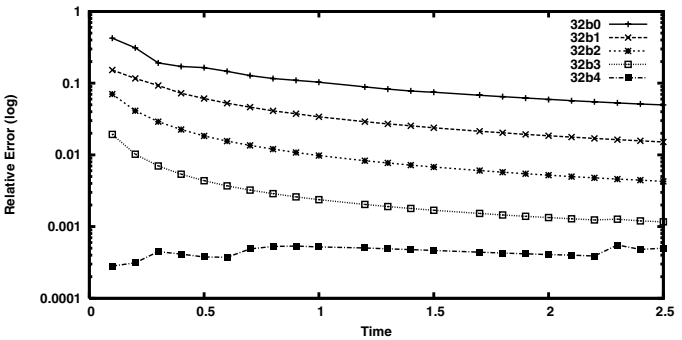


FIG. 6.2. *L2-norm of relative difference in locally refined solution against reference solution for a fixed base grid.*

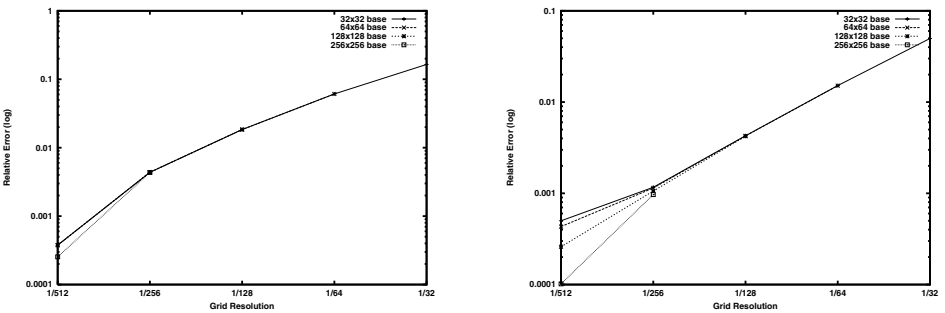


FIG. 6.3. *L2-norm of relative difference in locally refined solution against reference solution for varying base grids. Horizontal axis shows the mesh size of the finest grid level in any of the configurations. Time $t = 0.5$ is on the left, and time $t = 2.5$ is on the right.*

at the $x = 0$ boundary and $R = 0.25$ at the $x = 1$ boundary in (2.3), and set $\sigma = 0.1$ within the time step control algorithm. Performance results are presented for both BE and BDF2 computations.

6.2.1. Case 1. Time evolution of the solution and the grid hierarchy is shown in Figure 6.4. By $t = 0.25$, the incident energy has heated up the left side of the domain and a Marshak wave has propagated into the domain and begun to interact with the second material. Note that the finest level tracks the steepest part of the front, where the solution rapidly changes by three orders of magnitude. The finest level continues to track the front as it propagates further into the domain, and follows it as it wraps around the second material at time $t = 1$.

A comparison of the problem size needed for adaptive and uniform mesh calculations appears in Figure 6.5. In this example, the locally refined grid starts with a 32×32 coarse grid plus four refinement levels. Initially, refinement is automatically introduced only along the $x = 0$ boundary to resolve the increase in energy density due to the imposed radiation field. During this initial period the adaptive calculation requires less than 6% of the storage of the uniform grid calculation. As the front propagates into the domain, the refinement levels grow to capture the evolving front, and the storage required for the adaptive method increases until a maximum of 22% of the uniform grid calculation is needed. As the radiation energy in the interior

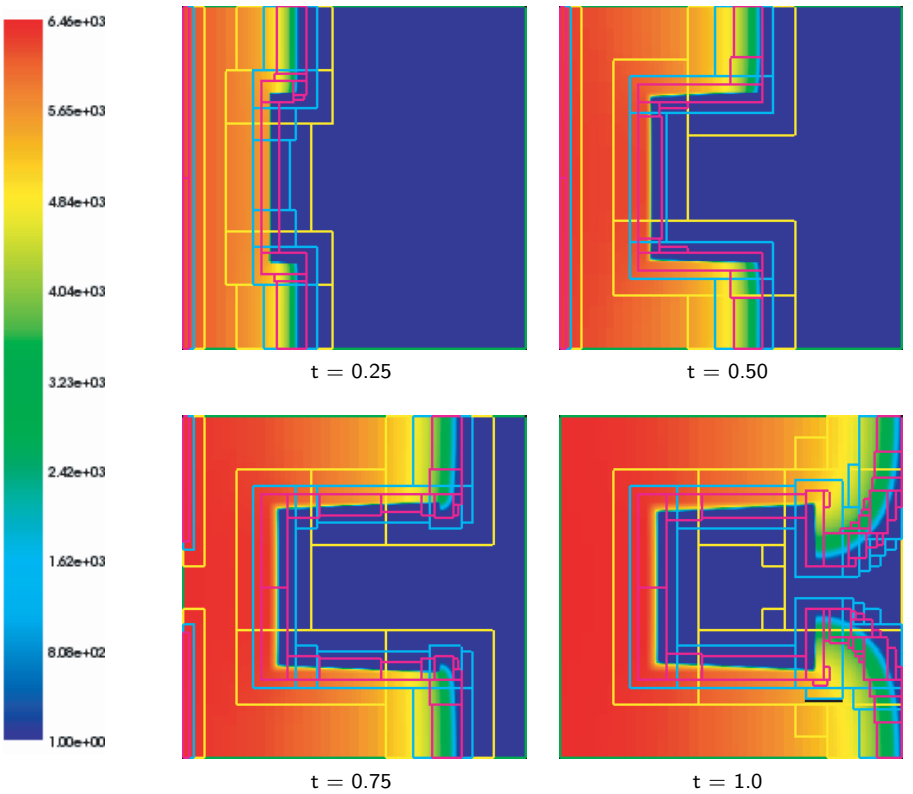


FIG. 6.4. *Evolution of solution and grid for Case* 1*, using a* 32×32 *base grid plus three refinement levels. Boundaries of refinement patches are superimposed on a pseudocolor plot of the solution using a logarithmic scale. The coarsest level is outlined in green; level* 1*: yellow; level* 2*: light blue; level* 3*: magenta.*

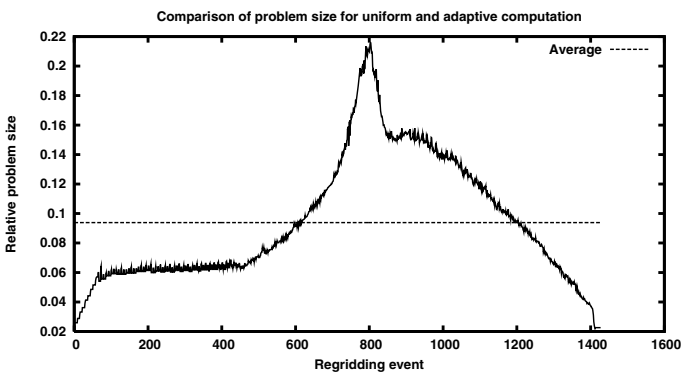


FIG. 6.5. *Comparison of problem size for adaptive vs. uniformly fine calculation.*

of the material increases, de-refinement takes place, until the simulation reaches close to steady state at the end of the calculation, and the adaptive calculation requires only 2% of the uniform grid calculation. The size of the adaptive calculation averages less than 10% of the size of the uniform calculation over the course of the simulation.

Similar behavior is observed when BDF2 is used for the time integration.

Next, we compare the number of iterations required by the adaptive and uniform mesh calculations for BE and BDF2 time integration. This is summarized as averages in Table 6.1. There is very little variation in the number of nonlinear iterations for a fixed base grid, and a slight decrease as the base grid is refined for a given number of refinement levels. Fixing resolution, we see a slight increase in the number of nonlinear iterations as the resolution of the base grid is increased. The number of linear iterations, which is a better reflection of the amount of work performed, shows similar trends, though with greater variation. This is not very surprising, as SMG's very robust semicoarsening/line relaxation algorithm on the global fine grid is being replaced with simple point relaxation, geometric interlevel transfers, and rediscretized operators at each refinement level. It must also be kept in mind that the larger iteration counts occur on much smaller problems. For example, we are comparing a combination of expensive SMG iterations on a 32×32 grid plus point Gauss–Seidel iterations on intermediate levels against SMG iterations on a 512×512 uniform grid.

TABLE 6.1

Summary of performance for Case 1. *The first column gives the size of the base grid; J is the number of refinement levels. Performance at a fixed finest resolution is obtained by reading diagonally from lower left to upper right. Grid configurations that were not run are denoted by –.*

Average number of nonlinear iterations per time step										
	BE					BDF2				
J	1	2	3	4	5	1	2	3	4	5
32×32	4.4	4.2	4.2	4.3	4.5	4.4	4.3	4.3	4.2	4.4
64×64	4.3	4.0	4.1	4.4	–	4.4	4.1	4.1	4.3	–
128×128	3.9	3.9	4.2	–	–	4.1	3.9	4.1	–	–
256×256	3.7	3.9	–	–	–	3.8	3.9	–	–	–
512×512	3.8	–	–	–	–	3.6	–	–	–	–

Average number of linear iterations per time step										
	BE					BDF2				
J	1	2	3	4	5	1	2	3	4	5
32×32	5.9	6.6	7.2	8.4	9.6	5.5	6.1	6.4	7.4	8.4
64×64	5.8	6.7	7.6	9.1	–	5.4	6.1	6.8	7.8	–
128×128	5.8	7.0	8.2	–	–	5.3	6.3	7.3	–	–
256×256	5.9	7.1	–	–	–	5.3	6.5	–	–	–
512×512	6.0	–	–	–	–	5.5	–	–	–	–

6.2.2. Case 2. Time evolution of the adaptive computation appears in Figure 6.6. By $t = 0.5$, the lower conductivity in Region 1 ($Z = 20$) has slowed the Marshak wave in comparison to the background material ($Z = 10$), which has been recognized by our refinement criteria. By $t = 0.75$, the front has begun to interact with Region 2 ($Z = 50$) and has begun to propagate towards Region 3 ($Z = 100$). Again, the finest region tracks this behavior. At $t = 1.0$, the front continues towards Region 3 and the thermal front has begun to wrap around Region 2 due to its lower conductivity. By $t = 1.25$, the Marshak wave has begun to interact with Region 3,

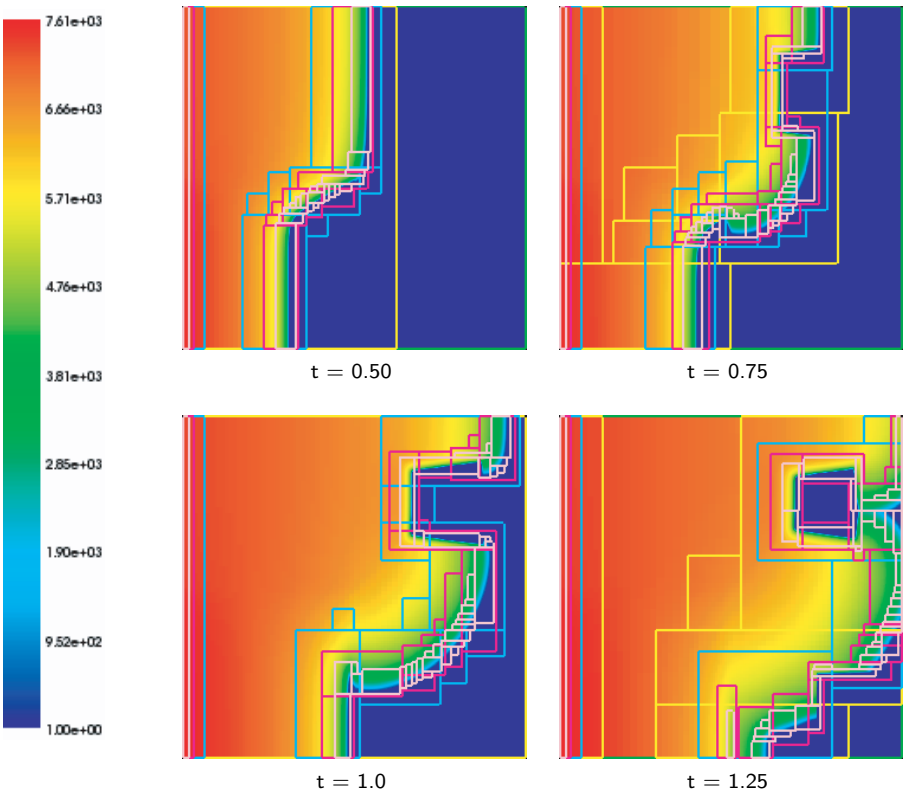


FIG. 6.6. *Evolution of solution and grid for Case* 2, *using a* 32×32 *base grid plus four refinement levels. Boundaries of refinement patches are superimposed on a pseudocolor plot of the solution using a logarithmic color scale. The coarsest level is outlined in green; level* 1: *yellow; level* 2: *light blue; level* 3: *magenta; level* 4: *peach.*

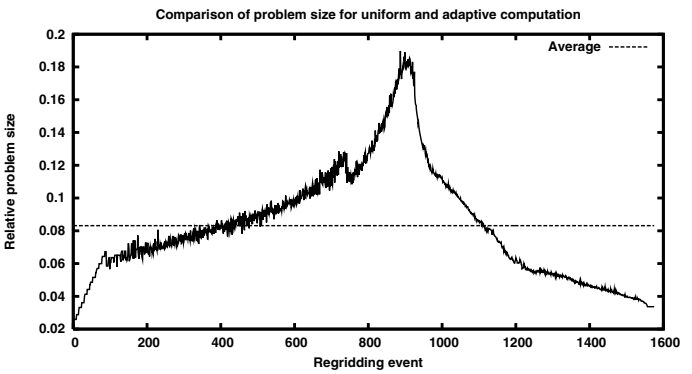


FIG. 6.7. *Comparison of problem size for adaptive vs. uniformly fine calculation.*

while the radiation energy in the interior of Region 2 begins increasing.

Comparison of problem sizes is presented in Figure 6.7. The situation is similar to what was seen for Case 1. This time, the number of cells in the adapted grid increases more quickly due to the presence of Region 1, adjacent to the $x = 0$ boundary.

Eventually there is a decrease in the size of the dynamic calculation as Region 1 is de-refined and before resolution is increased in Region 2. Two inflection points are seen in the size of the locally refined calculation, initially as Region 2 is fully resolved and resolution is increased around Region 3, and subsequently as Regions 2 and 3 are de-refined. The number of cells in the dynamic calculation peaks at less than 20% of the uniform grid calculation, then decreases steadily. On average the dynamic calculation is around 8% of the size of the uniform grid calculation.

Table 6.2 compares nonlinear and linear iteration counts per time step. Once again little variation is seen in the number of nonlinear iterations per time step for a fixed base grid size or for fixed finest resolution, and a small decrease is seen in this iteration count for a fixed number of refinement levels. In contrast, the number of linear iterations per time step increases slowly as more refinement levels are added, and increases by nearly half as we fix resolution and move from a global fine grid to a locally refined calculation. Again, this is likely due to the fact that operators on refinement levels are simply obtained by rediscretization, and interlevel transfer operators are purely geometric.

TABLE 6.2
Summary of performance for Case 2. *Notation is the same as for Table* 6.1.

Average number of nonlinear iterations per time step										
	BE					BDF2				
J	1	2	3	4	5	1	2	3	4	5
32×32	4.4	4.3	4.0	4.1	4.2	4.4	4.3	4.0	4.0	4.1
64×64	4.3	3.9	3.9	4.1	–	4.3	3.9	3.9	4.0	–
128×128	4.0	3.7	3.9	–	–	4.1	3.7	3.9	–	–
256×256	3.8	3.7	–	–	–	3.8	3.7	–	–	–
512×512	3.7	–	–	–	–	3.6	–	–	–	–

Average number of linear iterations per time step										
	BE					BDF2				
J	1	2	3	4	5	1	2	3	4	5
32×32	5.8	6.3	7.1	7.8	8.7	5.4	5.8	6.4	7.1	7.8
64×64	5.6	6.6	7.1	8.4	–	5.2	6.0	6.7	7.6	–
128×128	5.7	6.6	7.8	–	–	5.1	6.1	7.0	–	–
256×256	5.6	7.0	–	–	–	5.0	6.3	–	–	–
512×512	5.7	–	–	–	–	5.2	–	–	–	–

7. Conclusions and future work. The results presented demonstrate the feasibility of combining implicit time integration with AMR for equilibrium radiation diffusion problems. The numerical examples show that, for both first- and second-order time integration schemes, the multilevel FAC preconditioner effectively controls the number of linear iterations per time step. While good scalability with the number of refinement levels is also demonstrated, these results should be considered preliminary, since the refinement regions change as the number of levels is increased and the resolution of the coarsest level is decreased.

There are several issues in implicit AMR that merit improvement and further

exploration. In particular, the interlevel transfers use geometric interpolation, an approach that is known to lead to robustness problems in the presence of discontinuous coefficients. Coarse grid operators are defined via rediscretization rather than Galerkin coarsening. Evaluation of the parallel performance of the code is necessary, as is exploring the use of asynchronous multilevel preconditioners. The criteria for selecting refinement regions are based on feature detection and not estimation of spatial errors. A systematic evaluation of temporal and spatial convergence orders is needed to determine the efficacy of this approach. Before this can occur, the fact that previous information is no longer a solution after regridding must be addressed. We handle this by re-solving (2.1) on the new grid to update the most recent solution information. In [4] it was argued that this can be addressed with higher-order spatial interpolation, but this approach fails for this problem due to the presence of large spatial gradients. In [17] it is handled by reducing the size of the time step and the order of the integrator, but the effect on the order of temporal accuracy was not explored. We plan to investigate these issues in more detail in future work.

REFERENCES

[1] A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome, *A conservative adaptive projection method for the variable density incompressible Navier-Stokes equations*, J. Comput. Phys., 142 (1998), pp. 1–46.

[2] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith, *PETSc Users Manual*, Tech. Report ANL-95/11, Revision 2.1.6, Argonne National Laboratory, Argonne, IL, 2004.

[3] C. Baldwin, P. N. Brown, R. Falgout, F. Graziani, and J. Jones, *Iterative linear solvers in a 2D radiation-hydrodynamics code: Methods and performance*, J. Comput. Phys., 154 (1999), pp. 1–40.

[4] M. Berzins, P. Capon, and P. Jimack, *On spatial adaptivity and interpolation when using the method of lines*, Appl. Numer. Math., 26 (1998), pp. 117–133.

[5] R. Blaheta, *Adaptive composite grid methods for problems of plasticity*, Math. Comput. Simulation, 50 (1999), pp. 123–134.

[6] J. G. Blom, R. A. Trompert, and J. G. Verwer, *Algorithm* 758: *VLUGR2: A vectorizable adaptive-grid solver for PDEs in* 2*D*, ACM Trans. Math. Software, 22 (1996), pp. 302–329.

[7] P. N. Brown, *A local convergence theory for combined inexact-Newton/finite-difference projection methods*, SIAM J. Numer. Anal., 24 (1987), pp. 407–434.

[8] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.

[9] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice–Hall, Englewood Cliffs, NJ, 1983. Reprinted as Classics Appl. Math. 16, SIAM, Philadelphia, 1996.

[10] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

[11] R. E. Ewing, R. D. Lazarov, and P. S. Vassilevski, *Local refinement techniques for elliptic problems on cell–centered grids.* I: *Error analysis*, Math. Comp., 56 (1991), pp. 437–461.

[12] R. D. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners*, in Computational Science - CARS 2002 Part III, Lecture Notes in Comput. Sci. 2331, P. M. A. Sloot, C. J. K. Tan, J. J. Dongarra, and A. G. Hoekstra, eds., Springer-Verlag, New York, 2002, pp. 632–641.

[13] R. D. Hornung and S. Kohn, *Managing application complexity in the SAMRAI object-oriented framework*, Concurrency Comput.: Pract. Exp., 14 (2002), pp. 347–368.

[14] L. H. Howell and J. A. Greenough, *Radiation diffusion for multi-fluid Eulerian hydrodynamics with adaptive mesh refinement*, J. Comput. Phys., 184 (2003), pp. 53–78.

[15] D. A. Knoll, L. Chacón, L. G. Margolin, and V. A. Mousseau, *On balanced approximations for time integration of multiple time scale systems*, J. Comput. Phys., 185 (2003), pp. 583–611.

[16] D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357–397.

[17] S. Li, L. Petzold, and J. M. Hyman, *Solution adapted mesh refinement and sensitivity anal-*

ysis for parabolic partial differential equation systems, in Large-Scale PDE-Constrained Optimization, Lect. Notes Comput. Sci. Eng. 30, Springer-Verlag, Heidelberg, 2003, pp. 117–132.

[18] R. E. Marshak, *Effect of radiation on shock wave behavior*, Phys. Fluids, 1 (1958), pp. 24–29.

[19] S. F. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, Frontiers Appl. Math. 6, SIAM, Philadelphia, PA, 1989.

[20] S. F. McCormick and J. W. Thomas, *The fast adaptive composite grid (FAC) method for elliptic equations*, Math. Comp., 46 (1986), pp. 439–456.

[21] D. Mihalas and B. Weibel-Mihalas, *Foundations of Radiation Hydrodynamics*, Dover, New York, 1999.

[22] E. Minguez, P. Martel, M. Gil, J. G. Rubiano, and R. Rodriguez, *Analytical opacity formulas for ICF elements*, Fusion Eng. Des., 60 (2002), pp. 17–25.

[23] V. A. Mousseau, D. A. Knoll, and W. J. Rider, *Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys., 160 (2000), pp. 743–765.

[24] C. C. Ober and J. N. Shadid, *Studies on the accuracy of time-integration methods for the radiation-diffusion equations*, J. Comput. Phys., 195 (2004), pp. 743–772.

[25] M. Pernice and R. D. Hornung, *Newton-Krylov-FAC methods for problems discretized on locally refined grids*, Comput. Vis. Sci., 8 (2005), pp. 107–118.

[26] W. J. Rider, D. A. Knoll, and G. L. Olson, *A multigrid Newton-Krylov method for multimaterial equilibrium radiation diffusion*, J. Comput. Phys., 152 (1999), pp. 164–191.

[27] S. Schaffer, *A semicoarsening multigrid method for elliptic partial differential equations with highly discontinuous and anisotropic coefficients*, SIAM J. Sci. Comput., 20 (1998), pp. 228–242.

[28] L. Stals, *Comparison of non-linear solvers for the solution of radiation transport equations*, Electron. Trans. Numer. Anal., 15 (2003), pp. 78–93.

[29] R. A. Trompert and J. G. Verwer, *Analysis of the implicit Euler local uniform grid refinement method*, SIAM J. Sci. Comput., 14 (1993), pp. 259–278.

[30] R. A. Trompert and J. G. Verwer, *Runge-Kutta methods and local uniform grid refinement*, Math. Comp., 60 (1993), pp. 591–616.

[31] K.-H. Winkler, M. Norman, and D. Mihalas, *Implicit adaptive-grid radiation hydrodynamics*, in Multiple Time Scales, Academic Press, Orlando, FL, 1985, pp. 145–184.

[32] A. M. Winslow, *Extensions of asymptotic neutron diffusion theory*, Nuclear Sci. Eng., 32 (1968), pp. 101–110.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1727–1741
© 2006 Society for Industrial and Applied Mathematics

AN AGGREGATION MULTILEVEL METHOD USING SMOOTH ERROR VECTORS*

EDMOND CHOW†

Abstract. Many algebraic multilevel methods for solving linear systems assume that the slowly converging, algebraically smooth error is locally constant. This assumption is often not true and can lead to poor performance of the method. Other multilevel methods require a description of the algebraically smooth error via knowledge of the near-nullspace of the operator, but this information may not always be available. This paper presents an aggregation multilevel method for problems where the near-nullspace of the operator is not known. The method uses samples of low-energy error vectors to construct its interpolation operator. The basis vectors for an aggregate are computed via a singular value decomposition of the sample vectors locally over that aggregate. Compared to many other methods that automatically adjust to the near-nullspace, this method does not require that the element stiffness matrices are available.

Key words. preconditioning, multilevel and multigrid methods, smoothed aggregation

AMS subject classifications. 65F10, 65F35, 65F50

DOI. 10.1137/040608192

1. Introduction.

Algebraic multilevel methods use complementary smoothing and coarse-grid correction processes for solving linear systems from discretized partial differential equations (PDEs). In these methods, *slow-to-converge error*, or *algebraically smooth error*, is the error that remains after the smoother has been applied and that must be reduced at the next level. The *interpolation* or *prolongation* operator, P, must be able to represent this algebraically smooth error on the coarser level; i.e., P must be constructed such that the algebraically smooth error is in the range of P. To accomplish this, multilevel methods traditionally must make assumptions about the nature of the algebraically smooth error. In standard algebraic multigrid (AMG), algebraically smooth error is assumed to be locally constant or slowly varying along strong couplings [21].

In aggregation multilevel methods for elliptic PDEs, the near-nullspace of the discrete operator is assumed to locally form an approximate basis for the algebraically smooth error. For second- and fourth-order PDEs, these near-nullspace vectors are constants and linear functions [23]; for linear elasticity, these vectors are rigid body modes [9]. If it is known that the solution is physically smooth, then along with the constant vector, the geometric coordinate vectors of the grid points, x, y, and z, and monomial functions of these may locally represent the algebraically smooth error. Similarly, for p-version finite elements, appropriate basis vectors are also known [18].

Algebraic multilevel methods often fail because the above assumptions do not

*Received by the editors May 13, 2004; accepted for publication (in revised form) November 9, 2004; published electronically February 3, 2006. Lawrence Livermore National Laboratory Technical Report UCRL-JRNL-204139. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract W-7405-Eng-48. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/sisc/27-5/60819.html

†Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, L-560, Box 808, Livermore, CA 94551. Current address: D. E. Shaw Research and Development, LLP, 39th Floor, 120 W. 45th Street, New York, NY 10036 (edmond.chow@deshaw.com).

hold. For example, a scaling of the matrix on the right by a nonconstant diagonal matrix (different values along the diagonal) will change the near-nullspace. (See [6] for other examples.) In other cases, the near-nullspace and the nature of the algebraically smooth error are simply not known. Further, for aggregation methods, it may be desirable for some problems to use other low-energy vectors in addition to the near-nullspace vectors. A procedure is then needed to compute and incorporate these additional vectors.

Multilevel methods that are designed to be more robust and more general by not making assumptions about the algebraically smooth error for a given problem have been developed. The earliest methods constructed interpolation to fit global low-energy eigenvectors, which were computed approximately, or approximate solutions to the homogeneous error equation, $Ae = 0$. These methods date back to some of the original AMG papers [20, 21, 19] (see also [6] for additional early references).

Recently developed methods generally fall into two categories: (1) those that attempt to reduce some measure of the interpolation error and usually involve eigenvector calculations for aggregate matrices (also called element agglomerate matrices) and (2) adaptive methods based on approximately solving the homogeneous error equation.

The interpolation operator, P, may be constructed by minimizing a measure of the interpolation error [2, 1, 4],

$$\min_P \left(\max_u \min_v \frac{\|u - Pv\|_2}{\|u\|_A} \right) \quad \forall u \neq 0, \tag{1.1}$$

where it is assumed that A is scaled to have a unit diagonal. A better minimum leads to a better convergence rate for a two-level method. However, to be practical, the minimization (1.1) must be localized over an aggregate of elements or grid points. In an element-based AMG method [4], the interpolation weights are determined essentially by fitting the eigenvectors of the aggregate matrices (which sum to the global matrix, without boundary conditions). For aggregation multilevel methods, a local basis for the algebraically smooth error for an aggregate is formed by the low-energy eigenvectors of the aggregate matrix [13, 7, 4, 10]. Additional basis vectors can be used to improve the convergence rate [7, 10]. The above approaches, however, are designed for finite element discretizations and require access to the element stiffness matrices. (See, however, approaches that avoid the explicit use of element stiffness matrices [14].)

Related to minimizing a measure of the interpolation error is the energy minimization technique [24]. Here, P is constructed by directly optimizing a constant in the subspace correction framework [25], thus improving the convergence rate. The stability and approximation inequalities are satisfied, guaranteeing mesh-independent convergence.

In the category of adaptive methods are techniques that avoid element stiffness matrices and eigenvectors altogether. In these strategies, a few iterations of the "current" method are applied to the homogeneous problem, $Ae = 0$, to reveal the components of the error that are slow to converge. These components can then be used to update the current method. Versions based on both smoothed aggregation (SA) [6] and AMG [3, 16, 5] have been developed. In bootstrap AMG [3], the interpolation weights are constructed via a least-squares fit of a block of algebraically smooth error vectors. The method is currently being extended to high-order PDEs [15]. In αAMG [16, 5], an interpolation formula involving an algebraically smooth error vector is

used. Adaptive AMG methods can also be defined when AMG is accelerated by a Krylov subspace method. For example, in a conjugate gradient iteration, the Ritz vectors corresponding to the smallest eigenvalues can be used to update the multigrid method's interpolation operator [17].

In this paper, we present a new method of utilizing samples of algebraically smooth error to build an interpolation operator for the SA method. Like bootstrap AMG, the interpolation operator is built by fitting a block of algebraically smooth error vectors simultaneously. However, the SA framework which we use requires only one minimization per aggregate rather than one minimization per fine grid point. The basis vectors for the aggregates are computed via a truncated singular value decomposition (SVD) of the sample vectors locally over each aggregate. The method can be made adaptive by approximately solving $Ae = 0$ and updating the set of algebraically smooth error vectors, like the adaptive methods described above.

In section 2, the new interpolation operator is described. The interpolation is matrix dependent; anisotropies and physical jumps in the PDE problem are reflected in the interpolation operator. The number of sample vectors and the number of basis vectors per aggregate are also discussed. Section 3 presents numerical results on isotropic and anisotropic diffusion problems and a plane strain problem, showing detailed results with different parameters. The section also illustrates the use of a Lanczos process to generate the algebraically smooth sample vectors. Section 4 concludes this paper.

2. Interpolation using algebraically smooth error vectors.

2.1. New interpolation operator. In this section, we describe an interpolation operator based on aggregating variables or grid points that is constructed from samples of low-energy vectors rather than eigenvectors of aggregate matrices. Here, we assume that disjoint aggregates are given. We also suppose that the coefficient matrix, A, at each grid level has been reordered such that the rows and columns of A for an aggregate are ordered together and consecutively by aggregates. This will simplify the notation in this paper. Finally, we will discuss the two-level case and defer the extension to multiple levels to section 2.5.

For aggregate i, let n_i denote the number of variables that are aggregated. Each aggregate will be represented by k_i variables on the coarser level, with $k_i \leq n_i$ and each variable corresponding to a basis vector used to interpolate the error over that aggregate.

As in SA [22, 23], we will smooth the tentative interpolation operator in order to reduce the energy of the basis functions. Thus, we begin by seeking a rectangular tentative interpolation operator of the form

$$\tilde{P} = \begin{bmatrix} \tilde{P}_1 & & & & \\ & \ddots & & & \\ & & \tilde{P}_i & & \\ & & & \ddots & \\ & & & & \tilde{P}_J \end{bmatrix}$$

for J aggregates, where $\tilde{P}_i$ is composed of the k_i tentative basis vectors for aggregate i. The matrix $\tilde{P}$ is block diagonal because the grid point aggregates are disjoint.

Let $S = [s_1, \ldots, s_m]$ be a block of m algebraically smooth error vectors. These vectors may be generated by applying the smoother (to be used in the multigrid

solution process) to the homogeneous equations,

$$As_i = 0, \quad i = 1, \ldots, m$$

(or $AS = 0$ in block form), with a random initial guess for each s_i. The components of the random initial guesses are chosen uniformly from $(-1, 1)$. Choosing positive components is very advantageous for some problems but is less general. (Another method for generating S will be discussed in section 3.3.) When the near-nullspace of A is known, then S should be composed of these near-nullspace vectors. In this case, the method will be the same as the SA method [23] if all k_i equal the number of near-nullspace vectors. The samples, S, may be partitioned as

$$S = \begin{bmatrix} S_1 \\ \vdots \\ S_i \\ \vdots \\ S_J \end{bmatrix},$$

corresponding to the partitioning of $\tilde{P}$.

Given S_i, the portion of the sample vectors corresponding to aggregate i, we seek $\tilde{P}_i$, a rank k_i approximate basis for S_i. Formally, we seek

$$\min_{\tilde{P}_i, W} \|S_i - \tilde{P}_i W\|_2$$

for a given k_i, with $k_i \leq m$. The minimum is achieved when

$$\tilde{P}_i W = U_{k_i} \Sigma_{k_i} V_{k_i}^T,$$

where $U_{k_i} \Sigma_{k_i} V_{k_i}^T$ is the rank k_i truncated SVD of S_i. By matching variables, $\tilde{P}_i$ is defined to be U_{k_i}, the first k_i left singular vectors. We note that the computations are small, dense SVD computations.

This technique exploits the fact that a local portion (over an aggregate) of an algebraically smooth sample vector may have larger or smaller local energy than the same portion of other algebraically smooth sample vectors. The truncated SVD reduces the effect of the higher-energy portions while capturing the desirable low-energy components which are more typical. By using more sample vectors than k_i (the minimum required), the constructed bases are generally improved, as will be shown in section 3.

Recalling (1.1), samples with lower energy must be better approximated by the basis vectors than samples with higher energy; i.e., we wish to have a smaller approximation residual corresponding to samples with smaller energy. This can be accomplished, although imprecisely, by scaling each sample vector, s_j, by $(s_j^T A s_j)^{-1}$ before computing the singular value decomposition. (Empirically this scaling was more effective than $(s_j^T A s_j)^{-1/2}$.) This is particularly important if the sample vectors have very different energy norms. Ideally, we would like to scale the local portion of the sample vectors by their *local* energy norms. However, this would require the construction of aggregate matrices.

Finally, to reduce the energy of the basis functions, $\tilde{P}$ is smoothed one step by a Jacobi smoother to construct the final P. The Jacobi smoother is

$$I - \frac{4}{3\rho} D^{-1} A, \tag{2.1}$$

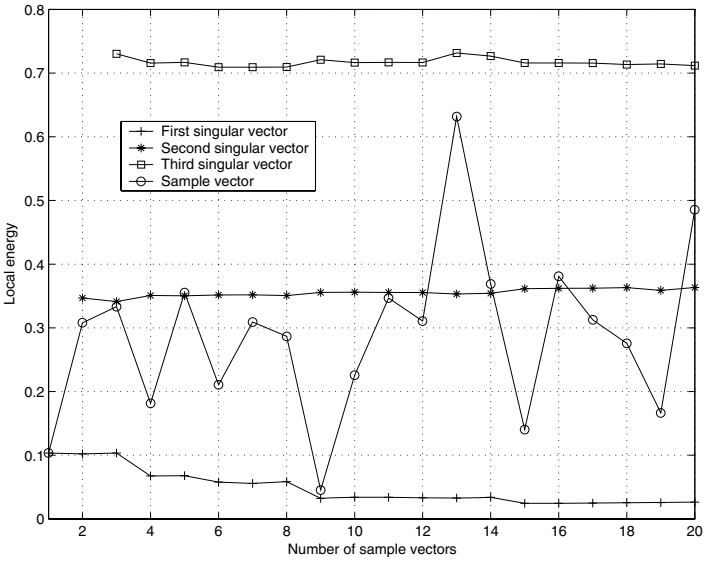
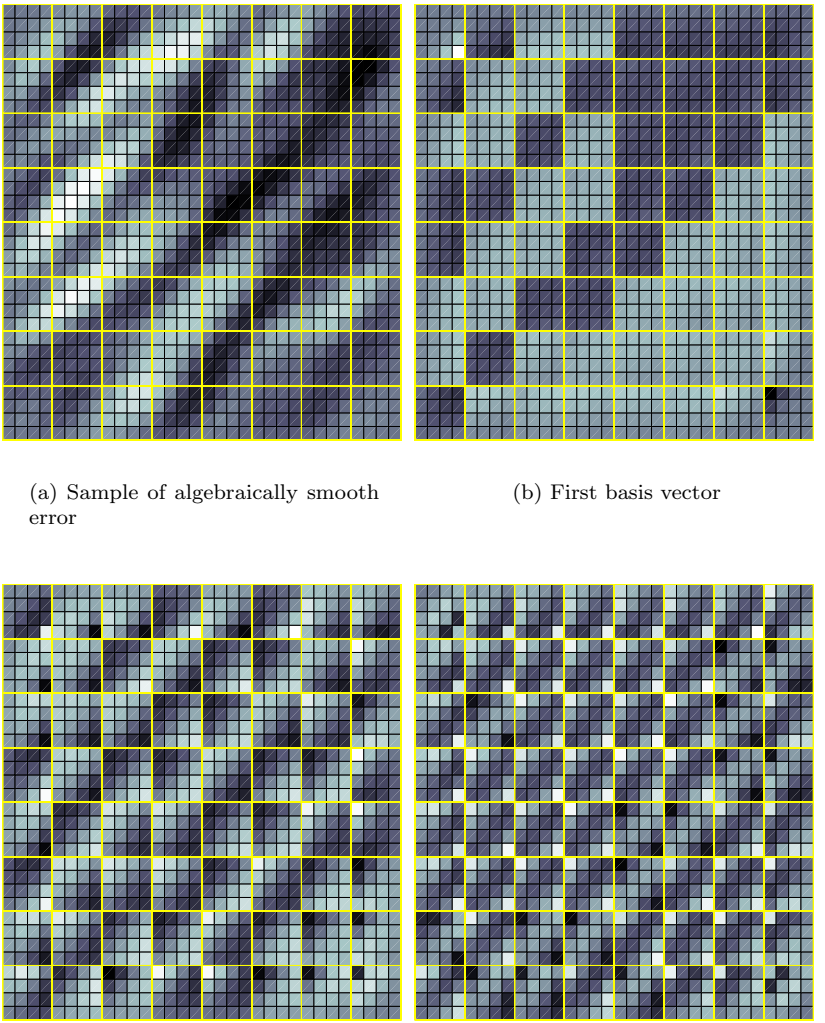


FIG. 2.1. *Local energy of samples and singular vectors versus number of sample vectors. The energy of the first singular vector generally decreases when more samples are used but decreases very slowly after a moderate number of samples. Note the decrease in the energy of the first singular vector when low-energy samples are generated at steps* 4, 9, *and* 15.

where D is the diagonal of A, and ρ is the spectral radius of $D^{-1}A$ estimated by a few steps of a Lanczos or Arnoldi method. This smoothing preserves the null vectors of A.

2.2. Number of sample vectors. The number of sample vectors is a parameter of this method. A larger number of sample vectors will improve the convergence rate, but this must be balanced with the cost of generating these sample vectors and the cost of larger SVD computations.

Figure 2.1 plots the local energy of each of 20 sample vectors for a 9-node aggregate of a one-dimensional isotropic diffusion operator. The sample vectors were generated by 3 steps of symmetric Gauss–Seidel (SGS) applied to the homogeneous error equation with random initial guesses. The figure also plots the local energies of the first, second, and third singular vectors as the number of sample vectors increases. It is evident that the first singular vector has the least energy compared to the second and third and that the energy of the first vector generally decreases when more sample vectors are used. Interestingly, the energy of this singular vector appears to decrease when a new sample vector with very low energy is generated. High-energy sample vectors do not dramatically affect the energy of the first few singular vectors. However, after a moderate number of sample vectors, the energy of the first singular vector then decreases very slowly. Thus, using a very large number of sample vectors is not efficient and should be balanced with the number of smoothing steps applied to each sample vector. This will also be reflected in the numerical tests of the convergence rate. We note that in these examples, the *local* energy of each sample is plotted. In practice, these local energies cannot be computed if the local aggregate matrices are not known.



(a) Sample of algebraically smooth error

(b) First basis vector

(c) Second basis vector

(d) Third basis vector

FIG. 2.2. *An anisotropic diffusion problem with Dirichlet boundary conditions. The first basis vectors* (b) *are nearly constant over each aggregate. The second* (c) *and third* (d) *basis vectors vary slowly in the direction of anisotropy and sharply in the cross direction, like the sample of algebraically smooth error* (a).

2.3. Example. Figure 2.2 shows a sample of algebraically smooth error and three basis vectors produced for an anisotropic diffusion problem with Dirichlet boundary conditions on a 32×32 grid. The direction of anisotropy is 45° (bottom left to top right). The aggregates, shown by the lighter boundary lines, are 4×4 grid points. The value at each grid point is indicated by the gray level of the grid point. For the illustration to be clear, 20 sample vectors were used, with 20 SGS smoothing steps for each vector.

As expected, in Figure 2.2(b), the first basis vector contains nearly locally con-

stant values over each aggregate. Some aggregates have positive values and others have negative values, which explains the two different gray levels predominant for this basis vector. It is also noticeable in the figure that at the Dirichlet boundaries there is a decay of the basis vectors toward zero. This problem-dependent behavior is a feature of this method.

Figure 2.2(c) shows that the second basis vector over each aggregate varies slowly in the direction of anisotropy, like the algebraically smooth error, and varies sharply in the cross direction, again like the smooth error. Figure 2.2(d) shows that the third basis vector is oscillatory in the cross direction, which is very helpful in representing the algebraically smooth error in this example. This quadratic-like function would not normally be chosen by other methods; the standard choice of three basis vectors for this example is composed of the constant and the x- and y-coordinate vectors.

2.4. Number of basis vectors per aggregate. In the SA method [23], the number of basis vectors is chosen beforehand, with each vector being a near-nullspace vector that is known for the problem. In spectral AMGe [10] and the method in [7], the number of basis vectors may be different for each aggregate, and may be chosen such that (1.1) is bounded. In the method presented here, the number of basis vectors may also be different for each aggregate, but no bound on (1.1) is possible.

For aggregate i, the number of basis vectors, k_i, may be chosen based on the singular values from the singular value decomposition of S_i. More basis vectors will improve interpolation for the given aggregate, but increase the cost of the method. One strategy for selecting k_i is to choose the singular vectors that correspond to singular values that are larger than δ times the largest singular value, with $0 \leq \delta < 1$. A sharp decay of the singular values indicates that only a few singular vectors suffice to accurately represent the sample vectors.

To illustrate this, consider two anisotropic diffusion problems on a 32×32 grid using 4×4 aggregates. The two problems have different angles of anisotropy, 0° and 45°. Figures 2.3(a) and 2.3(b) plot the first seven singular values for every aggregate for these two problems. Each curve represents an aggregate, with the singular values scaled such that the largest one for each aggregate is unity. For an angle of anisotropy of 0°, it is natural to use four basis vectors since there are four lines of grid points in each aggregate, and this is reflected in the singular values. For an angle of anisotropy of 45°, a smaller number of basis vectors for many aggregates appears adequate.

2.5. Extension to multiple levels. Once P has been defined for an operator A, the operator at the next coarser level may be defined by the Galerkin coarse-grid operator, $A_c = P^T AP$. The method can be extended to multiple levels by applying the method recursively to solve the coarse-grid correction equations involving A_c.

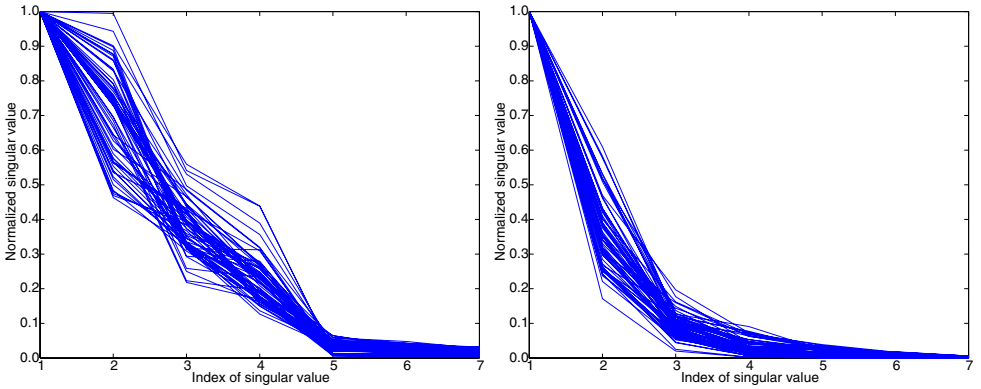
To apply the method to A_c, a set of algebraically smooth error vectors for A_c is needed. These vectors may be generated from scratch, but the following procedure is more effective.

Recall that S denotes m low-energy vectors for A. We seek T, which denotes m low-energy vectors for A_c. If T is constructed such that

$$S = PT,$$

then $A_c T = P^T AS \approx 0$ and the vectors in T have small energy. Assuming that $P \approx \tilde{P}$, then aggregatewise we have

$$S_i \approx \tilde{P}_i T_i, \tag{2.2}$$



(a) anisotropy angle = 0° (b) anisotropy angle = 45°

FIG. 2.3. *First seven singular values for each aggregate (each curve represents an aggregate), scaled such that the first singular value equals unity. The plot suggests using four and three basis vectors for* (a) *and* (b)*, respectively.*

where T_i is the portion of T corresponding to aggregate i. From the SVD of S_i already computed,

$$S_i = U_i \Sigma_i V_i^T \tag{2.3}$$

$$\approx \tilde{P}_i \Sigma_{k_i} V_{k_i}^T, \tag{2.4}$$

and, by matching variables in (2.2) and (2.4), we define

$$T_i = \Sigma_{k_i} V_{k_i}^T. \tag{2.5}$$

Once T is formed, it is generally not smooth enough due to the approximations made and the fact that this is a representation of a smooth vector on a coarser grid. The energy of the vectors in T can be further reduced by applying a few steps of a relaxation procedure to $A_c T = 0$ using T as an initial guess. We have found that for small problems, only a small number of steps is needed, and additional steps do not significantly improve the convergence rate. For large problems, additional steps can be advantageous. After some experimentation, we used two steps of SGS relaxation, which is a compromise between small and large problems.

For efficiency, it is possible to perform these smoothing steps along with the spectral radius estimation required by the prolongation smoother (2.1) by performing matrix-vector products with a block of vectors simultaneously.

3. Numerical tests. In this section, we test smooth-vector interpolation in a multigrid-preconditioned Krylov method. For the multigrid method, restriction is defined to be the transpose of interpolation, the coarse-grid operator is the Galerkin coarse-grid operator, and V(1,1) cycles are used, with SGS smoothing. Unstructured aggregation of the grid points was performed using the algorithm of the SA method [23], with a strength threshold of 0.08. For coarser levels and nonscalar problems, the degrees of freedom corresponding to a grid point are always aggregated together. We denote this multigrid method by SVMG.

The test problems are discretizations of two-dimensional unstructured isotropic and anisotropic diffusion equations,

$$\begin{aligned} au_{xx} + bu_{yy} = f \quad & \text{in } \Omega = (0,1)^2, \\ u = 0 \quad & \text{on } \partial\Omega, \end{aligned}$$

where $a = b = 1$ for the isotropic problems and $a = 1$ and $b = 1000$ for the anisotropic problems. Random right-hand sides were used. The discretization was linear triangular finite elements. Most of the tests were performed with the largest test problems, UNI7 and ANI7 (isotropic and anisotropic, respectively), which have 205,761 equations and 1,436,481 nonzeros. Smaller test problems were also used. In addition, one test with a plane strain problem was performed and will be described later.

Conjugate gradient acceleration was used, with a zero initial guess. The iterations were stopped when the preconditioned residual norm was decreased by 8 orders of magnitude. The experiments were run on a Linux 1.5 GHz Intel Xeon computer with 256 kbytes of cache memory and 512 Mbytes of main memory. Due to the use of randomness in the interpolation, the iteration counts vary by a few steps from test to test.

3.1. Comparison with SA. The SA method is a powerful method when the near-nullspace of the problem is known. For scalar problems, the practice is to use a single basis vector, the vector of all ones. For UNI7, the PCG-SA method converges in 18 iterations. ANI7 is a much more difficult problem, due to the strong anisotropy on an unstructured grid. Here PCG-SA method converges in 118 iterations.

The new method, SVMG, is designed for the case where the near-nullspace of the operator is not known. When the near-nullspace is known, the SA method using this near-nullspace will perform better. To illustrate this point, we use the UNI7 problem along with a scaled version. Given the original problem matrix, A, the scaled problem is DAD, with $D = \text{diag}(10^{d_1}, 10^{d_2}, \ldots)$ and with the real numbers, d_i, selected randomly from $(0, 6)$. This scaling changes the near-nullspace and is discarded and not known to the SVMG method.

Table 3.1 shows results for UNI7 and its scaled version using SA and SVMG. The table shows PCG iteration counts as well as the time to construct the multigrid preconditioner ("Setup") and the time for the PCG iterations ("Solve"). Five levels were used by the methods. SA used 3 basis vectors: the constant and the x- and y-coordinate vectors. SVMG also used 3 basis vectors, constructed from 6 sample vectors, which were themselves generated by 6 SGS relaxation steps. Three basis vectors were used because a single vector might not be adequate when the exact near-nullspace is not known. This, of course, depends on the accuracy of the basis vectors (see also [6], where a single vector is used).

As expected, the results show that for the scaled problem, SVMG performs better than SA. However, for the given parameters, the performance of SVMG on the scaled problem does not match the performance of SVMG or SA on the original, unscaled problem.

The standard basis vectors used in SA for the scaled problem leads to a coarse-grid matrix with sharply varying coefficients. This leads the aggregation procedure to generate many small aggregates at the coarse levels. This in turn leads to very full coarse matrices on subsequent levels and explains the very high setup time for SA on the scaled problem.

TABLE 3.1
PCG iteration counts and setup and solve timings (in seconds) with SA and with SVMG as the preconditioner for the unscaled and scaled UNI7 problem.

	UNI7			Scaled UNI7		
	Iterations	Setup	Solve	Iterations	Setup	Solve
SA	12	8.51	3.25	422	117.38	146.28
SVMG	26	11.43	6.76	63	14.90	17.27

TABLE 3.2
PCG-SVMG iteration counts and timings for UNI7 with various numbers of smooth sample vectors and smoothing steps per sample vector.

Smooth vectors	Smoothing steps	Iterations	Time (s)		
			Setup	Solve	Total
3	12	41	10.29	10.56	20.85
4	9	31	10.60	7.98	18.58
6	6	26	11.43	6.76	18.19
9	4	28	12.65	7.26	19.91
12	3	31	13.53	8.00	21.53
18	2	39	15.52	10.01	25.53
36	1	59	20.87	15.07	35.94
12	6	23	14.59	6.02	20.61
12	12	16	16.03	4.25	20.28
12	24	14	19.53	3.74	23.27
12	36	12	23.04	3.25	26.29

TABLE 3.3
PCG-SVMG iteration counts and timings for ANI7 with various numbers of smooth sample vectors and smoothing steps per sample vector.

Smooth vectors	Smoothing steps	Iterations	Time (s)		
			Setup	Solve	Total
3	12	221	13.77	58.91	72.68
4	9	187	14.15	49.92	64.07
6	6	164	14.88	43.72	58.60
9	4	158	16.25	42.25	58.50
12	3	156	17.61	41.89	59.50
18	2	165	19.82	44.47	64.29
36	1	198	25.85	53.68	79.53
12	6	126	17.81	33.71	51.52
12	12	109	19.55	29.07	48.62
12	24	97	22.82	25.96	48.78
12	36	86	26.31	23.01	49.32
12	48	80	29.80	21.41	51.21
12	60	76	33.23	20.35	53.58
12	72	68	37.02	18.27	55.29

3.2. Test results for isotropic and anisotropic problems. Tables 3.2 and 3.3 show SVMG test results with UNI7 and ANI7, respectively, with various numbers of smooth sample vectors and various numbers of SGS smoothing steps per vector. The top portion of each table shows results when a budget of 36 smoothing steps is used to generate the smooth vectors, either using fewer vectors and more smoothing steps per vector, or vice versa. Three basis vectors were used for each aggregate. Five levels were used in the method.

We express the storage used by the multilevel method in terms of the grid and operator complexities. These terms are common in the AMG literature, e.g., [8]. *Grid complexity* is the total number of degrees of freedom, on all grids, divided by

TABLE 3.4
PCG-SVMG iteration counts for UNI7 and ANI7 when the number of smoothing steps for T at the coarse levels is varied.

	Number of smoothing steps for T					
	0	1	2	3	4	5
UNI7	51	34	26	24	23	23
ANI7	193	169	164	163	157	153

TABLE 3.5
Test results for increasing problem sizes. Grid-independent convergence is not present, particularly for the anisotropic problems. It is expected that for larger problems, a larger number of smoothing steps are required to achieve a convergence rate comparable to that for smaller problems.

	Equations	Nonzeros	Levels	Iterations	Time (s)		
					Setup	Solve	Total
UNI4	3321	22761	2	7	0.25	0.03	0.28
UNI5	13041	90321	3	7	1.25	0.11	1.36
UNI6	51681	359841	4	7	5.57	0.49	6.06
UNI7	205761	1436481	5	13	23.12	3.49	26.61
ANI4	3321	22761	2	13	0.24	0.05	0.29
ANI5	13041	90321	3	22	1.30	0.32	1.62
ANI6	51681	359841	4	44	6.24	2.86	9.10
ANI7	205761	1436481	5	89	25.97	23.77	49.74

the number of degrees of freedom on the finest grid. *Operator complexity* is the total number of nonzero entries, in all coarse- and fine-grid matrices, divided by the number of nonzero entries in the fine-grid matrix. For the results in Tables 3.2 and 3.3, the grid and operator complexities were approximately 1.4 and 3.6, respectively. For scalar problems, these complexities are considered to be high, since a single basis vector is normally used, rather than three basis vectors as in our case. We use more than one basis vector because multiple basis vectors are readily available (from the SVD computation) and using multiple vectors generally improves the performance of the method when the near-nullspace is only approximated. A study using a smaller number of basis vectors will be given in Table 3.6.

Tables 3.2 and 3.3 show the main point of this paper: given a fixed budget of smoothing steps, it can be worthwhile to use more vectors that are less smooth rather than to apply all the smoothing steps to a small number of vectors. When conjugate gradient acceleration is not used, the effect is much more pronounced. In any code, the improvement depends on the portion of the preconditioner setup cost that is due to generating the smooth vectors. The disadvantage of using too many smooth vectors is the increased cost of the SVD calculations, and this is reflected in the setup timings.

Table 3.4 shows iteration counts for PCG-SVMG, varying the number of smoothing steps applied to T, the block of smooth vectors constructed from (2.5) at each coarse level. For UNI7 and ANI7, the results show no significant improvement if more than 2 smoothing steps are taken. As mentioned, we used 2 steps for the tests in this paper. For these results, 6 smooth sample vectors and 6 smoothing steps per sample vector were used.

Table 3.5 shows test results for increasing problem sizes, for both the isotropic and anisotropic problems. Again, 3 basis vectors were used for each aggregate, this time constructed from 12 smooth vectors, with each smooth vector generated using 36 SGS relaxation steps. The results show that grid-independent convergence is not present, particularly for the anisotropic problems. It is expected that for larger problems, a larger number of smoothing steps for the sample vectors is required to achieve a

TABLE 3.6

Test results using a variable number of basis vectors, selected via δ (defined in section 2.4) or k, the number of basis vectors. As the grid and operator complexities improve by using fewer basis vectors, the iteration counts increase.

		Grid complexity	Operator complexity	Iterations
UNI5	$\delta = 0.1$	1.40	3.21	9
	$\delta = 0.2$	1.29	2.17	12
	$\delta = 0.3$	1.20	1.55	16
	$\delta = 0.4$	1.16	1.33	20
	$\delta = 0.5$	1.14	1.25	21
	$k = 3$	1.40	3.03	11
	$k = 2$	1.26	1.90	16
	$k = 1$	1.13	1.22	23
ANI5	$\delta = 0.1$	1.61	6.48	25
	$\delta = 0.2$	1.44	3.86	37
	$\delta = 0.3$	1.37	2.93	47
	$\delta = 0.4$	1.32	2.40	59
	$\delta = 0.5$	1.27	1.97	76
	$k = 3$	1.40	3.03	46
	$k = 2$	1.26	1.90	89
	$k = 1$	1.13	1.22	124

convergence rate comparable to that for smaller problems.

Table 3.6 shows test results when a variable number of basis vectors is used for each aggregate. These tests were performed using a MATLAB code which had this functionality, but no timings are available. The test problems were UNI5 and ANI5 (see Table 3.5 for matrix information), and four levels were used in SVMG. We used a block of 12 sample vectors, each constructed from 3 SGS steps. The number of basis vectors was fixed at 1, 2, or 3, or was chosen based on δ (defined in section 2.4). Table 3.6 shows that a savings in storage can be achieved with a small impact on convergence rate, and in some cases a savings in time may be possible due to a lower operator complexity.

3.3. Using low-energy vectors from the Lanczos method. Instead of using SGS or another smoother to approximately solve the homogeneous error equation, the algebraically smooth error samples can be generated in other ways. For example, starting with a random vector, the Lanczos method can generate approximations to the extremal eigenvalues and eigenvectors of the fine-grid matrix. For SPD problems, it is natural to use the Ritz vectors corresponding to the smallest Ritz values to construct the smooth-vector interpolation operator. If a large number of Lanczos steps are desired, however, the method can be costly in terms of both storage (of the Lanczos vectors) and computation (forming the Ritz vectors).

We note also that SGS relaxation can be applied to a *block* of vectors, which is computationally very efficient. Block versions of the Lanczos method, on the other hand, generate poorer low-energy Ritz vectors, depending on the block size.

Tables 3.7 and 3.8 show test results using low-energy Ritz vectors for UNI7 and ANI7, respectively. Three basis vectors were used for each aggregate, and 5 levels were used in SVMG.

The tables show budgets of either 36 or 72 matrix-vector multiplies. (A budget of 72 corresponds to 36 SGS smoothing steps, but the computational cost is greater.) A different number of low-energy Ritz vectors were constructed. The results show that using more Ritz vectors will improve the convergence rate, and may also improve

TABLE 3.7
PCG-SVMG results for UNI7, using low-energy Ritz vectors.

Matvec budget	Ritz vectors	Iterations	Time (s)		
			Setup	Solve	Total
36	3	50	10.57	12.77	23.34
	4	45	11.24	11.53	22.77
	6	41	12.19	10.50	22.69
	9	36	13.83	9.24	23.07
	12	33	14.73	8.53	23.26
72	3	26	12.11	6.74	18.85
	4	24	12.75	6.24	18.99
	6	21	14.02	5.52	19.54
	9	21	15.63	5.48	21.11
	12	20	17.19	5.26	22.45

TABLE 3.8
PCG-SVMG results for ANI7, using low-energy Ritz vectors.

Matvec budget	Ritz vectors	Iterations	Time (s)		
			Setup	Solve	Total
36	3	300	15.30	80.69	95.99
	4	289	15.99	77.72	93.71
	6	282	17.30	75.89	93.19
	9	280	18.27	75.72	93.99
	12	281	19.44	75.61	95.05
72	3	204	15.49	54.33	69.82
	4	184	16.51	49.28	65.79
	6	176	17.75	47.12	64.87
	9	173	19.68	46.40	66.08
	12	170	21.22	45.45	66.67

the total time, depending on the increase in setup cost. In particular, for the ANI7 problem, the best timings are achieved when more than the minimum number of Ritz vectors are constructed.

3.4. Plane strain problem. We briefly consider a problem that strictly requires multiple basis vectors per aggregate, a plane strain problem on a square with homogeneous Dirichlet boundary conditions. The 3 rigid body modes are known from the geometry and grid for this problem and span the near-nullspace of the PDE operator.

The problem was discretized with 217×217 linear quadrilateral elements (93,312 equations). The nodes were aggregated regularly, using 3×3 node aggregates. Ten smooth vectors were used, with 10 SGS relaxation steps to generate each vector. Three basis vectors were used per aggregate. The multigrid method used 4 levels.

The original problem matrix, A, was generated, as well as a scaled matrix, DAD, with $D = \mathrm{diag}(10^{d_1}, 10^{d_2}, \ldots)$ and with the real numbers, d_i, selected randomly from $(0, 6)$.

Table 3.9 shows the results for the original problem matrix and the scaled matrix, using both the rigid body modes to construct the interpolation operator and the algebraically smooth vectors. Interpolation using the rigid body modes is ideal for the original matrix, but is entirely inappropriate for the scaled matrix. On the other hand, by using the algebraically smooth vectors, both problems can be solved. As expected, however, for the original matrix, the smooth-vector interpolation does not perform as well as interpolating using the rigid body modes.

TABLE 3.9

PCG iteration counts for the plane strain problem, using original and scaled matrices. SA or SVMG were compared as the preconditioner. 200+ *denotes that no convergence was achieved in* 200 *iterations.*

Poisson ratio	SA		SVMG	
	Original	Scaled	Original	Scaled
0.30	13	200+	34	64
0.40	18	200+	49	97
0.45	24	200+	87	161

4. Concluding remarks. The smooth-vector interpolation operator presented in this paper is constructed to interpolate samples of algebraically smooth error. The interpolation operator is matrix dependent, but particular entries in the matrix are not needed once the aggregates have been chosen. When the budget for generating the sample vectors is fixed, it can be beneficial to use more samples than the number of basis vectors, even if each sample vector has more energy. This technique is particularly advantageous when many basis vectors are required, since additional vectors are available (from the SVD) at no additional cost. Compared to other methods such as spectral AMGe [10], this method does not require that the stiffness matrices for each aggregate be available. Generating the sample vectors may also be very efficient if blocks of vectors are relaxed simultaneously, but this was not tested in this paper.

We note that the performance of the method cannot be better than using the exact near-nullspace vectors if they are known and available. An apparent disadvantage of the method is that for larger problems, smoother sample vectors are required to maintain the same rate of convergence. Thus it may be necessary to use a multigrid method to help generate the sample vectors themselves, as in the adaptive multilevel methods described in the introduction. SVMG can be made adaptive by approximately solving $Ae = 0$ and updating the set of algebraically smooth error vectors used to construct the interpolation operator. Preliminary results were reported in [11].

We also note that it is also possible to perform the node aggregation using samples of algebraically smooth error, instead of using matrix entries. As in [12], nodes may be aggregated if the samples of algebraically smooth error show a strong coupling between the nodes. Incorporating such a strategy may help develop multigrid methods that are matrix-free.

Acknowledgments. The author wishes to thank Marian Brezina, Robert Falgout, Oren Livne, Scott MacLachlan, Tom Manteuffel, Steve McCormick, Charles Tong, and Panayot Vassilevski for their help in this research and the preparation of this paper. The author is also grateful to two anonymous referees, whose detailed comments added significantly to the presentation and accuracy of this paper.

REFERENCES

[1] J. H. BRAMBLE, J. E. PASCIAK, J. WANG, AND J. XU, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., 57 (1991), pp. 23–45.

[2] A. BRANDT, *Algebraic multigrid theory: The symmetric case*, Appl. Math. Comput., 19 (1986), pp. 23–56.

[3] A. BRANDT, *Multiscale scientific computation: Review* 2001, in Multiscale and Multiresolution Methods: Theory and Applications, T. J. Barth, T. Chan, and R. Haimes, eds., Springer-Verlag, Heidelberg, 2001, pp. 1–96.

[4] M. BREZINA, A. J. CLEARY, R. D. FALGOUT, V. E. HENSON, J. E. JONES, T. A. MANTEUFFEL, S. F. MCCORMICK, AND J. W. RUGE, *Algebraic multigrid based on element interpolation*

(AMGe), SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.
[5] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive algebraic multigrid*, SIAM J. Sci. Comput., 27 (2006), pp. 1261–1286.
[6] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, *Adaptive smoothed aggregation (αSA) multigrid*, SIAM Rev., 47 (2005), pp. 317–346.
[7] M. Brezina, C. Heberton, J. Mandel, and P. Vanek, *An Iterative Method with Convergence Rate Chosen A Priori*, Tech. Report UCD/CCM 140, University of Colorado at Denver, 1999.
[8] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, 2000.
[9] V. E. Bulgakov, *Iterative aggregation technique for large-scale finite element analysis of mechanical systems*, Comput. & Structures, 52 (1994), pp. 829–840.
[10] T. Chartier, R. D. Falgout, V. E. Henson, J. Jones, T. Manteuffel, S. McCormick, J. Ruge, and P. S. Vassilevski, *Spectral AMGe (ρAMGe)*, SIAM J. Sci. Comput., 25 (2003), pp. 1–26.
[11] E. Chow, *An aggregation multilevel method based on smoothed error vectors*, in 2003 International Conference on Preconditioning Techniques for Large Sparse Matrix Problems in Scientific and Industrial Applications, Napa, CA, 2003.
[12] E. Chow, *An unstructured multigrid method based on geometric smoothness*, Numer. Linear Algebra Appl., 10 (2003), pp. 401–421.
[13] J. Fish and V. Belsky, Internat. J. Numer. Methods Engrg., 40 (1997), pp. 4341–4361.
[14] V. E. Henson and P. S. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.
[15] O. E. Livne, *private communication*, 2004.
[16] S. MacLachlan, *Improving Robustness in Multiscale Methods*, Ph.D. thesis, University of Colorado at Boulder, 2004.
[17] S. MacLachlan, T. Manteuffel, and S. McCormick, *Preconditioning with adaptive AMG*, in 2004 Copper Mountain Conference on Iterative Methods, 2004.
[18] J. Mandel, *Iterative methods for p-version finite elements: Preconditioning thin solids*, Comput. Methods Appl. Mech. Engrg., 133 (1996), pp. 247–257.
[19] S. F. McCormick and J. W. Ruge, *Algebraic multigrid methods applied to problems in computational structural mechanics*, in State-of-the-Art Surveys on Computational Mechanics, A. K. Noor and J. T. Oden, eds., ASME, New York, 1989, pp. 237–270.
[20] J. W. Ruge, *Algebraic multigrid (AMG) for geodetic survey problems*, in Preliminary Proceedings of the International Multigrid Conference, Institute for Computational Studies at Colorado State University, Fort Collins, CO, 1983.
[21] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, 1987, pp. 73–130.
[22] P. Vanek, *Acceleration of convergence of a two-level algorithm by smoothing transfer operators*, Appl. Math., 37 (1992), pp. 265–274.
[23] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.
[24] W. L. Wan, T. F. Chan, and B. Smith, *An energy-minimizing interpolation for robust multigrid methods*, SIAM J. Sci. Comput., 21 (2000), pp. 1632–1649.
[25] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1742–1759
© 2006 Society for Industrial and Applied Mathematics

A COMPARISON OF DEFLATION AND THE BALANCING PRECONDITIONER*

R. NABBEN† AND C. VUIK‡

Abstract. In this paper we compare various preconditioners for the numerical solution of partial differential equations. We compare the well-known balancing preconditioner used in domain decomposition methods with a so-called deflation preconditioner. We prove that the effective condition number of the deflated preconditioned system is always, i.e., for all deflation vectors and all restrictions and prolongations, below the condition number of the system preconditioned by the balancing preconditioner. Even more, we establish that both preconditioners lead to almost the same spectra. The zero eigenvalues of the deflation preconditioned system are replaced by eigenvalues which are one if the balancing preconditioner is used. Moreover, we prove that the A-norm of the errors of the iterates built by the deflation preconditioner is always below the A-norm of the errors of the iterates built by the balancing preconditioner. Depending on the implementation of the balancing preconditioner the amount of work of one iteration of the deflation preconditioned system is less than or equal to the amount of work of one iteration of the balancing preconditioned system. If the amount of work is equal, both preconditioners are sensitive with respect to inexact computations. Finally, we establish that the deflation preconditioner and the balancing preconditioner produce the same iterates if one uses certain starting vectors. Numerical results for porous media flows emphasize the theoretical results.

Key words. deflation, coarse grid correction, balancing, preconditioners, conjugate gradients, porous media flow, scalable parallel preconditioner

AMS subject classifications. 65F10, 65F50, 65N22

DOI. 10.1137/040608246

1. Introduction.

The conjugate gradient method is the most-used method to solve large linear systems of equations

$$Ax = b$$

whose coefficient matrices A are sparse and symmetric positive definite. Such systems are encountered, for example, when a finite volume/difference/element method is used to discretize an elliptic partial differential equation.

The convergence rate of the conjugate gradient method (CG-method) is bounded as a function of the condition number of the system matrix to which it is applied. If the condition number of A is large, it is advisable to solve, instead, a preconditioned system $M^{-1}Ax = M^{-1}b$, where the symmetric positive definite preconditioner M is chosen such that $M^{-1}A$ has a more clustered spectrum or a smaller condition number than that of A. Furthermore, system $Mz = r$ must be cheap to solve relative to the improvement it provides in convergence rate. A final desirable property in a preconditioner is that it should parallelize well, especially on distributed memory computers.

*Received by the editors May 14, 2004; accepted for publication (in revised form) September 6, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-5/60824.html
†TU Berlin Institut für Mathematik, MA 3-3, Strasse des 17. Juni 136, D-10623 Berlin, Germany (nabben@math.tu-berlin.de).
‡Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute of Applied Mathematics, P.O. Box 5031, 2600 GA Delft, The Netherlands (c.vuik@math.tudelft.nl).

In [16] two different preconditioners are compared, namely a deflation preconditioner and an additive coarse grid correction preconditioner. It is shown that the deflation preconditioner leads to a smaller condition number than a coarse grid correction preconditioner like the BPS preconditioner by Bramble, Pasciak, and Schatz [1]. Here we compare the deflation preconditioner with the balancing preconditioner proposed by Mandel [11]. In the following we give a brief introduction into both preconditioning techniques.

To describe the deflation method we define the projection P_D by

$$P_D = I - AZ(Z^T AZ)^{-1} Z^T, \quad Z \in \mathbb{R}^{n \times r}, \tag{1.1}$$

where the column space of Z is the deflation subspace, i.e., the space to be projected out of the residual, and I is the identity matrix of appropriate size. We assume that $r \ll n$ and that Z has rank r. Under this assumption $E \equiv Z^T AZ$ may be easily computed and factored and is symmetric positive definite. Since $x = (I - P_D^T)x + P_D^T x$ and because

$$(I - P_D^T)x = Z(Z^T AZ)^{-1} Z^T Ax = ZE^{-1} Z^T b \tag{1.2}$$

can be immediately computed, we need only to compute $P_D^T x$. In light of the identity $AP_D^T = P_D A$, we can solve the deflated system

$$P_D A\tilde{x} = P_D b \tag{1.3}$$

for $\tilde{x}$ using the CG-method, premultiply this by P_D^T, and add it to (1.2).

Obviously (1.3) is singular. But a positive semidefinite system can be solved by the CG-method as long as the right-hand side is consistent (i.e., as long as $b = Ax$ for some x) [9]. This is certainly true for (1.3), where the same projection is applied to both sides of the nonsingular system. Since the null space never enters the iteration, the corresponding zero eigenvalues do not influence the convergence [9, 22]. Motivated by this fact, we define the *effective condition number* of a positive semidefinite matrix $C \in \mathbb{R}^{n \times n}$ with r zero eigenvalues to be the ratio of its largest to smallest *positive* eigenvalues:

$$\kappa_{eff}(C) = \frac{\lambda_n}{\lambda_{r+1}}.$$

It is possible to combine both a standard preconditioning and preconditioning by deflation (for details see [7]). The convergence is then described by the effective condition number of $M^{-1} P_D A$. For more details about the deflation preconditioner see [17, 15, 4, 13, 10, 25, 2, 24, 26, 7, 23, 16].

We compare the preconditioned deflation operator with the balancing preconditioner proposed by Mandel [11] and Mandel and Brezina [12] and analyzed by Dryja and Widlund [3], Pavarino and Widlund [18], and Toselli and Widlund [21]. As the FETI algorithm [5, 6] the balancing Neumann-Neumann preconditioner is one of the domain decomposition methods that have been most carefully implemented and severely tested on the very largest existing parallel computer systems.

Applied to some specific symmetric positive definite problems the balancing Neumann-Neumann preconditioner leads to moderately growing condition numbers if the size of the systems increases [20]. Moreover, if an appropriate scaling is used, the condition numbers are independent of jumps in the coefficients in the matrices [20].

In our notation the balancing preconditioner is given by

$$P_B = (I - ZE^{-1}Z^TA)M^{-1}(I - AZE^{-1}Z^T) + ZE^{-1}Z^T, \tag{1.4}$$

where $Z \in \mathbb{R}^{n\times r}$, $E = Z^TAZ$, and M is a symmetric positive definite matrix. Note that P_B is symmetric and positive definite. For more details we refer the reader to [11] and the books [20, 19, 21].

As a first comparison of both preconditioners we observe that the balancing preconditioner needs per iteration 3 matrix vector products and the coarse grid operator is used 2 times. This makes the balancing preconditioner per iteration more expensive than the deflation approach. However, if an optimal implementation of the balancing preconditioner is used (see, e.g., [21]), the amount of work per iteration is the same.

In this article we give a detailed comparison of these two preconditioners. We prove that the effective condition number of the deflated preconditioned system $M^{-1}P_DA$ is always below the condition number of the system preconditioned by the balancing preconditioner P_BA. Even more, we establish that the spectrum of P_BA is the same as $M^{-1}P_DA$, except the r zero eigenvalues are replaced by eigenvalues which are one.

This implies that for all matrices $Z \in \mathbb{R}^{n\times r}$ and all positive definite preconditioners M^{-1} the effective condition number of the deflated preconditioned system is below or equal to the condition number of the system preconditioned by the balancing preconditioner! However, the condition number is not the only parameter which influences the convergence behavior of the CG-method. The convergence may be significantly faster if the eigenvalues of A are clustered [22]. But we obtain from the above-mentioned result that the clustering of the eigenvalues of the two different preconditioned systems is the same. However, we have a cluster at zero in one case and at one in the other case. These results are stated in section 2.

There are other properties which influence the convergence behavior of the CG-method, e.g., the starting vector, the right-hand side, and the location of the clusters of eigenvalues. Therefore, a more detailed comparison is given in section 3. There we prove that the A-norm of the errors of the iterates built by the deflation preconditioner is always below the A-norm of the errors of the iterates built by the balancing preconditioner. Moreover, we establish that the deflation preconditioner and the balancing preconditioner produce the same iterates if one uses certain starting vectors. More precisely we show which terms in the preconditioned CG-method are the same for both methods and which terms are different. At the end of section 3 we prove that the condition of the balancing preconditioned system decreases if one takes a finer grid as a coarse grid.

In section 4 numerical results emphasize our theoretical results.

2. Spectral properties. In this section we compare the effective condition number for the deflation and balancing preconditioned matrices. In section 2.1 we give some definitions and preliminary results. Thereafter a comparison is made if the projection vectors are equal to eigenvectors (in section 2.2) and for general projection vectors (in section 2.3).

2.1. Notations and preliminary results. In the following we denote by $\lambda_i(M)$ the eigenvalues of a matrix M. If the eigenvalues are real, the $\lambda_i(M)$'s are ordered increasingly.

For two Hermitian $n \times n$ matrices A and B we write $A \succeq B$, if $A - B$ is positive semidefinite.

Next we mention well-known properties of the eigenvalues of Hermitian matrices.

LEMMA 2.1. *Let $A, B \in \mathbb{C}^{n\times n}$ be Hermitian. For each $k = 1, 2, \ldots, n$ we have*

$$\lambda_k(A) + \lambda_1(B) \leq \lambda_k(A + B) \leq \lambda_k(A) + \lambda_n(B).$$

From the above lemma we easily obtain the next lemma.

LEMMA 2.2. *If $A, B \in \mathbb{C}^{n\times n}$ are positive semidefinite with $A \succeq B$, then $\lambda_i(A) \geq \lambda_i(B)$.*

Moreover, we will use the following lemma.

LEMMA 2.3. *Let $A, B \in \mathbb{C}^{n\times n}$ be Hermitian and suppose that B has rank at most r. Then*

- $\lambda_k(A + B) \leq \lambda_{k+r}(A), \quad k = 1, 2, \ldots, n - r,$
- $\lambda_k(A) \leq \lambda_{k+r}(A + B), \quad k = 1, 2, \ldots, n - r.$

Lemma 2.1, Lemma 2.2, and Lemma 2.3 can be found, e.g., as Theorem 4.3.1, Corollary 7.7.4, and Theorem 4.3.6, respectively, in [8].

2.2. Projection vectors chosen as eigenvectors. In this section we compare the effective condition number of $P_D A$ and $P_B A$ if the projection vectors are equal to the eigenvectors of A.

DEFINITION 2.4. *Let λ_i be the eigenvalues of A. Choose the eigenvectors v_k of A such that $v_k^T v_j = \delta_{kj}$, and define $Z = [v_1 \ldots v_r]$.*

THEOREM 2.5. *Using Z as given in Definition 2.4 and preconditioner M equal to the identity, the spectrum of $P_B A$ is*

$$\mathit{spectrum}(P_B A) = \{1, \ldots, 1, \lambda_{r+1}, \ldots, \lambda_n\}.$$

Proof. For this choice of Z it appears that

$$E = Z^T A Z = \mathit{diag}(\lambda_1, \ldots, \lambda_r). \tag{2.1}$$

We consider $P_B A v_k$. For $k = 1, \ldots, n$ we obtain

$$P_B A v_k = \left(I - Z\,\mathit{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_r}\right) Z^T A\right)\left(I - AZ\,\mathit{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_r}\right) Z^T\right)\lambda_k v_k + Z\,\mathit{diag}\left(\frac{1}{\lambda_1}, \ldots, \frac{1}{\lambda_r}\right) Z^T \lambda_k v_k.$$

Using the orthogonality properties of the eigenvectors one obtains

$$P_B A v_k = v_k, \text{ for } k = 1, \ldots, r.$$

For $k = r + 1, \ldots, n$ the same orthogonality properties lead to

$$P_B A v_k = \lambda_k v_k, \text{ for } k = r + 1, \ldots, n. \qquad \square$$

In order to compare both approaches we note that

$$\kappa_{eff}(P_D A) = \frac{\lambda_n}{\lambda_{r+1}} \tag{2.2}$$

and

$$\kappa(P_B A) = \frac{\max\{1, \lambda_n\}}{\min\{1, \lambda_{r+1}\}}. \tag{2.3}$$

From (2.2) and (2.3) it follows that $\kappa(P_B A) \geq \kappa_{eff}(P_D A)$, so the convergence bound based on the effective condition number implies that deflated CG never converges slower than CG combined with the balancing preconditioner if both methods use the eigenvectors corresponding to the r smallest eigenvalues as projection vectors.

2.3. Projection vectors chosen as general vectors. In the previous section we showed that the deflation technique leads to a smaller effective condition number than the balancing preconditioner, if eigenvectors are used. However, computing the r smallest eigenvalues is mostly very expensive. Moreover, in multigrid methods and domain decomposition methods special interpolation and prolongation matrices are used to obtain grid independent convergence rates. So a comparison only for eigenvectors is not enough. But in this section we generalize the results of the last section. We prove that the effective condition number of the deflated preconditioned system is always, i.e., for all matrices $Z \in \mathbb{R}^{n\times r}$ and all preconditioners M^{-1}, below the condition number of the system preconditioned by the balancing preconditioner. To do this we repeat some properties of the projection operator P_D used in the deflation method (see [7]). The operator P_D is defined as

$$P_D = I - AZE^{-1}Z^T, \text{ where } E = Z^TAZ. \tag{2.4}$$

Furthermore, the following identities hold:

$$P_D^2 = P_D, \;\; P_DAZ = 0, \;\; Z^TP_D = P_D^TZ = 0, \;\; \text{and} \;\; AP_D^T = P_DA.$$

We start with a result for the deflation preconditioner which helps to compare the deflation and the balancing preconditioner.

PROPOSITION 2.6. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Let $Z \in \mathbb{R}^{n\times r}$ with rank$Z = r$. Then*

$$\sigma(P_D^TM^{-1}P_DA) = \sigma(M^{-1}P_DA).$$

Proof. Since $AP_D^T = P_DA$ the following identities hold:

$$\sigma(P_D^TM^{-1}P_DA) = \sigma(M^{-1}P_DAP_D^T) = \sigma(M^{-1}P_D^2A) = \sigma(M^{-1}P_DA). \qquad \square$$

Using Proposition 2.6 we obtain the following theorem.

THEOREM 2.7. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Let $Z \in \mathbb{R}^{n\times r}$ with rank$Z = r$. Then the preconditioner defined in* (1.1) *and* (1.4) *satisfies*

$$\lambda_n(M^{-1}P_DA) \le \lambda_n(P_BA), \tag{2.5}$$

$$\lambda_{r+1}(M^{-1}P_DA) \ge \lambda_1(P_BA). \tag{2.6}$$

Proof. We can write P_B as

$$P_B = P_D^TM^{-1}P_D + ZE^{-1}Z^T.$$

Thus

$$A^{\frac{1}{2}}P_BA^{\frac{1}{2}} = A^{\frac{1}{2}}P_D^TM^{-1}P_DA^{\frac{1}{2}} + A^{\frac{1}{2}}ZE^{-1}Z^TA^{\frac{1}{2}}.$$

Since $A^{\frac{1}{2}}ZE^{-1}Z^TA^{\frac{1}{2}}$ is a symmetric positive semidefinite matrix of rank r, we obtain with Lemma 2.2

$$\lambda_i(P_BA) = \lambda_i(A^{\frac{1}{2}}P_BA^{\frac{1}{2}}) \ge \lambda_i(A^{\frac{1}{2}}P_D^TM^{-1}P_DA^{\frac{1}{2}}) = \lambda_i(P_D^TM^{-1}P_DA).$$

Using Lemma 2.3 we get

$$\lambda_{r+1}(P_D^TM^{-1}P_DA) \ge \lambda_1(P_BA).$$

Using Proposition 2.6 we get the desired result. $\square$

It follows from Theorem 2.7 that

$$\kappa(P_B A) \geq \kappa_{eff}(M^{-1}P_D A),$$

so the convergence bound based on the effective condition number implies that preconditioned deflated CG never converges slower than CG preconditioned by the balancing preconditioner.

It appears that the results given in Theorem 2.5 can be generalized to general projection vectors.

THEOREM 2.8. *Suppose that the spectrum of $M^{-1}P_D A$ is given by*

$$spectrum(M^{-1}P_D A) = \{0, \ldots, 0, \mu_{r+1}, \ldots, \mu_n\}.$$

Then

$$spectrum(P_B A) = \{1, \ldots, 1, \mu_{r+1}, \ldots, \mu_n\}.$$

Proof. We know that $M^{-1}P_D AZ = 0$, so the eigenvectors corresponding to the zero eigenvalues of $M^{-1}P_D A$ are equal to $\{z_1, \ldots, z_r\}$. On the other hand, it is easy to check that

$$P_B AZ = (P_D^T M^{-1} P_D + ZE^{-1}Z^T)AZ = P_D^T M^{-1} P_D AZ + ZE^{-1}Z^T AZ = Z.$$

This implies that $\{z_1, \ldots, z_r\}$ are the eigenvectors corresponding to the eigenvalues of $P_B A$, which are equal to 1.

Now we consider the eigenvalue μ_i, with $r+1 \leq i \leq n$. Suppose v_i is the corresponding eigenvector of $M^{-1}P_D A$, and thus $M^{-1}P_D Av_i = \mu_i v_i$. Since

$$M^{-1}P_D Av_i = M^{-1}P_D^2 Av_i = M^{-1}P_D AP_D^T v_i = \mu_i v_i \neq 0,$$

the vector $P_D^T v_i$ is nonzero. Using this vector, it follows that

$$P_B A(P_D^T v_i) = (P_D^T M^{-1} P_D + ZE^{-1}Z^T)AP_D^T v_i \tag{2.7}$$

$$= P_D^T M^{-1} P_D AP_D^T v_i + ZE^{-1}Z^T AP_D^T v_i \tag{2.8}$$

$$= P_D^T M^{-1} P_D^2 Av_i = P_D^T M^{-1} P_D Av_i = \mu_i P_D^T v_i. \tag{2.9}$$

So the vectors $P_D^T v_i$ are eigenvectors of $P_B A$ corresponding to the eigenvalues μ_i. ∎

Thus both preconditioners lead to almost the same spectra with the same clustering. The zero eigenvalues of the deflation preconditioned system are replaced by eigenvalues which are one if the balancing preconditioner is used.

3. Comparing the norm of the residuals. In order to make a more detailed comparison of the deflation operator and the balancing preconditioner for general projection vectors we start to compare the vector spaces which contain the approximations of both methods. Using CG with P_B as preconditioner and start vector $x_{0,B} = 0$, it is well known that

$$x_{k,B} \in K^k\{P_B A, P_B b\},$$

where the Krylov subspace $K^k\{P_B A, P_B b\} = span\{P_B b, P_B AP_B b, \ldots, (P_B A)^{k-1}P_B b\}$.

THEOREM 3.1. *The Krylov space used in the CG method with P_B as preconditioner and start vector $x_{0,B} = 0$ has the following property:*

$$K^k\{P_B A, P_B b\} \subset span\{ZE^{-1}Z^T b, P_D^T M^{-1} P_D b, \ldots, P_D^T (M^{-1}P_D A)^{k-1} M^{-1} P_D b\}. \tag{3.1}$$

Proof. To start the proof we first note that

$$P_B b = P_D^T M^{-1} P_D b + Z E^{-1} Z^T b.$$

Therefore, the property holds for $k = 1$. For $k = 2$ we note that

$$P_B A P_B b = (P_D^T M^{-1} P_D + Z E^{-1} Z^T) A (P_D^T M^{-1} P_D + Z E^{-1} Z^T) b.$$

Writing out the various terms on the right-hand side one obtains

$$Z E^{-1} Z^T A Z E^{-1} Z^T b = Z E^{-1} Z^T b,$$

$$P_D^T M^{-1} P_D A P_D^T M^{-1} P_D b = P_D^T M^{-1} P_D P_D A M^{-1} P_D b = P_D^T M^{-1} P_D A M^{-1} P_D b,$$

where we have used that $A P_D^T = P_D A$ and $P_D^2 = P_D$. Finally, the terms

$$P_D^T M^{-1} P_D A Z E^{-1} Z^T \text{ and } Z E^{-1} Z^T A P_D^T M^{-1} P_D$$

are both zero because they contain the combination $P_D A Z = 0$ or $Z^T A P_D^T = (P_D A Z)^T = 0$. Repeating this argument for $(P_B A)^i P_B b$ for $i = 3, \ldots, k-1$ proves the theorem. ∎

With respect to the approximation using preconditioned CG combined with deflation, we note that $x = (I - P_D^T)x + P_D^T x = Z E^{-1} Z^T b + P_D^T x$. So after k iterations of preconditioned CG applied to $A P_D^T x = P_D A x = P_D b$ we get the approximation $\tilde{x}_{k,D}$. The approximation $x_{k,D}$ of the solution vector x is then given by $x_{k,D} = Z E^{-1} Z^T b + P_D^T \tilde{x}_{k,D}$. The vector $x_{k,D}$ is contained in the following space:

$$x_{k,D} \in Z E^{-1} Z^T b + span\{P_D^T M^{-1} P_D b, \ldots, P_D^T (M^{-1} P_D A)^{k-1} M^{-1} P_D b\}.$$

This implies that both approximations are elements of the same space. So the difference in quality of the approximation only depends on which norm is minimized.

LEMMA 3.2. *For the deflation iterates $x_{k,D}$ and $\tilde{x}_{k,D}$ with start vector $\tilde{x}_{0,D} = 0$ the following optimality property holds:*

$$\|x - x_{k,D}\|_A = \|P_D^T (x - \tilde{x}_{k,D})\|_A = \min_{\xi \in K^k\{M^{-1} P_D A, M^{-1} P_D b\}} \|P_D^T (x - \xi)\|_A. \tag{3.2}$$

Proof. The first equality follows from the fact that $x = (I - P_D^T)x + P_D^T x$. If CG is applied to the preconditioned system

$$L^{-1} P_D A L^{-T} y = L^{-1} P_D b, \tag{3.3}$$

the following expression holds:

$$\|\tilde{y} - y_k\|_{L^{-1} P_D A L^{-T}} = \min_{\eta \in K^k\{L^{-1} P_D A L^{-T}, L^{-1} P_D b\}} \|\tilde{y} - \eta\|_{L^{-1} P_D A L^{-T}}, \tag{3.4}$$

where $\tilde{y}$ is a solution of (3.3). Note that $\tilde{x} = L^{-T} \tilde{y}$ is a solution of $P_D A x = P_D b$. Rewriting (3.4) with $\xi = L^{-T} \eta$ leads to

$$\|L^T (\tilde{x} - \tilde{x}_{k,D})\|_{L^{-1} P_D A L^{-T}} = \min_{\xi \in K^k\{M^{-1} P_D A, M^{-1} P_D b\}} \|L^T (\tilde{x} - \xi)\|_{L^{-1} P_D A L^{-T}}.$$

Using the equalities

$$\|L^T (\tilde{x} - \tilde{x}_{k,D})\|^2_{L^{-1} P_D A L^{-T}} = (\tilde{x} - \tilde{x}_{k,D})^T P_D A (\tilde{x} - \tilde{x}_{k,D})$$
$$= (\tilde{x} - \tilde{x}_{k,D})^T P_D^2 A (\tilde{x} - \tilde{x}_{k,D}) = \|P_D^T (\tilde{x} - \tilde{x}_{k,D})\|^2_A = \|P_D^T (x - \tilde{x}_{k,D})\|^2_A$$

leads to the proof of the lemma. ∎

THEOREM 3.3. *Let $x_{k,D}$ and $\tilde{x}_{k,D}$ be the deflation iterates with start vector $\tilde{x}_{0,D} = 0$. For every $x_k \in span\{ZE^{-1}Z^Tb, P_D^TM^{-1}P_Db, \ldots, P_D^T(M^{-1}P_DA)^{k-1}M^{-1}P_Db\}$ the following inequality holds:*

$$\|x - x_{k,D}\|_A \leq \|x - x_k\|_A.$$

Proof. We decompose x_k as follows:

$$x_k = \alpha ZE^{-1}Z^Tb + P_D^T\xi, \text{ where } \xi \in K^k\{M^{-1}P_DA, M^{-1}P_Db\}.$$

Substituting this into $\|x - x_k\|_A^2$ shows that

$$\|x - x_k\|_A^2 = \|x - \alpha ZE^{-1}Z^Tb - P_D^T\xi\|_A^2.$$

Using the equation $x = (I - P_D^T)x + P_D^Tx = ZE^{-1}Z^Tb + P_D^Tx$ we obtain

$$\begin{aligned}\|x - x_k\|_A^2 &= \|(1-\alpha)ZE^{-1}Z^Tb - P_D^T(x-\xi)\|_A^2 \\ &= (1-\alpha)^2\|ZE^{-1}Z^Tb\|_A^2 + \|P_D^T(x-\xi)\|_A^2 \\ &\quad + (1-\alpha)b^TZE^{-1}Z^TAP_D^T(x-\xi) \\ &\quad + (1-\alpha)(x-\xi)^TP_DAZE^{-1}Z^Tb.\end{aligned}$$

The last two terms are equal to zero, because $Z^TAP_D^T = (P_DAZ)^T = 0$. For $x_{k,D}$ we know that $\alpha = 1$. This together with Lemma 3.2 implies

$$\|x - x_{k,D}\|_A^2 \leq (1-\alpha)^2\|ZE^{-1}Z^Tb\|_A^2 + \|P_D^T(x-\xi)\|_A^2 = \|x - x_k\|_A^2,$$

where $\xi \in K^k\{M^{-1}P_DA, M^{-1}P_Db\}$. ∎

Theorems 3.1 and 3.3 imply the following theorem.

THEOREM 3.4. *The iterates $x_{k,D}$ and $x_{k,B}$ of the CG-method with start vector zero and preconditioned by the deflation preconditioner and the balancing preconditioner, respectively, satisfy*

$$\|x - x_{k,D}\|_A \leq \|x - x_{k,B}\|_A.$$

Next we are able to prove that using a certain start vector the iterates $x_{k,D}$ are equal to the $x_{k,B}$.

THEOREM 3.5. *Using $x_{0,B} = ZE^{-1}Z^Tb$ and $\tilde{x}_{0,D} = 0$ it follows that $x_{k,D} = x_{k,B}$.*

Proof. Using the start vector $x_{0,B} = ZE^{-1}Z^Tb$ it appears that

$$r_{0,B} = b - Ax_{0,B} = (I - AZE^{-1}Z^T)b = P_Db.$$

This implies that the Krylov subspace is given by $K^k\{P_BA, P_BP_Db\}$. For $k = 1$ it follows from $P_D^2 = P_D$ and $Z^TP_D = Z^T(I - AZE^{-1}Z^T) = 0$ that

$$(P_D^TM^{-1}P_D + ZE^{-1}Z^T)P_Db = P_D^TM^{-1}P_Db.$$

For $k = 2$ we know from the proof of Theorem 3.1 that

$$P_BAP_BP_Db = ZE^{-1}Z^TP_Db + P_D^TM^{-1}P_DAM^{-1}P_D^2b.$$

Note that $Z^TP_D = Z^T(I - AZE^{-1}Z^T) = 0$, and thus

$$P_BAP_BP_Db = P_D^TM^{-1}P_DAM^{-1}P_Db.$$

Repeating this argument shows that

$$\begin{aligned} K^k\{P_BA, P_BP_Db\} &= span\{P_D^TM^{-1}P_Db, \ldots, P_D^T(M^{-1}P_DA)^{k-1}M^{-1}P_Db\} \\ &= P_D^TK^k\{M^{-1}P_DA, M^{-1}P_Db\}. \end{aligned}$$

We again use the fact that CG combined with the balancing preconditioner minimizes

$$(x - x_{k,B})^TA(x - x_{k,B}),$$

where

$$x_{k,B} = ZE^{-1}Z^Tb + P_D^T\xi, \text{ and } \xi \in K^k\{M^{-1}P_DA, M^{-1}P_Db\}$$

due to the choice of the start vector $x_{0,B} = ZE^{-1}Z^Tb$. We have that

$$x - x_{k,B} = x - ZE^{-1}Z^Tb - P_D^T\xi = P_D^T(x - \xi).$$

But by Lemma 3.2 the optimal ξ is nothing other than $\tilde{x}_{k,D}$. Thus we obtain

$$x - x_{k,B} = P_D^T(x - \tilde{x}_{k,D}).$$

Since $x = ZE^{-1}Z^Tb + P_D^Tx$ we get

$$x_{k,D} = ZE^{-1}Z^Tb + P_D^T\tilde{x}_{k,D} = x_{k,B}. \qquad \square$$

Using the identity $x_{k,D} = x_{k,B}$ it is easy to see that Theorem 2.11 of [16] implies that the balancing preconditioner with $x_{0,B} = ZE^{-1}Z^Tb$ never converges slower than the additive coarse grid preconditioner.

In the following we give a more detailed analysis of the preconditioned CG-method for both preconditioners if the above start vectors are used. We prove which quantities in the preconditioned CG-algorithm (PCG) are the same for both preconditioners and which are different. To make this paper self-contained we repeat the PCG-algorithm.

PCG-ALGORITHM for $Ax = b$ with preconditioner M^{-1}.
$r_0 := b - Ax_0, \quad z_0 = M^{-1}r_0, \quad p_0 := z_0$
For $j = 0, 1, \ldots$ until convergence, do
$\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
$x_{j+1} := x_j + \alpha_jp_j$
$r_{j+1} := r_j - \alpha_jAp_j$
$z_{j+1} := M^{-1}r_{j+1}$
$\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
$p_{j+1} := z_{j+1} + \beta_jp_j$
end

Moreover, we need the next proposition.

PROPOSITION 3.6. *Let P_D, P_B, and M^{-1} be defined as above. Then*

$$P_D^TP_BP_D = P_D^TM^{-1}P_D = P_D^TP_B = P_BP_D. \tag{3.5}$$

Proof. Since $P_D = I - AZE^{-1}Z^T$ we have $P_D^TZ = Z - ZE^{-1}Z^TAZ = 0$. Hence,

$$P_D^TP_BP_D = P_D^{T^2}M^{-1}P_D^2 + P_D^TZE^{-1}ZP_D = P_D^TM^{-1}P_D.$$

Similarly,

$$P_D^TP_B = P_D^{T^2}M^{-1}P_D + P_DZE^{-1}ZP_D = P_D^TM^{-1}P_D.$$

Since $P_D^TM^{-1}P_D$ is symmetric we also have $P_D^TM^{-1}P_D = P_BP_D$. $\square$

Now we can prove the following theorem.

THEOREM 3.7. *Using the PCG-algorithm with the balancing preconditioner P_B and $x_{0,B} = ZE^{-1}Z^Tb$ on one side and with the deflation preconditioner $M^{-1}P_D$ and $\tilde{x}_{0,D} = 0$ on the other side we have, for all j,*

$$\begin{aligned}(r_{j,D}, z_{j,D}) &= (r_{j,B}, z_{j,B}),\\ (P_DAp_{j,D}, p_{j,D}) &= (Ap_{j,B}, p_{j,B}),\\ r_{j+1,D} &= r_{j+1,B},\\ z_{j+1,B} &= P_D^T z_{j+1,D},\\ p_{j+1,B} &= P_D^T p_{j+1,D},\\ \beta_{j,B} &= \beta_{j,D},\\ x_{j+1,B} &= x_{j+1,D} = ZE^{-1}Z^T + P_D^T\tilde{x}_{j+1,D}.\end{aligned}$$

Proof. If we use PCG for

$$P_DAx = P_Db$$

with preconditioner M^{-1} and start vector $x_0 = 0$, we obtain

$$\begin{aligned}&x_{0,D} = 0, \quad r_{0,D} = P_Db, \quad z_{0,D} = M^{-1}P_Db,\\ &p_{0,D} = z_{0,D} = M^{-1}P_Db, \quad \alpha_{0,D} = \frac{(r_{0,D}, z_{0,D})}{(P_DAp_{0,D}, p_{0,D})},\\ &\tilde{x}_{1,D} = 0 + \alpha_{0,D}M^{-1}P_Db,\\ &x_{1,D} = ZE^{-1}Z^Tb + \alpha_{0,D}P_D^TM^{-1}P_Db.\end{aligned}$$

If we use PCG for

$$Ax = b$$

with preconditioner P_B and start vector $x_0 = ZE^{-1}Z^Tb$, we obtain

$$\begin{aligned}x_{0,B} = ZE^{-1}Z^Tb, \quad r_{0,B} = P_Db, \quad z_{0,B} = P_BP_Db,\\ p_{0,B} = z_{0,B} = P_BP_Db, \quad \alpha_{0,B} = \frac{(r_{0,B}, z_{0,B})}{(Ap_{0,B}, p_{0,B})},\\ x_{1,B} = ZE^{-1}Z^Tb + \alpha_{0,B}P_BP_Db.\end{aligned}$$

Obviously, we have for all iterates

$$P_Dr_{j+1,D} = P_D(P_Db - P_DAx_{j+1,D}) = r_{j+1,D}. \tag{3.6}$$

The identity

$$P_BP_D = P_D^TM^{-1}P_D^2 + ZE^{-1}Z^TP_D = P_D^TM^{-1}P_D$$

is frequently used in the following analysis.

Next, we prove the following identities by induction:

$$\begin{aligned}(r_{j,D}, z_{j,D}) = (r_{j,B}, z_{j,B}), \quad (r_{j+1,D}, z_{j+1,D}) = (r_{j+1,B}, z_{j+1,B}),\\ (P_DAp_{j,D}, p_{j,D}) = (Ap_{j,B}, p_{j,B}),\end{aligned}$$

$$
\begin{aligned}
r_{j+1,D} &= r_{j+1,B},\\
z_{j+1,B} &= P_D^T z_{j+1,D},\\
p_{j+1,B} &= P_D^T p_{j+1,D},\\
\beta_{j,B} &= \beta_{j,D},\\
x_{j+1,B} &= x_{j+1,D} = ZE^{-1}Z^T + P_D^T \tilde{x}_{j+1,D}.
\end{aligned}
$$

In the following we use Proposition 3.6 and (3.6). For $j = 0$ we have

$$
(r_{0,D}, z_{0,D}) = b^T P_D^T M^{-1} P_D b = b^T P_D^T P_B P_D b = (r_{0,B}, z_{0,B}).
$$

$$
\begin{aligned}
(P_D A p_{0,D}, p_{0,D}) &= b^T P_D^T M^{-1} P_D A M^{-1} P_D b\\
&= b^T P_D^T M^{-1} P_D A P_D^T M^{-1} P_D b\\
&= b^T P_D^T P_B A P_B P_D b\\
&= (A p_{0,B}, p_{0,B}).
\end{aligned}
$$

Hence, $\alpha_{0,D} = \alpha_{0,B}$.

$$
\begin{aligned}
r_{1,D} &= P_D b - \alpha_{0,D} P_D A M^{-1} P_D b = P_D b - \alpha_{0,D} A P_D^T M^{-1} P_D b\\
&= P_D b - \alpha_{0,B} A P_B P_D b\\
&= r_{1,B}.
\end{aligned}
$$

$$
\begin{aligned}
x_{1,D} &= ZE^{-1}Z^T b + \alpha_{0,D} P_D^T M^{-1} P_D b\\
&= ZE^{-1}Z^T b + \alpha_{0,B} P_B P_D b = x_{1,B}.
\end{aligned}
$$

$$
\begin{aligned}
P_D^T z_{1,D} &= P_D^T M^{-1} r_{1,D} = P_D^T M^{-1} P_D r_{1,D}\\
&= P_B P_D r_{1,D} = P_B r_{1,B} = z_{1,B}.
\end{aligned}
$$

Thus

$$
\begin{aligned}
(r_{1,B}, z_{1,B}) &= (r_{1,D}, P_D^T z_{1,D})\\
&= (P_D r_{1,D}, z_{1,D}) = (r_{1,D}, z_{1,D}).
\end{aligned}
$$

Hence $\beta_{0,D} = \beta_{0,B}$. Next,

$$
\begin{aligned}
p_{1,B} &= z_{1,B} + \beta_{0,B} p_{0,B}\\
&= P_D^T z_{1,D} + \beta_{0,B} P_B P_D b\\
&= P_D^T z_{1,D} + \beta_{0,B} P_D^T M^{-1} P_D b\\
&= P_D^T (z_{1,D} + \beta_{0,D} p_{0,D})\\
&= P_D^T p_{1,D}.
\end{aligned}
$$

Now assume that the above identities hold for $j-1$ and that $(r_{j,B}, z_{j,B}) = (r_{j,D}, z_{j,D})$ holds. We then have

$$
\begin{aligned}
(A p_{j,B}, p_{j,B}) &= (A P_D^T p_{j,D}, P_D^T p_{j,D})\\
&= p_{j,D}^T P_D A P_D^T p_{j,D}\\
&= (P_D A p_{j,D}, p_{j,D}).
\end{aligned}
$$

Hence, $\alpha_{j,D} = \alpha_{j,B}$. Since

$$\begin{aligned} x_{j+1,B} &= x_{j,B} + \alpha_{j,B} p_{j,B}, \\ \tilde{x}_{j+1,D} &= \tilde{x}_{j,D} + \alpha_{j,D} p_{j,D}, \end{aligned}$$

we obtain

$$\begin{aligned} x_{j+1,D} &= ZE^{-1}Z^T + P_D \tilde{x}_{j,D} + \alpha_{j,D} P_D^T p_{j,D} \\ &= x_{j,B} + \alpha_{j,B} p_{j,B} \\ &= x_{j+1,B}. \end{aligned}$$

$$\begin{aligned} r_{j+1,B} &= r_{j,B} - \alpha_{j,B} A p_{j,B} \\ &= P_D r_{j,D} - \alpha_{j,D} A P_D^T p_{j,D} \\ &= P_D r_{j,D} - \alpha_{j,D} P_D A p_{j,D} \\ &= P_D r_{j+1,D} \\ &= r_{j+1,D}. \end{aligned}$$

Moreover,

$$\begin{aligned} z_{j+1,B} &= P_B r_{j+1,B} = P_B P_D r_{j+1,D} \\ &= P_D^T M^{-1} P_D r_{j+1,D} = P_D^T M^{-1} r_{j+1,D} \\ &= P_D^T z_{j+1,D}. \end{aligned}$$

$$\begin{aligned} (r_{j+1,B}, z_{j+1,B}) &= (P_D r_{j+1,D}, P_D^T z_{j+1,D}) \\ &= r_{j+1,D}^T P_D^T z_{j+1,D} \\ &= (r_{j+1,D}, z_{j+1,D}). \end{aligned}$$

Hence, $\beta_{j,B} = \beta_{j,D}$. Next we have

$$\begin{aligned} p_{j+1,B} &= z_{j+1,B} + \beta_{j,B} p_{j,B} \\ &= P_D^T z_{j+1,D} + \beta_{j,D} P_D^T p_{j,D} \\ &= P_D^T p_{j+1,D}, \end{aligned}$$

which completes the proof. ∎

In the following we show how the eigenvalues and the condition number of the system preconditioned by balancing behave if we choose a finer coarse grid. Therefore, let $Z_1 \in \mathbb{R}^{n\times r}$ and $Z_2 \in \mathbb{R}^{n\times s}$ with $rank Z_1 = r$ and $rank Z_2 = s$. Define

$$\begin{aligned} E_1 := Z_1^T A Z_1 \quad \text{and} \quad E_2 := Z_2^T A Z_2, \\ P_{D_1} = I - A Z_1 E_1^{-1} Z_1^T \quad \text{and} \quad P_{D_2} = I - A Z_2 E_2^{-1} Z_2^T. \end{aligned}$$

Moreover, let

$$P_{B_1} = P_{D_1}^T M^{-1} P_{D_1} + Z_1 E_1^{-1} Z_1^T \text{ and } P_{B_2} = P_{D_2}^T M^{-1} P_{D_2} + Z_2 E_2^{-1} Z_2^T. \tag{3.7}$$

We then have the following theorem.

THEOREM 3.8. *Let A and $M \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Let P_{B_1} and P_{B_2} be defined as in* (3.7). *If* $\mathrm{Im} Z_1 \subseteq \mathrm{Im} Z_2$, *then*

$$\lambda_n(P_{B_1} A) \geq \lambda_n(P_{B_2} A), \tag{3.8}$$

$$\lambda_{r+1}(P_{B_1} A) \leq \lambda_{s+1}(P_{B_2} A). \tag{3.9}$$

Moreover,

$$cond(P_{B_1}A) \geq cond(P_{B_2}A).$$

Proof. Theorem 2.12 in [16] states that

$$\lambda_n(M^{-1}P_{D_1}A) \geq \lambda_n(M^{-1}P_{D_2}A),$$
$$\lambda_{r+1}(M^{-1}P_{D_1}A) \leq \lambda_{s+1}(M^{-1}P_{D_2}A).$$

Thus, with Theorem 2.8 we get

$$\begin{aligned} cond(P_{B_1}A) &= \frac{\max(1, \lambda_n(M^{-1}P_{D_1}A))}{\min(1, \lambda_{r+1}(M^{-1}P_{D_1}A))} \\ &\geq \frac{\max(1, \lambda_n(M^{-1}P_{D_2}A))}{\min(1, \lambda_{s+1}(M^{-1}P_{D_2}A))} \\ &= cond(P_{B_2}A). \qquad \square \end{aligned}$$

If a finer grid is used as a coarse grid in the balancing preconditioner, the amount of work to solve the coarse grid system is increased. But then Theorem 3.8 states that the condition number of the system preconditioned by the balancing preconditioner decreases. In general this leads to fewer iterations, although more work is needed on the coarse grid problem.

4. Numerical experiments. In all our numerical experiments, the multiplication $y = E^{-1}b$ is done by solving y from $Ey = b$, where E is decomposed in its Cholesky factor. The choice of the boundary conditions is such that all problems have as exact solution the vector with components equal to 1. In order to make the convergence behavior representative for general problems, we chose a random vector as the starting solution instead of the zero start vector.

4.1. Artificial test problems. We apply both methods (deflation and balancing) to the Poisson equation. It appears that in the numerical experiments $\|x - x_{k,D}\|_A \leq \|x - x_{k,B}\|_A$, but for well-scaled problems the differences are very small. From Theorem 2.8 it follows that the spectrum of the balancing preconditioner consists of two parts: in one part the eigenvalues are equal to 1, and in the other part the eigenvalues are equal to the nonzero eigenvalues of the deflated matrix. This suggests that if the eigenvalues equal to 1 are interior eigenvalues, the convergence is close to the convergence of the preconditioned deflation method; otherwise these eigenvalues may influence the convergence.

Scaling properties. Note that P_DA is scaling invariant, whereas P_BA is not scaling invariant. This means that if deflation is applied to a system $\gamma Ax = \gamma b$ the effective condition number of $P_{D\gamma A}\gamma A = (I - \gamma AZ(Z^T\gamma AZ)^{-1}Z^T)\gamma A$ is independent of the scalar γ; i.e.,

$$\kappa_{eff}(P_{D\gamma A}\gamma A) = \frac{\gamma\lambda_n(P_{DA}A)}{\gamma\lambda_{r+1}(P_{DA}A)} = \kappa_{eff}(P_{DA}A),$$

whereas the condition number of $P_B\gamma A$ depends on the choice of γ,

$$\kappa(P_{B\gamma A}\gamma A) \neq \kappa(P_{BA}A).$$

To check this in practice, we do experiments with balancing using various values of γ. From Figure 4.1 it appears that the convergence of the balancing preconditioner

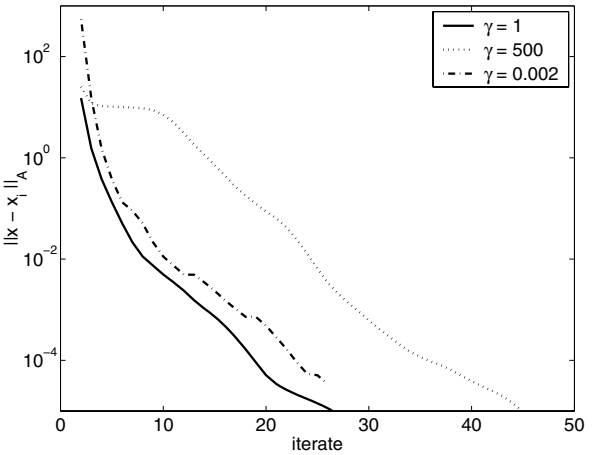


FIG. 4.1. *Comparison of the balancing preconditioner for various values of γ. The convergence of deflation is identical to the convergence of the balancing preconditioner with $\gamma = 1$.*

is worse if $\gamma \neq 1$. We note that the deflation method (for all values of γ) has the same convergence as the balancing method with $\gamma = 1$. In most preconditioning techniques an automatic scaling is used, which makes the combination scaling invariant. However, the resulting spectrum is not always clustered around 1. We used this example to illustrate what happens if the spectrum is far away from 1.

Inaccurate solution. If the dimensions of matrix E become large (i.e., many projection vectors are used), it seems to be a good idea to compute E^{-1} approximately (by an iterative method or a recursive procedure). It appears that the balancing preconditioner is insensitive to the accuracy of the approximation of E^{-1}, while deflation is sensitive to it.

To illustrate this we consider the same Poisson problem. In both examples seven projection vectors are used. We replace E^{-1} by $\tilde{E}^{-1} = (I + \epsilon R)E^{-1}(I + \epsilon R)$, where R is a symmetric $r \times r$ matrix with random elements chosen from the interval $[-\frac{1}{2}, \frac{1}{2}]$. From Figure 4.2 it follows that the convergence of the deflation preconditioner is good as long as $|\epsilon| < 10^{-6}$.

Starting solution for the balancing preconditioner. In Theorem 3.5 we have proven that $x_{k,B} = x_{k,D}$ if $x_{0,B} = ZE^{-1}Z^T b$ and $\tilde{x}_{0,D} = 0$. In this paragraph we illustrate this by numerical examples. In Figure 4.3 we plot the convergence of the balancing preconditioner with start vector $x_{0,B} = ZE^{-1}Z^T b$. It appears that the choice $\gamma = 500$ leads to the same results as $\gamma = 1$ (and deflation). Furthermore, the convergence for the choice $\gamma = 0.002$ is initially also the same, but later on the convergence becomes worse. This can be explained by rounding errors. Using the choice $\gamma = 0.002$ the eigenvalues equal to 1 are large with respect to the other eigenvalues. Initially, due to the start vector the components of the corresponding eigenvectors are zero or small. During the iterations, the perturbations in large eigencomponents increase, which leads to the same convergence as if the method is started with $x_{0,B} = 0$. To enlarge the rounding error effect we have also done experiments where the matrix E^{-1} is replaced by $\tilde{E}^{-1}$ with $\epsilon = 10^{-2}$. The results are given in Figure 4.4. Note that the same effect now appears for both values of $\gamma \neq 1$.

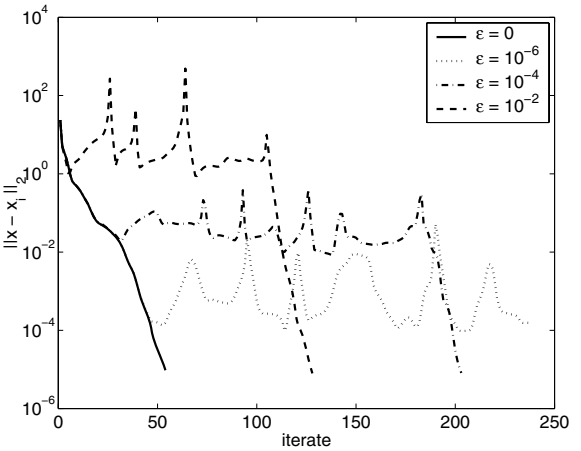


FIG. 4.2. *Convergence behavior of the deflated ICCG including perturbations.*

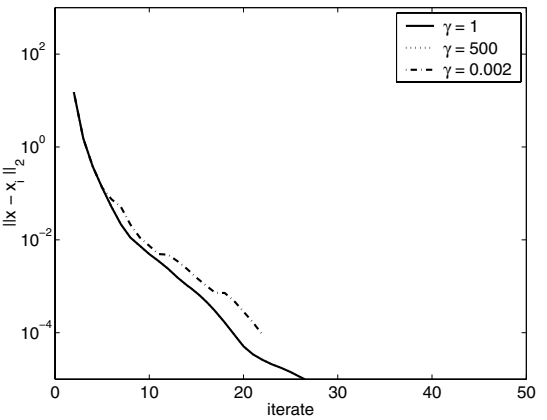


FIG. 4.3. *Convergence of the balancing preconditioner with* $x_{0,B} = ZE^{-1}Z^Tb$ *and* $\epsilon = 0$. *The convergence of deflation is identical to the convergence of the balancing preconditioner with* $\gamma = 1$.

4.2. Porous media flows. In this section we simulate a porous media oil flow in a 3-dimensional layered geometry, where the layers vary in thickness and orientation (see Figures 4.5 and 4.6 for a 4-layer problem). The fluid pressure and permeability are denoted by p and σ, respectively. The pressure p satisfies the equation

$$-\text{div}(\sigma \nabla p) = 0 \text{ on } \Omega, \tag{4.1}$$

with boundary conditions

$$p = 1 \text{ on } \partial\Omega^D \text{ (Dirichlet) and } \frac{\partial p}{\partial n} = 0 \text{ on } \partial\Omega^N \text{ (Neumann)},$$

where $\partial\Omega = \partial\Omega^D \cup \partial\Omega^N$. In this problem $\partial\Omega^D$ is the top boundary of the domain. Figure 4.5 shows a part of the earth's crust. The depth of this part varies between 3 and 6 kilometers, whereas horizontally its dimensions are 40×60 kilometers. The upper layer is a mixture of sandstone and shale and has a permeability of 10^{-4}. Below this layer, shale and sandstone layers are present with permeabilities of 10^{-7} and 10,

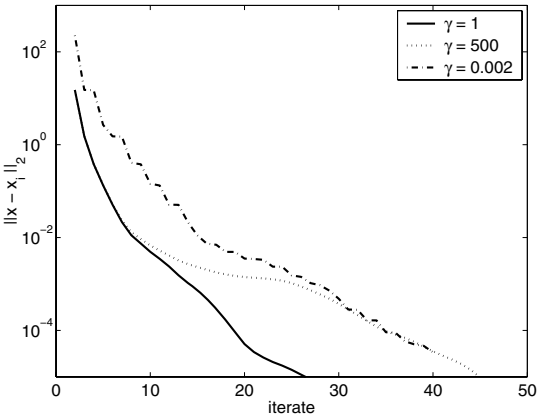


FIG. 4.4. *Convergence of the balancing preconditioner with* $x_{0,B} = ZE^{-1}Z^Tb$ *and* $\epsilon = 10^{-2}$.

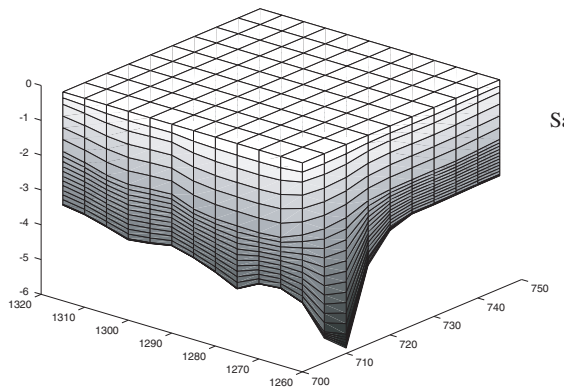


FIG. 4.5. *The geometry of an oil flow problem with 4 layers.*

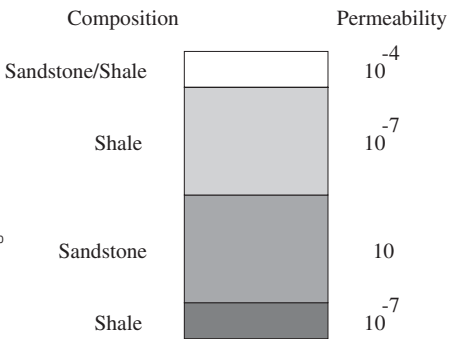


FIG. 4.6. *Permeabilities for each layer.*

TABLE 4.1
The results for the oil flow problem.

Method	Deflation	Balancing
Iterations	36	36
CPU time in seconds	6.3	9.8

respectively. An incomplete Cholesky factorization with no fill in is used as preconditioner [14]. We consider a problem with nine layers. Five sandstone layers are separated by four shale layers. Due to the Dirichlet boundary condition at the top, the preconditioned matrix has four small eigenvalues. We use four *physical* projection vectors [26] and stop if $\|r_k\|_2 \leq 10^{-5}$. Trilinear hexahedral elements are used and the total number of gridpoints is equal to 148185. The results are given in Table 4.1. It appears that the norms of the residuals for both preconditioners are the same. Due to extra work per iteration the balancing preconditioner costs more CPU time. We note that it is possible to implement the balancing preconditioner such that the costs are the same as those of deflation [11]. However, this implementation also leads to the same difficulties as deflation if the matrix E^{-1} is perturbed. The computations

are performed on an AMD Athlon, 1.4 GHz processor with 256 Mb of RAM. The code is compiled with FORTRAN g77 on LINUX.

5. Conclusions. In this paper we compared various preconditioners for the numerical solution of partial differential equations.

We have given a detailed comparison of the well-known balancing preconditioner used in domain decomposition methods and the deflation preconditioner.

We proved that both preconditioners lead to almost the same spectra. The zero eigenvalues of the deflation preconditioned system are replaced by eigenvalues which are one if the balancing preconditioner is used. Thus the effective condition number of the deflated preconditioned system is always, i.e., for all deflation vectors and all restrictions and prolongations, below or equal to the condition number of the system preconditioned by the balancing preconditioner. Moreover, we proved that the A-norm of the errors of the iterates built by the deflation preconditioner is always below the A-norm of the errors of the iterates built by the balancing preconditioner. Hence, the CG-method applied to the deflated preconditioned system never converges slower than the CG-method applied to the system preconditioned by the balancing preconditioner. Additionally, the amount of work of one iteration of the deflation preconditioned system is less than the amount of work of one iteration of the balancing preconditioned system. Hence the deflation preconditioner leads to fewer iterations and each iteration having less work.

Moreover, we established that the deflation preconditioner and the balancing preconditioner produce the same iterates if one uses certain starting vectors. More precisely, we showed which terms in the PCG-method are the same for both methods and which terms are different. Numerical results for porous media flows emphasized the theoretical results.

REFERENCES

[1] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring.* I, Math. Comp., 47 (1986), pp. 103–134.

[2] H. De Gersem and K. Hameyer, *A deflated iterative solver for magnetostatic finite element models with large differences in permeability*, Eur. Phys. J. Appl. Phys., 13 (2000), pp. 45–49.

[3] M. Dryja and O. B. Widlund, *Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems*, Comm. Pure Appl. Math., 48 (1995), pp. 121–155.

[4] M. Eiermann, O. G. Ernst, and O. Schneider, *Analysis of acceleration strategies for restarted minimal residual methods*, J. Comput. Appl. Math., 123 (2000), pp. 261–292.

[5] C. Farhat and F.-X. Roux, *A method of finite element tearing and interconnecting and its parallel solution algorithm*, Internat. J. Numer. Methods Engrg., 32 (1991), pp. 1205–1227.

[6] C. Farhat and F.-X. Roux, *Implicit parallel processing in structural mechanics*, Comput. Mech. Adv., 2 (1994), pp. 1–124.

[7] J. Frank and C. Vuik, *On the construction of deflation-based preconditioners*, SIAM J. Sci. Comput., 23 (2001), pp. 442–462.

[8] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[9] E. F. Kaasschieter, *Preconditioned conjugate gradients for solving singular systems*, J. Comput. Appl. Math., 24 (1988), pp. 265–275.

[10] L. Yu. Kolotilina, *Preconditioning of systems of linear algebraic equations by means of twofold deflation.* I. *Theory*, J. Math. Sci. (New York), 89 (1998), pp. 1652–1689.

[11] J. Mandel, *Balancing domain decomposition*, Comm. Numer. Methods Engrg., 9 (1993), pp. 233–241.

[12] J. Mandel and M. Brezina, *Balancing domain decomposition for problems with large jumps in coefficients*, Math. Comp., 216 (1996), pp. 1387–1401.

[13] L. Mansfield, *Damped Jacobi preconditioning and coarse grid deflation for conjugate gradient iteration on parallel computers*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1314–1323.

[14] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[15] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[16] R. Nabben and C. Vuik, *A comparison of deflation and coarse grid correction applied to porous media flow*, SIAM J. Numer. Anal., 42 (2004), pp. 1631–1647.

[17] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM J. Numer. Anal., 24 (1987), pp. 355–365.

[18] L. F. Pavarino and O. B. Widlund, *Balancing Neumann-Neumann methods for incompressible Stokes equations*, Comm. Pure Appl. Math., 55 (2002), pp. 302–335.

[19] A. Quarteroni and A. Valli, *Domain Decomposition Methods for Partial Differential Equations*, Oxford Science Publications, Oxford, UK, 1999.

[20] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition*, Cambridge University Press, Cambridge, UK, 1996.

[21] A. Toselli and W. Widlund, *Domain Decomposition Methods*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.

[22] A. van der Sluis and H. A. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.

[23] F. Vermolen, C. Vuik, and A. Segal, *Deflation in preconditioned conjugate gradient methods for finite element problems*, in Conjugate Gradient and Finite Element Methods, M. Křižek, P. Neittaanmäki, R. Glowinski, and S. Korotov, eds., Springer-Verlag, Berlin, 2004, pp. 103–129.

[24] C. Vuik, A. Segal, J. A. Meijerink, and G. T. Wijma, *The construction of projection vectors for a deflated ICCG method applied to problems with extreme contrasts in the coefficients*, J. Comput. Phys., 172 (2001), pp. 426–450.

[25] C. Vuik, A. Segal, and J. A. Meijerink, *An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients*, J. Comput. Phys., 152 (1999), pp. 385–403.

[26] C. Vuik, A. Segal, L. El Yaakoubi, and E. Dufour, *A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients*, Appl. Numer. Math., 41 (2002), pp. 219–233.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1760–1786
© 2006 Society for Industrial and Applied Mathematics

A COMPARATIVE STUDY OF ITERATIVE SOLVERS EXPLOITING SPECTRAL INFORMATION FOR SPD SYSTEMS*

LUC GIRAUD†, DANIEL RUIZ‡, AND AHMED TOUHAMI§

Abstract. When solving the symmetric positive definite (SPD) linear system $\mathbf{Ax}^\star = \mathbf{b}$ with the conjugate gradient method, the smallest eigenvalues in the matrix $\mathbf{A}$ often slow down the convergence. Consequently if the smallest eigenvalues in $\mathbf{A}$ could somehow be "removed," the convergence may be improved. This observation is of importance even when a preconditioner is used, and some extra techniques might be investigated to further improve the convergence rate of the conjugate gradient on the given preconditioned system. Several techniques have been proposed in the literature that consist of either updating the preconditioner or enforcing conjugate gradient to work in the orthogonal complement of an invariant subspace associated with the smallest eigenvalues. The goal of this work is to compare several of these techniques in terms of numerical efficiency. Among various possibilities, we exploit the PARTIAL SPECTRAL FACTORIZATION algorithm presented in [M. Arioli and D. Ruiz, Technical Report RAL-TR-2002-021, Rutherford Appleton Laboratory, Atlas Center, Didcot, Oxfordshire, England, 2002] to compute an orthonormal basis of a near-invariant subspace of $\mathbf{A}$ associated with the smallest eigenvalues. This eigeninformation is used in combination with different solution techniques. In particular we consider the deflated version of conjugate gradient. As representative of techniques exploiting the spectral information to update the preconditioner we consider also the approaches that attempt to shift the smallest eigenvalues close to one where most of the eigenvalues of the preconditioned matrix should be located. Finally, we consider an algebraic two-grid scheme inspired by ideas from the multigrid philosophy. In this paper, we describe these various variants and we compare their numerical behavior on a set of model problems from Matrix Market or arising from the discretization via the finite element technique of some two-dimensional (2D) heterogeneous diffusion PDE problems. We discuss their numerical efficiency, computational complexity, and sensitivity to the accuracy of the eigencalculation.

Key words. Chebyshev polynomials, block Lanczos method, conjugate gradient method, filtering, deflation, spectral preconditioning, two-grid schemes, iterative methods

AMS subject classifications. 65F10, 65F15, 65F50, 65H10, 65N22

DOI. 10.1137/040608301

1. Introduction.

Many problems in engineering and physics require solving successively linear systems of the form

$$\mathbf{Ax}^\star = \mathbf{b} \tag{1.1}$$

with the same coefficient matrix $\mathbf{A}$ but different right-hand sides $\mathbf{b}$ not available simultaneously. We can reduce the cost of solving subsequent systems in the sequence by recycling some spectral information about the matrix $\mathbf{A}$ extracted from previous solves. This can be of particular interest, for instance, in the asymptotic development of solutions of thick plates problems. Indeed, this application requires solving successive symmetric positive definite (SPD) linear systems with the same matrix, but where each solution at a given order in the development depends upon the previous

*Received by the editors May 14, 2004; accepted for publication (in revised form) October 28, 2005; published electronically February 3, 2006. This work has been partially supported by the ACI GRID TLSE project from the French Ministry of Research and Education grant n° 02 2 0678.
http://www.siam.org/journals/sisc/27-5/60830.html

†CERFACS, 42 Avenue G. Coriolis, 31057 Toulouse Cedex, France. Current address: ENSEEIHT-IRIT, 2 rue C. Camichel, 31071 Toulouse Cedex, France (luc.giraud@enseeiht.fr).

‡ENSEEIHT-IRIT, 2 rue C. Camichel, 31071 Toulouse Cedex, France (daniel.ruiz@enseeiht.fr).

§Corresponding author. ENSEEIHT-IRIT, 2 rue C. Camichel, 31071 Toulouse Cedex, France (Ahmed.Touhami@enseeiht.fr).

order [22]. This type of situation can also occur in the solution of nonlinear systems by means of an approximate Newton method, where the approximate Jacobian matrix is frozen during several iterations and just the right-hand side is updated sequentially.

When the matrix $\mathbf{A}$ is large, sparse, and SPD, the Krylov solver of choice is the conjugate gradient [20], which constructs the kth iterate $\mathbf{x}^{(k)}$ as an element of

$$\mathbf{x}^{(0)} + \mathrm{Span}\{\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{(\mathrm{k}-1)}\mathbf{r}^{(0)}\}$$

so that the $\mathbf{A}$-norm of the error, $||\mathbf{e}^{(k)}|| = ||\mathbf{x}^{(k)} - \mathbf{x}^{\star}||_{\mathbf{A}}$, is minimized, where $\mathbf{x}^{\star}$ is the exact solution of (1.1). The convergence of the conjugate gradient method is governed by the eigenvalues distribution in $\mathbf{A}$ [37]. A coarse characterization is given by the bound on the rate of convergence of the conjugate gradient method given by Concus, Golub, and O'Leary [8] viz.

$$||\mathbf{e}^{(k)}||_{\mathbf{A}} \leq \mathbf{2}\, ||\mathbf{e}^{(0)}||_{\mathbf{A}} \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^{k}, \tag{1.2}$$

where $\kappa = \lambda_{\max}/\lambda_{\min}$ denotes the condition number of $\mathbf{A}$. From this bound, it can be seen that increasing the size of the smallest eigenvalues will substantially decrease the upper bound (1.2) on the convergence rate of the conjugate gradient.

Additionally, in the presence of clusters of eigenvalues at the extreme of the spectrum of $\mathbf{A}$, combined with some ill-conditioning, the classical conjugate gradient may show a convergence history with sequences of plateaux and sharp drops. These plateaux (see [37]) are well understood and correspond to the discovery of separate eigenvalues within a cluster. The problem is that these plateaux can be rather large, in terms of the number of iterations, even when these extreme clusters incorporate only a few eigenvalues. Block conjugate gradient can help to reduce the size of these plateaux but in general cannot remove them completely (see, e.g., [1, 6, 30]). Thus removing the bad effect of small eigenvalues in the matrix $\mathbf{A}$ might have a beneficial effect on the convergence.

In the following, we assume that $\mathbf{A}$ is a large SPD preconditioned matrix with a spectrum largely clustered, and the goal is to solve several linear systems in sequence, with the same matrix $\mathbf{A}$ but with changing right-hand sides that are computed sequentially. In this context, several techniques have been proposed in the literature to improve the convergence of conjugate gradient type methods on the solution of each system; these techniques consist of either updating the preconditioner or enforcing conjugate gradient to work in the orthogonal complement of an invariant subspace associated with the smallest eigenvalues. The aim of this work is to compare several of these techniques in terms of numerical efficiency.

In this paper, as one possible choice amongst several, we exploit the "partial spectral factorization" algorithm introduced in [2] to compute an orthonormal basis $\mathbf{W}$ of a near invariant subspace of $\mathbf{A}$ associated with the smallest eigenvalues. This basis is then exploited in combination with different solution techniques. In particular we consider the deflated version of conjugate gradient described in [34]. This algorithm exploits the link between the Lanczos algorithm and the standard conjugate gradient algorithm, and is mathematically equivalent to the Nicolaide's algorithm which is derived from a "deflated" Lanczos procedure [29]. In [23] Kolotilina uses a twofold deflation technique for simultaneously deflating the k largest and the k smallest eigenvalues using an appropriate deflating subspace of dimension k. An alternative technique consists in applying the deflation to eliminate the bad effects on

the convergence caused by the remaining small eigenvalues [10, 39]. This technique is used in [9] to solve electromagnetic problems with large jumps in the coefficients. Similar ideas have also been applied to nonsymmetric problems using the GMRES algorithm (see, e.g., [3, 7, 12, 25, 26, 27]). As representative of techniques exploiting spectral information to update the preconditioner, we consider the approach proposed in [4] that attempts to shift the smallest eigenvalues close to one, where most of the eigenvalues of the preconditioned matrix should be located. Finally, we consider an algebraic two-grid scheme inspired by ideas from the multigrid philosophy.

The outline of the paper is as follows. We devote section 2 to various solution techniques that exploit some spectral information extracted from the matrix $\mathbf{A}$, and we indicate how they can be combined with "partial spectral factorization" to construct efficient iterative solvers. In section 3, we compare these techniques in terms of numerical behavior on a set of model problems from Matrix Market, or arising from the discretization via the finite element technique of some heterogeneous diffusion PDEs in two dimensions. In section 4, we analyze in more detail the computational complexity of these different solution techniques, and we investigate how they can be used efficiently to solve a sequence of linear systems with the same matrix but changing right-hand sides. Finally, we conclude with some remarks and perspectives in section 5.

2. Alternative techniques for the solution of linear systems exploiting partial spectral decomposition. In this section, we consider solution techniques that exploit some spectral information of the matrix $\mathbf{A}$. The common general idea beneath all such techniques, as we will see in detail in the following paragraphs, is to use some precomputed *near*-invariant subspace $\mathbf{W}$, associated with the smallest eigenvalues in $\mathbf{A}$, to build a projection onto the orthogonal complement of $\mathbf{W}$ in some appropriate norm. This projector is then exploited to maintain, implicitly or explicitly, throughout the iterations, the computed residuals in this orthogonal complement. In this section, we discuss six different solution techniques that exploit this general idea, but each in a different manner.

A first idea can be to decompose the solution of (1.1) into two parts. The first can be obtained directly with an oblique projection of the right-hand side $\mathbf{b}$ onto the orthogonal complement of $\mathbf{W}$. The second part of this solution can then be obtained with the use of conjugate gradient with the first part of this solution as a starting guess, as detailed in section 2.1.

The following three techniques make use of such a projector explicitly throughout the iterations. The first one, as discussed in section 2.2, is to augment the Krylov space generated in the conjugate gradient iterations with the basis $\mathbf{W}$, and resumes in a deflation at each iteration of the generated Krylov vectors. The second one, introduced in section 2.3, is to run the conjugate gradient algorithm onto the projected system $\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$, where $\mathcal{P}$ is a projector onto the orthogonal complement of $\mathbf{W}$ that commutes in some way with $\mathbf{A}$. The last technique, detailed in section 2.4, aims at building a preconditioner in $\mathbf{A}$ that shifts, by adding the value 1, those eigenvalues linked to the *near*-invariant subspace $\mathbf{W}$, and to run a preconditioned conjugate gradient in the usual way to compute the solution of (1.1).

In section 2.5, we describe the Chebyshev-based filtering technique combined with some Lanczos process used for computation of the partial spectral factorization with some chosen level of accuracy.

In section 2.6 Chebyshev iteration can also be used to compute the part of the solution associated with the largest eigenvalues in $\mathbf{A}$. The second part of the solution

is obtained with an oblique projection of the initial residual onto the *near*-invariant subspace, in order to get those eigencomponents in the solution corresponding to the smallest eigenvalues of $\mathbf{A}$, and exploits the precomputed partial spectral factorization of $\mathbf{A}$.

Finally, in section 2.7, we introduce algebraic two-grid methods which provide an alternative interpretation of some of the techniques discussed above and offer the possibility of deriving a family of iterative techniques based on the precomputation of the approximate invariant subspace $\mathbf{W}$.

2.1. Conjugate gradient with an oblique projection for the initial guess. As proposed in [32, 33, 34, 38], once an approximation of the invariant subspace $\mathbf{W}$ associated with the smallest eigenvalues is obtained, we can use it for the computation of further solutions. The idea is to perform an oblique projection of the initial residual onto this invariant subspace in order to get the eigencomponents in the solution corresponding to the smallest eigenvalues. The update of the initial guess is defined by (2.1). A classical conjugate gradient is performed to compute the remaining part of the solution vector. We call this algorithm INIT-CG (see, e.g., [13]).

$$\mathbf{x}^{(0)} = \mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{b}. \tag{2.1}$$

Note that we do not project the given matrix in any way but that we just run the classical conjugate gradient with the original matrix and an initial guess which we expect to remove all the troubles that the classical conjugate gradient algorithm would encounter otherwise.

2.2. The deflated conjugate gradient. In this subsection, we consider the deflated version of conjugate gradient described in [34] and called DEF-CG. This algorithm exploits the link between the Lanczos algorithm and the standard conjugate gradient algorithm, and is mathematically equivalent to Nicolaide's algorithm which is derived from a "deflated" Lanczos procedure [29].

Let p real vectors $\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(p)}$ be given, along with a unit vector $\mathbf{v}^{(1)}$ that is orthogonal to $\mathbf{w}^{(i)}$ for $i = 1, 2, \ldots, p$. Define $\mathbf{W} = [\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(p)}]$. We assume that $[\mathbf{w}^{(1)}, \mathbf{w}^{(2)}, \ldots, \mathbf{w}^{(p)}]$ is a set of linearly independent vectors. The deflated conjugate gradient algorithm builds a sequence $\{\mathbf{v}^{(k)}\}_{k=1,2,\ldots}$ of vectors such that

$$\mathbf{v}^{(k+1)} \perp \mathrm{Span}\{\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(\mathrm{k})}\} \text{ and } ||\mathbf{v}^{(\mathrm{k}+1)}||_2 = 1.$$

Assume an initial guess $\mathbf{x}^{(0)}$ is given such that $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)} \perp \mathbf{W}$. We set $\mathbf{v}^{(1)} = \mathbf{r}^{(0)}/||\mathbf{r}^{(0)}||_2$. Define

$$\mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) = \mathrm{Span}\{\mathbf{W}, \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots, \mathbf{v}^{(\mathrm{k})}\},$$

where the $\mathbf{v}^{(k)}$ are the generated Krylov vectors at step k.

At the kth step of our projection method, we seek an approximate solution $\mathbf{x}^{(k)}$ with

$$\mathbf{x}^{(k)} \in \mathbf{x}^{(0)} + \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}) \quad \text{such that} \quad \mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)} \perp \mathcal{K}_{p,k}(\mathbf{A}, \mathbf{W}, \mathbf{r}^{(0)}).$$

This technique resumes in performing, at the beginning of the conjugate gradient algorithm, an oblique projection of the initial residual onto the basis $\mathbf{W}$ and an update of the initial residual viz.

$$\mathbf{x}^{(0)} = \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{b} \quad \text{and} \quad \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}, \tag{2.2}$$

$$\mathbf{p}^{(0)} = \mathbf{r}^{(0)} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{r}^{(0)}. \tag{2.3}$$

We then project, at each iteration, the newly constructed residual vector $\mathbf{r}^{(k)}$ onto the $\mathbf{A}$-orthogonal complement of the basis $\mathbf{W}$,

$$\mathbf{z}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\mathbf{A}\mathbf{r}^{(k)}, \tag{2.4}$$

before building the next entry $\mathbf{p}^{(k)}$ in the $\mathbf{A}$-orthogonal basis of the current Krylov sequence. We refer to [34] for a complete description.

2.3. Deflation as a preconditioner. To describe this approach, which we call Proj-CG, we define the projection $\mathcal{P}$ by

$$\mathcal{P} = \mathbf{I} - \mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T,\ \mathbf{W} \in \mathbb{R}^{n\times p}, \tag{2.5}$$

where Span($\mathbf{W}$) is the deflation subspace, i.e., the space to be projected out of the residual, and $\mathbf{I}$ is the identity matrix. We assume $p \ll n$. We split the solution vector $\mathbf{x}$ into two parts:

$$\mathbf{x} = (\mathbf{I} - \mathcal{P}^T)\mathbf{x} + \mathcal{P}^T\mathbf{x}.$$

The first part $(\mathbf{I} - \mathcal{P}^T)\mathbf{x}$ is the component of $\mathbf{x}$ contained in $\mathbf{W}$, whereas the second part $\mathcal{P}^T\mathbf{x}$ is orthogonal to $\mathbf{W}$. The first part is computed from

$$\Big(\mathbf{I} - \mathcal{P}^T\Big)\mathbf{x} = \Big(\mathbf{A}\mathbf{W}(\mathbf{W}^T\mathbf{A}\mathbf{W})^{-1}\mathbf{W}^T\Big)^T\mathbf{x} = \mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{b}.$$

To compute the second part $\mathcal{P}^T\mathbf{x}$, we use the property $\mathbf{A}\mathcal{P}^T\mathbf{x} = \mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}$, and we then solve the system

$$\mathcal{P}\mathbf{A}\mathbf{x} = \mathcal{P}\mathbf{b}. \tag{2.6}$$

Kaasschieter [21] notes that a positive semidefinite system can be solved by conjugate gradient as long as the right-hand side is consistent (i.e., as long as $\mathbf{b} = \mathbf{A}\mathbf{x}$ for some $\mathbf{x}$). This is certainly true for (2.6), where the same projection is applied to both sides of the nonsingular system. Furthermore, he notes that, because the null space never enters the iteration, the corresponding zero-eigenvalues do not influence the convergence. We refer the reader to [28] for a complete description.

2.4. A spectral low-rank update preconditioner. In this subsection, we consider the approach proposed in [4] that attempts to remove the effect of the smallest eigenvalues in magnitude in the preconditioned matrix, which potentially can slow down the convergence of Krylov solvers. Roughly speaking, the proposed technique consists in building the preconditioner

$$\mathbf{M} = \mathbf{I} + \mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T \tag{2.7}$$

that aims to shift, by adding the value 1, those eigenvalues in the matrix $\mathbf{A}$ linked to the *near*-invariant subspace $\mathbf{W}$.

We can then use this spectral preconditioner $\mathbf{M}$ to run the preconditioned conjugate gradient, that will benefit from the reduced condition number of the preconditioned matrix $\mathbf{MA}$, thus yielding the algorithm which we call SLRU (spectral low-rank update; see [4]).

2.5. A Chebyshev-based partial spectral factorization. The purpose of this subsection is to briefly describe the combination of Chebyshev filtering polynomials with a Lanczos-type technique (see, e.g., [17, 31, 36]) to derive a *near*-invariant subspace of $\mathbf{A}$ associated with the smallest eigenvalues in $\mathbf{A}$ with some chosen level of accuracy. This technique corresponds to the CHEBYSHEV-PSF algorithm detailed in [2], [14, sect. 2], and [35, sect. 2.3] and can be summarized as follows.

The CHEBYSHEV-PSF algorithm exploits Chebyshev polynomials in $\mathbf{A}$ to damp, in a set of vectors, the eigencomponents associated with all the eigenvalues in some predetermined range $[\mu, \lambda_{\max}(\mathbf{A})]$, where μ is some fixed (user-given) positive number. Let us denote by

$$\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T = \mathbf{U}_1\Lambda_1\mathbf{U}_1^T + \mathbf{U}_2\Lambda_2\mathbf{U}_2^T$$

the eigendecomposition of the SPD matrix $\mathbf{A}$, Λ_1 being the diagonal matrix made with all eigenvalues of $\mathbf{A}$ less than μ, $\mathbf{U}_1$ the unitary matrix whose columns are the corresponding orthonormalized eigenvectors in matrix form, and Λ_2 and $\mathbf{U}_2$ the complementary corresponding matrices. We can then introduce the Chebyshev polynomials (see, e.g., [19]), $\mathcal{F}_m(\lambda)$, that have a minimum l_∞-norm on the interval $[\mu, \lambda_{\max}(\mathbf{A})]$ over all polynomials Q_m of degree less than or equal to m and satisfying $Q_m(0) = 1$. Let now $\mathbf{P}$ be a set of vectors, and consider

$$\mathbf{Z} = \mathcal{F}_m(\mathbf{A})\,\mathbf{P} = \mathbf{U}_1\mathcal{F}_m(\Lambda_1)\mathbf{U}_1^T\,\mathbf{P} + \mathbf{U}_2\mathcal{F}_m(\Lambda_2)\mathbf{U}_2^T\,\mathbf{P}. \tag{2.8}$$

For given values of μ, $\lambda_{\max}(\mathbf{A})$, and ε, we can fix the degree m of $\mathcal{F}_m$ such that $||\mathcal{F}_m(\Lambda_2)||_\infty < \varepsilon$, and using (2.8) we get

$$||\mathbf{U}_2^T\,\mathbf{Z}||_2 \leq ||\mathcal{F}_m(\Lambda_2)||_2\,||\mathbf{U}_2^T\,\mathbf{P}||_2 \leq \varepsilon\,||\mathbf{U}_2^T\,\mathbf{P}||_2. \tag{2.9}$$

Equation (2.9) explains explicitly how the action of the Chebyshev polynomials in $\mathbf{A}$ applied on a set of vectors can reduce, by some given factor ε, the eigencomponents with respect to the invariant subspace $\mathbf{U}_2$ relatively to the others. The idea in the CHEBYSHEV-PSF algorithm is then to combine such Chebyshev polynomials in $\mathbf{A}$ with Lanczos techniques (see, e.g., [17, 31, 36]) to directly build a *near*-invariant basis for the invariant subspace associated with all eigenvalues in the range $[\lambda_{\min}(\mathbf{A}), \mu]$.

As opposed to classical polynomial preconditioning techniques, and as explained by (2.9), the use of Chebyshev filtering polynomials in the context of the Lanczos iteration can be viewed as a way to perform implicitly some sort of deflation with respect to the invariant subspace linked with the largest eigenvalues in $\mathbf{A}$. Chebyshev filtering polynomials, in that respect, do not present the same nice properties as projectors used commonly in such deflation, which have explicit eigenvalues equal to 0 and 1. Still, they can partly mimic these properties, and offer an alternative to achieve the same behavior in the Lanczos iteration as with deflation techniques, but without having to effectively compute the basis vectors associated with this deflation.

Of course, to be efficient, the method relies on the fact that the spectrum of $\mathbf{A}$ is mostly concentrated in the range $[\mu, \lambda_{\max}(\mathbf{A})]$, with reasonable value of μ (like $\lambda_{\max}/100$ or $\lambda_{\max}/10$, for instance) so that the number of Chebyshev iterations required to reach a given filtering level ε are not too large. In that respect, a preliminary preconditioner applied to $\mathbf{A}$, with classical preconditioning techniques, can also be useful to transform and clusterize better the spectrum in the iteration matrix.

Additionally, the CHEBYSHEV-PSF algorithm is designed to control explicitly the level of filtering ε of the eigencomponents between $\mathbf{U}_1$ and $\mathbf{U}_2$ in the computed

approximate invariant subspace basis $\mathbf{W}$ (function of ε), as indicated in (2.9). In the following, we compare the different proposed approaches for the solution of linear systems with $\mathbf{A}$, combined with the computation of a *near*-invariant orthonormal basis $\mathbf{W}$ obtained with this Chebyshev-Partial Spectral Factorization. We shall also vary the level of filtering ε in this Chebyshev-PSF algorithm, and evaluate the performance of all these solution techniques with respect to this control of the numerical quality of the basis $\mathbf{W}(\varepsilon)$.

2.6. Combination of oblique projection and Chebyshev iteration. In this subsection, we describe how we use the *near*-invariant subspace $\mathbf{W}$ linked to the smallest eigenvalues to compute the solution of the linear system with $\mathbf{A}$.

The idea is to perform an oblique projection of the initial residual ($\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$) onto this *near*-invariant subspace in order to get the eigencomponents in the solution corresponding to the smallest eigenvalues, viz.

$$\begin{aligned} &\mathbf{r}^{(\mathrm{proj})} = \mathbf{r}^{(0)} - \mathbf{A}\mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}, \\ \text{with} \quad &\mathbf{x}^{(\mathrm{proj})} = \mathbf{x}^{(0)} + \mathbf{W}\Big(\mathbf{W}^T\mathbf{A}\mathbf{W}\Big)^{-1}\mathbf{W}^T\mathbf{r}^{(0)}. \end{aligned} \tag{2.10}$$

To compute the remaining part of the solution vector $\mathbf{A}\widetilde{\mathbf{x}} = \mathbf{r}^{(\mathrm{proj})}$, one can then use the classical Chebyshev algorithm with eigenvalue bounds given by μ and $\lambda_{\max}(\mathbf{A})$. This technique is called Init-Chebyshev (for technical details, see [14, sect. 3.1]).

Note that, if $\mathbf{W}$ exactly spans the invariant subspace generated by $\mathbf{U}_1$, the Chebyshev iteration and the oblique projection steps in this solution phase can be performed, though sequentially, in any order a priori. However, since Span($\mathbf{W}$) is only an approximation of this invariant subspace, we prefer to perform the Chebyshev step first, followed then by the oblique projection, because this helps to increase the accuracy of the oblique projection by *minimizing* the influence of the eigencomponents in $\mathbf{U}_2$ relative to those in $\mathbf{U}_1$ in the scalar products $\mathbf{W}^T\mathbf{r}$ in (2.10). This is of particular interest when the filtering level ε used to compute the basis $\mathbf{W}$, as in section 2.5, is not close to machine precision.

The number of Chebyshev iterations to achieve a given level of filtering ε is directly related to the rate of convergence of Chebyshev polynomials on the interval $[\mu, \lambda_{\max}]$ (see, e.g., [15, 19]) which depends only on the ratio $\lambda_{\max}/\mu$. This technique also enables us to control a priori, with some forward error analysis (see, e.g., [2]), the $\mathbf{A}$-norm of the error in subsequent solutions of linear systems with the same matrix.

2.7. Algebraic two-grid cycle. The multigrid methods are among the fastest techniques to solve a linear system (1.1) arising from the discretization of elliptic PDEs [18]. The core of the multigrid algorithms is a two-grid procedure that is applied recursively. A classical two-grid cycle can be briefly described as follows.

On the fine grid a few iterations of a presmoother are applied that attempt to reduce the high frequencies of the error (i.e., the components of the error in the space spanned by the vectors associated with the largest eigenvalues of $\mathbf{A}$). The residual is then projected on the coarse grid where the low frequencies, that are the components associated with the smallest eigenvalues, can be captured and the coarse error equation is solved. The coarse error is prolongated back to the fine grid to update the approximation computed by the presmoother and a few more steps of the smoother are applied. Finally, if the new iterate is not accurate enough, this two-grid

cycle is applied iteratively, viz.

$$\begin{aligned} \mathbf{x}_1 &= \mathbf{S}(\mathbf{x}, \mathbf{b}, \xi), \\ \mathbf{x}_2 &= \mathbf{x}_1 + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_1), \\ \mathbf{x} &= \mathbf{S}(\mathbf{x}_2, \mathbf{b}, \xi), \end{aligned} \tag{2.11}$$

where $\mathbf{S}(\mathbf{x}, \mathbf{b}, \xi)$ defines the smoother that performs ξ iterations on the $\mathbf{x}$ to solve $\mathbf{A}\mathbf{x}^\star = \mathbf{b}$.

In classical multigrid, the coarse space is not defined explicitly through the knowledge of the eigenspaces but by the selection of a space that is expected to capture them. In this work, we explicitly define the coarse space by computing the eigenvectors $\mathbf{W}$ associated with the smallest eigenvalues of $\mathbf{A}$. For symmetric problems, the restriction operator is denoted $\mathbf{R} = \mathbf{W}^T$ and the prolongation operator is its transpose. The matrix associated with the coarse grid error problem is defined by a Galerkin formula $\mathbf{A}_c = \mathbf{R}\,\mathbf{A}\,\mathbf{R}^T$. In this scheme, different smoothers (CG, Chebyshev, damped Richardson) can be used. We refer to [5], where this approach is used to design preconditioners and where the spectral transformation associated with this scheme is analyzed.

3. Numerical experiments. In this section, we show some numerical results that compare the different approaches in terms of numerical efficiency for the solution of linear systems with the matrix $\mathbf{A}$. We discuss also their computational complexity and sensitivity to the accuracy of the spectral information. The precomputation of a *near*-invariant orthonormal basis $\mathbf{W}$ is obtained by means of the Chebyshev-Partial Spectral Factorization algorithm introduced in section 2.5. We shall also evaluate the impact of varying the level of filtering ε in the precomputation of the *near*-invariant basis $\mathbf{W}$(function of ε) on the performance of all these solution techniques.

In Table 3.1 we summarize for each matrix the size, the number of nonzero elements in $\mathbf{A}$, and the application in which this matrix arises. Matrices BCSSTK are obtained from the Harwell–Boeing collection [11]. PDE1 and PDE2 are extracted from the finite element discretization using **pdetool** under MATLAB of the partial differential equation problem

$$\left\{ \begin{array}{rcl} -\mathrm{div}\Big(\Lambda(x) \cdot \nabla u\Big) & = & f \quad \text{in} \quad \Omega, \\ u_{|\partial\Omega} & = & 0, \end{array} \right.$$

where $\Omega \subset \mathbb{R}^2$ is an L-shaped region as described in Figure 3.1. The major differences between PDE1 and PDE2 are in the settings for $\Lambda(x)$ and in the size of discretization.

In PDE1, $f = 10$, and the function $\Lambda(x) \in L^\infty(\Omega)$ takes different scalar values in each subdomain:

$$\Lambda(x) = \left\{ \begin{array}{ll} 1, & x \in \Omega_1 \cup \Omega_4, \\ 10^6, & x \in \Omega_2, \\ 10^4, & x \in \Omega_3. \end{array} \right.$$

In PDE2, $f = 200$, and the PDE problem incorporates some heterogeneity and

TABLE 3.1
Set of SPD test matrices.

Name	Size	**nnz(A)**	Short description
BCSSTK14	1,806	63,454	static analyses in structural engineering Roof of the Omni Coliseum, Atlanta
BCSSTK15	3,948	117,816	from the BCSSTRUC2 group of the Harwell Boeing Collection
PDE1	7,969	55,131	2D heterogeneous diffusion equation discretized by finite element in a L shape region
PDE2	161,313	1,125,897	2D heterogeneous and anisotropic diffusion equation discretized by finite element in a L shape region

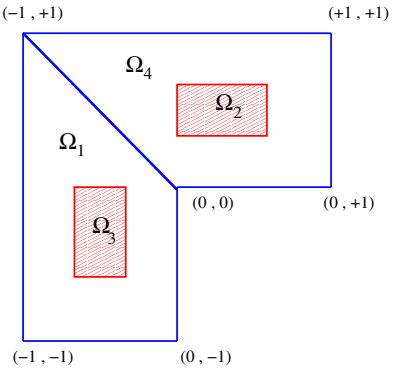


FIG. 3.1. *Geometry of the domain* Ω.

anisotropy, with the matrix $\Lambda(x)$ that takes the following values:

$$\Lambda(x) = \begin{cases} \begin{bmatrix} 1 & 4\,\lambda_1 \\ 4\,\lambda_1 & \lambda_1 \end{bmatrix} & \text{if } x \in \Omega_1 \quad \text{and} \quad \lambda_2\,\mathbf{I}_2 \quad \text{if } x \in \Omega_2, \\ \begin{bmatrix} \lambda_1 & -2\,\lambda_1 \\ -2\,\lambda_1 & 1 \end{bmatrix} & \text{if } x \in \Omega_4 \quad \text{and} \quad \lambda_3\,\mathbf{I}_2 \quad \text{if } x \in \Omega_3, \end{cases}$$

where $\lambda_1 = 6 \cdot 10^{-2}$, $\lambda_2 = 1 \cdot 10^{6}$, $\lambda_3 = 1 \cdot 10^{2}$, and $\mathbf{I}_2$ denotes the (2×2) identity matrix.

For all the numerical experiments reported in this section, a first level of preconditioner $\mathbf{M}_1 = \mathbf{L}\mathbf{L}^T$ is constructed using incomplete Cholesky factorization IC(t) of $\mathbf{A}$ with t as the threshold. The purpose of this first preliminary preconditioning of the given linear system is to better cluster the eigenvalues so that the invariant subspace associated with the smallest eigenvalues is of small dimension. We denote by $\widehat{\mathbf{b}} = \mathbf{L}^{-1}\mathbf{b}$ and $\widehat{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}\mathbf{L}^{-T}$ the matrix symmetrically preconditioned with IC(t) which is SPD and similar to the matrix $\mathbf{M}_1^{-1}\mathbf{A}$. Additionally, in the different figures illustrating the experimental results, we shall denote by *classical CG* the preconditioned conjugate gradient with the incomplete Cholesky factorization as the first level of preconditioner only.

In Table 3.2, we give some insight about the effect of the incomplete Cholesky factorization on the eigenvalue distribution of the preconditioned matrix $\widehat{\mathbf{A}}$. For instance, we first notice the high classical condition number of order 10^{10} for the original BCSSTK14's matrix. We can also observe that the condition number of this preconditioned matrix is reduced to about 10^2. As mentioned before, a clustered

TABLE 3.2
Distribution of eigenvalues.

Matrix	Unpreconditioned matrix		Preconditioned matrix				Number of eigenvalues below the $\mu = \lambda_{\max}/10$
	Smallest eig	Largest eig	Threshold IC(t)	**nnz(L)**	Smallest eig	Largest eig	
BCSSTK14	1	$1.19 \cdot 10^{10}$	10^{-2}	21,197	$1.41 \cdot 10^{-2}$	3.13	23
BCSSTK15	1	$6.53 \cdot 10^{9}$	10^{-3}	109,860	$5.35 \cdot 10^{-3}$	1.77	4
PDE1	$2.92 \cdot 10^{-9}$	2.17	10^{-2}	64,566	$1.79 \cdot 10^{-7}$	1.11	3
PDE2	$7.69 \cdot 10^{-5}$	$1.00 \cdot 10^{7}$	0	643,605	$6.78 \cdot 10^{-10}$	2.03	16*

*In this case we take $\mu = \lambda_{\max}/2000$.

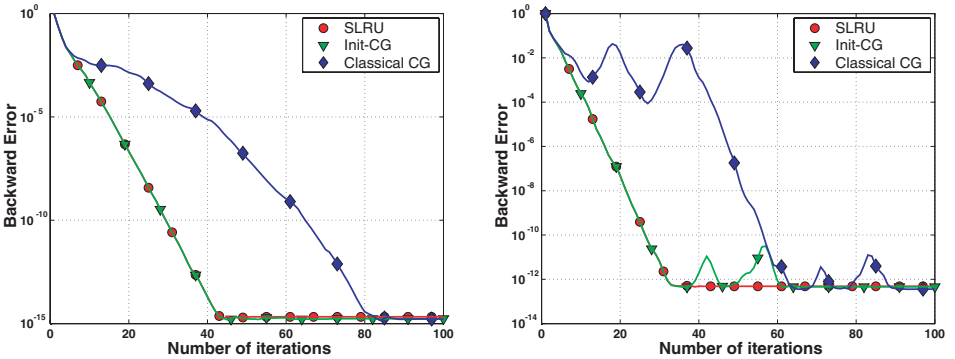
spectrum of the preconditioned matrix is usually considered a desirable property for fast convergence of Krylov solvers and makes the numerical techniques considered in this paper affordable because only a few eigencomponents have to be handled by a complementary approach.

The stopping criterion used in our tests is the backward error $||\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\mathbf{x}^{(i)}||_2/||\widehat{\mathbf{b}}||_2$. In all our runs, we have monitored this backward error down to close to machine precision, when possible, to illustrate and study the complete convergence history of the schemes. All experiments were performed in 64 bits floating-point arithmetic using MATLAB. The initial guess is always $\mathbf{x}^{(0)} = 0$ and the right-hand side is chosen so that the solution of the initial linear system is the vector of all ones. The cut-off eigenvalue μ in the Chebyshev polynomial filters has been fixed to $\lambda_{\max}(\widehat{\mathbf{A}})/10$ whenever this was suitable, e.g., when the number of eigenvalues of the range $[0, \mu]$ was reasonably small. In particular, in the case of the larger size problem PDE2, we have preferred to keep under control the cost of the first level of preconditioning $\mathbf{M}_1$ (IC(0) in this case), and to move the cut-off value μ closer to zero ($\mu = \lambda_{\max}(\widehat{\mathbf{A}})/2000$) to incorporate only a small set of eigenvalues in the second level of preconditioning.

3.1. Comparison of the different techniques for the solution phase. In this subsection, we compare the numerical convergence behavior of the various algorithms described in section 2. As we will see in the following, the experiments show that the different algorithms exhibit very similar numerical behavior. This is particularly true when the precomputed *near*-invariant basis $\mathbf{W}(\varepsilon)$ is obtained with filtering level ε close to machine precision. In this case, this numerical behavior corresponds to the one that we would obtain in general with a linear system where the smallest eigenvalue is given by the cut-off value μ in the CHEBYSHEV-PSF algorithm, with a resulting reduced condition number of $\lambda_{\max}(\mathbf{A})/\mu$.

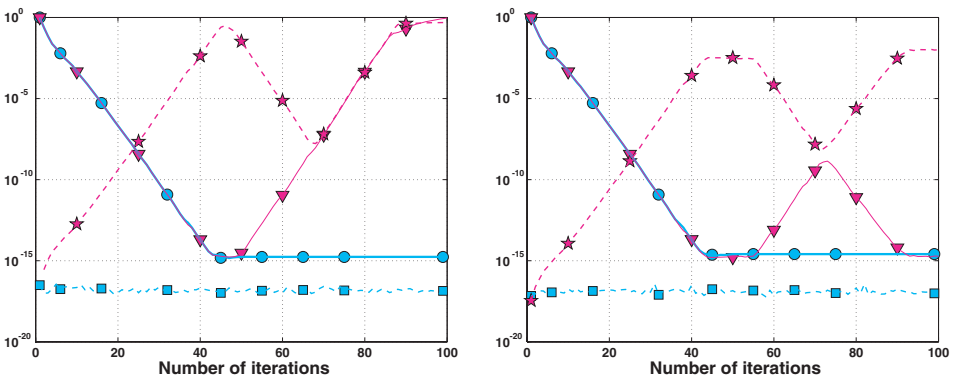
Figure 3.2 illustrates the convergence history of the conjugate gradient versus INIT-CG and SLRU obtained with a basis $\mathbf{W}(\varepsilon)$ computed with a level of filtering ε equal to 10^{-16}. The backward error of the standard conjugate gradient method tends to decrease very rapidly in the first few iterations, then decreases more slowly in an intermediate phase, during which oscillations may also appear, and finally stagnates. As opposed to this, INIT-CG and SLRU present convergence histories that coincide almost completely, except perhaps after stagnation.

DEF-CG and PROJ-CG also present the same type of convergence, at least in the first part of their convergence histograms as illustrated by the results in Figure 3.3. However, we can observe that these are also subject to some numerical instability, in particular after reaching the stagnation phase. Theoretically, the residuals $\mathbf{r}^{(k)}$ are supposed to stay orthogonal to all the columns of $\mathbf{W}(\varepsilon)$, but in practice, this



(a) *BCSSTK*14. (b) *PDE*1.

FIG. 3.2. *Convergence history for a level of filtering* ε *equal to* 10^{-16}.



(a) *convergence history of* DEF-CG. (b) *convergence history of* PROJ-CG.

★ ortho(k) without correction of $\mathbf{r}^{(k)}$
□ ortho(k) with correction of $\mathbf{r}^{(k)}$
▽ backward error without correction of $\mathbf{r}^{(k)}$
○ backward error with correction of $\mathbf{r}^{(k)}$

FIG. 3.3. *Orthogonality of* $\mathbf{r}^{(k)}$ *with respect to* $\mathbf{W}(\varepsilon)$ *precomputed with a level of filtering* ε *equal to* 10^{-16} *and convergence histories with or without the correction of* $\mathbf{r}^{(k)}$ *on the BCSSTK*14 *matrix.*

orthogonality is gradually lost as the iterations progress. Figure 3.3 shows, for matrix BCSSTK14, for instance, the histogram of

$$\text{ortho(k)} = \max_{\text{j}=1,2,\ldots,\text{p}} \left(\frac{\mathbf{w}^{(\text{j})^{\text{T}}} \mathbf{r}^{(\text{k})}}{||\mathbf{w}^{(\text{j})}||_2 ||\mathbf{r}^{(\text{k})}||_2} \right),$$

computed at each iteration k in DEF-CG and PROJ-CG. Both curves of the function ortho(k) show that loss of orthogonality is inversely proportional to the convergence of the backward error. One remedy to recover orthogonality as mentioned in [34] is to add the reorthogonalization step

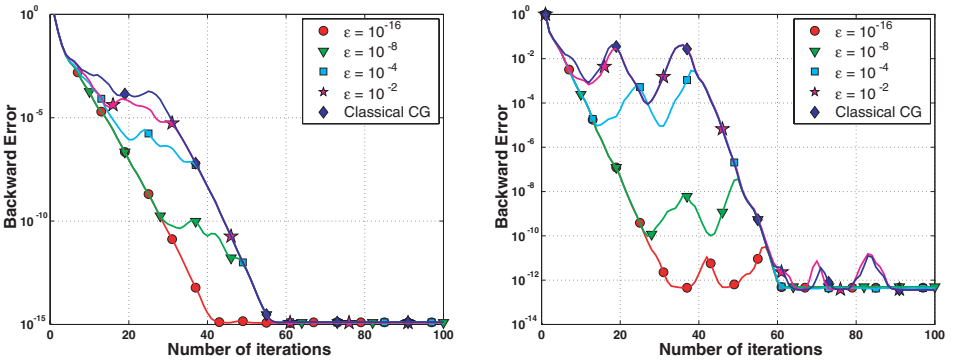
$$\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \mathbf{W}\left(\mathbf{W}^T\mathbf{W}\right)^{-1}\mathbf{W}^T\mathbf{r}^{(k)} \tag{3.1}$$

right after $\mathbf{r}^{(k)}$ is computed in DEF-CG and PROJ-CG. The computational cost of this extra step is of order $\mathcal{O}(p * n)$ per iteration.

The convergence curves with the additional correction of $\mathbf{r}^{(k)}$ are plotted with a (○, solid line) in Figure 3.3. In this case, the convergence history of DEF-CG and PROJ-CG are very much close to that of INIT-CG and SLRU as in Figure 3.2, with no more instability after convergence is reached.

In summary, when the *near*-invariant subspace basis $\mathbf{W}(\varepsilon)$ is computed with very high accuracy (i.e., with a filtering level ε close to machine precision), the INIT-CG is the algorithm of choice because it is less expensive, since it avoids the use of oblique projectors within the conjugate gradient iterations explicitly.

3.2. When the near-invariant subspace basis W is less accurate. The different solution techniques can however present very different convergence behavior when the computation of the *near*-invariant basis vectors $\mathbf{W}(\varepsilon)$ is obtained with larger values of level of filtering ε.

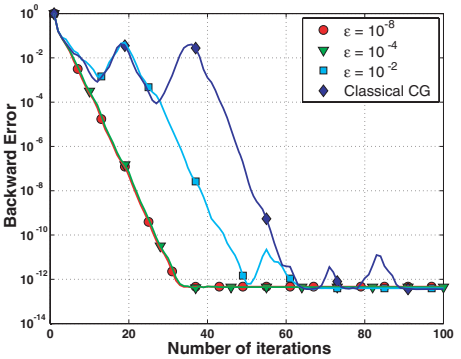


(a) *BCSSTK*15. (b) *PDE*1.

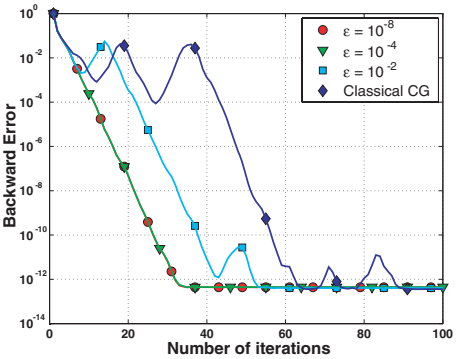
FIG. 3.4. *Convergence history of* INIT-CG *for different values of level of filtering* ε.

Figure 3.4 shows the impact of varying the filtering level ε (when computing the *near*-invariant $\mathbf{W}(\varepsilon)$ basis) on the convergence of INIT-CG. For example, when the level of filtering ε is equal to 10^{-8}, the phenomenon of plateaux occurs again, but at intermediate levels. Indeed, INIT-CG (conjugate gradient with initial oblique projection) shows two different phases in its numerical behavior. The first stage follows very closely the convergence history observed in Figure 3.2 when the computation of the $\mathbf{W}(\varepsilon)$ basis is very accurate. Then, the speed of convergence is disrupted at an intermediate level of the backward error closely related to the value of the filtering level ε, and the convergence history resembles again that of the conjugate gradient on the original system. The reason for this is that INIT-CG provides an initial residual to the conjugate gradient that has larger eigencomponents in the orthogonal complement of the basis $\mathbf{W}(\varepsilon)$ and eigencomponents of order ε in the $\mathbf{W}(\varepsilon)$ subspace. After the residual norm has been reduced to about the value of ε, the conjugate gradient is left again with a Krylov direction with eigencomponents all of about the same magnitude, and then exhibits the same type of convergence behavior linked to the complete spectrum of the given matrix. For efficiency, it is clear that one should then stop the iterations in INIT-CG at a backward error about the filtering level ε, before reaching this intermediate plateau.

As we can observe in Figure 3.6 (a), when the size of the linear system becomes very large, the phenomenon of plateaux can also occur in the numerical behavior of the INIT-CG algorithm, even with a very accurate *near*-invariant subspace basis

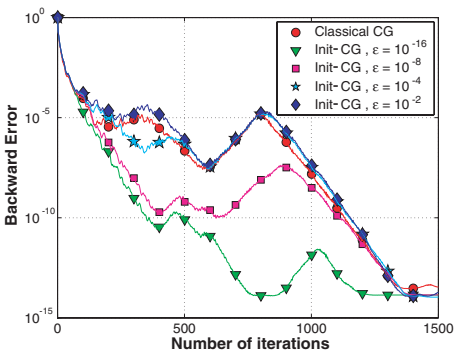


(a) *convergence history of* SLRU.

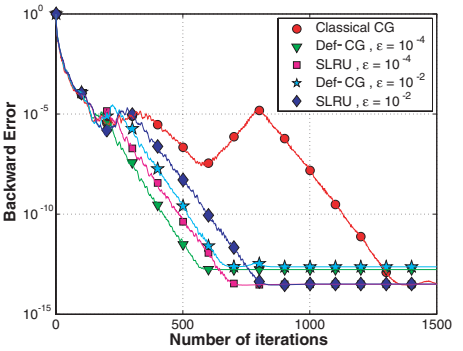


(b) *convergence history of* DEF-CG.

FIG. 3.5. *Convergence history for different values of* ε *on PDE*1.



(a) *convergence history of* INIT-CG.



(b) *convergence history of* SLRU *and* DEF-CG.

FIG. 3.6. *Convergence history of* INIT-CG, DEF-CG, *and* SLRU *algorithms with different values of level of filtering* ε *on PDE*2.

$\mathbf{W}(10^{-16})$. This is due to the sensitivity of the projector to the total size of the linear system [2].

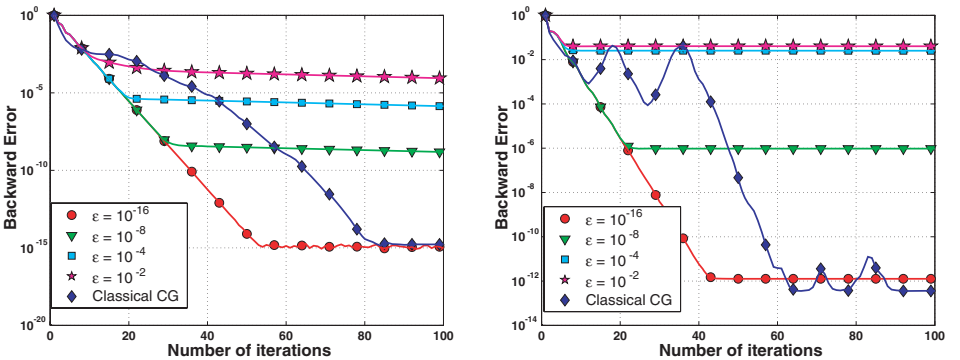
The convergence history of DEF-CG and SLRU, depicted in Figure 3.5 for different values of the level of filtering ε, shows that these two algorithms exhibit very close numerical behaviors in any case. We can observe anyway that if the accuracy in the computation of the invariant subspace $\mathbf{W}(\varepsilon)$ is strongly deteriorated, as with a value $\varepsilon = 10^{-2}$, for instance, then the speed of convergence may vary slightly. Still, as opposed to INIT-CG, these algorithms show a constant numerical behavior close to the one observed in Figures 3.2 and 3.3, even when the filtering level is in the range 10^{-8} to 10^{-4}, as illustrated in Figure 3.5. This is the benefit of performing at each iteration of the conjugate gradient an oblique projection with the $\mathbf{W}(\varepsilon)$ basis that maintains throughout the iterations a fixed gap between the eigencomponents corresponding to the smallest eigenvalues and the others.

In Figure 3.6 (b), we compare the numerical behavior of DEF-CG and SLRU when the level of filtering ε is large. We can see that DEF-CG converges a little faster than SLRU. Knowing that the convergence of DEF-CG and PROJ-CG are very similar, a possible explanation of this (as mentioned in [28]) can be that the effective

condition number of the projected system is always below the condition number of the system preconditioned by the corresponding coarse grid correction.

3.3. Combination of oblique projection and Chebyshev iteration. We have postponed in this section the discussion concerning the results for INIT-CHEBYSHEV, since it does not belong to Krylov techniques as the others. INIT-CHEBYSHEV basically corresponds to the juxtaposition of a Chebyshev iterative solver and an oblique projection that act independently on two separate subparts in the spectrum of $\mathbf{A}$. In that respect, it can be viewed either as a direct method, if the eigenspace is computed very accurately, or as a form of two-space cycle, where the smoother is defined by a few steps of Chebyshev iteration (see section 2.6).

For the sake of comparison with the results shown for the other Krylov-based iterative solution techniques, we present in Figure 3.7 the computed backward error obtained after each Chebyshev iteration combined directly with the application of the final oblique projection. Of course, this is not the way INIT-CHEBYSHEV should work, since the oblique projection should be applied only once at the end, after the termination of the Chebyshev iterations.



(a) *BCSSTK*14. (b) *PDE*1.

FIG. 3.7. *Convergence history of* INIT-CHEBYSHEV *for different values of level of filtering* ε.

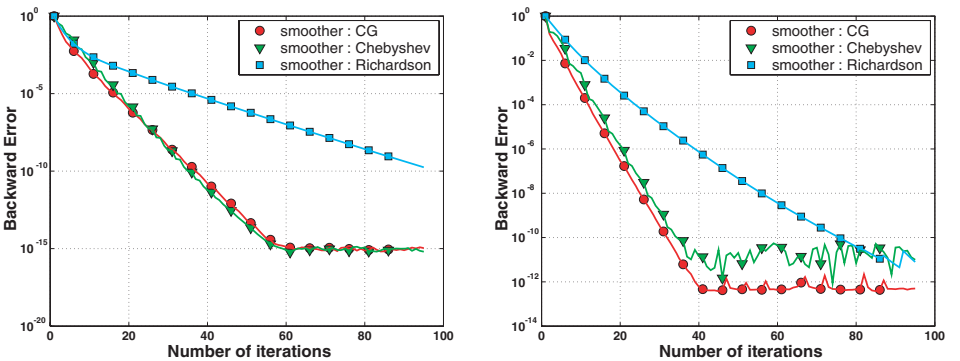
In Figure 3.7, we illustrate the effect of varying the filtering level ε (when computing the invariant basis $\mathbf{W}(\varepsilon)$) on the convergence behavior of INIT-CHEBYSHEV. When the level of filtering ε is equal to 10^{-16}, INIT-CHEBYSHEV behaves like the other algorithms, as illustrated in the previous sections, but with a little difference in the number of iterations. This is simply due to the different rates of convergence between the Chebyshev semi-iterative method and the conjugate gradient method in general.

However, for larger values of the filtering level ε, we can observe that the convergence history of INIT-CHEBYSHEV is disrupted, with a stagnation of the backward error at a level close to the actual value of ε. The reason for this is simply that the *near*-invariant basis $\mathbf{W}(\varepsilon)$ has eigencomponents of order ε with respect to those eigenvectors corresponding to eigenvalues in the range $[\mu, \lambda_{\max}]$. Obviously, this level of information is incorporated explicitly in the oblique projection onto this subspace that is applied as a final stage, and perturbs the eigencomponents in the final residual at about that level. For a more detailed analysis of this phenomenon, we refer the reader to [2].

It is also worth mentioning that this combination of oblique projection and Chebyshev iteration requires only matrix-vector products plus some vector updates, but no

dot-products, as opposed to the other Krylov-based solution techniques. This remark is also of some importance in the context of parallel computing, and in particular in distributed memory environments where the computation of the dot-product requires particular attention.

3.4. Algebraic two-grid. As mentioned in section 2.7, one way to exploit the precomputed spectral information is to define a two-grid cycle that is run as a stationary iterative solver. This approach makes sense only when the spectral information is not computed with very high accuracy. In that latter case, the schemes based on a deflation to define the initial guess should be preferred because they require only one oblique projection. To illustrate the numerical behavior of these schemes, we depict in Figure 3.8 the convergence history of this algebraic two-grid solver using different smoothers. More precisely, for these experiments, we have implemented in our experiments a few steps of conjugate gradient, Chebyshev, or damped Richardson. For the Richardson iteration we select $\alpha_{opt} = 2/(\lambda_{\max}+\mu)$ and minimize the spectral radius of the iteration matrix (equal to $\max\{|1-\alpha\,\lambda_{\max}|, |1-\alpha\,\mu|\}$), assuming that the coarse grid correction has removed the effect of the eigenvalues in the range $[\lambda_{\min}, \mu]$. For the experiments illustrated in Figure 3.8, we have performed $\xi = 5$ smoothing steps. It can be seen that the three implementations exhibit a linear rate of convergence, and that conjugate gradient and Chebyshev have the same smoothing properties.

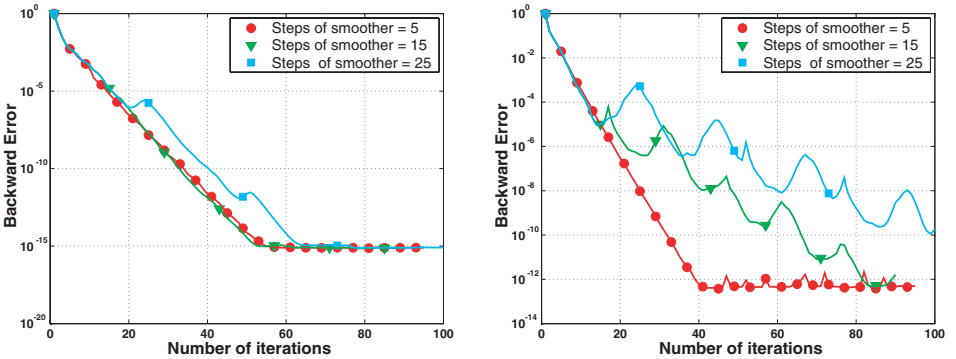


(a) *BCSSTK*14. (b) *PDE*1.

FIG. 3.8. *Convergence history of the two-grid cycle solver with different smoothers using a level of filtering* $\varepsilon = 10^{-4}$.

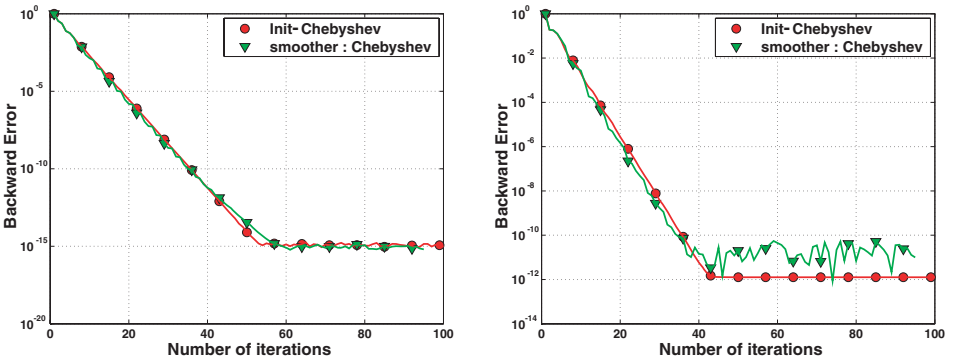
For the sake of efficiency one might want to use a value of ξ (number of smoothing steps) as large as possible in order to reduce the number of cycles, and consequently the number of oblique projections, while preserving the same rate of convergence. In Figure 3.9 we display the convergence history of the two-grid solver when the number of smoothing steps is varied; conjugate gradient is the smoother considered for those experiments. We can observe that using too large a value of ξ might delay the convergence. These situations occur when the smoother has succeeded in reducing most of the high frequencies of the error and makes only slight progress in reducing the low frequencies. Applying a coarse grid correction in that case is an efficient way to speed up the convergence. Based on this observation, a heuristic could be implemented to adapt the number of smoothing steps automatically, but we do not pursue this aspect further in this work.

The INIT-CHEBYSHEV algorithm can also be viewed as one step of a two-grid



(a) *BCSSTK*14. (b) *PDE*1.

FIG. 3.9. *Convergence history of the two-grid cycle solver when the number of smoothing steps using conjugate gradient is varied. The level of filtering is* $\varepsilon = 10^{-4}$.



(a) *BCSSTK*14. (b) *PDE*1.

FIG. 3.10. *Convergence history of the* INIT-CHEBYSHEV *algorithm obtained with a basis* $\mathbf{W}(\varepsilon)$ *computed with a level of filtering* ε *equal to* 10^{-16} *versus the Chebyshev-based two-grid solver with a number of smoothing steps* $\xi = 5$ *and a* near*-invariant* $\mathbf{W}(\varepsilon)$ *basis precomputed with a level* ε *of* 10^{-4}.

cycle, where the smoother corresponds to the Chebyshev iteration. In Figure 3.10, we compare the convergence history of the INIT-CHEBYSHEV algorithm, obtained with a basis $\mathbf{W}(\varepsilon)$ computed with a level of filtering ε equal to 10^{-16}, versus the Chebyshev-based two-grid solver with a number of smoothing steps $\xi = 5$, and a *near*-invariant $\mathbf{W}(\varepsilon)$ basis precomputed with a level ε of 10^{-4}. As mentioned in section 3.3, the convergence history of INIT-CHEBYSHEV is disrupted, with a stagnation of the backward error at a level close to the actual value of ε. As opposed to this, the two-grid cycle solver, with Chebyshev as the smoother, shows the same curve of convergence as that of the INIT-CHEBYSHEV algorithm obtained with a very accurately precomputed $\mathbf{W}(\varepsilon)$ basis ($\varepsilon = 10^{-16}$). This illustrates the benefit of using a two-grid cycle scheme when the spectral information is not very accurate.

In the case of our test examples, we have seen that the convergence rate of the two-grid cycle scheme, using conjugate gradient as the smoother and a suitable value of ξ (i.e., not too large), has a convergence history very similar to the one obtained with SLRU when ε is small enough (i.e., less than or equal to 10^{-4}). Consequently, the two-grid cycle scheme outperforms the SLRU algorithm in terms of floating-

point operations, since it requires far fewer oblique projections. This observation is no longer true when the *near*-invariant eigenspace poorly approximates the eigenspace (i.e., with $\varepsilon = 10^{-2}$, in our experiments). In that situation, the two-grid cycle scheme does not converge anymore with any of the smoothers, because the coarse grid correction systematically corrupts the iterates computed by the smoother. We recall (see Figure 3.5) that SLRU and DEF-CG still succeed in taking advantage of the approximate eigenspace in that case, as was also observed in [4].

Finally, we mention that the two-grid cycle using a stationary method as a smoother can also be used to define a preconditioner for the conjugate gradient; we refer the reader to [5] for more details on that approach.

4. Practical considerations. We first consider operations count. We assume that $p \ll n$ so that we neglect terms not containing n (p is the dimension of the *near*-invariant basis $\mathbf{W}$). All costs are evaluated in terms of the number of floating-point operations. The cost for a matrix-vector product is given by

$$\mathcal{C}_{\mathbf{A}} \approx 2\, nnz(\mathbf{A}) - n, \tag{4.1}$$

where $nnz(\mathbf{A})$ is the number of nonzero elements in $\mathbf{A}$.

The different algorithms compared in this paper perform an oblique projection of the initial residual onto the *near*-invariant basis $\mathbf{W}$ of size p, involving the same type of operator $\mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T\mathbf{r}^{(0)}$, where $\mathbf{A}_c = \mathbf{W}^T\mathbf{A}\mathbf{W}$ is a dense matrix of order p. The computation of $\mathbf{A}_c$ has a cost of

$$\mathcal{C}_{\mathbf{A}_c} \approx n\,p^2 + 2\, nnz(\mathbf{A})\, p - n\,p. \tag{4.2}$$

We note that $\mathbf{A}_c$ is factorized once at the beginning of each algorithm, and the factors are used for any subsequent solution with $\mathbf{A}_c$.

In addition to the sparse matrix-vector product with $\mathbf{A}$ at each iteration, the CG iteration adds merely two dot-products (`_DOT`) and three vector updates (`_AXPY`). The cost of these extra operations is given by

$$\mathcal{C}_{\texttt{_AXPY}} \approx 2\,n \text{ and } \mathcal{C}_{\texttt{_DOT}} \approx 2\,n. \tag{4.3}$$

In the schemes DEF-CG and PROJ-CG (see sections 2.2 and 2.3), a computation of an oblique projection $\mathcal{P}\,\mathbf{r}^{(k)}$ (see formulas (2.4) and (2.5)) is also involved at each iteration. This can be done using common BLAS2 operations viz. $\mathbf{z}^{(k)} = (\mathbf{A}\mathbf{W})^T\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k)} - \mathbf{W}\mathbf{A}_c^{-1}\mathbf{z}^{(k)}$. Considering that $\mathbf{A}\mathbf{W}$ and $\mathbf{A}_c$ are already computed and stored at the beginning of the algorithm, the cost of these operations is the same as that of the application of the SLRU preconditioner, $\mathbf{M}_{\text{SLRU}}$, which implements a rank-p update of the identity matrix of the form $\mathbf{I} + \mathbf{W}\mathbf{A}_c^{-1}\mathbf{W}^T$. The total cost of these BLAS2 operations is represented by

$$\mathcal{C}_{\text{Proj}} \approx 4\,(p+1)\,n. \tag{4.4}$$

The INIT-CHEBYSHEV algorithm does not belong to the Krylov technique like the others do. It basically corresponds to the juxtaposition of a Chebyshev iterative solver which involves three vector updates, one sparse matrix-vector product with $\mathbf{A}$, and an oblique projection at the end. For ease of comparison, we shall consider the cost of this last oblique projection in the count of the operations performed "at the beginning." In Table 4.1, we summarize the total cost of floating-point operations for each algorithm, with an initial cost, and a cost per iteration.

TABLE 4.1
Cost in floating-point operations for different methods.

	The cost in floating-point operations	
	At the beginning	At each iteration
INIT-CG	$\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$	$\mathcal{C}_{\mathbf{A}} + 0 + 3\mathcal{C}_{_\text{AXPY}} + 2\mathcal{C}_{_\text{DOT}}$
DEF-CG	$\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}}$	$\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{_\text{AXPY}} + 2\mathcal{C}_{_\text{DOT}}$
PROJ-CG	$\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$	$\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{_\text{AXPY}} + 2\mathcal{C}_{_\text{DOT}}$
SLRU	$\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + 0 + 0$	$\mathcal{C}_{\mathbf{A}} + \mathcal{C}_{\text{Proj}} + 3\mathcal{C}_{_\text{AXPY}} + 2\mathcal{C}_{_\text{DOT}}$
INIT-CHEBYSHEV	$\mathcal{C}_{\mathbf{A}_c} + \mathcal{C}_{\text{Proj}} + \mathcal{C}_{\mathbf{A}} + 0$	$\mathcal{C}_{\mathbf{A}} + 0 + 3\mathcal{C}_{_\text{AXPY}} + 0$

One of the major differences is that DEF-CG, PROJ-CG, and SLRU use this operator at each iteration whereas INIT-CG exploits this only at the beginning (for computing a starting vector $\mathbf{x}^{(0)}$). The two-grid algorithm is in between, in the sense that it performs an oblique projection for the coarse grid correction every other ξ steps of the smoother (which can be the conjugate gradient, Chebyshev, or Richardson iteration). In that respect, INIT-CG and INIT-CHEBYSHEV are less expensive, since they avoid the use of oblique projectors at each iteration, but, as we have seen previously in the experiments, they can suffer from some misconvergence when the basis $\mathbf{W}$ is not close to being *near*-invariant.

A final comment about the memory requirements of these five solution techniques (INIT-CG, INIT-CHEBYSHEV, DEF-CG, PROJ-CG, and SLRU): All these algorithms require about the same amount of extra storage, of order $2np$, to store $\mathbf{AW}$ and $\mathbf{W}$, the basis of the *near*-invariant subspace.

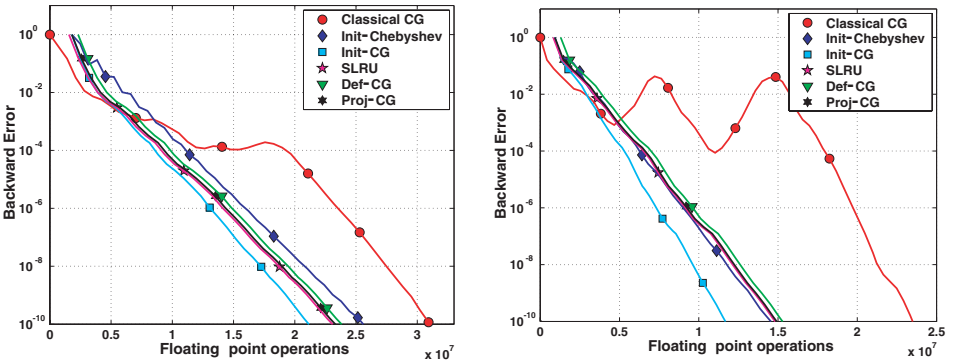
4.1. Comparison of the different techniques in terms of floating-point operations. In this subsection, we analyze the cost of the different algorithms introduced previously. As mentioned at the beginning of section 3, a first level of preconditioner $\mathbf{M}_1$ is also used, the purpose of which is to better clusterize the spectrum of our iteration matrix. The cost of the multiplication of $\mathbf{M}_1$ by a vector is represented by $\mathcal{C}_{\mathbf{M}_1}$. Since, in our experiments, $\mathbf{M}_1$ is constructed by means of the incomplete Cholesky factorization, $\mathcal{C}_{\mathbf{M}_1}$ can be estimated as

$$\mathcal{C}_{\mathbf{M}_1} \approx 4\, nnz(\mathbf{L}) - 2\, n, \tag{4.5}$$

where $nnz(\mathbf{L})$ is the number of nonzero elements in the factor $\mathbf{L}$ from the incomplete Cholesky factorization. The cost of this first level of preconditioning $\mathbf{M}_1$ must be incorporated in the floating-point counts detailed in Table 4.1. This simply resumes in adding $\mathcal{C}_{\mathbf{M}_1}$ to $\mathcal{C}_{\text{Proj}}$ (4.4) wherever it appears, or explicitly at each iteration for INIT-CG or INIT-CHEBYSHEV.

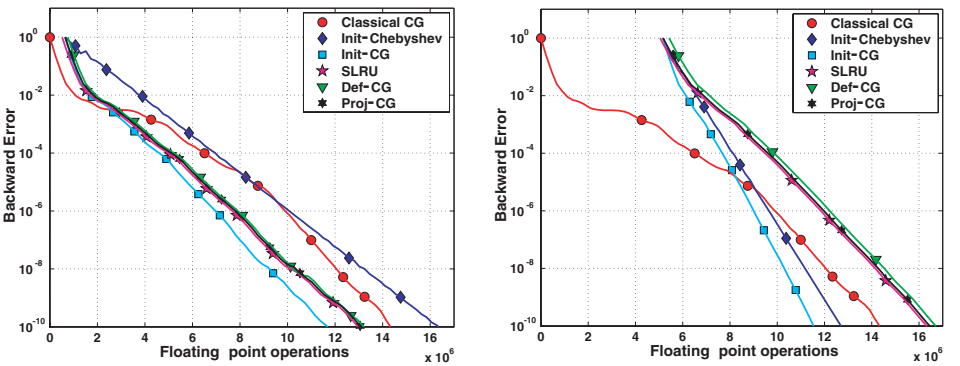
For simplicity, we exploit only the basis $\mathbf{W}(\varepsilon)$ obtained with maximum quality (level of filtering $\varepsilon = 10^{-16}$) so that we can directly observe and compare the behavior of the algorithms for a large range of the backward error, without any break in the slope of convergence due to potential lack of accuracy in the *near*-invariant subspace $\mathbf{W}(\varepsilon)$, as described and discussed already in the previous sections.

In Figures 4.1 and 4.2, we plot the histogram of the backward error versus the number of floating-point operations. The gap between the conjugate gradient and the other algorithms, at the beginning of these histograms, denotes the additional initial cost necssary to incorporate the $\mathbf{W}(\varepsilon)$ basis and to build the associated oblique projector. This gap is more or less large depending on the dimension of the basis $\mathbf{W}(\varepsilon)$. For instance, this dimension is of order 3–4 in Figures 4.1 (a), (b) and Figure



(a) *BCSSTK*15. (b) *PDE*1.

FIG. 4.1. *Floating-point operations with a level of filtering* ε *equal to* 10^{-16}.



(a) $\mu = \lambda_{\max}/40$. (b) $\mu = \lambda_{\max}/10$.

FIG. 4.2. *Floating-point operations of various algorithms with a level of filtering* ε *equal to* 10^{-16} *on the BCSSTK*14 *matrix.*

4.2 (a), and of order 23 in Figure 4.2 (b). We can also observe the different slopes in these histograms, with major differences between INIT-CG or INIT-CHEBYSHEV and the other three (DEF-CG, PROJ-CG, and SLRU). Indeed, INIT-CG and INIT-CHEBYSHEV exploit the oblique projection only at the beginning, and the other three at each iteration.

When the basis $\mathbf{W}(\varepsilon)$ is of small dimension, and this is one of the desired features for our approach (with respect to the use of the CHEBYSHEV-PARTIAL SPECTRAL FACTORIZATION), it is clear that the extra cost at the beginning is not extremely important and can be quickly compensated after very few iterations, and this for reasonable backward error. When the basis is large, and this is the case in Figure 4.2 (b) where $\mathbf{W}(\varepsilon)$ is of dimension 23, the cost of using the oblique projector is very important, not only at the beginning but also within the iterations, and the gains in terms of speed of convergence may not be good enough to compensate for this extra cost, as we can observe for DEF-CG, PROJ-CG, and SLRU. Only those techniques like INIT-CG or INIT-CHEBYSHEV can eventually do better than the classical preconditioned conjugate gradient, since they perform the oblique projection only at the beginning. As a last comment, we mention that in the case of the experiments in Figure 4.2, the preconditioned conjugate gradient performs 65 iterations to reach a backward error

of 10^{-10}; the other Krylov techniques perform 51 iterations when $\mu = \lambda_{\max}/40$ and $\mathbf{W}(\varepsilon)$ is of size 3, and 30 iterations when $\mu = \lambda_{\max}/10$ and $\mathbf{W}(\varepsilon)$ is of size 23; and INIT-CHEBYSHEV performs 74 iterations in the first case and 36 iterations in the latter case (for the same final backward error).

There is of course a trade-off to manage, and this is problem dependent, between the overhead that is directly linked to the size of the *near*-invariant basis $\mathbf{W}(\varepsilon)$, and the resulting reduced condition number of the iteration matrix. As mentioned in section 2.5, the parameter μ splits the spectrum of the matrix into two subsets and determines the dimension p of the invariant subspace that will be computed in the partial spectral factorization phase, depending of course on the actual eigenvalue distribution in the iteration matrix $\widehat{\mathbf{A}}$. A direct consequence of this choice of μ can be observed in the behavior of the INIT-CHEBYSHEV algorithm, which is explicitly linked to the reduced condition number $\lambda_{\max}/\mu$, as opposed to other Krylov techniques which rely more on the global distribution of eigenvalues. Indeed, in Figure 4.2, we can see that INIT-CHEBYSHEV may not be a good competitor despite the small size of the $\mathbf{W}(\varepsilon)$ basis when μ is set to $\lambda_{\max}/40$ (and the $\mathbf{W}(\varepsilon)$ basis is of size 3) as opposed to $\lambda_{\max}/10$ (with a $\mathbf{W}(\varepsilon)$ basis is of size 23).

Finally, the different histograms show that the three techniques DEF-CG, PROJ-CG, and SLRU exhibit very similar behavior. A fine grain analysis based on the values given in Table 4.1 can help to conclude that SLRU is a little cheaper than the other two. However, if the *near*-invariant basis $\mathbf{W}(\varepsilon)$ is very accurate (independently of its size), INIT-CG is the algorithm of choice with respect to these three techniques, and even with respect to INIT-CHEBYSHEV, which will at any rate suffer from a slower speed of convergence.

4.2. Cost benefit and remarks. The purpose of this subsection is to compare, in terms of cost benefits, the various iterative solution techniques introduced in section 2. We now consider the situation where one is interested in consecutively solving several linear systems with the same coefficient matrix but with changing right-hand sides that are computed sequentially. We illustrate how the gains obtained at each solve can reduce quite substantially the total computational cost in the long run.

Concerning the precomputation of the *near*-invariant basis $\mathbf{W}(\varepsilon)$, we have used the CHEBYSHEV-PSF algorithm proposed by [2]. The main reason for the choice of the CHEBYSHEV-PSF algorithm was to enable the computation of an approximation of the invariant subspace linked to all eigenvalues in a given range $[\lambda_{\min}, \mu[$, without any a priori knowledge of their count, and to target directly a reduced condition number $\lambda_{\max}/\mu$. The only risk regarding the choice of the parameter μ is that, if it is not well located with respect to the actual clustering of the eigenvalues in the iteration matrix $\mathbf{A}$, the size of the constructed Krylov space can be large. However, and this is one requirement for the efficiency of the approach studied in this paper, the iteration matrix should have a spectrum well clusterized, and a first level of preconditioning should in general be incorporated to achieve this. Additionally, the CHEBYSHEV-PSF algorithm also offers the possibility to control the accuracy of the precomputed $\mathbf{W}(\varepsilon)$ basis and to reduce substantially the cost in this precomputation.

Other iterative techniques, like those available in ARPACK [24], for instance, can also be used to approximate a given number of eigenvalues or eigenvectors in the SPD iteration matrix $\mathbf{A}$. The difference here is that the ARPACK equivalent routines target in fact some fixed number of approximate eigenvalues. However, if this predetermined number of eigenvalues (denoted by p) is not large enough with respect to the eigenvalue distribution in $\mathbf{A}$, one may be left with a reduced condition number

$\lambda_{\max}(\mathbf{A})/\lambda_{p+1}(\mathbf{A})$ that is still rather large. The claim here is not that one method is better than the other, but just that the target is different since with CHEBYSHEV-PSF we control directly the range of eigenvalues, and with the latter technique, one can control better in general the total number of computations.

The precomputation of the *near*-invariant basis $\mathbf{W}(\varepsilon)$ has a cost which we denote by $\mathcal{C}_{\text{PSF}}$ and that depends on the dimension and on the accuracy of the computed *near*-invariant subspace. To be effective, the gains obtained in the acceleration of the convergence of the given iterative solvers must compensate, in some way, for the extra cost of the computation of this spectral information.

In the CHEBYSHEV-PSF algorithm, we perform the Chebyshev filtering polynomials in $\mathbf{A}$ at the beginning, to generate an initial filtered vector $\mathbf{w}^{(0)}$, and at each step, to maintain, under the level of filtering ε, the relative gap in the eigencomponents associated with all eigenvalues in the range $[\mu, \lambda_{\max}(\mathbf{A})]$ in the newly computed Lanczos vectors $\mathbf{w}^{(k+1)}$. The cost of one Chebyshev inner iteration is $\mathcal{C}_{\mathbf{A}} + 3\,\mathcal{C}_{\texttt{_AXPY}}$ (see the formulas (4.1), (4.3)). The CHEBYSHEV-PSF algorithm also involves the computation of $\left(\mathbf{I} - \mathbf{W}\mathbf{W}^T\right)\mathbf{A}\mathbf{w}^{(k)}$ at each Lanczos step, to get the next entry $\mathbf{w}^{(k+1)}$ in the Krylov basis $\mathbf{W}$, before filtering it again. This computation is a BLAS2 operation whose total cost for $(p-1)$ Lanczos steps is given by

$$\mathcal{C}_{\text{Lanczos}} \approx (p-1)\,\mathcal{C}_{\mathbf{A}} + 2\,p\,(p-1)\,n. \tag{4.6}$$

In addition, each iteration also requires two dot-products, for a total cost of $2\,(p-1)\,\mathcal{C}_{\texttt{_DOT}}$ (see (4.3)). Finally, at each iteration, the filtered Krylov entry $\mathbf{z}^{(k+1)}$ must again be orthogonalized against the current orthonormal basis $\mathbf{W}$ to build the new vector $\mathbf{w}^{(k+1)}$ in the *near*-invariant basis $\mathbf{W}$ (i.e., $\mathbf{z}^{(k+1)} - \mathbf{W}\mathbf{W}^T\mathbf{z}^{(k+1)}$). The total cost of these last operations after $(p-1)$ global iterations can be estimated as

$$\mathcal{C}_{\text{Ortho}} \approx 2\,n\,p^2 - n\,p - n. \tag{4.7}$$

Thus, the cost of precomputing the orthonormal basis $\mathbf{W}$ with the CHEBYSHEV-PSF algorithm is of order

$$\mathcal{C}_{\text{PSF}} \approx \left(\mathcal{C}_{\mathbf{A}} + 3\,\mathcal{C}_{\texttt{_AXPY}}\right) \times \texttt{ChebIt} + 2\,(p-1)\,\mathcal{C}_{\texttt{_DOT}} + \mathcal{C}_{\text{Ortho}} + \mathcal{C}_{\text{Lanczos}}, \tag{4.8}$$

where `ChebIt` is the total number of Chebyshev filtering iterations performed in the CHEBYSHEV-PSF algorithm.

In Table 4.2, we report the number of floating-point operations (in `Mflop`, and in `Gflop` for Problem PDE2 with largest size) and the total number of Chebyshev filtering steps in the CHEBYSHEV-PSF algorithm, and this for each matrix, and with different values for the filtering level ε.

In 4.3, 4.4, and 4.5, we illustrate, in the case of PDE1, BCSSTK15, and PDE2 test problems, the cost benefit of the various algorithms described in section 2. Let us consider, for instance, that we stop the iterations when the relative residual norm $||\widehat{\mathbf{b}} - \widehat{\mathbf{A}}\mathbf{x}^{(i)}||_2/||\widehat{\mathbf{b}}||_2$ is below 10^{-10}. For each level of filtering ε, we indicate in these tables the number of conjugate gradient iterations (`Nit`), and the number of floating-point operations (`Mflop`) to reach the above backward error.

We also compute the number of right-hand sides that have to be considered in subsequent consecutive solves before the extra cost $\mathcal{C}_{\text{PSF}}$ for the computation of the *near*-invariant basis $\mathbf{W}(\varepsilon)$ is compensated. This is indicated as the number of amortization vectors (`Amor.Vec`). To compute this number, we must compare the number of floating-point operations needed to reach the given level of backward error

TABLE 4.2
Cost of CHEBYSHEV-PSF *for different values of level of filtering* ε.

MATRIX	SIZE OF	VALUE OF FILTERING LEVEL ε							
		$\varepsilon = 10^{-16}$		$\varepsilon = 10^{-8}$		$\varepsilon = 10^{-4}$		$\varepsilon = 10^{-2}$	
	W	`ChebIt`	`Mflop`	`ChebIt`	`Mflop`	`ChebIt`	`Mflop`	`ChebIt`	`Mflop`
BCSSTK14	23	592	135.17	470	108.68	326	77.42	209	52.02
BCSSTK15	4	136	94.37	91	63.46	59	41.48	42	29.80
PDE1	3	108	42.87	70	27.96	49	19.72	32	13.04
		`ChebIt`	`Gflop`	`ChebIt`	`Gflop`	`ChebIt`	`Gflop`	`ChebIt`	`Gflop`
PDE2	16	9911	52.83	6027	32.21	3100	16.66	1970	10.66

TABLE 4.3
Cost and benefits of different approaches for different values of level of filtering ε. *The conjugate gradient iterations are stopped when the relative residual is below* 10^{-10}.

	PDE1									
	INIT-CG		DEF-CG		PROJ-CG		SLRU		INIT-CHEBYSHEV	
ε	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`
10^{-16}	27	11.88	27	15.08	27	14.95	27	14.85	36	14.58
10^{-8}	27	11.88	27	15.08	27	14.95	27	14.85	∞	∞
10^{-4}	57	24.61	27	15.08	27	14.95	27	14.85	∞	∞
10^{-2}	57	24.61	38	21.15	38	21.02	45	24.78	∞	∞
ε	`Amor. Vec`		`Amor. Vec`		`Amor. Vec`		`Amor. Vec`		`Amor. Vec`	
10^{-16}	4		5		5		5		5	
10^{-8}	3		4		4		4		∞	
10^{-4}	∞		3		3		3		∞	
10^{-2}	∞		5		5		∞		∞	
	BCSSTK15									
	INIT-CG		DEF-CG		PROJ-CG		SLRU		INIT-CHEBYSHEV	
ε	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`	`Nit`	`Mflop`
10^{-16}	29	21.40	29	23.26	29	23.18	29	22.95	36	25.77
10^{-8}	29	21.40	29	23.26	29	23.18	29	22.95	45	31.95
10^{-4}	45	32.65	29	23.26	29	23.18	29	22.95	∞	∞
10^{-2}	45	32.65	32	25.61	32	25.53	32	25.30	∞	∞
ε	`Amor. Vec`		`Amor. Vec`		`Amor. Vec`		`Amor. Vec`		`Amor. Vec`	
10^{-16}	9		11		11		10		14	
10^{-8}	6		7		7		7		∞	
10^{-4}	∞		5		5		5		∞	
10^{-2}	∞		5		5		5		∞	

with the conjugate gradient algorithm, and the number of floating point operations to reach the same level with any of the other techniques. Note that $\mathbf{A}_c$ is factorized once at the end of the spectral precomputation, and these factors are used for any solution with $\mathbf{A}_c$. Consequently, the number of amortization vectors (`Amor. Vec`) is

TABLE 4.4

Cost and benefits of different approaches on the PDE2 anisotropic problem for different values of level of filtering ε. The conjugate gradient iterations are stopped when the relative residual is below 10^{-10}.

	INIT-CG			DEF-CG			SLRU		
ε	Nit	Gflop	Amor. Vec	Nit	Gflop	Amor. Vec	Nit	Gflop	Amor. Vec
10^{-16}	283	1.77	12	283	4.87	35	283	4.86	35
10^{-8}	326*	2.11*	8*	292	5.02	24	310	5.32	31
10^{-4}	1060	6.49	∞	353	6.05	50	413	7.06	∞
10^{-2}	1060	6.49	∞	450	7.70	∞	533	9.09	∞

*In this case we use CG based Two-Grid solver with only one coarse grid correction.

TABLE 4.5

Number of amortization vectors. The computation of amortization vectors is relative to CHEBYSHEV-*based* TWO-GRID *solver.*

	CHEBYSHEV BASED TWO-GRID SOLVER							
	PDE1'S MATRIX				BCSSTK15'S MATRIX			
ε	ξ	Nit	Mflop	Amor. Vec	ξ	Nit	Mflop	Amor. Vec
10^{-8}	15	36	16.28	4	20	36	27.50	13
10^{-4}	5	36	19.69	5	15	36	29.23	13

given by

$$\texttt{Amor. Vec} = \left\lfloor \frac{\mathcal{C}_{\text{PSF}} + \mathcal{C}_{\mathbf{A}_c}}{\mathcal{C}_{\text{CG}} - (\mathcal{C}_{Algo} - \mathcal{C}_{\mathbf{A}_c})} \right\rfloor + 1, \tag{4.9}$$

where $\mathcal{C}_{Algo}$ is the total cost in floating-point operations for the algorithm *Algo* (summarized in Table 4.1).

For instance, to reach a backward error of 10^{-10}, the conjugate gradient algorithm performs 57 iterations in the case of the PDE1 test problem with a cost of 23.76 `Mflop`, 45 iterations in the case of BCSSTK15 with a cost of 31.62 `Mflop`, and, in the case of the PDE2 anisotropic problem, 1060 iterations with a cost of 6.31 `Gflop`. If we consider the case of Problem PDE1, in particular, 42.87 `Mflop` are needed to compute a basis with a filtering level of $\varepsilon = 10^{-16}$ (see Table 4.2), out of which INIT-CG convergence is achieved in 27 iterations, with a cost of 11.88 `Mflop` (see Table 4.3). This yields a reduction of 47% compared to the run which does not use this spectral information, and the extra 42.87 `Mflop` in the spectral precomputation are paid back after only four consecutive runs of INIT-CG.

Tables 4.3 and 4.4 also show the impact of varying the filtering level ε on the actual number of amortization vectors. We can observe that the number of amortization vectors (`Amor. Vec`) is about the same for the three Krylov techniques (DEF-CG, PROJ-CG, and SLRU) for a given level of filtering ε. For example, when the *near*-invariant subspace $\mathbf{W}(\varepsilon)$ is computed with very high accuracy (i.e., with a filtering level ε close to machine precision), the extra cost is compensated in few consecutive runs, and this occurs with any of the iterative solvers considered here. As mentioned already in such a case, INIT-CG appears explicitly to be the algorithm of choice.

The INIT-CHEBYSHEV algorithm is particular since it does not belong to Krylov techniques as the others do. When the level of filtering ε is equal to 10^{-16}, INIT-

CHEBYSHEV overcomes the extra cost of the eigencomputation like the other Krylov algorithms (DEF-CG, PROJ-CG, and SLRU), as illustrated in Table 4.3, but with a little difference in the number of amortization vectors. This can be explained by the different rates of convergence between the Chebyshev method and the conjugate gradient method in general. However, when the spectral information is not extremely accurate, i.e., the level of filtering is in the range 10^{-8} to 10^{-4}, the simple use of a projector only at the beginning is not sufficient to reach a great accuracy in the Chebyshev solve.

The two-grid cycle scheme proposed in section 2.7 can be an alternative to still benefit from the available spectral information (despite its poor numerical quality), but without the expense of performing an oblique projection at each iteration as with DEF-CG, PROJ-CG, or SLRU. In Table 4.5, we indicate the cost benefit of such a scheme only with the Chebyshev iteration as the smoother, knowing that the convergence with conjugate gradient is very comparable. We also vary the number of smoothing steps in each cycle, depending on the actual accuracy of the precomputed basis $\mathbf{W}(\varepsilon)$, in order to minimize the number of coarse grid corrections (i.e., oblique projections) versus the total number of iterations.

The various numerical results illustrate that, when a great accuracy is required to solve several linear systems in sequence, the precomputed *near*-invariant subspace with a level of filtering ε higher than 10^{-2} should not be used. Furthermore, the best strategy to address this situation is to compute, if possible, an accurate *near*-invariant subspace basis $\mathbf{W}(\varepsilon)$, i.e., $(10^{-16} \leq \varepsilon \leq 10^{-8})$, and to carry out the INIT-CG algorithm or the conjugate gradient based TWO-GRID solver. We also note that, if we have a *near*-invariant subspace basis $\mathbf{W}(\varepsilon)$ not extremely but sufficiently accurate (for instance, $\varepsilon = 10^{-4}$), the use of the three other Krylov techniques (DEF-CG, PROJ-CG, and SLRU) is in general more efficient, and the TWO-GRID cycle scheme can even be a good alternative.

5. Conclusions and perspectives. In this paper, we have compared various iterative solution techniques that exploit partial spectral information about the matrix $\mathbf{A}$. The common general idea in all these techniques is to use some precomputed *near*-invariant subspace $\mathbf{W}(\varepsilon)$ to build a projection onto the orthogonal complement of $\mathbf{W}(\varepsilon)$ in some appropriate norm, and to exploit such a projector to maintain, implicitly or explicitly, throughout the iterations, the computed residuals in this orthogonal complement.

We have considered, for instance, the deflated version of the conjugate gradient algorithm [34], which consists in augmenting the generated Krylov space with the basis $\mathbf{W}(\varepsilon)$, and resumes in projecting at each conjugate gradient iteration the generated Krylov vectors. An alternative technique consists in applying the conjugate gradient algorithm to a projected system [28]. A third approach is based on a spectral low-rank update preconditioning technique [4], which aims to shift close to one the smallest eigenvalues in the original system. These three Krylov-based techniques enabled faster convergence in the conjugate gradient, with very similar linear rates of convergence that could be achieved even when the precomputed *near*-invariant basis $\mathbf{W}(\varepsilon)$ was obtained with very low accuracy.

For the sake of comparison, we have also experimented with three other *simpler* approaches, namely, the classical conjugate gradient algorithm and the Chebyshev and Richardson polynomial iterative methods, all three combined with a starting guess obtained with an oblique projection of the initial residual onto the orthogonal complement of $\mathbf{W}(\varepsilon)$. We have seen that, when the basis $\mathbf{W}(\varepsilon)$ is computed with an

accuracy close to machine precision, these three other techniques also present similar numerical behavior, with linear rates of convergence that differ slightly at any rate since their intrinsic numerical properties are different.

The resulting linear rates of convergence observed in all cases are directly connected to the cut-off value μ, which corresponds to the smallest eigenvalue in $\mathbf{A}$ that is not included in the set of eigenvalues approximated by the basis $\mathbf{W}(\varepsilon)$. More precisely, these rates correspond to those one would obtain with matrices having a reduced condition number equal to $\lambda_{\max}(\mathbf{A})/\mu$. In that respect, these last three techniques are very appealing since they exploit the oblique projection with the basis $\mathbf{W}(\varepsilon)$ only at the beginning. Finally, despite their relatively higher linear rates of convergence, the Chebyshev and Richardson polynomial iterative techniques present the additional advantage of not using any dot-products in their iterations, which can be quite beneficial when considering parallel implementations on distributed memory multiprocessor environments.

In summary, when the *near*-invariant subspace basis $\mathbf{W}(\varepsilon)$ is computed with very high accuracy (i.e., with a filtering level ε close to machine precision), the coarse-grid correction needs to be applied only once and algorithms of the type INIT-CG and INIT-CHEBYSHEV are the algorithms of choice, because they avoid the use of oblique projectors at each iteration and are much less expensive. When the computed *near*-invariant subspace basis $\mathbf{W}(\varepsilon)$ is not very accurate, the simple use of a projector only at the beginning is not sufficient to ensure a linear rate of convergence throughout the iterations. In particular, when the *near*-invariance of the precomputed basis is very poor, no improvements might be observed. As opposed to that, the first three Krylov techniques, which perform at each iteration an oblique projection with the $\mathbf{W}(\varepsilon)$ basis, do not suffer from these misconvergence problems and show a constant numerical behavior in almost every case. In the intermediate situations, e.g., when the precomputed *near*-invariant basis is not extremely but sufficiently accurate, we can use a two-grid cycle scheme to still benefit from the spectral information without the expense of performing a coarse grid correction (oblique projection) at each iteration.

The computational costs analyzed in section 4.2 have also shown that, when the spectrum of the iteration matrix is well clusterized and the $\mathbf{W}(\varepsilon)$ basis is small, the gains can be rather effective and the cost for precomputing this basis can rapidly be compensated when solving several linear systems with the same matrix but changing right-hand sides computed sequentially.

Our perspectives are to investigate the combination of conjugate gradient with Chebyshev filtering polynomials applied to a part of the spectrum of the coefficient matrix as preconditioners that help to put the conjugate gradient in a better situation for convergence. The expectation is to be capable of directly constructing a small dimensional Krylov basis naturally rich with respect to the smallest eigenvalues/eigenvectors [16].

Acknowledgments. We would like to thank the anonymous referees for their valuable remarks, questions, and comments that enabled us to substantially improve this paper.

REFERENCES

[1] M. ARIOLI AND D. RUIZ, *Block conjugate gradient with subspace iteration for solving linear systems*, in Iterative Methods in Linear Algebra, II, Proceedings of the Second IMACS International Symposium on Iterative Methods in Linear Algebra, IMACS Series in Com-

putational and Applied Mathematics 3, S. D. Margenov and P. S. Vassilevski, eds., IMACS, New Brunswick, NJ, 1995.

[2] M. Arioli and D. Ruiz, *A Chebyshev-Based Two-Stage Iterative Method as an Alternative to the Direct Solution of Linear Systems*, Technical Report RAL-TR-2002-021, Rutherford Appleton Laboratory, Atlas Center, Didcot, Oxfordshire, England, 2002.

[3] J. Baglama, D. Calvetti, G. H. Golub, and L. Reichel, *Adaptively preconditioned GMRES algorithms*, SIAM J. Sci. Comput., 20 (1998), pp. 243–269.

[4] B. Carpentieri, I. Duff, and L. Giraud, *A class of spectral two-level preconditioners*, SIAM J. Sci. Comput., 25 (2003), pp. 749–765.

[5] B. Carpentieri, L. Giraud, and S. Gratton, *Additive and Multiplicative Spectral Two-Level Preconditioners for General Linear Systems*, Technical Report TR/PA/04/38, CERFACS, Toulouse, France, 2004.

[6] T. Chan and W. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[7] A. Chapman and Y. Saad, *Deflated and augmented Krylov subspace techniques*, Numer. Linear Algebra Appl., 4 (1997), pp. 43–66.

[8] P. Concus, G. Golub, and D. O'Leary, *A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, Academic Press, New York, 1976, pp. 309–332.

[9] H. de Gersem and K. Hameyer, *A deflated iterative solver for magnetostatic finite element models with large differences in permeability*, Eur. Phys. J. Appl. Phys., 13 (2000), pp. 45–49.

[10] Z. Dostál, *Conjugate gradient method with preconditioning by projector*, Int. J. Comput. Math., 23 (1988), pp. 315–323.

[11] I. Duff, R. Grimes, and J. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[12] J. Erhel, K. Burrage, and B. Pohl, *Restarted GMRES preconditioned by deflation*, J. Comput. Appl. Math., 69 (1996), pp. 303–318.

[13] J. Erhel and F. Guyomarc'h, *An augmented conjugate gradient method for solving consecutive symmetric positive definite linear systems*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1279–1299.

[14] L. Giraud, D. Ruiz, and A. Touhami, *Krylov based and polynomial iterative solvers combined with partial spectral factorization for SPD linear systems*, in Vector and Parallel Processing, Lecture Notes in Comput. Sci. 3402, M. Daydé, J. M. Palma, J. Dongarra, and V. Hernández, eds., Springer-Verlag, Berlin, Heidelberg, 2005, pp. 635–655.

[15] G. Golub and M. Kent, *Estimates of eigenvalues for iterative methods*, Math. Comp., 53 (1989), pp. 249–263.

[16] G. Golub, D. Ruiz, and A. Touhami, *A Hybrid Approach Combining Chebyshev Filter and Conjugate Gradient for Solving Linear Systems with Multiple Right-Hand Sides*, Technical Report TR/TLSE/05/10, ENSEEIHT-IRIT, Toulouse, France, 2005; Technical Report SCCM-05-08, Stanford University, Stanford, CA, 2005.

[17] G. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III (Proceedings of a Symposium Conducted by the Mathematics Research Center, the University of Wisconsin, Madison, WI, 1977), Publ. Math. Res. Center 39, J. R. Rice, ed., Academic Press, New York, 1977, pp. 361–377.

[18] W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, New York, 1985.

[19] L. Hageman and D. Young, *Applied Iterative Methods*, Academic Press, New York, London, 1981.

[20] M. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–435.

[21] E. Kaasschieter, *Preconditioned conjugate gradient for solving singular systems*, J. Comput. Appl. Math., 24 (1988), pp. 265–275.

[22] R. Kail, F. Léné, and Y.-H. de Roeck, *An asymptotic model of laminated plates*, Rev. Européenne Élém. Finis, 6 (1997), pp. 23–42.

[23] L. Yu. Kolotilina, *Preconditioning of systems of linear algebraic equations by means of twofold deflation* I. *Theory*, Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI), 229 (1995), Chisl. Metody i Voprosy Organ. Vychisl. 11, pp. 95–152, 323 (in Russian).

[24] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, Software Environ. Tools 6, SIAM, Philadelphia, 1998.

[25] D. Loghin, D. Ruiz, and A. Touhami, *Adaptive Preconditioners for Nonlinear Systems of Equations*, Technical Report TR/PA/04/42, CERFACS, Toulouse, France, 2004; ENSEEIHT-IRIT Technical Report RT/TLSE/04/02; J. Comput. Appl. Math., to appear.

[26] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1154–1171.

[27] R. B. Morgan, *Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1112–1135.

[28] R. Nabben and C. Vuik, *A comparison of deflation and coarse grid correction applied to porous media flow*, SIAM J. Numer. Anal., 42 (2004), pp. 1631–1647.

[29] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM J. Numer. Anal., 24 (1987), pp. 355–365.

[30] D. O'Leary, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[31] D. O'Leary, *A generalized conjugate gradient algorithm for solving a class of quadratic programming problems*, Linear Algebra Appl., 34 (1980), pp. 371–399.

[32] B. Parlett, *A new look at the Lanczos algorithm for solving symmetric systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 323–346.

[33] Y. Saad, *On the Lanczos method for solving symmetric linear systems with several right-hand sides*, Math. Comp., 178 (1987), pp. 651–662.

[34] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM J. Sci. Comput., 21 (2000), pp. 1909–1926.

[35] A. Touhami, *Utilisation des Filtres de Tchebycheff et Construction de Préconditonneurs Spectraux pour l'Accélération des Méthodes de Krylov*, Ph.D. thesis, INPT–ENSEEIHT, Toulouse, France, 2005.

[36] R. Underwood, *An Iterative Block Lanczos Method for the Solution of Large Sparse Symmetric Eigenproblems*, Ph.D. Thesis STAN-CS-75-496, Computer Science Department, Stanford University, Stanford, CA, 1975.

[37] A. van der Sluis and H. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.

[38] H. van der Vorst, *An iterative solution method for solving $f(A)x = b$ using Krylov subspace information obtained for the symmetric positive definite matrix A*, J. Comput. Appl. Math., 18 (1987), pp. 249–263.

[39] F. Vermolen and C. Vuik, *The Influence of Deflation Vectors at Interfaces on the Deflated Conjugate Gradient Method*, Report 01-13, Department of Applied Mathematical Analysis, Delft University of Technology, Delft, The Netherlands, 2001.

SIAM J. SCI. COMPUT.
Vol. 27, No. 5, pp. 1787–1801
© 2006 Society for Industrial and Applied Mathematics

SCHURRAS: A RESTRICTED VERSION OF THE OVERLAPPING SCHUR COMPLEMENT PRECONDITIONER*

ZHONGZE LI† AND YOUSEF SAAD‡

Abstract. This paper presents a preconditioner based on solving approximate Schur complement systems with the overlapping restricted additive Schwarz (RAS) methods. The construction of the preconditioner, called SchurRAS, is as simple as in the standard RAS. The communication requirements for each application of the preconditioning operation are also similar to those of the standard RAS approach. In the particular case when the degree of overlap is one, then SchurRAS and RAS involve exactly the same communication volume per step. In addition, SchurRAS has the same degree of parallelism as RAS. In some numerical experiments with a model problem, the convergence rate of the method was found to be similar to that of the multiplicative Schwarz (MS) method. The Schur-based RAS usually outperforms the standard RAS both in terms of iteration count and CPU time. For a few two-dimensional scaled problems, SchurRAS was about twice as fast as the stardard RAS on 64 processors.

Key words. parallel preconditioning, distributed sparse linear systems, Schur complement techniques, domain decomposition, additive Schwarz preconditioner, restricted additive Schwarz preconditioner

AMS subject classifications. 65F10, 65N55, 65Y05, 65F50

DOI. 10.1137/040608350

1. Introduction.

In the past few decades, *incomplete LU* (ILU) preconditioners, combined with Krylov subspace acceleration, have yielded techniques which offer a good compromise between robustness and efficiency when solving general large sparse linear systems of equations; see, e.g., [14, 21]. The extension of these techniques to parallel computing environments has followed two distinct paths. The first is to extract parallelism via a reordering of the unknowns. The second is to resort to domain decomposition ideas. Multicoloring is a standard way of uncovering parallel tasks in an algorithm such as Gaussian elimination. For example, parallel ILU-type preconditioners based on graph partitioning and multicoloring were recently presented [8, 6, 12]. Constructing a parallel ILU preconditioner with multicoloring starts with a coloring of the interface variables. The elimination proceeds with the interior nodes first, and this phase is perfectly parallel. Then the interface variables are eliminated in a certain order which is based on the colors of the interfaces. This basic idea was already exploited for a parallel implementation of ILU(0) in 1994 [11] (see also the 1996 edition of [14]).

Concerning the second class of methods mentioned above, domain decomposition methods (DDM) have given rise to a general-purpose strategy for constructing global

*Received by the editors May 15, 2004; accepted for publication (in revised form) November 16, 2005; published electronically February 3, 2006.

http://www.siam.org/journals/sisc/27-5/60835.html

†State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, China (lzz@lsec.cc.ac.cn). This author's work was supported by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education of Ministry, China, and by the National Basic Research Program of China under grant 2005CB321702.

‡Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455 (saad@cs.umn.edu). This author's work was supported by the NSF under grant ACI-0305120 and by the Minnesota Supercomputing Institute.

preconditioners for sparse linear systems [14, 21]. These methods were initially developed for solving partial differential equations (PDEs), but there is an increasing interest in DDM-like preconditioners for general sparse linear systems obtained from a purely algebraic viewpoint; see, e.g., [2, 13, 7, 10]. In DDM, the initial problem domain is partitioned into a number of subdomains, possibly with overlapping interfaces. Such extensions lead to *distributed sparse systems* [14], whereby the equations and associated unknowns are assigned to processors with the help of a graph partitioning approach. Using this viewpoint, many of the DDM can be retrofitted to the solution of distributed sparse linear systems. The idea is to compute an approximate solution from solutions in the subdomains, exploiting knowledge about the operator, discretization, and other properties of the problem when possible. In the simplest case of the additive Schwarz methods (ASM), the iterative process consists of simply solving independent problems and updating residuals in each domain after each iteration. Variants of ASM are obtained by varying the amount of overlap and the subdomain solvers. If the subdomains do not overlap, the procedure results in a method that essentially amounts to a *block Jacobi* preconditioner. It is important to note that Schwarz-type procedures are straightforward to extend to general sparse linear systems, at least from an algorithmic point of view. Though the body of theory that is available in DDM for PDEs does not easily extend without additional information, domain decomposition gives rise to fairly efficient methods for sparse linear systems, which have the advantage of being easy to implement [2].

In [3] Cai and Sarkis introduced an effective variant of the Schwarz procedure called the restricted additive Schwarz (RAS) method for the case where subdomains are overlapped and each subdomain problem is approximately solved using an incomplete factorization. The initial motivation that led to the discovery of RAS was to reduce communication costs—by omitting half of the communication in the standard overlapping ASM. However, it turns out that this omission often has the unexpected effect of also reducing the number of iterations.

In practice, ASM-type preconditioners are preferable to multicoloring ILU preconditioners in that they yield a better performance in most cases [4, 1]. ASM usually outperforms nonoverlapping block Jacobi preconditioners both in terms of iteration counts and execution times because the block Jacobi preconditioner ignores all interdomain couplings. To improve the performance for nonoverlapping domain decomposition, Schur complement-based preconditioners were advocated as alternatives to the simple block Jacobi techniques [15, 10]. Though Schur complement preconditioners take fewer iterations to converge, they lead to higher execution times to converge in some cases because of the additional costs incurred by the inner iterations when solving the Schur complement systems.

This paper presents a technique (SchurRAS) which, in effect, brings together the best of ASM on the one hand and Schur complement methods on the other. As in RAS, the new algorithm combines overlapping and nonoverlapping iterations. However, these iterations now involve Schur complements; i.e., they work on interface variables.

The rest of the paper is organized as follows. Section 2 introduces distributed sparse systems and Schwarz procedures, as well as Schur complements. Section 3 presents the restricted version of the Schur complement preconditioner, and section 4 gives some details on the parallel implementation of SchurRAS. Numerical experiments are shown in section 5, and some conclusions are drawn in section 6.

2. Distributed systems and distributed preconditioners. When solving a linear system $Ax = b$ on a distributed memory computer, it is common to start by partitioning the adjacency graph of the coefficient matrix A by a graph partitioner [9]. Recall that the *adjacency graph* of A is a graph $G = (V, E)$, where V is the set of all vertices representing the unknowns of the system, and E is the set of all edges representing the binary relation "a_{ij} is a nonzero entry" between vertices i and j [14]. Assuming that the matrix A has a symmetric nonzero pattern, the adjacency graph G is undirected. The partitioning of G will result in p nonoverlapping subgraphs $V_1, \ldots, V_p$. Typically, the group of equations associated with the set V_i will be assigned to processor i, along with the corresponding unknowns. Thus, each processor holds a set of equations (rows of the linear system) and vector components associated with these rows. Note that we assume a "vertex-based" partitioning. This means that vertices are assigned to processors (instead of edges or elements in a finite element problem). In this situation, unknowns can be categorized into three classes: (1) interior variables which are coupled only with local variables by the equations; (2) local interface variables which are coupled with nonlocal (external) variables as well as local variables; and (3) external interface variables (which are unknowns in other processors) that are coupled with local variables. Figure 1 illustrates this.

When putting together the equations assigned to a given processor, one observes that the coefficient matrix consists of two pieces: The first, called A_i, acts on local variables, and the other, called X_i, acts on external variables. Thus, the global linear system $Ax = b$ can be rewritten in each processor as

$$A_i x_i + X_i x_{i,ext} = b_i, \tag{2.1}$$

where x_i are the vector of local unknowns, $x_{i,ext}$ are the external interface variables, and b_i is the corresponding local part of the right-hand side vector. The above system is referred to as the "local system."

For convenience, a local reordering of the data is often exploited by splitting the local vector x_i into two parts: the subvector u_i of interior nodes followed by the subvector v_i of local interface nodes. The local right-hand side vector b_i is conformally split into the subvectors f_i and g_i, so that

$$x_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}, \quad b_i = \begin{pmatrix} f_i \\ g_i \end{pmatrix}. \tag{2.2}$$

With this reordering, (2.1) takes the form

$$\underbrace{\begin{pmatrix} B_i & E_i \\ F_i & C_i \end{pmatrix}}_{A_i} \begin{pmatrix} u_i \\ v_i \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \sum_{j \in N_i} E_{ij} v_j \end{pmatrix}}_{X_i x_{i,ext}} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}. \tag{2.3}$$

Here, N_i is the set of indices for subdomains that are adjacent to the subdomain i. The term $E_{ij}v_j$ is the contribution to the local equation from neighboring subdomain j. The zero block in the second term of the left-hand side of (2.3) is due to the fact that local internal nodes are not coupled with external nodes.

2.1. Schwarz procedures for distributed systems. Let the global domain Ω be decomposed into p nonoverlapping subdomains Ω_i, obtained from a partitioning of V into p nonoverlapping subgraphs V_i^0. Here the superscript 0 is added to indicate that the domains do not overlap. Each subdomain Ω_i can then be extended to a larger

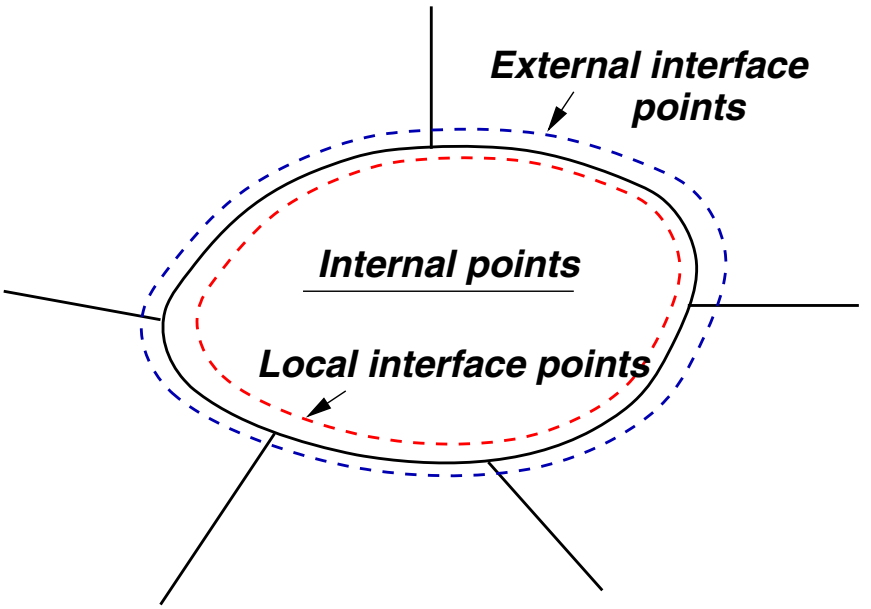


FIG. 1. *A local view of a distributed sparse matrix.*

subdomain Ω_i^1 to form a one-level overlapped subdomain by including all immediate neighboring nodes of V_i^0. We may similarly define a δ-level overlapped subdomain Ω_i^δ with the set of nodes V_i^δ. Associated with each subdomain Ω_i^δ, we can define the restriction operator

$$\begin{aligned} R_{i,\delta} : \Omega &\longrightarrow \Omega_i^\delta, \\ z &\longrightarrow z_{i,\delta}, \end{aligned}$$

where $z_{i,\delta} = R_{i,\delta} z$ is the vector of Ω_i^δ which consists of only components of z belonging to Ω_i^δ. It is common to use the transpose $R_{i,\delta}^T$ as the corresponding *extension* operator

$$\begin{aligned} R_{i,\delta}^T : \Omega_i^\delta &\longrightarrow \Omega, \\ z_{i,\delta} &\longrightarrow z, \end{aligned}$$

where $z = R_{i,\delta}^T z_{i,\delta}$ is the zero-extension of a vector $z_{i,\delta}$ on Ω_i^δ to Ω. The local matrix $A_{i,\delta}$ associated with the subdomain Ω_i^δ can be expressed as $A_{i,\delta} = R_{i,\delta}^T A R_{i,\delta}$. The additive Schwarz preconditioner can be written as [19]

$$\mathcal{B}_{[\delta,\delta]} = \sum_{i=1}^{p} R_{i,\delta}^T A_{i,\delta}^{-1} R_{i,\delta}. \tag{2.4}$$

In the case when there is no overlap, i.e., when $\delta = 0$, we obtain the special case of the *block Jacobi* preconditioner

$$\mathcal{B}_{[0,0]} = \sum_{i=1}^{p} R_{i,0}^T A_{i,0}^{-1} R_{i,0}. \tag{2.5}$$

Finally, using the same notation, the RAS proposed by Cai and Sarkis [3] can be written as

$$\mathcal{B}_{[0,\delta]} = \sum_{i=1}^{p} R_{i,0}^T A_{i,\delta}^{-1} R_{i,\delta}. \tag{2.6}$$

The solve with the matrix $A_{i,\delta}$ is rarely performed exactly. Instead, one can obtain an incomplete LU factorization and solve the systems with a preconditioned Krylov method. Examining the incomplete factorization carefully will also yield an approach based on Schur complements.

2.2. Schur complement-based preconditioners. Eliminating u_i from the top part of (2.3) leads to the relation

$$u_i = B_i^{-1}[f_i - E_i v_i], \tag{2.7}$$

which upon substitution into the bottom part, results in the following equation for v_i:

$$S_i v_i + \sum_{j \in N_i} E_{ij} v_j = g_i - F_i B_i^{-1} f_i \equiv g_i', \quad i = 1, \ldots, p, \tag{2.8}$$

where S_i is the "local" Schur complement matrix

$$S_i = C_i - F_i B_i^{-1} E_i. \tag{2.9}$$

Note that this elimination of the u_i's is local in that it can be carried out on each processor in parallel since interior unknowns of each subdomain are not coupled with variables of other subdomains.

Equations (2.8) for all subdomains constitute the global Schur complement system which involves only all interface variables v_i, $i = 1, \ldots, p$. The global Schur complement matrix has a natural block structure,

$$\begin{pmatrix} S_1 & E_{12} & E_{13} & \cdots & E_{1p} \\ E_{21} & S_2 & E_{23} & \cdots & E_{2p} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ E_{p1} & E_{p2} & E_{p3} & \cdots & S_p \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ \vdots \\ v_p \end{pmatrix} = \begin{pmatrix} g_1' \\ g_2' \\ \vdots \\ \vdots \\ g_p' \end{pmatrix}. \tag{2.10}$$

The above system can be written as

$$Sv = g',$$

where v consists of all interface variables $v_1, v_2, \ldots, v_p$ stacked into a long vector. The matrix S is the global Schur complement matrix.

If the global Schur complement system (2.10) is solved exactly, the u variables can be obtained from (2.7), and the exact solution of the whole system will then be obtained. Preconditioning techniques can be obtained by solving the Schur complement system approximately.

3. RAS versions of Schur complement preconditioners. Once interface variables v_i are available, all interior variables u_i can be computed via (2.7). This process can be carried out in parallel. If we simply drop the terms E_{ij} in (2.10), v_i can be computed via $S_i^{-1}g_i'$ without any communication among processors. This amounts to a block Jacobi preconditioner. In passing, we note that the block Jacobi preconditioner for the original linear system can be induced by the block Jacobi preconditioner for the global Schur complement system [14, 15]. Due to the local nature of the block Jacobi preconditioner, the resulting iteration will usually converge slowly, since one may expect that changes in a given domain will take quite a few steps before propagating to all domains.

One way to cure this is to attack the global Schur complement system directly, using Krylov subspace methods with block Jacobi or distributed ILU(0) as preconditioners [15, 10]. Solving the global Schur complement system more accurately will result in a preconditioner that is usually superior to the standard block Jacobi preconditioner for the original system. This superiority comes from the fact that Schur complement-based preconditioners tend to require fewer iterations than ASM to converge. However, note that because the inner steps for solving the global Schur complement systems are more expensive, the overall performance may be poorer in some cases [10].

Part of the problem with the Schur complement-based methods developed in [10, 15] lies in the ineffectiveness of the block Jacobi preconditioner which is used to solve the global Schur complement system. Since the RAS shows better performance than the block Jacobi for the original linear system, one can think of solving the global Schur complement system by using an RAS-like procedure. We will refer to this approach as SchurRAS. Because from some classes of problems the condition number of the Schur complement matrix is better than that of the original matrix [5, 19] and because RAS is a better preconditioner than the block Jacobi preconditioner, we may expect to obtain a more "exact" solution for the global Schur complement system. Therefore, SchurRAS should outperform the block Jacobi preconditioner. This will be borne out by the experiments.

3.1. The SchurRAS preconditioner. The local matrix $A_{i,\delta}$ can be written in the following block form:

$$A_{i,\delta} = \begin{pmatrix} B_{i,\delta} & E_{i,\delta} \\ F_{i,\delta} & C_{i,\delta} \end{pmatrix}, \tag{3.1}$$

where $B_{i,\delta}$ is the matrix associated with interior nodes of subdomain Ω_i^δ. Recall that interior nodes are those nodes which are coupled with unknowns from other domains. Assume that the ILU factorization of $B_{i,\delta}$ is available as

$$B_{i,\delta} \approx L_{i,\delta} U_{i,\delta}$$

and consider the following block ILU factorization of $A_{i,\delta}$:

$$\begin{aligned} A_{i,\delta} &\approx \begin{pmatrix} L_{i,\delta} & 0 \\ F_{i,\delta}U_{i,\delta}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,\delta} & L_{i,\delta}^{-1}E_{i,\delta} \\ 0 & S_{i,\delta} \end{pmatrix} \\ &= \begin{pmatrix} L_{i,\delta} & 0 \\ F_{i,\delta}U_{i,\delta}^{-1} & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & S_{i,\delta} \end{pmatrix} \begin{pmatrix} U_{i,\delta} & L_{i,\delta}^{-1}E_{i,\delta} \\ 0 & I \end{pmatrix} \\ &\equiv \mathcal{L}_{i,\delta}\mathcal{D}_{i,\delta}\mathcal{U}^{i,\delta} \ . \end{aligned} \tag{3.2}$$

Then, using this approximation for $A_{i,\delta}$, the preconditioner (2.4) can be expressed as

$$\mathcal{B}_{[\delta,\delta]} = \sum_{i=1}^{p} R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1}\ \mathcal{D}_{i,\delta}^{-1}\ \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}\ . \tag{3.3}$$

We can treat $\mathcal{U}_{i,\delta}$, $\mathcal{D}_{i,\delta}$ and $\mathcal{L}_{i,\delta}$ as three separate operators, in which case it helps to expand each of them as an operator on the whole space. Then (3.3) can be recast as

$$\mathcal{B}_{[\delta,\delta]}\ = \sum_{i=1}^{p} (R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}). \tag{3.4}$$

Similarly, in the nonoverlapping case the block Jacobi preconditioner (2.5) becomes

$$\mathcal{B}_{[0,0]} = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{D}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0}), \tag{3.5}$$

while the RAS preconditioner takes the form

$$\mathcal{B}_{[0,\delta]}\ = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta}). \tag{3.6}$$

Note that in the above equations, the solves with $\mathcal{D}_{i,\delta}$ correspond to solving Schur complement systems. The operations $(R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta})$ and $(R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})$ are akin to a restriction and a prolongation operator, respectively. If we assume for a moment that the factorization $B_{i,\delta} = L_{i,\delta} U_{i,\delta}$ is exact, then one can expect that all that is required is an exact solution to the Schur complement system, to obtain an exact solution of the global problem. Keeping in mind the interpretation of $(R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0})$ and $(R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})$ as "prolongation" and "restriction" operators, a careful look at (3.5) reveals that only the term $R_{i,0}^T \mathcal{D}_{i,0}^{-1} R_{i,0}$ needs to be improved to lead to an improvement to the global preconditioner. Recall that the assumption here is that the factorization of $B_{i,0}$ is exact. We can therefore focus on this operator and obtain a more exact solution by possibly using a preconditioned GMRES iteration [15, 10]. What is known about RAS can now be exploited, and we arrive naturally at the following definition of the SchurRAS preconditioner:

$$\mathcal{B} = \sum_{i=1}^{p} (R_{i,0}^T \mathcal{U}_{i,0}^{-1} R_{i,0})\ (R_{i,0}^T \mathcal{D}_{i,\delta}^{-1} R_{i,\delta})\ (R_{i,0}^T \mathcal{L}_{i,0}^{-1} R_{i,0}). \tag{3.7}$$

One may ask why we do not use the overlapping versions of the prolongation and restriction operators $(R_{i,\delta}^T \mathcal{L}_{i,\delta}^{-1} R_{i,\delta})$ and $(R_{i,\delta}^T \mathcal{U}_{i,\delta}^{-1} R_{i,\delta})$ in the above expressions. The purpose of SchurRAS is to construct a preconditioner for the original linear system by solving the Schur complement system with RAS. It is the nonoverlapping versions of the prolongation and restriction operators that lead to the Schur complement system, not the overlapping ones.

Comparing (3.6) and (3.7), we can see that SchurRAS and RAS have almost the same computational costs. Only the operator $R_{i,\delta}$ involves communications between neighboring processors. In particular, this means that SchurRAS and RAS involve the same communication when applying the preconditioner to a vector. According to

(2.3), the term E_{ij} remains unchanged after the (block) Gaussian elimination process. Accordingly, the couplings of the unknowns among neighboring subdomains for the global Schur complement system are the same as those of the original linear system. In particular, when $\delta = 1$, SchurRAS and RAS involve exactly the same volume of communication.

4. Parallel implementation. This section discusses some implementation details of the preprocessing phase in which the preconditioner is computed and the iteration phase in which the preconditioner is applied at each step.

4.1. Constructing the preconditioning matrix. Consider the local matrix A_i defined by (2.3). To compute the preconditioner, we make use of a relationship that was exploited in [15], between the Schur complement of A_i with respect to C_i, i.e., $S_i = C_i - F_i B_i^{-1} E_i$, and the block LU factorization of A_i. Specifically, we have

$$A_i = L_i U_i \equiv \begin{pmatrix} L_{i,B} & 0 \\ F_i U_{i,B}^{-1} & I \end{pmatrix} \begin{pmatrix} U_{i,B} & L_{i,B}^{-1} E_i \\ 0 & S_i \end{pmatrix}. \tag{4.1}$$

What is interesting is that the above factorization can be obtained via a simple variation of Gaussian elimination referred to as the "restricted" or "partial" Gaussian elimination [17, 16]. The algorithm proceeds as any form of Gaussian elimination but does not pursue the elimination in the (2,2) block of the matrix A_i. It is described next.

ALGORITHM 4.1. *Restricted IKJ version of Gaussian elimination*

1. *For* $i = 2, n$, *Do*
2. *For* $k = 1, \min(i\text{-}1, m)$, *Do*
3. $a_{i,k} := a_{i,k}/a_{k,k}$
4. *For* $j = k{+}1, n$, *Do*
5. $a_{i,j} := a_{i,j} - a_{i,k} * a_{k,j}$
6. *End Do*
7. *End Do*
8. *End Do*

Clearly, dropping will be used in this approximate factorization, resulting in an approximation $\tilde{S}_i$ to S_i. For clarity, we use the same symbol S_i for this approximation.

Once S_i is computed we then extend it by including rows associated with all external variables v_j that are adjacent to subdomain i. The extended matrix S_i' for the Schur complement matrix S_i has the following form:

$$S_i' = \begin{pmatrix} S_i & E_{i,ext} \\ G_i & H_i \end{pmatrix}. \tag{4.2}$$

In the above definition, $E_{i,ext}$ simply denotes the interdomain coupling matrix. The matrices G_i and H_i are matrices from neighboring subdomains which act on overlapping nodes. The matrix H_i is a Schur complement corresponding to overlapping variables with respect to external interface variables from other processors. Algorithm 4.1 describes the procedure for constructing the preconditioning matrix of SchurRAS.

ALGORITHM 4.2. *Construction of SchurRAS preconditioner*

1. *Construct approximate local Schur complement matrix* S_i.
2. *Extend* S_i *to* S_i' *by including rows corresponding to external variables received from neighboring processors.*
3. *Compute the ILU factorization of* S_i'.

It is also interesting to note that the SchurRAS procedure for two subdomains can be thought of as a variant of the global ILU factorization performed in parallel on each subdomain with different global orderings. First, S_1 and S_2 can be constructed in parallel on each processor as described in section 2.2. This corresponds to eliminating u_1 and u_2 from a global point of view. Then, we can obtain the extended Schur complement matrices S_1' and S_2', which have the following forms:

$$S_1' = \begin{pmatrix} S_1 & E_{12} \\ E_{21} & S_2 \end{pmatrix}, \qquad S_2' = \begin{pmatrix} S_2 & E_{21} \\ E_{12} & S_1 \end{pmatrix}. \tag{4.3}$$

Last, the ILU factorizations for S_1' and S_2' are performed in parallel on each processor. It is easy to understand that the whole process corresponds to a global ILU factorization performed on processor 1 with the order u_1, u_2, v_1, v_2 and on processor 2 with the order of u_2, u_1, v_2, v_1.

In contrast, the additive Schwarz preconditioner (with overlap) processes the ILU factorization for the rows associated with the overlapping variables differently: It does not take into account contributions from interior nodes to the external overlapping variables. In other words, the S_2 block in the left-hand side of (4.3) would be replaced by a matrix similar to the C_2 matrix on processor 2 (albeit with a different ordering). Similarly for the right-hand side matrix S_2'.

This argument should make it clear that we expect the Schur-based preconditioner SchurRAS to outperform the additive Schwarz, at least on two processors.

4.2. Applying the preconditioner. Once the Schur complement system (2.8) has been computed, we could also extend the right-hand side into the vector $(g_i', g_{i,ext}')$. Here, $g_{i,ext}'$ is obtained from neighboring subdomains. The extended local equation on subdomain i for the global Schur complement system $SV = g'$ can be written as follows:

$$\begin{pmatrix} S_i & E_{i,ext} \\ G_i & H_i \end{pmatrix} \begin{pmatrix} v_i \\ v_{i,ext} \end{pmatrix} = \begin{pmatrix} g_i' \\ g_{i,ext}' \end{pmatrix}, \tag{4.4}$$

where $v_{i,ext}$ corresponds to the components of external variables on neighboring subdomains of subdomain i, and $g_{i,ext}'$ denotes components of the associated right-hand side vectors.

After (4.4) is approximately solved (e.g., using an ILU factorization), we obtain v_i by simply dropping the overlapping term $v_{i,ext}$; see [3]. Finally, u_i can be computed by back substitution from (2.7). The whole process is described in the next algorithm.

ALGORITHM 4.3. *Applying SchurRAS to a vector* z

1. *Compute* $g_i' = L_i^{-1} z$.
2. *Exchange interface variables in* g_i'.
3. *Compute* v_i *by solving* (4.4).
4. *Compute* u_i *using* (2.7).

5. Numerical experiments. This section presents a few experiments with the SchurRAS preconditioning technique. The first test is with a model test problem on a regular rectangular grid. The second considers a more realistic linear system related to a magnetohydrodynamics (MHD) problem. The experiments were performed on an SGI Origin 3800 with 64 CPUs at the Institute of Computational Mathematics and Scientific/Engineering Computing, Chinese Academy of Sciences, China.

5.1. Scaled problem: Two-dimensional case. Consider the elliptic PDE

$$-\Delta u + 100\frac{\partial}{\partial x}\left(e^{xy}u\right) + 100\frac{\partial}{\partial y}\left(e^{-xy}u\right) - 10u = f \tag{5.1}$$

on a square region with Dirichlet boundary conditions, discretized with a five-point centered finite-difference scheme on an $n_x \times n_y$ grid, excluding boundary points. The mesh is mapped to a virtual $p_x \times p_y$ grid of processors, such that a subrectangle of $r_x = n_x/p_x$ points in the x direction and $r_y = n_y/p_y$ points in the y direction is mapped to a processor. In the following experiments, the mesh size in each processor is kept constant at $r_x = 100$, $r_y = 100$. As the number of processors increases, the problem size increases proportionally. For example, on 4 processors the mesh is $(2r_x, 2r_y)$ leading to a problem size of $40,000$, and on 16 processors the mesh is $(4r_x, 4r_y)$ leading to a problem size of $160,000$. The resulting problems become harder as the number of processors grows. In a perfectly scalable situation, the final execution time for solving the problem with size $n = 160,000$ on 64 processors should be identical with the time for solving the problem of size $n = 40,000$ on 4 processors. Note that in order to maintain the aspect ratio of the physical domain, we need to consider *square* processor grids of increasing size.

In the following tests, ILU with threshold (ILUT) [14] is used as the local preconditioner. The fill-in parameter is 20, and the dropping tolerance is set to 0.001. Table 1 shows a comparison of preconditioners. Here, *its* indicates the number of iterations, t_{set} denotes the time for building the preconditioner, and t_{it} is the iteration time. BJ, RAS, and SchurRAS stand for the block Jacobi, the restricted additive Schwarz, and the Schur additive Schwarz method, respectively. The amount of overlap is 1 for RAS and SchurRAS. Flexible GMRES (FGMRES) [14] with the subspace size of 60 was used as an accelerator in all tests. The iteration is stopped when the residual norm is reduced by a factor of 10^6.

As shown in Table 1, the construction phase for all three preconditioners exhibits good scalability: The time for constructing preconditioners is almost a constant for three preconditioners as the number of processors increases. It is relatively more expensive to construct the preconditioning for SchurRAS due to the extra cost of constructing the approximate Schur complement matrix with Algorithm 4.1. If one of the methods in ARMS (Algebraic Recursive Multilevel Solvers) [16] is used as the local preconditioner instead of ILUT, then this additional cost can be reduced because the Schur complement matrix can be naturally obtained as a by-product of the ARMS preconditioning matrix. Table 1 also shows that when iteration counts and iteration times are considered, SchurRAS is the fastest of the three preconditioners. As the number of processors increases, the increase in the iteration counts for SchurRAS is more moderate than for the other two methods. The number of iterations taken by the SchurRAS on 64 processors is almost exactly half that required by RAS, which is itself almost exactly half that required by block Jacobi. The observations regarding iteration times are similar: The resulting iteration time for SchurRAS on 64 processors is about half that of RAS, which is about 55% that of BJ. This means in particular that the computational costs to apply the preconditioner to a vector is almost the same for SchurRAS and RAS. According to [19], for the Poisson equation, the multiplicative Schwarz (MS) preconditioner requires about half the iterations of the additive Schwarz preconditioner. Since the operator in this example is not quite a Laplacian, one cannot strictly make a comparison of the convergence rates of SchurRAS and MS though we expect them to be similar.

TABLE 1
Comparison of preconditioners without inner iteration for a two-dimensional PDE problem with fixed problem size. The times are in seconds.

	BJ			RAS			SchurRAS		
np	its	t_{set}	t_{it}	its	t_{set}	t_{it}	its	t_{set}	t_{it}
4	28	0.105	0.284	18	0.127	0.167	12	0.149	0.112
9	42	0.117	0.482	27	0.138	0.278	18	0.176	0.175
16	54	0.118	0.732	37	0.139	0.430	25	0.174	0.267
25	68	0.123	0.992	49	0.140	0.620	32	0.179	0.385
36	91	0.125	1.366	57	0.142	0.939	39	0.180	0.547
49	138	0.127	2.223	74	0.147	1.248	47	0.180	0.786
64	232	0.127	3.589	115	0.147	1.972	56	0.185	0.982

5.2. Using inner iterations. It is natural to ask whether the performance can be improved by employing an inner iteration for the Schur complement system. Specifically we can use the GMRES accelerator to solve the Schur complement using SchurRAS as a preconditioner. The test problem is the same as the one in the previous section. The subspace size of GMRES is 5 and the tolerance for inner iteration is 10^6 (stop iteration when initial residual norm is reduced by a factor of 10^6). Table 2 compares two different preconditioners. One is the block Jacobi combined with GMRES(5) for solving the Schur complement system. The corresponding induced preconditioner, denoted by Schur–BJ, was presented in [15]. The other preconditioner, denoted by Schur–SchurRAS, is the preconditioner induced by solving the Schur complement system with GMRES(5) with SchurRAS as a preconditioner. A comparison between Tables 1 and 2 reveals that the numbers of iterations for SchurRAS and Schur–BJ are essentially identical. At the same time, and to our surprise, Table 2 shows that Schur–SchurRAS also takes about the same number of iterations to converge as SchurRAS. The interpretation of what has happened is as follows. In all cases, the resulting Schur complement system is the same since it involves the same (interface) variables and the same approximations coming from the same ILUT. By iterating on the Schur complement system, this system is solved to an accuracy that has become better than the accuracy used to compute the Schur complement system itself (ILUT). While these iterations are helpful in the BJ case (they solve the Schur complement more accurately), they seem to be useless in the SchurRAS case. The only explanation for this is that SchurRAS solves the complement systems *very accurately, i.e., within the range of accuracy with which the Schur complement matrices are computed*. The end result is that the SchurRAS is still the fastest since it involves no inner iterations.

TABLE 2
Comparison of preconditioners with inner iteration for a two-dimensional PDE problem with fixed problem size. The times are in seconds.

	Schur–BJ		Schur–SchurRAS	
np	its	t_{it}	its	t_{it}
4	12	0.127	12	0.137
9	19	0.223	17	0.243
16	25	0.308	24	0.354
25	31	0.414	30	0.474
36	39	0.625	38	0.561
49	47	0.885	45	0.967
64	56	1.076	54	1.167

5.3. Test with a matrix from an MHD simulation. The test matrix used in the following experiment originates from the simulation of MHD flows. The flow equations are represented as coupled Maxwell equations and the Navier–Stokes equations. The conservative MHD system is modeled by the Maxwell equations, written as

$$\frac{\partial \mathbf{B}}{\partial t} - \nabla \times (\mathbf{u} \times \mathbf{B}) + \eta \nabla \times (\nabla \times \mathbf{B}) + \nabla \mathbf{q} = 0, \tag{5.2}$$

$$\nabla \cdot \mathbf{B} = 0, \tag{5.3}$$

where η, $\mathbf{B}$, $\mathbf{u}$, and q are, respectively, the magnetic diffusivity coefficient, magnetic induction field, velocity field, and the scalar Lagrange multiplier for the magnetic-free divergence constraint. In fully coupled MHD, this system is solved along with the incompressible Navier–Stokes equations. In the tests that follow, we solve problems which arise from the Maxwell equations only. A preset periodic induction field $\mathbf{u}$ is used in the Maxwell equation (5.2). The physical region is the three-dimensional unit cube $[-1, 1]^3$. The discretization uses tetrahedral elements, with a Galerkin–least-squares stabilized formulation. The magnetic diffusivity coefficient is $\eta = 1$. Details on the discretization, the time-stepping procedure, and the choice of various numerical parameters can be found in [20] as well as in [18]. The matrix is of size $n = 470,596$ and has $nnz = 23,784,208$ nonzero entries. Note that in contrast with the scaled performance experiment of section 5.1 the problem size remains the same in this experiment, regardless of the number of processors used.

The following input parameters have been selected to solve this problem: The residual is to be reduced by 10^6, the FGMRES restart value is 60, the fill-in parameter used for ILUT is 40, and the dropping tolerance is 0.001. SchurRAS always takes fewer iterations than RAS and BJ. Note, for example, that SchurRAS takes 76 iterations to converge on two processors, which is even less than the number of iterations—78—required by BJ on one processor. The execution times are 84.46s and 161.27s, respectively. The efficiency is 0.95. It is verified that SchurRAS on two processors is a variant of global ILU factorization. It takes the same number of iterations (or even fewer) as BJ on one processor. RAS and BJ converge in 86 and 117 iterations on two processors, respectively. On the SGI Origin 3800, this amounts to 128.18s and 94.43s, respectively. The corresponding efficiencies are 0.629 and 0.854, respectively. SchurRAS always takes fewer iterations than BJ and RAS. As the number of processors increases, the number of iterations required by SchurRAS exhibits moderate growth, as shown in Figure 2. Because the computational costs of SchurRAS are similar to those of RAS, the reduction in the iteration number also leads to a reduction in the execution time as shown in Figure 3.

6. Conclusions. We presented a new preconditioner, called SchurRAS, which consists of solving the approximate Schur complement system with the restricted form of the additive Schwarz method. This preconditioner can be thought of as the combination of the overlapping additive Schwarz preconditioner and the nonoverlapping Schur complement preconditioner. The new technique has two main advantages. First, the construction of the preconditioning matrix for SchurRAS is not only easy to code from existing routines but also nicely scalable. This is in contrast with parallel ILU-based preconditioners or more standard Schur complement techniques. Second, SchurRAS has the same degree of parallelism as the additive Schwarz.

The experimental tests show that SchurRAS is superior to the widely used additive Schwarz preconditioner and the Schur complement-based preconditioner. For a

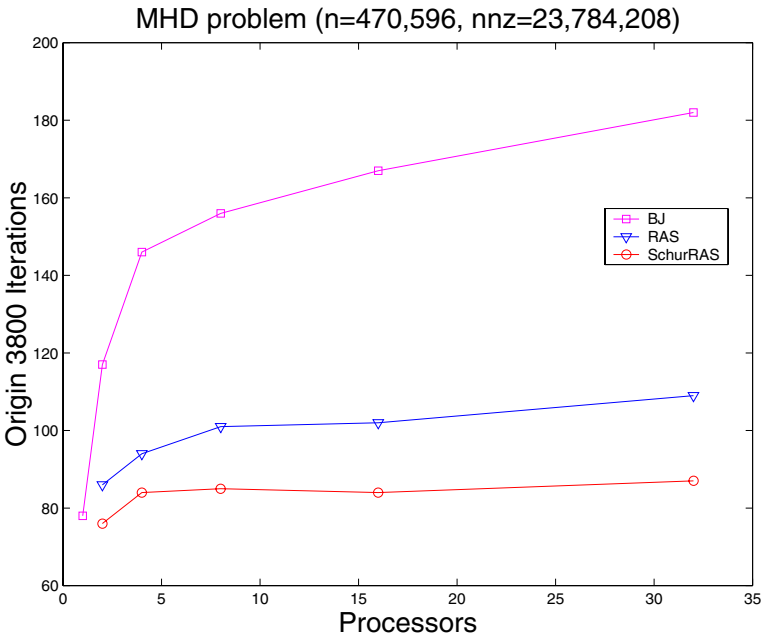


FIG. 2. *Iterations required by three methods for solving the MHD problem.*

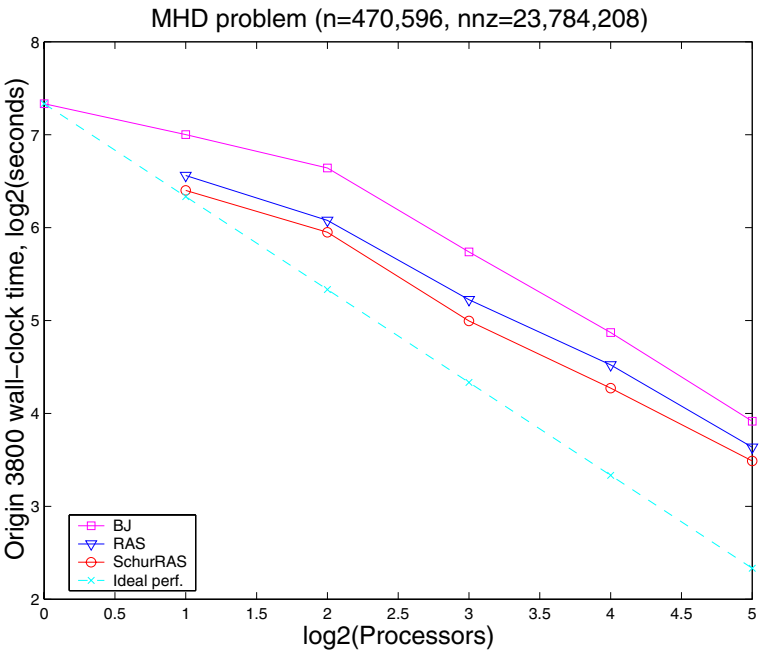


FIG. 3. *Origin seconds required by three methods for solving the MHD problem.*

model two-dimensional scaled problem, the convergence rate of SchurRAS is similar to that of the MS preconditioner. For the same model problem and using 64 processors, we found that SchurRAS is about twice as fast as the RAS.

Work is under way to implement a fully recursive multilevel version of SchurRAS. Once the approximate Schur complement is obtained by performing partial Gaussian elimination, two neighboring subdomains can be merged into a larger subdomain, as in substructuring methods [19]. Some of the interface unknowns for the above two subdomains become interior unknowns for the newly merged larger subdomain. One may construct the approximate Schur complement system for the merged subdomains, and the process can be performed recursively. The recursion is stopped when the last approximate Schur complement system is small enough. Finally, a multilevel SchurRAS can be constructed by applying RAS to the last approximate Schur complement system.

Acknowledgments. The computing resources for this work were provided by the State Key Laboratory of Scientific/Engineering Computing (LSEC), Chinese Academy of Sciences, China. The authors would like to thank Prof. Azzeddine Soulaimani (Ecole de Technologie Superieure, Montreal, Canada) for providing them with the MHD test example used in the experiments.

REFERENCES

[1] M. Benzi, W. D. Joubert, and G. Mateescu, *Numerical experiments with parallel orderings for ILU preconditioners*, Electron. Trans. Numer. Anal., 8 (1999), pp. 88–114.

[2] X.-C. Cai and Y. Saad, *Overlapping domain decomposition algorithms for general sparse matrices*, Numer. Linear Algebra Appl., 3 (1996), pp. 221–237.

[3] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[4] T. F. Chan and D. Goovaerts, *A note on the efficiency of domain decomposed incomplete factorizations*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 794–803.

[5] H. C. Elman and G. H. Golub, *Iterative methods for cyclically reduced nonself-adjoint linear systems*, Math. Comput., 54 (1990), pp. 671–700.

[6] D. Hysom and A. Pothen, *A scalable parallel algorithm for incomplete factor preconditioning*, SIAM J. Sci. Comput., 22 (2001), pp. 2194–2215.

[7] I. E. Kaporin and I. N. Konshin, *A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems*, Numer. Linear Algebra Appl., 9 (2002), pp. 141–162.

[8] G. Karypis and V. Kumar, *Parallel threshold-based ILU factorization*, in Proceedings of Supercomputing '97, San Jose, CA, IEEE Computer Society, Los Alamitos, CA, 1997; also available online at http://www.cs.umn.edu/~karypis.

[9] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[10] Z. Li, Y. Saad, and M. Sosonkina, *pARMS: A parallel version of the algebraic recursive multilevel solver*, Numer. Linear Algebra Appl., 10 (2003), pp. 485–509.

[11] S. Ma and Y. Saad, *Distributed ILU*(0) *and SOR Preconditioners for Unstructured Sparse Linear Systems*, Technical report ahpcrc-94-027, Army High Performance Computing Research Center, University of Minnesota, Minneapolis, MN, 1994.

[12] O. Yu. Milyukova, *Parallel approximate factorization method for solving discrete elliptic equations*, Parallel Comput., 27 (2001), pp. 1365–1379.

[13] M. Magolu monga Made and H. A. van der Vorst, *Parallel incomplete factorizations with pseudo-overlapped subdomains*, Parallel Comput., 27 (2001), pp. 989–1008.

[14] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.

[15] Y. Saad and M. Sosonkina, *Distributed Schur complement techniques for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 1337–1356.

[16] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.

[17] Y. Saad and J. Zhang, *BILUTM: A domain-based multilevel block ILUT preconditioner for general sparse matrices*, SIAM J. Matrix Anal. Appl., 21 (1999), pp. 279–299.

[18] N. Ben Salah, A. Soulaïmani, W. G. Habashi, and M. Fortin, *A conservative stabilized finite element method for the magneto-hydrodynamics equations*, Internat. J. Numer. Methods Fluids, 29 (1999), pp. 535–554.

[19] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.

[20] A. Soulaimani, N. B. Salah, and Y. Saad, *Enhanced GMRES acceleration techniques for some CFD problems*, Int. J. Comput. Fluid Dyn., 16 (2002), pp. 1–20.

[21] H. A. van der Vorst, *Iterative Krylov Methods for Large Linear Systems*, Cambridge University Press, Cambridge, UK, 2003.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1803–1816
© 2006 Society for Industrial and Applied Mathematics

COMPUTATIONAL STUDY OF FAST METHODS FOR THE EIKONAL EQUATION*

PIERRE A. GREMAUD† AND CHRISTOPHER M. KUSTER†

Abstract. A computational study of the fast marching and the fast sweeping methods for the eikonal equation is given. It is stressed that both algorithms should be considered as "direct" (as opposed to iterative) methods. On realistic grids, fast sweeping is faster than fast marching for problems with simple geometry. For strongly nonuniform problems and/or complex geometry, the situation may be reversed. Finally, fully second order generalizations of methods of this type for problems with obstacles are proposed and implemented.

Key words. Hamilton–Jacobi, eikonal, viscosity solution, fast marching, fast sweeping

AMS subject classifications. 65N06, 65Y20, 49L25, 35F30

DOI. 10.1137/040605655

1. Introduction. Steady state Hamilton–Jacobi equations play a major role in countless applications. While the *discretization* of those equations is approaching maturity (at least for single equations and convex Hamiltonians), the *resolution* of the nonlinear discretized systems is an active topic of research.

In this paper, two methods are compared: the fast marching (FM) method [13, 14, 17] and the fast sweeping (FS) method [4, 8, 16, 19]. The two-dimensional eikonal equation is taken as a test problem: given a slowness field F (a positive function), we look for a time of propagation u satisfying

$$|\nabla u| = F \qquad \text{in } \Omega \subset \mathrm{R}^2, \tag{1.1}$$

$$u = 0 \qquad \text{on } \Gamma \subset \mathrm{R}^2. \tag{1.2}$$

The two-dimensional domain Ω is the computational domain; Γ is a source, i.e., a closed subset of $\bar{\Omega}$ in which u is zero. The slowness field F can be unbounded, corresponding to the presence of obstacles.

We refer the reader to [8, 15] for issues related to the application of the above methods to the more general problem

$$H(x, \nabla u) = F. \tag{1.3}$$

While clearly of importance, those considerations are related more to discretization than they are to resolution and are not the focus of this paper. Discretization algorithms are only briefly reviewed in section 2. In section 3, three nonlinear solvers are described (Gauss–Jacobi, FM, and FS/nonlinear Gauss–Seidel). Section 4 presents an adaption of the FM and FS methods to problems with obstacles. The relative efficiency of FM and FS is discussed in section 5. Concluding remarks are offered in section 6.

*Received by the editors March 25, 2004; accepted for publication (in revised form) February 4, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-6/60565.html

†Department of Mathematics and Center for Research in Scientific Computation, North Carolina State University, Raleigh, NC 27695-8205 (gremaud@unity.ncsu.edu, cmkuster@unity.ncsu.edu). The work of the first author was partially supported by National Science Foundation grants DMS-0204578, DMS-0244488, and DMS-0410561. The work of the second author was partially supported by National Science Foundation grant DMS-0244488.

2. Discretization. The basic construction principles for numerical Hamiltonians are well known [10] and are easily described on a Cartesian grid. There, the basic scheme takes the form

$$\mathcal{H}(D_+^x U_{ij}, D_-^x U_{ij}; D_+^y U_{ij}, D_-^y U_{ij}) = \mathcal{F}_{ij}, \tag{2.1}$$

with the obvious notation. In that case, we recall that $\mathcal{H}$ is consistent if

$$\mathcal{H}(p,p;q,q) = H(p,q) \qquad \forall p,q, \tag{2.2}$$

where H is the Hamiltonian of the problem under consideration (here $H(p,q) = \sqrt{p^2+q^2}$). The numerical Hamiltonian is monotone if nonincreasing in its first and third arguments and nondecreasing in the other two: $\mathcal{H}(\downarrow,\uparrow,\downarrow,\uparrow)$ [6]. Consistency ensures that the correct problem is approximated while monotonicity guarantees convergence to the correct viscosity solution [5, 12].

Many numerical Hamiltonians can be found in the literature; most of them are derived from corresponding numerical fluxes for conservation laws. The main examples are the Godunov flux $\mathcal{H}^G$ and the Lax–Friedrichs flux $\mathcal{H}^{LF}$, which here take the form

$$\begin{aligned}
\mathcal{H}^G(p_+,p_-;q_+,q_-) &= \sqrt{\max\{p_-^+, p_+^-\}^2 + \max\{q_-^+, q_+^-\}^2}, \\
\mathcal{H}^{LF}(p_+,p_-;q_+,q_-) &= \sqrt{\left(\frac{p_+ + p_-}{2}\right)^2 + \left(\frac{q_+ + q_-}{2}\right)^2} \\
&\quad - \sigma_x \frac{p_+ - p_-}{2} - \sigma_y \frac{q_+ - q_-}{2},
\end{aligned}$$

where $\sigma_x \geq \max|\frac{\partial H}{\partial p}|$ and $\sigma_y \geq \max|\frac{\partial H}{\partial q}|$, where $(\cdot)^+ = \max\{\cdot, 0\}$ and $(\cdot)^- = -\min\{\cdot, 0\}$. Upwind discretizations such as the Godunov method are especially simple to implement for (1.1) and for convex Hamiltonians in general. However, $\mathcal{H}^G$, for instance, becomes very involved for general Hamiltonians. While centered methods such as Lax–Friedrichs do not suffer from that problem, they require special treatment at the boundary of the computational domain [8]. The discretization used below is essentially $\mathcal{H}^G$.

Many applications call for the use of non-Cartesian or unstructured meshes; see [1, 7, 9, 14, 15] for a few examples of such methods. Following [14, 15], consider a node X_0 at which the solution U is to be computed and two neighboring nodes X_1 and X_2 at which the values of U (U_ℓ, $\ell = 1,2$) and its derivatives ($\partial_x U_\ell, \partial_y U_\ell$, $\ell = 1,2$) are known or have already been computed. We set

$$N_\ell = \frac{X_0 - X_\ell}{|X_0 - X_\ell|}, \quad \ell = 1,2, \quad \text{and} \quad \mathcal{N} = \left[\frac{N_1}{N_2}\right],$$

where $\mathcal{N}$ is a 2×2 nonsingular matrix, assuming X_0, X_1, and X_2 are not lined up. The directional derivatives in the directions N_1 and N_2 are approximated by

$$D_\ell U = \frac{U_0 - U_\ell}{|X_0 - X_\ell|}, \qquad \ell = 1,2 \quad \text{(first order formula)}, \tag{2.3}$$

$$D_\ell U = 2\frac{U_0 - U_\ell}{|X_0 - X_\ell|} - N_\ell \cdot [\partial_x U_\ell, \partial_y U_\ell], \qquad \ell = 1,2 \quad \text{(second order formula)}. \tag{2.4}$$

Those approximate directional derivatives are linked to the gradient by

$$D_\ell U = N_\ell \cdot \nabla U + \mathcal{O}(|X_0 - X_i|^\alpha) \qquad \text{or} \qquad DU = \mathcal{N}\nabla U + \mathcal{O}(h^\alpha), \tag{2.5}$$

where $DU = [D_1U, D_2U]$, $h = \max\{|X_0 - X_1|, |X_0 - X_2|\}$, and $\alpha = 1$ or 2 depending on whether (2.3) or (2.4), respectively, is used. Solving for ∇U and plugging into (1.1), one finds the (quadratic) equation defining the unknown U_0,

$$DU^t(\mathcal{N}\mathcal{N}^t)^{-1}DU = (F(X_0))^2. \tag{2.6}$$

The above methods are well understood theoretically: convergence to the viscosity solution when the Godunov flux $\mathcal{H}^G$ is used was established in [12]; see also [15] for additional results and references.

3. Resolution. The system of coupled quadratic equations corresponding to imposing (2.6) at all nodes has to be solved. If θ denotes the angles between N_1 and N_2, the matrix $\mathcal{N}\mathcal{N}^t$ is symmetric positive definite provided $0 < \theta < \pi$; its condition number is $\frac{1+\cos\theta}{1-\cos\theta}$. Therefore, regardless of the choice of stencil (first or second order), (2.6) has two real solutions. This situation is generic for Hamilton–Jacobi equations.

Let $(U_1, U_2, \ldots, U_N)$ be the unknowns for an arbitrary ordering of the nodes. The resulting system can be written symbolically as

$$f(U) = f(U_1, U_2, \ldots, U_N) = 0. \tag{3.1}$$

An iterative scheme, essentially a fixed point method or Gauss–Jacobi method, was proposed in [12]. One step of the algorithm is as follows.

Gauss–Jacobi algorithm.

- Choose a node and regard the neighboring values as fixed.
- Solve (2.6) for the value at the considered node; the largest of the two possible solutions is selected, in agreement with the viscosity criterion.
- Repeat until convergence (fixed point).

The corresponding pseudocode is

```
k = 0
U^0 = (0, ..., 0)
while  U^k ≠ U^{k+1}  or  k ≠ 0
       k = k + 1
       for  ℓ = 1  to  N
            solve f(U_1^{k-1}, ..., U_{ℓ-1}^{k-1}, U_ℓ^k, U_{ℓ+1}^{k-1}, ..., U_N^{k-1}) = 0  for  U_ℓ^k
       end
end
```

This approach is obviously slow, as each node has to be revisited several times before the numerical solution settles down. Further, no advantage is taken of the propagation character of the problem.

With marching methods, the nodes are considered in an order consistent with the way the wave fronts propagate, i.e., with the Huygens principle [13, 14, 17]. This leads to (quasi) single pass algorithms. The fast marching algorithm, FM, is based on the monotonicity of the solution along the characteristics. The nodes are divided into three sets: the already-computed nodes, the not-yet-considered nodes, and the nodes "on the propagation front."

Fast Marching algorithm.

- The node with lowest value in the "front" set is removed from it and added to the "computed" set; its not-yet-computed neighbors are added to the "front."
- The values of all the nodes in the "front" are recomputed using (2.6).
- Repeat until all the nodes are "computed."

The list operations are handled by the heap sort algorithm [3, 18], which is the key to the speed of FM. This type of sort has the advantage that every tree operation (insertion, removal, update) is of order $\log_2 n$, where n is the number of elements in the tree, i.e., the number of "front" nodes at that step. Note that n is of order $\sqrt{N}$ at every step, where N is the total number of nodes in the mesh. By placing the "front" nodes into this structure, the computational complexity is kept at the order of $N \log_2 N$. The pseudocode for FM is

$\mathtt{U} = (\infty, \ldots, \infty)$
"front" = "source"
while "front" $\neq \emptyset$
    $\mathtt{U}_{\min} = \min \mathtt{U} \in$ "front"
    "front" = "front" + neighbors of $\mathtt{U}_{\min}$
    remove $\mathtt{U}_{\min}$ from "front"
    for $\mathtt{U}_\ell \in$ "front"
        solve $\mathtt{f}(\mathtt{U}_1, \ldots, \mathtt{U}_{\ell-1}, \mathtt{U}_\ell, \mathtt{U}_{\ell+1}, \ldots, \mathtt{U}_\mathtt{N}) = 0$ for $\mathtt{U}_\ell$
    end
end

A third type of method has recently been proposed, the fast sweeping (FS) method; see among others [4, 8, 16, 19]. FS aims at improving on the first Gauss–Jacobi method by using a Gauss–Seidel-type update process. This change alone would not improve on the performance of Gauss–Jacobi very significantly because of the upwind condition (see (4.1) below): an ordering process is needed. Instead of proceeding in a way consistent with the underlying wave propagation, FS considers sweeps in predetermined directions. The directions of sweep can correspond, for instance, to the direction of the coordinate axes. For two-dimensional problems, four sweeping directions can be considered. More precisely, in an $M \times M$ matrix, we would have $i = 1 : M, j = 1 : M$ (upper left to lower right, ULLR); $j = 1 : M, i = M : 1$ (lower left to upper right, LLUR); $i = M : 1, j = M : 1$ (lower right to upper left, LRUL); $j = M : 1, i = 1 : M$ (upper right to lower left, URLL).

Fast Sweeping algorithm.

- Choose a direction of sweep and a corresponding ordering.
- Loop through of all the nodes in the chosen order and successively solve (2.6) for each of them.
- Repeat for the other directions of sweep.
- Repeat until convergence.

The corresponding pseudocode is

$\mathtt{k} = 0$
$\mathtt{U}^0 = (\infty, \ldots, \infty)$
$\mathtt{order} = [\mathtt{ULLR}, \mathtt{LLUR}, \mathtt{LRUL}.\mathtt{URLL}]$
while $\mathtt{U}^\mathtt{k} \neq \mathtt{U}^{\mathtt{k}+1}$ or $\mathtt{k} = 0$
    $\mathtt{k} = \mathtt{k} + 1$
    $\mathtt{i} = \mathtt{mod}(\mathtt{k} - 1, 4) + 1$
    $\mathtt{ordering} = \mathtt{order}[\mathtt{i}]$

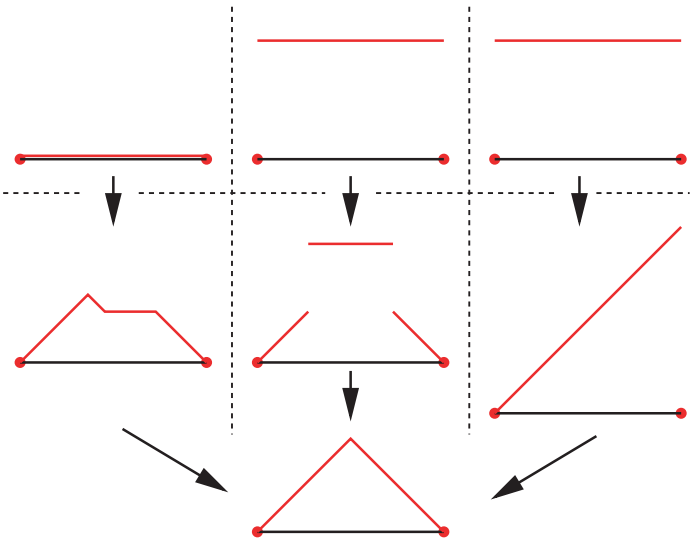


FIG. 1. *Schematic representation of the update process for Gauss–Jacobi (left), FM (center), and FS (right) for a one-dimensional eikonal equation with homogenous boundary conditions.*

```
        for  ℓ = 1  to  N
             solve f(U_1^k, ..., U_{ℓ-1}^k, U_ℓ^k, U_{ℓ+1}^{k-1}, ..., U_N^{k-1}) = 0  for  U_ℓ^k
        end
    end
```

The update process for the above three methods is schematically represented in Figure 1. Also, as a result of the absence of a true ordering process (only the way arrays are accessed changes), the complexity of the above algorithm can be shown to be of order N [19], as opposed to $N \log_2 N$ for FM. However, whether this translates into a computational advantage depends on the geometry of the problem; see section 5.

Unlike what is proposed in [16, 19], we consider FS as a direct method for two reasons. First, as shown in section 4, the algorithm has to be run to its completion. Second, the following result gives a natural stopping criterion.

LEMMA 3.1. *Let U^k, $k = 0, \ldots, \infty$, be the iterates generated by the FS method applied to* (1.1), (1.2). *Then there exists $\bar{k}$ such that $U^k = U^{\bar{k}}$ for all $k \geq \bar{k}$.*

Proof. Consider an arbitrary node i at which the solution has been computed. There exists a path made of successive upwind neighbors from i to Γ. By construction, each successive sweep determines the final value of at least one node on that path, starting on Γ and moving successively to i. This clearly leads to a finite algorithm. The numerical solution at node i depends only on the values at its neighbors. If those values do not change at step $\bar{k}+1$, then $U_i^k = U_i^{\bar{k}}$ for $k \geq \bar{k}$. □

Essentially, the three methods differ in the order in which the nodes are computed. Gauss–Jacobi takes no particular order, FM uses an ordering based on partial numerical solutions, while FS uses a mesh-based ordering. As described above, FS makes full use of the Cartesian structure of the mesh.

4. Problems with obstacles and implementation. An obstacle is a domain in which waves propagate at infinitely slow speed: $F(x, y) = \infty$ if and only if (x, y) belongs to the obstacle. Problems with obstacles require additional care if both accuracy and efficiency, i.e., marching character, of the methods are to be preserved.

For future reference, we now introduce three specific test problems used in this paper. In all three cases, we consider (1.1), (1.2) with $\Omega = (0,6)^2$, $\Gamma = (0,0)$.

Example 1. $F(x,y) = C$ for all $(x,y) \in \Omega$, no obstacle.[1]

Example 2.

$$F(x,y) = \begin{cases} \infty & \text{if } (x-3)^2 + (y-3)^2 <= 1, \\ C & \text{otherwise,} \end{cases}$$

circular obstacle.

Example 3.

$$F(x,y) = \begin{cases} \infty & \text{if } (x,y) \in \Xi, \\ C & \text{otherwise,} \end{cases}$$

where Ξ is the "rings" of Figure 7.

If no local adaption is made when treating problems with obstacles such as Examples 2 and 3, accuracy is lost. Taking, for instance, Example 2, a straightforward discretization would consist of applying any of the above three methods *outside* the circular obstacle using a simple Cartesian mesh. The left half of Table 1 illustrates the loss accuracy incurred by the first and second order FM methods, respectively, i.e., using (2.3) and (2.4).

TABLE 1

Convergence study for formally first and second order methods in the presence of a circular obstacle (Example 2); M measures the number of nodes on one edge of Ω, i.e., total number of nodes $N = \mathcal{O}(M^2)$.

First order methods												
	Standard method						Modified method					
M	L^1	Rate	L^2	Rate	L^∞	Rate	L^1	Rate	L^2	Rate	L^∞	Rate
25	2.91(0)		9.00(-1)		6.11(-1)		1.13(0)		3.04(-1)		1.44(-1)	
50	1.71(0)	.77	5.24(-1)	.78	3.52(-1)	.79	7.58(-1)	.58	2.00(-1)	.60	9.18(-2)	.65
100	1.07(0)	.68	3.27(-1)	.68	2.23(-1)	.66	4.41(-1)	.78	1.13(-1)	.82	4.92(-2)	.88
200	6.42(-1)	.74	2.00(-1)	.71	1.46(-1)	.61	2.39(-1)	.88	6.10(-2)	.89	2.65(-2)	.91
400	3.76(-1)	.77	1.20(-1)	.74	9.36(-2)	.64	1.27(-1)	.92	3.24(-2)	.91	2.02(-2)	.39
800	2.21(-1)	.77	7.24(-2)	.73	6.07(-2)	.62	6.38(-2)	.99	1.64(-2)	.99	1.54(-2)	.39
1600	1.31(-1)	.76	4.40(-2)	.72	3.82(-2)	.67	3.07(-2)	1.1	7.91(-3)	1.0	1.14(-2)	.43
3200	7.80(-2)	.75	2.68(-2)	.72	2.44(-2)	.65	1.49(-2)	1.0	3.88(-3)	1.0	6.77(-3)	.75
Second order methods												
	Standard method						Modified method					
M	L^1	Rate	L^2	Rate	L^∞	Rate	L^1	Rate	L^2	Rate	L^∞	Rate
25	2.52(0)		1.07(0)		8.99(-1)		1.17(0)		2.84(-1)		1.16(-1)	
50	1.45(0)	.80	5.94(-1)	.85	7.39(-1)	.28	2.83(-1)	2.0	9.59(-2)	1.6	4.54(-2)	1.4
100	6.36(-1)	1.2	2.45(-1)	1.3	3.10(-1)	1.3	1.23(-1)	1.2	4.26(-2)	1.2	1.97(-2)	1.2
200	3.02(-1)	1.1	1.20(-1)	1.0	1.73(-1)	.84	3.29(-2)	1.9	1.14(-2)	1.9	1.22(-2)	.70
400	1.52(-1)	.99	6.16(-2)	.97	8.49(-2)	1.0	1.52(-2)	1.1	5.32(-3)	1.1	5.93(-3)	1.0
800	6.91(-2)	1.1	2.91(-2)	1.1	4.41(-2)	.95	2.06(-3)	2.9	8.22(-4)	2.7	3.11(-3)	.94
1600	3.42(-2)	1.0	1.45(-2)	1.0	2.31(-2)	.93	4.82(-4)	2.1	2.47(-4)	1.7	1.68(-3)	.89
3200	1.48(-2)	1.2	6.36(-3)	1.2	1.12(-2)	1.0	1.15(-4)	2.1	7.64(-5)	1.7	8.83(-4)	.93

As can be seen from those results, the standard first order method converges with an average order of only about .75 (in the L^1 norm) while the standard formally second order method loses a full order (the average rate in the L^1 norm is about 1.0).

[1] In all the calculations that follow, C was chosen as $\tan 40^o$; see [2] for a related application to granular piles. This choice does not affect the numerical results.

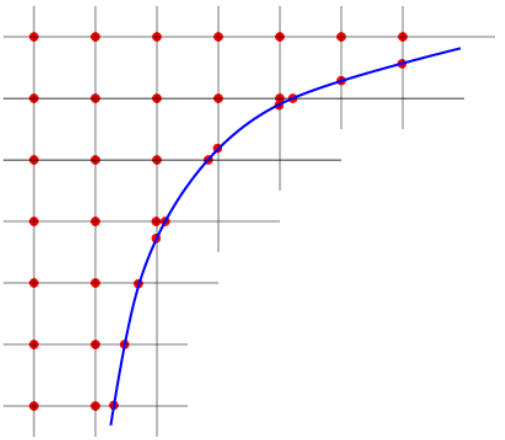


FIG. 2. *Mesh structure around the obstacle.*

The results in the L^2 norm are comparable, while the L^∞ rates are significantly lower. This is not surprising since the underlying solution presents a "shock" at the back of the obstacle (discontinuity of the first derivatives).

In [2], we started the study of a generalization of the FM method from [14, 15] to problems with obstacles. The nodes are defined on a uniform Cartesian mesh away from the obstacle. Near the obstacle, additional nodes corresponding to the intersection of the obstacle's boundary with the mesh lines are added; see Figure 2. The detailed solving process is as follows for the modified FM algorithm.

Let X_0 be a node to be updated. Let X_1 and X_2 be a pair of primary neighbors of X_0. Following [15], an upwinding criterion is considered: the characteristic direction should point into the simplex defined by X_0, X_1, and X_2. This is equivalent to requiring the approximate gradient defined by (2.5) to have positive components when expressed in terms of the unit vectors $N_1 = \frac{X_0 - X_1}{|X_0 - X_1|}$ and $N_2 = \frac{X_0 - X_2}{|X_0 - X_2|}$. In other words,

$$(\mathcal{N}\mathcal{N}^t)^{-1} DU > 0. \tag{4.1}$$

The value U_0 at X_0 is then determined as follows:

$$U_0 = \min \begin{cases} \text{solution of (2.6) with second order stencil (2.4) if (4.1) is true,} \\ \text{solution of (2.6) with first order stencil (2.3) if (4.1) is true,} \\ \min_{i=1,2}\{U_i + |X_0 - X_i| F(X_0)\}. \end{cases} \tag{4.2}$$

Regardless of the choice of stencil (first or second order), (2.6) has two real solutions, from which we always consider the larger one only, to be consistent with causality. In the first two cases, the values of the partial derivatives of U at X_0 are updated using $\nabla U = \mathcal{N}^{-1} DU$. In the third case, we set $\nabla U = N_i F(X_0)$, where i is the index corresponding to the case of lowest value for U_0. This process is repeated for all pairs of primary neighbors X_1 and X_2, and the lowest resulting value of U_0 is kept.

As checking (4.1) accounts for a sizable part of the runtime, the following result is useful in practice.

LEMMA 4.1. *Let θ, $0 < \theta < \pi$, be the angle between N_1 and N_2. The upwind condition* (4.1) *is equivalent to*

$$| D_1 U - D_2 U | < (1 - \cos\theta)\, F(X_0). \tag{4.3}$$

Proof. The result is easy to verify geometrically. Setting $d_i = D_i U$, $i = 1, 2$, the scheme (2.6) takes the form

$$d_1^2 - 2\cos\theta\, d_1 d_2 + d_2^2 = \sin^2\theta\, F(X_0)^2.$$

This defines an ellipse in the $d_1 d_2$-plane with major (respectively, minor) semiaxis of length $\frac{\sin\theta}{\sqrt{1-\cos\theta}} F(X_0)$ (respectively, $\frac{\sin\theta}{\sqrt{1+\cos\theta}} F(X_0)$) in the direction $[1, 1]$ (respectively, $[1, -1]$). The upwind condition (4.1) reads as

$$d_1 - \cos\theta\, d_2 > 0 \text{ and } d_2 - \cos\theta\, d_1 > 0.$$

As Figure 3 and a little algebra show, the portion of the ellipse that satisfies the above conditions can also be characterized by $|\, d_1 - d_2\,| < (1 - \cos\theta) F(X_0)$. Note that this last condition also admits a set of solutions opposite the desired ones on the ellipse depicted in Figure 3. Those solutions are, however, never encountered in practice since only the largest of the two roots of the quadratic equation is retained. ∎

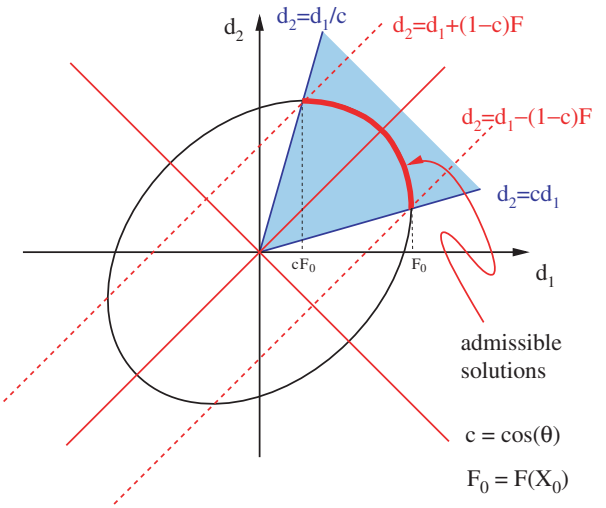


FIG. 3. *Upwind condition; see Lemma* 4.1 *and its proof.*

For general meshes, the above condition requires the calculation of U_0 and has to be checked a posteriori. For a uniform Cartesian mesh of size Δx, (4.3) reads as

$$|\, U_1 - U_2\,| < \Delta x\, F(X_0)$$

and can be checked a priori, i.e., *without* having to solve the quadratic equation (2.6). This explains the large discrepancies in runtimes in favor of the low accuracy method on uniform Cartesian grids reported in Table 2 (see Figure 5 for a comparison of runtime versus accuracy).

As observed in [11], in the presence of a point source, a singularity of the travel time field is clearly located at the source itself. Unless special care is taken near that source, a loss of accuracy ensues. The problem can be fixed by defining a mesh-

TABLE 2

Runtimes illustrating the overhead due to higher accuracy for the FM method applied to Example 2; see text for full details.

M	Time 1	Time 2	Time 3
100	0.02	0.57	0.60
200	0.10	2.20	2.38
400	0.43	8.93	9.45
800	2.02	34.5	38.1
1600	10.1	143	154
3200	53.0	610	654

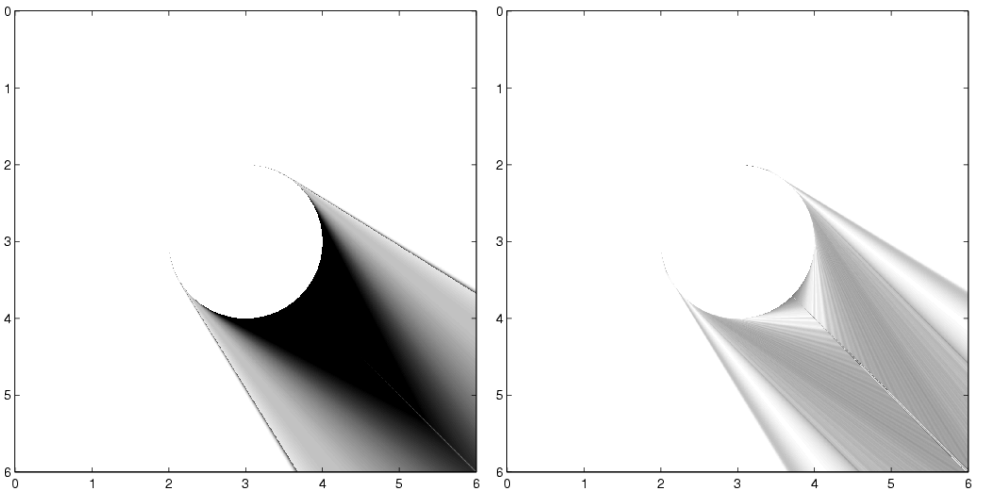


FIG. 4. *Error propagation downstream of the obstacle for the standard (left) and modified (right) second order FM method (Example 2, $M = 1600$, $N = \mathcal{O}(M^2)$). The source is located at $(0, 0)$.*

independent domain around the source and initializing the values at the nodes there to the corresponding values of the exact solution. This fix was applied to all methods under study here. As illustrated in Table 1, the modified scheme is found to preserve first and second order accuracy in the L^1 norm even in the presence of obstacles. For the second order version, the L^2 rates of convergence are slightly below 2, while the method appears to converge with order 1 in the maximum norm. In all cases and all norms, the modified method exhibits much better accuracy than the standard method. Error propagation downstream from the obstacle is illustrated in Figure 4 for both standard and modified methods.

It should be noted that second order convergence for the finer meshes was obtained only after rewriting the quadratic formula for solving (2.6) in a form that limits cancellation effects. (Standard IEC559 double precision was used throughout.)

Higher accuracy does not come for free. In Table 2, the overhead corresponding to maintaining optimal accuracy is described in terms of computational time and number of operations.

Table 2 compares three different algorithms. Time 1 corresponds to the computational time and number of operations for the standard method where full advantage has been taken of the uniform Cartesian mesh in the implementation: simplification

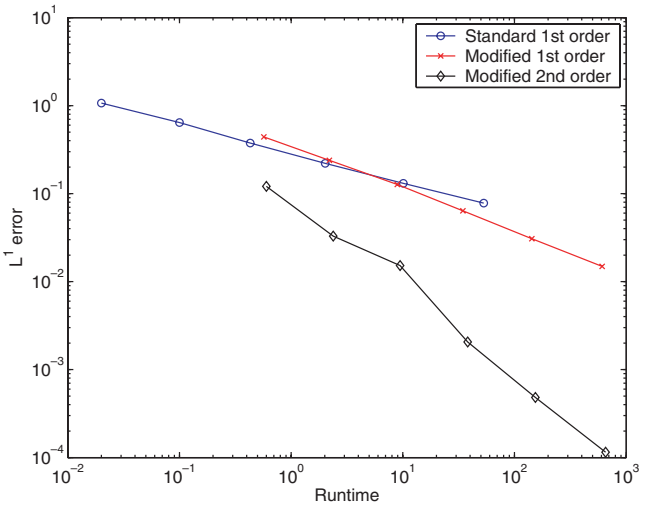


FIG. 5. *L^1 error versus runtime for the standard "first" order method, the modifed first order method, and the modified second order method applied to Example* 2.

of the quadratic equation (2.6) and of the upwinding condition (see remarks above). The results for the fully first and second order methods correspond, respectively, to Time 2 and Time 3. The fully second order method is found to be only marginally slower than the fully first order one. It is clear from Table 2 that a significant price has to be paid for the increase in accuracy. However, Figure 5 shows that the advantage in speed for the standard first order is offset by its relative lack of accuracy and slow convergence as the mesh is refined.

FS as described in [16, 19], for instance, is proposed as an iterative method, i.e., the algorithm is stopped before completion according to a stopping criterion of the type $\|u^{k+1} - u^k\|_{L^1} < \delta = 10^{-10}$. Figure 6 illustrates the convergence history of the numerical solution toward the exact solution of the *discretized* problem for Example 3. The discrepancy falls down to zero if one additional step is taken. It is observed that stopping on small decreases, as suggested by the previous criterion, is not advisable. Therefore, running FS until completion appears to be the best strategy, and the algorithm has to be considered as a "direct" method.

For both FM and FS, obstacles can be implemented as domains with truly infinite speed (in which case, no calculations are made at nodes inside the obstacle), or a sufficiently large but finite slowness can be assigned to nodes inside the obstacle and the solution can be computed there. Our implementation uses the former approach. As can be expected and is illustrated in Figure 6, those choices do not affect the solutions of the discretized problems.

5. Algorithms comparison. The three examples from section 4 were implemented. As the implementation on FS on a non-Cartesian grid was not attempted, first order FM and FS were compared on Cartesian grids. Therefore in Table 3, both methods share the same (in)accuracy. Note that each algorithm was run until "convergence." For FM, this means until exhaustion of the nodes in the "front," while for FS it means until a fixed point has been reached (no change after a sweep; see Lemma 3.1). As can be seen from that table, FS is faster for the first (trivial) example;

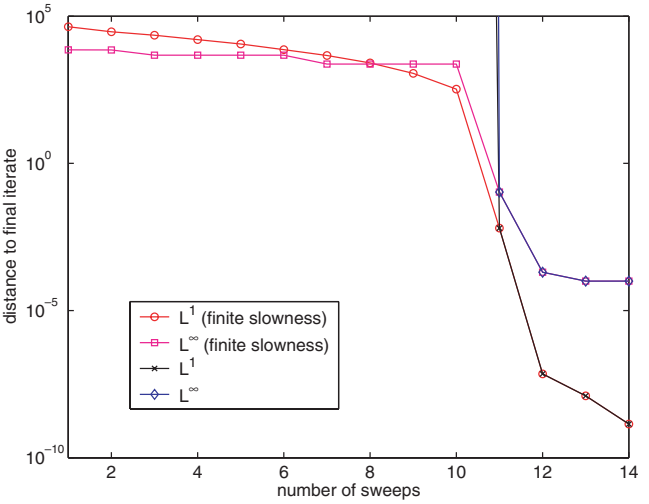


FIG. 6. *L^1 and L^∞ norms of the difference between the exact solution of the* discretized *problem with the FS method and iterates after each sweep; rings (Example* 3*); $M = 1600$. The curves with "finite slowness" are obtained by treating the obstacle as a domain with large slowness; the other curves correspond to a genuine obstacle (slowness $= \infty$).*

both methods perform in a way very similar to Example 2, with a slight advantage for FS, while on Example 3 FM is faster for all but the finest mesh.[2]

The results from Table 3 also show that, for moderate mesh sizes, the more complicated the domain is, the better FM performs with respect to FS. Indeed, while FM continually advances the wavefront, FS has to do another set of sweeps every time the direction of propagation changes. In the case of Example 3, for instance, that direction changes several times; see Figure 7. In Table 4, the runtimes for a modified Example 3 are reported: the number of partial rings was increased from zero to a total of five as displayed in Figure 7. It is observed that the runtime is roughly constant for FM as the complexity of the domain (number of rings) increases. For FS, however, the runtime increases, although in a nonuniform way. This nonuniformity results from both the geometry of the domain (odd/even number of rings) and our specific implementation.

As argued in [19], FS has an asymptotic computational complexity of $\mathcal{O}(N)$ while FM's is $\mathcal{O}(N \log_2 N)$. Eventually, FS overcomes FM, even for nontrivial examples such as Example 3, as the mesh gets fine enough. The constants of proportionality involved in those asymptotic relations are not easy to analyze. For specific examples, those constants can be estimated numerically. Assuming runtimes for FS and FM to be, respectively, equal to $C_{FS} mN$ and $C_{FM} N \log_2 N$, where m is the number of sweeps required by FS, one can use results similar to those from Table 3 to roughly determine the two constants C_{FS} and C_{FM}. An empirical condition under which FS is faster than FM can then be derived. Using best fits to the above examples and cases, one finds here

$$m \lessapprox .6 \log_2 N.$$

[2]In an initial version of this paper (technical report), the reported runtimes had been obtained by codes that were not fully optimized for use on uniform Cartesian meshes. One of those optimization steps removed unnecessary checks from FS; those redundant operations were, by construction, not present in the FM algorithm. Therefore, the two codes were not equally affected. To the authors' knowledge, the current codes are fully optimized for uniform Cartesian grids.

TABLE 3
Runtime for first order FM and FS on a uniform Cartesian mesh for Examples 1, 2, *and* 3 *($N = \mathcal{O}(M^2)$).*

M	FM (time)	FS (time)	# sweeps
Example 1: no obstacle			
100	0.01	0.01	2
200	0.09	0.03	2
400	0.43	0.13	2
800	2.03	0.54	2
1600	10.1	2.16	2
3200	53.6	8.59	2
Example 2: circular obstacle			
100	0.02	0.01	5
200	0.10	0.09	6
400	0.43	0.37	6
800	2.02	1.45	6
1600	10.1	5.91	6
3200	53.0	23.5	6
Example 3: "rings"			
100	0.02	0.05	16
200	0.10	0.22	16
400	0.48	0.91	16
800	2.28	3.44	16
1600	11.4	14.3	16
3200	57.5	56.8	16

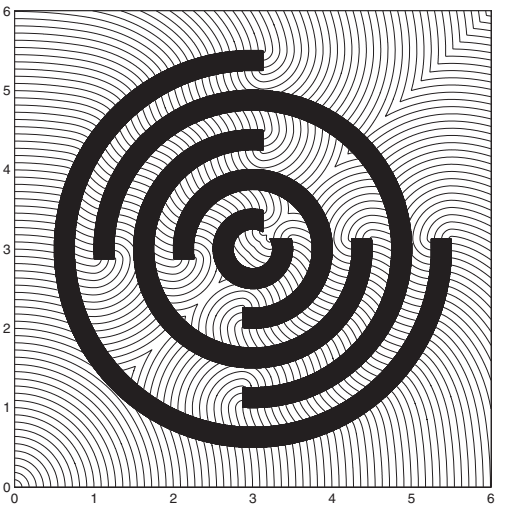


FIG. 7. *Propagation around an obstacle: Example* 3*, FM with $M = 800$. The source is located at $(0, 0)$.*

In other words, for a "reasonable" 800×800 mesh ($N = 800^2$), FS will be faster than FM as long as the number of sweeps necessary for convergence does not exceed about 11. This is in good agreement with Table 4.

6. Conclusion. We have brought to the fore three main points. First, ordered methods for the eikonal equation can be modified to retain optimal convergence rates (first or second order), even on low regularity problems, while retaining their efficiency. This can be done in a natural way for FM, but it is more cumbersome for FS, which is

TABLE 4
Runtime and operation count with differing numbers of concentric partial rings ($M = 800$).

Rings	FM (time)	FS (time)	# sweeps
0	2.03	0.54	2
1	2.05	2.15	8
2	2.12	2.09	8
3	2.12	3.02	12
4	2.24	2.89	12
5	2.28	3.44	16

much less flexible. Second, the examples considered show that both methods should be thought of as direct and not iterative algorithms. This is obvious for FM. For FS, a natural stopping criterion is proposed and justified. Finally, FS and FM have asymptotic orders of complexity of, respectively, $\mathcal{O}(mN)$ and $\mathcal{O}(N \log_2 N)$, where m is the number of sweeps and N is the total number of nodes. On a realistic grid, FS will be faster than FM for problems with simple geometry. For strongly nonuniform problems and/or complex geometry, the situation may be reversed.

Both methods can be relatively easily extended to more general convex Hamiltonians [15, 16]. Behaviors similar to those reported here are expected. The case of nonconvex Hamiltonians is more complex, even with respect to the construction of appropriate numerical fluxes. Very few schemes have been proposed (see, for instance, [8], where a Lax–Friedrichs sweeping method is introduced); more needs to be done to develop fast and accurate methods for those problems.

Acknowledgments. The authors are grateful to Hongkai Zhao for his helpful comments and to the referees for remarks that led to a better presentation of the results.

REFERENCES

[1] R. ABGRALL, *Numerical discretization of the first order Hamilton-Jacobi equations on triangular meshes*, Comm. Pure Appl. Math., 49 (1996), pp. 1339–1377.
[2] S.A. AHMED, R. BUCKINGHAM, P.A. GREMAUD, C.D. HAUCK, C.M. KUSTER, M. PRODANOVIC, T.A. ROYAL, AND V. SILANTYEV, *Volume determination for bulk materials in bunkers*, Internat. J. Numer. Methods Engrg., 61 (2004), pp. 2239–2249.
[3] J. BENTLEY, *Thanks, heaps*, Comm. ACM, 28 (1985), pp. 245–250.
[4] M. BOUÉ AND P. DUPUIS, *Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control*, SIAM J. Numer. Anal., 36 (1999), pp. 667–695.
[5] M. CRANDALL, L. EVANS, AND P.-L. LIONS, *Some properties of viscosity solutions of Hamilton-Jacobi equations*, Trans. Amer. Math. Soc., 282 (1984), pp, 487–502.
[6] M. CRANDALL AND P.-L. LIONS, *Two approximations of solutions of Hamilton-Jacobi equations*, Math. Comp., 43 (1984), pp. 1–19.
[7] C. HU AND C.-W. SHU, *A discontinuous Galerkin method for Hamilton–Jacobi equations*, SIAM J. Sci. Comput., 21 (1999), pp. 666–690.
[8] C.-Y. KAO, S. OSHER, AND J. QIAN, *Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations*, J. Comput. Phys., 196 (2004), pp. 367–391.
[9] G. KOSSIORIS, C. MAKRIDAKIS, AND P. SOUGANIDIS, *Finite volume schemes for Hamilton-Jacobi equations*, Numer. Math., 83 (1999), pp. 427–442.
[10] S. OSHER AND C.-W. SHU, *High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations*, SIAM J. Numer. Anal., 28 (1991), pp. 907–922.
[11] J. QIAN AND W.W. SYMES, *An adaptive finite difference method for traveltimes and amplitudes*, Geophysics, 67 (2002), pp. 167–176.
[12] E. ROUY AND A. TOURIN, *A viscosity solutions approach to shape-from-shading*, SIAM J. Numer. Anal., 29 (1992), pp. 867–884.

[13] J.A. Sethian, *Fast marching methods*, SIAM Rev., 41 (1999), pp. 199–235.
[14] J.A. Sethian and A. Vladimirsky, *Fast methods for the Eikonal and related Hamilton-Jacobi equations on unstructured meshes*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 5699–5703.
[15] J.A. Sethian and A. Vladimirsky, *Ordered upwind methods for static Hamilton–Jacobi equations: Theory and algorithms*, SIAM J. Numer. Anal., 41 (2003), pp. 325–363.
[16] Y.H.R. Tsai, L.T. Cheng, S. Osher, and H.K. Zhao, *Fast sweeping algorithms for a class of Hamilton–Jacobi problems*, SIAM J. Numer. Anal., 41 (2003), pp. 673–694.
[17] J.N. Tsitsiklis, *Efficient algorithms for globally optimal trajectories*, IEEE Trans. Automat. Control, 40 (1995), pp. 1528–1538.
[18] J.W.J Williams, *Heapsort*, Comm. ACM, 7 (1964), pp. 347–348.
[19] H.K. Zhao, *Fast sweeping method for eikonal equations*, Math. Comp., 74 (2005), pp. 603–627.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1817–1843
© 2006 Society for Industrial and Applied Mathematics

COMPOSITION METHODS FOR DIFFERENTIAL EQUATIONS WITH PROCESSING*

S. BLANES†, F. CASAS‡, AND A. MURUA§

Abstract. We construct numerical integrators for differential equations up to order 12 obtained by composition of basic integrators. The following cases are considered: (i) composition for a system separable in two solvable parts, (ii) composition using as basic methods a first-order integrator and its adjoint, (iii) composition using second-order symmetric methods, and (iv) composition using fourth-order symmetric methods. Each scheme is implemented with a processor or corrector to improve their efficiency, and this can be done virtually cost-free.

Key words. composition methods, geometric integrators, processing technique, cheap postprocessor, initial value problems

AMS subject classifications. 65L05, 65L70, 65D30

DOI. 10.1137/030601223

1. Introduction.

A widely used technique in the geometric integration of ordinary differential equations (ODEs) is to compose one or more low-order basic one-step methods (usually first or second order) with appropriately chosen weights to achieve a higher-order scheme. The resulting composition method then inherits the relevant geometric properties that the basic scheme shares with the exact solution (if these properties are preserved by composition). In close connection with composition is the splitting idea: if the differential equation can be split into two or more parts that are either solvable or simpler to integrate than the original system, it is also possible to build integrators by composition of the corresponding flows. These splitting methods are frequently used in celestial mechanics, quantum mechanics, molecular dynamics, and accelerator physics and, in general, for solving numerically Hamiltonian systems, as well as Poisson systems and reversible differential equations [8, 16, 21].

It has been widely recognized that the class of numerical integrators which preserve the geometric properties of the exact flow (the so-called *geometric integrators*) provide a better description of the system than standard methods, even with respect to the accumulation of numerical errors along the evolution, making them the best option for carrying out a long time integration. However, for short time integrations they are typically less efficient. For this reason it is important to build more efficient high-order geometric integrators which are also competitive in this setting.

*Received by the editors October 16, 2003; accepted for publication (in revised form) February 18, 2005; published electronically February 3, 2006. The work of the first and second authors was partially supported by Fundació Caixa Castelló–Bancaixa through project P1.1B2002-22 and by Generalitat Valenciana through project GRUPOS03-002.

http://www.siam.org/journals/sisc/27-6/60122.html

†Instituto de Matemática Multidisciplinar, Universidad Politécnica de Valencia, 46022 Valencia, Spain (serblaza@imm.upv.es). This author's work was supported by Ministerio de Ciencia y Tecnología (Spain) through a contract in the Programme Ramón y Cajal 2001.

‡Departament de Matemàtiques, Universitat Jaume I, 12071 Castellón, Spain (Fernando.Casas@uji.es).

§Konputazio Zientziak eta A. A. saila, Informatika Fakultatea, EHU/UPV, Donostia/San Sebastián, Spain (ander@si.ehu.es). This author's work was supported by Ministerio de Educación y Ciencia under project MTM2004-00535, cofinanced by the European Regional Development Fund (ERDF).

Given the ODE

$$x' = f(x), \qquad x_0 = x(0) \in \mathbb{R}^D, \tag{1.1}$$

with $f : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ and associated vector field (or Lie operator associated with f)

$$F = \sum_{i=1}^{D} f_i(x) \frac{\partial}{\partial x_i}, \tag{1.2}$$

a one-step numerical integrator for a time step h, $\psi_h : \mathbb{R}^D \longrightarrow \mathbb{R}^D$, can be regarded as a smooth family of maps with parameter h such that ψ_0 is the identity map. The integrator ψ_h is said to be of order q if $\psi_h = \varphi_h + \mathcal{O}(h^{q+1})$ as $h \to 0$, where φ_h is the h-flow of the ODE (1.1). In other words, an approximation to the exact solution $x(h)$ is given by $x_h = \psi_h(x_0) = \varphi_h(x_0) + \mathcal{O}(h^{q+1})$.

In this work we consider methods obtained by enhancing an integrator ψ_h with processing, i.e., algorithms of the form

$$\hat{\psi}_h = \pi_h \circ \psi_h \circ \pi_h^{-1}. \tag{1.3}$$

Here ψ_h is referred to as the *kernel* and the map $\pi_h : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ as the *postprocessor* or *corrector*. The method ψ_h is of *effective order* q if a postprocessor π_h exists for which $\hat{\psi}_h$ is of (conventional) order q [6]. Application of $\hat{\psi}_h$ over n steps with constant step size h leads to

$$\hat{\psi}_h^n = \pi_h \circ \psi_h^n \circ \pi_h^{-1}.$$

The computation of the preprocessor π_h^{-1} has to be done only at the beginning of the integration; then the kernel ψ_h acts once per step, and finally π_h is evaluated when output is required.

The analysis of the order conditions of the method $\hat{\psi}_h$ has shown that many of them can be satisfied by π_h, so that ψ_h must fulfill a much reduced set of restrictions [1, 3, 7, 14]. This allows us to take kernels of effective order q involving far fewer function evaluations than a conventional integrator of order q.

In the course of constructing composition methods of a given order, the processing technique also presents additional advantages. In general, it is necessary to solve numerically a system of nonlinear polynomial equations in the coefficients (the order conditions), whose number and complexity grow very rapidly with the order. Numerically finding every solution and eventually determining the optimal solution turns into a very complicated task for nonprocessed methods, even for relatively small orders. However, this problem simplifies for processed composition schemes because the analysis has to be carried out only for the kernel. Since the number of order conditions for the kernel is smaller, one can analyze in more detail the set of solutions, even for moderate orders, and eventually find very efficient methods [25, 3, 1, 15].

Once a kernel is chosen, it remains for us to build a postprocessor such that the whole method has the required order. We can also impose additional conditions to reduce the leading order error terms. Then the coefficients of the postprocessor π_h usually have to satisfy a large number of conditions. Many different solutions can be found, and since usually their different contributions affect only higher-order terms, we can choose one with moderately small coefficients (to avoid large higher-order error terms), which considerably simplifies the numerical search.

In any case, very often the postprocessors thus constructed are more expensive than the kernel ψ_h, so that the use of processed schemes in this form has to be confined to problems in which intermediate results are not frequently required; otherwise the overall efficiency of the methods is decreased. To overcome this difficulty, a technique is developed in [2] which allows us to obtain approximations to the postprocessor virtually cost-free and without loss of accuracy. The key point is to replace π_h by a new integrator $\hat{\pi}_h \simeq \pi_h$ obtained from the intermediate stages in the computation of ψ_h. It is recommended that one have a very accurate preprocessor π_h^{-1} but, on the other hand, the postprocessor can safely be replaced by a cheap approximation satisfying only the order conditions.

In the analysis of composition methods it is useful to apply the notion of adjoint of a given integrator ψ_h [21]. By definition, this is the method ψ_h^* such that $\psi_h^* = \psi_{-h}^{-1}$. A method that is its own adjoint is called self-adjoint or (time-)*symmetric*. In this case, $\psi_{-h} \circ \psi_h = \mathrm{id}$. The order of a symmetric method is necessarily even. In addition, only even powers of h appear in the asymptotic expansion of the global error [21] so that the order equations are considerably simplified and its number is reduced. Time-symmetry is often a desirable property when one aims to discretize the differential equation in such a way that qualitative features of the system are preserved [8, 16]. In the context of processed methods, if the kernel ψ_h is symmetric and the postprocessor π_h is an even function of h ($\pi_{-h} = \pi_h$), then the processed integrator $\hat{\psi}_h$ has also the time-symmetry property.

In this paper we construct and analyze new high-order processed geometric integrators. We consider several families of kernels which are symmetric compositions of different basic methods of first, second, and fourth order and whose postprocessors can be implemented essentially at no additional computational cost. The procedure is largely based on the theoretical analysis of the processing technique done in [2], to which we will refer often in what follows.

2. Families of composition methods.

2.1. Classes of methods. Next we present the different classes of schemes analyzed, whose common feature is that they are defined as the composition of simpler maps.

1. Let us assume that the ODE (1.1) can be written as $x' = f_a(x) + f_b(x)$ and that the vector field F is split accordingly as $F = F_a + F_b$. Suppose that the corresponding h-flows $\varphi_h^{[a]}$ and $\varphi_h^{[b]}$ can be computed exactly. Then one can form the following group $\mathcal{G}_1$ of schemes:

   (2.1)
   $$\mathcal{G}_1 = \left\{ \varphi^{[b]}_{\beta_{2s}h} \circ \varphi^{[a]}_{\beta_{2s-1}h} \circ \cdots \circ \varphi^{[b]}_{\beta_2 h} \circ \varphi^{[a]}_{\beta_1 h} : \ s \geq 1, \ (\beta_1, \ldots, \beta_{2s}) \in \mathbb{R}^{2s} \right\}.$$

2. Let $\chi_h : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ be any first-order integration scheme. An important class $\mathcal{G}_2$ of integrators is obtained by considering compositions of χ_h and its adjoint χ_h^*, namely,

   (2.2)
   $$\mathcal{G}_2 = \left\{ \chi_{\alpha_{2s}h} \circ \chi^*_{\alpha_{2s-1}h} \circ \cdots \circ \chi_{\alpha_2 h} \circ \chi^*_{\alpha_1 h} : \ s \geq 1, \ (\alpha_1, \ldots, \alpha_{2s}) \in \mathbb{R}^{2s} \right\}.$$

   This family $\mathcal{G}_2$ of integrators clearly has (with the composition) a group structure. As shown in [13], if $\sum_{i=1}^{s} \beta_{2i-1} = \sum_{i=1}^{s} \beta_{2i}$, $\mathcal{G}_2$ is closely related to $\mathcal{G}_1$

when $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]}$, $\beta_i = \alpha_i + \alpha_{i-1}$ $(i = 1, \dots, 2s)$, and $\alpha_0 = \alpha_{2s} = 0$. Consistency of the schemes requires that $\sum_{i=1}^{s} \beta_{2i-1} = \sum_{i=1}^{s} \beta_{2i} = \sum_{i=1}^{2s} \alpha_i = 1$, and the order conditions at higher orders are equivalent in both cases [13]. Methods in $\mathcal{G}_2$ are more general and can also be used if the system is split into more than two parts, i.e., $x' = f_a + f_b + f_c + \cdots$, just taking $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]} \circ \varphi_h^{[c]} \circ \cdots$ $(\chi_h^* = \cdots \circ \varphi_h^{[c]} \circ \varphi_h^{[b]} \circ \varphi_h^{[a]})$.

3. Another well-known class $\mathcal{G}_3$ of integrators [13] is introduced as follows. If $\mathcal{S}_h^{[2]} : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ is any second-order self-adjoint integrator, then

$$\mathcal{G}_3 = \left\{ \mathcal{S}^{[2]}_{\alpha_s h} \circ \cdots \circ \mathcal{S}^{[2]}_{\alpha_1 h} : \ s \geq 1, \ (\alpha_1, \dots, \alpha_s) \in \mathbb{R}^s \right\}. \tag{2.3}$$

This family of integrators also has group structure. In fact, if $\mathcal{S}_h^{[2]}$ is chosen as $\mathcal{S}_h^{[2]} = \chi_{h/2} \circ \chi_{h/2}^*$, then $\mathcal{G}_3$ is a subgroup of $\mathcal{G}_2$ (in [18], it is shown that any self-adjoint integrator can be written in this form; the choice of χ_h is not unique, however).

4. Finally, if $\mathcal{S}_h^{[4]} : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ is any fourth-order self-adjoint integrator, then another class $\mathcal{G}_4$ of integrators is

$$\mathcal{G}_4 = \left\{ \mathcal{S}^{[4]}_{\alpha_s h} \circ \cdots \circ \mathcal{S}^{[4]}_{\alpha_1 h} : \ s \geq 1, \ (\alpha_1, \dots, \alpha_s) \in \mathbb{R}^s \right\}. \tag{2.4}$$

Methods in $\mathcal{G}_4$ could be compositions of schemes in $\mathcal{G}_l$, $l = 1, 2, 3$. They can be useful in some problems where, due to their particular structure, it is possible to build a very efficient integrator $\mathcal{S}_h^{[4]}$, as is the case for the second-order differential equation $x'' = g(x)$ [11].

Composition integrators $\psi_h \in \mathcal{G}_l$ $(l = 1, 2, 3, 4)$ are time-symmetric whenever they have left-right palindromic sequences of coefficients α_i—more specifically, if the following apply:

1. In $\mathcal{G}_1$, if either $\beta_{2s} = 0$, $\beta_{2s-i} = \beta_i$ or $\beta_1 = 0$, $\beta_{2s+1-i} = \beta_{i+1}$. Then one of the maps $\varphi^{[a]}_{\beta_i h}$ and $\varphi^{[b]}_{\beta_i h}$ is computed s times and the other $s-1$ times. Now the computations done for the last map can be reused for the first map in the following step. (This feature is known as the FSAL (first same as last) property.) Then both maps $\varphi^{[a]}_{\beta_i h}$ and $\varphi^{[b]}_{\beta_i h}$ are computed $s-1$ times.
2. In $\mathcal{G}_l$, $l = 2, 3, 4$, if $\alpha_{\bar{s}+1-i} = \alpha_i$ with $\bar{s} = 2s$ or $\bar{s} = s$.

2.2. Associated graded Lie algebra. It is well known that for each infinitely differentiable map $g : \mathbb{R}^D \longrightarrow \mathbb{R}$, $g(\varphi_h(x))$ admits an expansion of the form [19]

$$g(\varphi_h(x)) = e^{hF}[g](x) = g(x) + \sum_{k \geq 1} \frac{h^k}{k!} F^k[g](x), \qquad x \in \mathbb{R}^D,$$

where F is the vector field (1.2). Similarly, for the maps $\hat{\psi}_h$, ψ_h, and π_h one has

$$g(\hat{\psi}_h(x)) = e^{\hat{F}_h}[g](x), \qquad g(\psi_h(x)) = e^{F_h}[g](x), \qquad g(\pi_h(x)) = e^{P_h}[g](x),$$

with

$$\hat{F}_h = \sum_{k \geq 1} h^k \hat{F}_k, \qquad F_h = \sum_{k \geq 1} h^k F_k, \qquad P_h = \sum_{k \geq 1} h^k P_k;$$

i.e., $\hat{\psi}_h$, ψ_h, and π_h are formally the exact 1-flows of the vector fields $\hat{F}_h$, F_h, and P_h, respectively.

If the kernel ψ_h belongs to $\mathcal{G}_l$ ($l = 1, 2, 3, 4$), then the corresponding (h-dependent) vector fields $h^k F_k$ belong to a certain graded Lie algebra $\mathcal{L} = \bigoplus_{k\geq 1} \mathcal{L}_k$ such that $hF \in \mathcal{L}_1$, $h^k F_k \in \mathcal{L}_k$ for each $k \geq 1$ and $[\mathcal{L}_k, \mathcal{L}_m] \subset \mathcal{L}_{k+m}$. Each $\mathcal{L}_k$ ($k \geq 1$) is the subspace $\mathcal{L}$ of vector fields affected by a kth power of h, and we denote $n_k = \dim \mathcal{L}_k$. We give $\mathcal{L}$ explicitly in each case.

1. If $\psi_h \in \mathcal{G}_1$, then $\mathcal{L} = \bigoplus_{k\geq 1} \mathcal{L}_k$ is the graded Lie algebra generated by the vector fields $\{hF_a, hF_b\}$. Here, $hF_1 + h^2 F_2 + \cdots$ can be obtained, for instance, by repeated application of the Baker–Campbell–Hausdorff (BCH) formula [24] to the expression

$$\Psi_h = \exp(h\beta_1 F_a) \exp(h\beta_2 F_b) \cdots \exp(h\beta_{2s-1} F_a) \exp(h\beta_{2s} F_b)$$

since, formally, $g \circ \psi_h = \Psi_h[g]$ and $F_h = \log(\Psi_h)$ [2, 8].

2. For $\mathcal{G}_2$, we have for the basic method $g(\chi_h(x)) = e^{Y_h}[g](x)$ with $Y_h = \sum_{k\geq 1} h^k Y_k$ and for its adjoint $g(\chi_h^*(x)) = e^{-Y_{-h}}[g](x)$. Then

$$\Psi_h = \exp(-Y_{-h\alpha_1}) \exp(Y_{h\alpha_2}) \cdots \exp(-Y_{-h\alpha_{2s-1}}) \exp(Y_{h\alpha_{2s}}),$$

and F_h can be obtained by repeated application of the BCH formula, which shows that $h^k F_k \in \mathcal{L}_k$ for each $k \geq 1$ and $\mathcal{L} = \bigoplus_{k\geq 1} \mathcal{L}_k$ is the graded Lie algebra generated by the vector fields $\{hY_1, h^2 Y_2, h^3 Y_3, \dots\}$ where, by consistency, $Y_1 = F$.

3. For $\mathcal{G}_3$, the series of differential operators S_h associated with the integrator $\mathcal{S}_h^{[2]}$, i.e., such that $g \circ \mathcal{S}_h^{[2]} = S_h[g]$, can be written as $S_h = \exp(Y_h)$, where $Y_h = hY_1 + h^3 Y_3 + h^5 Y_5 + \cdots$, $Y_1 = F$, and

$$\Psi_h = \exp(Y_{h\alpha_1}) \cdots \exp(Y_{h\alpha_s}).$$

One similarly [13] arrives at an expansion $hF_1 + h^3 F_3 + h^4 F_4 + \cdots$, where $h^k F_k \in \mathcal{L}_k$ for the graded Lie algebra $\mathcal{L} = \bigoplus_{k\geq 1} \mathcal{L}_k$ generated by the vector fields $\{hY_1, h^3 Y_3, h^5 Y_5, \dots\}$.

4. The case of $\mathcal{G}_4$ can be considered as a particular case of $\mathcal{G}_3$ with $Y_3 = 0$.

2.3. Effective order conditions. According to the theory developed in [2] the following theorem applies to the processed methods considered here.

THEOREM 1 (see [2]). *An integrator ψ_h has effective order $p \geq q$ if and only if there exist vector fields $P_1, \dots, P_{q-1}$ such that*

$$F_1 = F, \quad [P_{n-1}, F] = F_n + R_n(P_1, \dots, P_{n-2}, F_1, \dots, F_{n-1}), \quad 1 < n \leq q, \tag{2.5}$$

hold, where

$$R_n = -\sum_{j=1}^{n-2} [P_j, F_{n-j}] + \sum_{k\geq 2} \frac{(-1)^k}{k!} \sum_{j_1+\cdots+j_{k+1}=n} [P_{j_1}, [P_{j_2}, \dots [P_{j_k}, F_{j_{k+1}}] \dots]].$$

Under the premises of Theorem 1, $[P_k, F] \in \mathcal{L}_{k+1}$. As shown in [2], if such vector fields P_k ($1 \leq k \leq q-1$) exist, then they are not unique. We will hereafter consider vector fields P_k belonging to $\mathcal{L}_k$. Uniqueness is then achieved by requiring that $P_k \in \mathcal{L}_k^*$, where $\mathcal{L}_k^*$ is a subspace of $\mathcal{L}_k$ of minimal dimension, n_k^*, among all

subspaces V_k of $\mathcal{L}_k$ such that $[F, V_k] = [F, \mathcal{L}_k]$. It must be stressed that such a requirement does not restrict the choice of the processed integrator [2]. Since the graded Lie algebras associated with $\mathcal{G}_l$ ($l = 1, 2, 3, 4$) are free, then $\mathcal{L}_1 = \mathcal{L}_1^* \oplus F$, $\mathcal{L}_k^* = \mathcal{L}_k$ for $k > 1$, and $P_1 \in \mathcal{L}_1^*$. Therefore, $n_1^* = n_1 - 1$ and $n_k^* = n_k$ for $k > 1$.

Let us denote by $\{E_{k,i}\}_{i=1}^{n_k}$ a basis of $\mathcal{L}_k$ with $E_{1,1} = F$. Then one can write

$$F_k = \sum_{i=1}^{n_k} f_{k,i} E_{k,i}, \qquad R_k = \sum_{i=1}^{n_k} r_{k,i} E_{k,i}, \qquad P_{k-1} = \sum_{i=1}^{n_{k-1}^*} p_{k-1,i} E_{k-1,i}, \tag{2.6}$$

where $P_0 = 0$, $P_1 = \sum_{i=1}^{n_1} p_{1,i} E_{1,i}$ with $p_{1,1} = 0$, and the coefficients $r_{k,i}$ can be written as polynomials in the indeterminates $f_{l,j}, p_{l-1,j}$, $l < k$.

The order conditions (2.5) can be expressed as a system of (polynomial) equations in $f_{k,i}, p_{k-1,i}$. Such equations have a particularly simple form if the basis of each $\mathcal{L}_{k+1}$ is constructed (from a given basis of $\mathcal{L}_k$) in such a way that

$$E_{k+1,n_{k+1}-n_k^*+i} = [F, E_{k,i}], \qquad 1 \le i \le n_k^*, \quad k \ge 1. \tag{2.7}$$

THEOREM 2 (see [2]). *Assume that the chosen basis of each $\mathcal{L}_k$ is such that* (2.7) *holds. The kernel ψ_h has effective order $p \ge q$ if and only if it is consistent ($F_1 = F$) and the vector fields* (2.6) *in Theorem* 1 *satisfy the equations*

$$f_{k,i} = -r_{k,i}, \qquad 1 \le i \le l_k := n_k - n_{k-1}^*, \quad 2 \le k \le q, \tag{2.8}$$

$$p_{k-1,i} = -f_{k,l_k+i} - r_{k,l_k+i}, \qquad 1 \le i \le n_{k-1}^*, \quad 2 \le k \le q. \tag{2.9}$$

If ψ_h is self-adjoint, then for even values of k, conditions (2.8) *are automatically satisfied and equations* (2.9) *reduce to $p_{k-1,i} = 0$.*

When the coefficients $f_{k,i}$ are written in terms of the parameters $\boldsymbol{\alpha} \equiv (\alpha_1, \dots, \alpha_{\bar{s}})$ ($\bar{s} = 2s$ or $\bar{s} = s$) of the kernel ψ_h, repeated substitution of (2.9) into the recursive expressions (2.8)–(2.9) leads to an equivalent system of equations where (2.8) is replaced by a polynomial equation in the parameters α_j, $N_{k,i}(\boldsymbol{\alpha}) = 0$, and (2.9) is transformed into an expression of the form $p_{k-1,i} = Q_{k,i}(\boldsymbol{\alpha})$, with $Q_{k,i}$ a polynomial in $\boldsymbol{\alpha}$.

The total number of equations $N_{k,i}(\boldsymbol{\alpha}) = 0$ required to achieve effective order q depends on the dimensions n_k in the following way.

COROLLARY 2.1. *There exists a system of $s(q) = n_q + 1$ polynomial equations on the parameters $\boldsymbol{\alpha}$ that guarantees that the corresponding kernel $\psi_h \in \mathcal{G}_1$ (resp., $\psi_h \in \mathcal{G}_l$, $l = 2, 3, 4$) has effective order $q > 1$ for arbitrary basic vector fields $\{F_a, F_b\}$ (resp., $\{Y_k\}_l$). If ψ_h is self-adjoint, this number reduces to $\hat{s}(2) = n_1$, $\hat{s}(2n) = n_1 + \sum_{k=2}^{n} (n_{2k-1} - n_{2k-2})$, $n > 1$.*

The numbers of effective order conditions given by Corollary 2.1 for each class of integrators are displayed in Table 1 for nonsymmetric, $s(q)$, and symmetric, $\hat{s}(q)$, kernels.

For a kernel of effective order q (i.e., satisfying (2.8) for $k \le q$ but not for $k = q+1$) one can in principle determine a postprocessor such that equations (2.9) hold also for all $k > q$. We refer to that postprocessor as optimal, as it causes many terms of each $\hat{F}_k = \sum_{i=1}^{n_k} \hat{f}_{k,i} E_{k,i}$ to cancel ($\hat{f}_{k,n_k-n_{k-1}^*+i} = 0$, $i = 1, \dots, n_{k-1}^*$), and denote by $\mathbb{P}_k$ the set of maps $\pi_h : \mathbb{R}^D \longrightarrow \mathbb{R}^D$ whose Taylor expansion coincides with the optimal postprocessor up to order k (i.e., their difference is $\mathcal{O}(h^{k+1})$). With this notation, the processed integrator $\hat{\psi}_h$ in (1.3) is of order q if the kernel ψ_h is of effective order q and the postprocessor $\pi_h \in \mathbb{P}_{q-1}$. Note that optimality of a postprocessor depends on the chosen basis in each $\mathcal{L}_k$ (see [2] for more details).

TABLE 1
Total number of effective order conditions for nonsymmetric ($s(q)$) and symmetric ($\hat{s}(q)$) kernels of effective order q.

Effective order	$\mathcal{G}_1$		$\mathcal{G}_2$		$\mathcal{G}_3$		$\mathcal{G}_4$	
q	$s(q)$	$\hat{s}(q)$	$s(q)$	$\hat{s}(q)$	$s(q)$	$\hat{s}(q)$	$s(q)$	$\hat{s}(q)$
1	2	–	1	–	1	–	1	–
2	2	2	2	1	1	1	1	1
3	3	–	3	–	2	–	1	–
4	4	3	4	2	2	2	1	1
5	7	–	7	–	3	–	2	–
6	10	6	10	5	3	3	2	2
7	19	–	19	–	5	–	3	–
8	31	15	31	14	6	5	3	3
9	57	–	57	–	9	–	4	–
10	100	40	100	39	12	8	4	4
11	187	–	187	–	19	–	6	–
12	335	127	335	126	26	15	7	6

Remark. For schemes in $\mathcal{G}_1$ it is clear that $\mathcal{L}_1 = \mathrm{span}(E_{1,1} = F = F_a + F_b,\ E_{1,2} = F_a - F_b)$, $\mathcal{L}_1^* = \mathrm{span}(E_{1,1}^* = E_{1,2})$, and thus $n_1 = 2$, $n_1^* = 1$, whereas in $\mathcal{G}_2$ one has $\mathcal{L}_1 = \mathrm{span}(E_{1,1} = F = Y_1)$, $\mathcal{L}_1^* = \emptyset$, so that $n_1 = 1$, $n_1^* = 0$. Moreover, n_k for $k \geq 2$ and $s(k)$ for $k \geq 3$ are identical in $\mathcal{G}_1$ and $\mathcal{G}_2$, but $s(1) = s(2) = 2$ in $\mathcal{G}_1$ and $s(1) = 1$, $s(2) = 2$ in $\mathcal{G}_2$. In consequence, *any first-order method in $\mathcal{G}_1$ has always effective order 2, but this is not the case in $\mathcal{G}_2$.*

For example, let us consider the first-order scheme $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]}$ belonging to $\mathcal{G}_1$ and $\mathcal{G}_2$. If we take $\pi_h = \varphi_{h/2}^{[b]} \in \mathcal{G}_1$, then the processed method $\pi_h \circ \chi_h \circ \pi_h^{-1}$ is the well-known second-order leapfrog scheme. On the other hand, because a general postprocessor in $\mathcal{G}_1$, $\exp(P_h) = \exp\big(h(p_{1,1}F_a + p_{1,2}F_b) + \mathcal{O}(h^2)\big)$, cannot be written as a composition in $\mathcal{G}_2$ unless $p_{1,1} = p_{1,2}$, the effective order of χ_h in $\mathcal{G}_2$ is only one. However, if the kernel is a second-order method (as is the case of symmetric kernels), then $p_{1,1} = p_{1,2} = 0$ and any composition postprocessor in $\mathcal{G}_1$ can be expressed as a composition in $\mathcal{G}_2$; the effective order conditions are equivalent in both cases. □

2.4. Cost saving using the processing technique. For the families of composition methods considered in this work, the number of stages is frequently used as a measure of the computational cost. Here and in the following, the number of basic maps involved in a composition method or kernel will be referred to as the number of stages (i.e., $2s$ in (2.1) and (2.2) and s in (2.3) and (2.4)). In general, the minimum number of stages to obtain a method of order, or effective order, q is given by the total number of order conditions, $N_q = \sum_{k=1}^{q} n_k$, or effective order conditions, $s(q) = n_q + 1$. Then, the use of processed schemes is more advantageous as the value of

$$r_q = \frac{N_q}{s(q)} = \frac{\sum_{k=1}^{q} n_k}{n_q + 1}$$

increases. Observe that r_q depends on how fast $n_k = \dim \mathcal{L}_k$ grows, and thus it is important to analyze its asymptotic behavior.

Let us suppose that $\{X_1, X_2, \dots\}$ are a countable set of generators of a graded free Lie algebra with grades $w_i = w(X_i)$ and suppose that the formal sum $1 - \sum_{i=1}^{\infty} T^{w_i}$ converges to a rational function $P(T) = q(T)/r(T)$, as is the case for the Lie algebras previously considered. If the roots of $q(T)$ and $r(T)$ are denoted by $\lambda_1, \dots, \lambda_m$ and

$\gamma_1, \dots, \gamma_{\bar{m}}$, respectively, then n_k is given as [9, 16, 17]

$$n_k = \frac{1}{k} \sum_{d|k} \mu(d) \left(\sum_{j=1}^{m} \lambda_j^{-k/d} - \sum_{i=1}^{\bar{m}} \gamma_i^{-k/d} \right), \tag{2.10}$$

where the first sum ranges over all integers which divide k, and $\mu(d)$ is the Möbius function: $\mu(1) = 1$, $\mu(d) = (-1)^q$ if d is the product of q distinct prime factors, and $\mu(d) = 0$ otherwise. Let $\rho = 1/\min\{\lambda_1, \dots, \lambda_m\} > 1$ denote the *entropy* of the system, so that $n_k \sim \rho^k/k$ [16]. By instead assuming $n_k \sim \rho^k$, a rough estimation of the value of r_q is obtained as

$$N_q = \sum_{k=1}^{q} n_k \sim \sum_{k=1}^{q} \rho^k \sim \frac{\rho^{q+1}}{\rho - 1} \quad \Longrightarrow \quad r_q \sim \frac{\rho}{\rho - 1}, \tag{2.11}$$

and thus *the cost saving of the processing technique increases when ρ decreases.*

For $\mathcal{G}_1$ and $\mathcal{G}_2$ one has $P(T) = 1 - 2T$ and $P(T) = (1-2T)/(1-T)$, respectively, and $\rho = 2$. In $\mathcal{G}_3$, $P(T) = (1 - T - T^2)/(1 - T^2)$, so $\rho = 1.618$, and in $\mathcal{G}_4$, $P(T) = (1 - T - T^2 + T^3 - T^5)/(1 - T^2)$, so $\rho = 1.443$. It is worth noticing that the values of n_k (and thus the number of order conditions) collected in Table 1 can be obtained through the respective functions $P(T)$ and (2.10).

2.5. Order conditions for methods in $\mathcal{G}_3$. For the sake of illustration we next consider kernels $\psi_h \in \mathcal{G}_3$. Recall that in that case $\mathcal{L} = \bigoplus_{k \geq 1} \mathcal{L}_k$ is the graded free Lie algebra generated by the symbols $\{Y_1, Y_3, Y_5, \dots\}$ ($Y_1 = F$). Now $\mathcal{L}_1^* = \emptyset$, $\mathcal{L}_2 = \emptyset$, so that $n_k^* = n_k$ for all k. We derive the effective order conditions up to order 11 and the corresponding postprocessor coefficients when the basis of $\mathcal{L}_k$, $k \leq 11$, is chosen according to (2.7) (see Table 2 for the particular basis taken).

TABLE 2
Basis of $\mathcal{L}_k$ ($k \leq 11$) chosen for kernels in $\mathcal{G}_3$.

$\mathcal{L}_i$	Basis of $\mathcal{L}_i$		
$\mathcal{L}_1$	$E_{1,1} = Y_1 = F$		
$\mathcal{L}_3$	$E_{3,1} = Y_3$		
$\mathcal{L}_4$	$E_{4,1} = [F, E_{3,1}]$		
$\mathcal{L}_5$	$E_{5,1} = Y_5$	$E_{5,2} = [F, E_{4,1}]$	
$\mathcal{L}_6$	$E_{6,i} = [F, E_{5,i}]$	$i = 1, 2$	
$\mathcal{L}_7$	$E_{7,1} = Y_7$	$E_{7,2} = [E_{3,1}, E_{4,1}]$	
	$E_{7,2+i} = [F, E_{6,i}]$	$i = 1, 2$	
$\mathcal{L}_8$	$E_{8,1} = [E_{3,1}, E_{5,1}]$	$E_{8,i+1} = [F, E_{7,i}]$	$i = 1, \dots, 4$
$\mathcal{L}_9$	$E_{9,1} = Y_9$	$E_{9,i+1} = [E_{3,1}, E_{6,i}]$	$i = 1, 2$
	$E_{9,3+j} = [F, E_{8,j}]$	$j = 1, \dots, 5$	
$\mathcal{L}_{10}$	$E_{10,i} = [E_{3,1}, E_{7,i}]$	$i = 1, \dots, 3$	
	$E_{10,3+j} = [F, E_{9,j}]$	$j = 1, \dots, 8$	
$\mathcal{L}_{11}$	$E_{11,1} = Y_{11}$	$E_{11,i+1} = [E_{3,1}, E_{8,i}]$	$i = 1, \dots, 5$
	$E_{11,7} = [E_{5,1}, E_{6,1}]$	$E_{11,7+j} = [F, E_{10,j}]$	$j = 1, \dots, 11$

As previously stated, if $\psi_h \in \mathcal{G}_3$ the corresponding series of differential operators can be written as $\Psi_h = \exp(Y_{h\alpha_1}) \cdots \exp(Y_{h\alpha_s}) \equiv \exp(F_h)$, with $F_h = hF_1 + h^3 F_3 + h^4 F_4 + \cdots$, and each $f_{k,i}$ in (2.6) is a polynomial in the coefficients $\alpha_1, \dots, \alpha_s$. Consistency ($F_1 = F$) is achieved by requiring that $\sum \alpha_i = 1$. Since $n_1 = 1$, $n_1^* = n_2 = 0$, then $F_2 = 0$, $P_1 = P_2 = 0$ [2]. If we consider a self-adjoint

kernel, a straightforward computation shows that

$$R_2 = R_3 = R_4 = R_5 = R_6 = 0, \qquad R_7 = f_{3,1}p_{4,1}E_{7,2}, \qquad R_8 = 0,$$
$$R_9 = f_{5,1}p_{4,1}(E_{9,2} - E_{9,4}) + \left(f_{5,2}p_{4,1} + \frac{1}{2}p_{4,1}^2 + f_{3,1}p_{6,2}\right) E_{9,3} - p_{4,1}\left(f_{5,2} + \frac{1}{2}p_{4,1}\right) E_{9,6},$$

with similar (but larger) expressions for R_{10}, R_{11}. Equations (2.8) and (2.9) finally lead, after rearranging and simplifying terms, to the order conditions collected in Table 3.

TABLE 3

Order conditions for the kernel and postprocessor up to order q in $\mathcal{G}_3$ when the kernel is self-adjoint ($f_{2k,i} = 0,\ p_{2k-1,j} = 0$).

q	Eff. order conditions	Postprocessor conditions
1	$f_{1,1} = 1$	
3	$f_{3,1} = 0$	
5	$f_{5,1} = 0$	$p_{4,1} = -f_{5,2}$
7	$f_{7,1} = f_{7,2} = 0$	$p_{6,i} = -f_{7,i+2}, \quad i = 1, 2$
9	$f_{9,1} = f_{9,2} = 0$	$p_{8,i} = -f_{9,i+3}, \quad i = 1, 2, 4, 5$
	$f_{9,3} = \frac{1}{2}f_{5,2}^2$	$p_{8,3} = -f_{9,6} - \frac{1}{2}f_{5,2}^2$
11	$f_{11,i} = 0,\ i = 1, 2, 3, 4, 7$	$p_{10,i} = -f_{11,i+7}, \quad i = 1, 2, 4, 7, 8, 10, 11$
	$f_{11,5} = f_{5,2}f_{7,3}$	$p_{10,3} = -f_{11,10} - \frac{3}{2}f_{5,2}f_{7,3}; \quad p_{10,5} = \frac{1}{2}f_{5,2}f_{7,3} - f_{11,12}$
	$f_{11,6} = f_{5,2}f_{7,4}$	$p_{10,6} = -f_{11,13} - \frac{5}{2}f_{5,2}f_{7,4}; \quad p_{10,9} = \frac{3}{2}f_{5,2}f_{7,4} - f_{11,16}$

3. Construction of the processor. Once a kernel satisfying the effective order conditions is proposed, there is still much freedom in the choice of the preprocessor and postprocessor π_h to ensure that $\hat{\psi}_h$ has the required order. The kernel fixes the accuracy that is possible to reach. This can be attained with the optimal postprocessor, defined as the exact 1-flow of an infinite series of vector fields P_h. In practice, one must look for a preprocessor and a postprocessor close to this optimal one (an approximate map in $\mathbb{P}_{q-1}$ or preferably in $\mathbb{P}_q$), and at the same time it has to be easy and/or cheap to compute. In the following we propose two different techniques which satisfy the above requirements.

3.1. Composition postprocessors, $\pi_h^{(c)}$. Since the kernel is a composition of basic methods in $\mathcal{G}_l$, it seems appropriate to approximate the optimal map π_h by a similar composition, $\pi_h^{(c)}$, so that the algorithm employed to compute the kernel can also be used. The resulting processed methods are also in $\mathcal{G}_l$. The coefficients of the composition $\pi_h^{(c)}$ are chosen such that (2.9) are verified. It turns out, however, that finding real solutions for the resulting system of polynomial equations can be difficult, and the composition requires typically at least as many basic integrators as the kernel.

As we mentioned in the introduction, if the kernel is time-symmetric and $\pi_{-h} = \pi_h$, then the processed method is also time-symmetric, but this equality cannot be satisfied by $\pi_h^{(c)}$. Instead we impose that $\pi_{-h}^{(c)} - \pi_h^{(c)} = \mathcal{O}(h^l)$, $l \geq q$, for a postprocessor $\pi_h^{(c)} \in \mathbb{P}_{q-1}$. In this sense, it is sometimes helpful to construct $\pi_h^{(c)}$ as a composition of the form $\pi_h^{(c)} = \omega_h \circ \omega_{-h}$, or even of the form $\pi_h^{(c)} = \omega_{-h} \circ \omega_h^2 \circ \omega_{-h}$. With these special structures many order conditions of the form $p_{2i-1,j} = 0$ in Theorem 2 are automatically satisfied, but the order conditions (2.9) of even k must still be enforced, which still account for more equations than the kernel itself. Since the preprocessor

is computed only once, it makes sense to use $(\pi_h^{(c)})^{-1}$ as an accurate approximation to π_h^{-1}, whereas $\pi_h^{(c)}$ should be used if intermediate output is not frequently required.

3.2. Cheap postprocessors, $\hat{\pi}_h$. For a number of problems, the output has to be computed for most steps. It is therefore reasonable to look for an approximation $\hat{\pi}_h$ to the optimal postprocessor as cheap to compute as possible. Usually $\hat{\pi}_h$ will be a less accurate approximation to π_h than the composition $\pi_h^{(c)}$, but the error $\hat{\pi}_h(z_n) - \pi_h^{(c)}(z_n)$ is of local character and will eventually be overtaken by the global error of the underlying optimally processed integrator (if the global error grows at least linearly in time) [2].

Here we approximate π_h by reusing intermediate calculations obtained when evaluating the kernel ψ_h [2], so that the procedure can be considered virtually cost-free. More specifically, if $z_n \equiv \psi_h^n(\pi_h^{-1}(x_0))$, then the numerical solution $x_n = \pi_h(z_n)$ is approximated by the linear combination

$$x_n \approx \sum_{i=-\bar{s}}^{\bar{s}} w_i Z_i \tag{3.1}$$

of intermediate values Z_i computed when evaluating $z_n = \psi_h(z_{n-1})$ and $z_{n+1} = \psi_h(z_n)$. In particular, we always take $Z_{-\bar{s}} = z_{n-1}$, $Z_0 = z_n$, and $Z_{\bar{s}} = z_{n+1}$. For the set $\mathcal{G}_l$ $(l = 1, 2, 3, 4)$, the intermediate values Z_i that we propose for approximating x_n are the following.

1. In the case of $\mathcal{G}_1$, we have $\bar{s} = 2s$, and

$$Z_{-2s} = z_{n-1}, \quad Z_{2i-1} = \varphi^{[a]}_{\beta_{2i-1}h}(Z_{2i-2}), \quad Z_{2i} = \varphi^{[b]}_{\beta_{2i}}(Z_{2i-1}), \tag{3.2}$$

   where $-(s-1) \le i \le s$ and $\beta_{j-2s} = \beta_j$ for each j.
2. For $\mathcal{G}_2$,

$$Z_{-2s} = z_{n-1}, \quad Z_{2i-1} = \chi^*_{\alpha_{2i-1}h}(Z_{2i-2}), \quad Z_{2i} = \chi_{\alpha_{2i}}(Z_{2i-1}), \tag{3.3}$$

   with $-(s-1) \le i \le s$ and $\alpha_{j-2s} = \alpha_j$ for each j.
3. For $\mathcal{G}_3$ and $\mathcal{G}_4$, we take $\bar{s} = s$, and

$$Z_{-s} = z_{n-1}, \quad Z_i = \mathcal{S}^{[2]}_{\alpha_i h}(Z_{i-1}) \qquad (\text{resp., } Z_i = \mathcal{S}^{[4]}_{\alpha_i h}(Z_{i-1})), \tag{3.4}$$

   with $-(s-1) \le i \le s$ and $\alpha_{j-s} = \alpha_j$ for each j.

Remark. As we will see, the number of conditions to be imposed on the coefficients w_i so that the cheap postprocessor (3.1) achieves a given accuracy is smaller in $\mathcal{G}_2$ than in $\mathcal{G}_1$. Nevertheless, implementing $\psi_h \in \mathcal{G}_1$ as an integrator in $\mathcal{G}_2$ considering the relation between (2.1) and (2.2) is in general more costly because explicitly obtaining the intermediate values Z_i in (3.3) requires the computation of additional basic flows, which does not in general pay off for the extra accuracy of the postprocessor. An important exception is the case in which f_a and f_b come from a partitioned ODE of the form $q' = f_1(p)$, $p' = f_2(q)$. Then an integrator in $\mathcal{G}_1$ can be seen as an explicit partitioned Runge–Kutta method. In that case, there is no practical overhead in implementing the kernel $\psi_h \in \mathcal{G}_1$ as an integrator in $\mathcal{G}_2$ because it does not require additional evaluations of $f_1(p)$ and $f_2(q)$, and then it is recommended to do it that way. □

Each Z_i can be written as $Z_i = \phi_h^{(i)}(z_n)$, where $\phi_h^{(i)} \in \mathcal{G}_l$ $(l = 1, 2, 3, 4,\ -\bar{s} \le i \le \bar{s})$ implies that there exists a series $F_h^{(i)} = \sum_{k\ge1} h^k F_k^{(i)} \in \mathcal{L}$ such that formally

$g \circ \phi_h^{(i)} = \exp(F_h^{(i)})[g]$ for each infinitely differentiable map $g : \mathbb{R}^D \longrightarrow \mathbb{R}$ (see [2] for more details). Clearly, (3.1) can be expressed as

$$x_n \approx \hat{\pi}_h(z_n), \qquad \text{where} \qquad \hat{\pi}_h(z) = \sum_{i=-\bar{s}}^{\bar{s}} w_i \phi_h^{(i)}(z). \tag{3.5}$$

Notice that, in general, $\hat{\pi}_h \notin \mathcal{G}_l$. Our goal is to find coefficients w_i in such a way that $\hat{\pi}_h \in \mathbb{P}_l$ with l as large as possible. This is guaranteed for a given $l \geq 1$ if

$$\sum_{i=-\bar{s}}^{\bar{s}} w_i \exp(F_h^{(i)}) = \exp(P_h) + \mathcal{O}(h^{l+1}), \tag{3.6}$$

where $P_h = \sum_{k\geq 1} h^k P_k \in \mathcal{L}$ is the vector field corresponding to the optimal postprocessor.

The exponential of a formal series in the graded Lie algebra $\mathcal{L}$ can be represented as an element of the universal enveloping algebra $\mathcal{A}$ of $\mathcal{L}$. The algebra $\mathcal{A}$ is also graded, with $\mathcal{A} = \bigoplus_{k\geq 0} \mathcal{A}_k$, and $\mathcal{A}_0 = \text{span}(I)$. Let us now consider a fixed basis $\{D_{k,j}\}_{j=1}^{m_k}$ of the homogeneous subspace $\mathcal{A}_k$ for each $k \geq 1$, with $m_k = \dim \mathcal{A}_k$. The values of m_k for $\mathcal{G}_l$ ($l = 1, 2, 3, 4$) are displayed in Table 4 [2]. Then $\exp(P_h)$ and $\exp(F_h^{(i)})$ can be expressed as

$$\exp(P_h) = I + \sum_{k\geq 1} h^k \sum_{j=1}^{m_k} \pi_{k,j} D_{k,j}, \qquad \exp(F_h^{(i)}) = I + \sum_{k\geq 1} h^k \sum_{j=1}^{m_k} \phi_{k,j}^{(i)} D_{k,j}$$

for $-\bar{s} \leq i \leq \bar{s}$, where $\pi_{k,j}$ and $\phi_{k,j}^{(i)}$ are polynomials in the coefficients α_j, $1 \leq j \leq \bar{s}$. Hence, (3.6) is equivalent to a system of linear equations on the unknowns w_i, i.e.,

$$\sum_{i=-\bar{s}}^{\bar{s}} w_i \phi_{k,j}^{(i)} = \pi_{k,j}, \qquad 1 \leq j \leq m_k, \qquad 0 \leq k \leq l. \tag{3.7}$$

In particular, $\hat{\pi}_h \in \mathbb{P}_0$ requires that $\sum_{i=-\bar{s}}^{\bar{s}} w_i = 1$. The total number of equations (3.7) required for $\hat{\pi}_h \in \mathbb{P}_l$ is $1 + m_1 + \cdots + m_l$ ($m_0 = 1$).

TABLE 4

Dimensions m_k of $\mathcal{A}_k$ for each class of integrators. If the kernel is self-adjoint, only the dimensions m_{2j} are relevant (bold numbers).

	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8
$\mathcal{G}_1$	2	**4**	8	**16**	32	**64**	128	**256**
$\mathcal{G}_2$	1	**2**	4	**8**	16	**32**	64	**128**
$\mathcal{G}_3$	1	**1**	2	**3**	5	**8**	13	**21**
$\mathcal{G}_4$	1	**1**	1	**1**	2	**3**	5	**7**

For self-adjoint composition kernels, $P_{-h} = P_h$ and $F_h^{(-i)} = F_{-h}^{(i)}$, which implies that $\phi_{k,j}^{(-i)} = (-1)^k \phi_{k,j}^{(i)}$. The choice $w_{-i} = w_i$ for all i in (3.1) then makes sense, as this guarantees that (3.7) is automatically satisfied for odd values of k. If in addition $w_0 = 1 - 2\sum_{i=1}^{\bar{s}} w_i$ (so that $\hat{\pi}_h \in \mathbb{P}_l$ at least with $l = 0$), (3.7) holds if

$$2\sum_{i=1}^{\bar{s}} w_i \phi_{2k,j}^{(i)} = \pi_{2k,j}, \qquad 1 \leq j \leq m_{2k}, \quad 1 \leq k \leq \frac{l}{2}. \tag{3.8}$$

Hence, the number of equations that remain to be satisfied by the unknowns $w_1, \ldots, w_{\bar{s}}$ in order that $\hat{\pi}_h \in \mathbb{P}_{2r+1}$ is $m_2 + \cdots + m_{2r}$.

When the number of unknowns w_i is larger than the number of equations (3.7) required so that $\hat{\pi}_h \in \mathbb{P}_l$ for a given l, then one can use this freedom to minimize the difference with the optimal postprocessor at higher orders.

4. New processed methods. In this section we construct self-adjoint composition kernels in $\mathcal{G}_l$ satisfying certain optimization criteria and a simplifying choice of the coefficients. For the kernels collected here we also provide in Appendix A the corresponding postprocessor, built according to the procedure of section 3.

4.1. Construction of the kernel. For building a specific kernel ψ_h of effective order q by composition in $\mathcal{G}_l$, one typically has to take as many basic integrators as the total number, $s(q) = n_q + 1$, of effective order conditions. As a matter of fact, more stages are frequently required (a) due to the nonexistence of real solutions or (b) to diminish the size of higher ($> q+1$)–order error terms (usually of large magnitude) [5, 13, 15]. In any case, introducing extra parameters makes more difficult the task of finding the numerical values of $\boldsymbol{\alpha}$ which solve the equations $N_{k,i}(\boldsymbol{\alpha}) = 0$. For this reason one usually considers self-adjoint kernels: although this symmetry fixes half of the coefficients, it also solves (in most cases) approximately the same number of equations.

Here the following symmetric compositions in $\mathcal{G}_l$, $l = 2, 3, 4$ (schemes in $\mathcal{G}_1$ and $\mathcal{G}_2$ are studied together), at different orders are considered:

$$\psi_h^{[2]} = \chi_{\alpha_1 h} \circ \chi^*_{\alpha_2 h} \circ \cdots \circ \chi_{\alpha_2 h} \circ \chi^*_{\alpha_1 h}, \tag{4.1}$$

$$\psi_h^{[3]} = \mathcal{S}^{[2]}_{\alpha_1 h} \circ \cdots \circ \mathcal{S}^{[2]}_{\alpha_{m-1} h} \circ \mathcal{S}^{[2]}_{\alpha_m h} \circ \mathcal{S}^{[2]}_{\alpha_{m-1} h} \circ \cdots \circ \mathcal{S}^{[2]}_{\alpha_1 h}, \tag{4.2}$$

$$\psi_h^{[4]} = \mathcal{S}^{[4]}_{\alpha_1 h} \circ \cdots \circ \mathcal{S}^{[4]}_{\alpha_{m-1} h} \circ \mathcal{S}^{[4]}_{\alpha_m h} \circ \mathcal{S}^{[4]}_{\alpha_{m-1} h} \circ \cdots \circ \mathcal{S}^{[4]}_{\alpha_1 h}, \tag{4.3}$$

where $\psi_h^{[2]}$ involves $2m$ stages (or m different evaluations of $\varphi_h^{[a]}$ and $\varphi_h^{[b]}$ if $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]}$), whereas $\psi_h^{[3]}$ and $\psi_h^{[4]}$ have $(2m-1)$ stages, and all of them have $\alpha_1, \ldots, \alpha_m$ to solve the effective order conditions.

Among the different solutions, one is interested in those that minimize the $l_{q+1} = n_{q+1} - n_q^*$ noncorrectable terms at order $q+1$, according to some criterion previously adopted. For instance, one could consider a norm in $\mathcal{L}_{q+1}$ and use it to devise some objective function of the parameters α_i to be minimized, aimed at measuring the size of the error. Unfortunately, it is by no means obvious how to characterize the performance of the integrators applied to all initial value problems, since for different problems the dominant error terms are not necessarily the same.

At least two objective functions have been used in the literature:

$$E_1(\boldsymbol{\alpha}) = \sum_{i=1}^{\bar{m}} |\alpha_i| \qquad \text{and} \qquad E_2(\mathbf{d}, \boldsymbol{\alpha}) = \bar{m} \, \|\mathbf{d} \cdot \mathbf{f}_{q+1}\|^{1/q}, \tag{4.4}$$

where $\bar{m} = 2m$ or $2m-1$ (depending on the set $\mathcal{G}_l$ considered), $\alpha_{\bar{m}+1-i} = \alpha_i$, and $\mathbf{f}_{q+1} = (f_{q+1,1}, \ldots, f_{q+1,l_{q+1}})$ and $\mathbf{d} = (d_1, \ldots, d_{l_{q+1}})$ are weight parameters fixed in advance. Methods with small values of E_1 usually have large stability domains and small error terms. It has been successfully used to find efficient nonprocessed methods in $\mathcal{G}_3$ up to order 10 [10, 8], although it is not possible to compare the efficiency of schemes with different number of stages. On the other hand, the optimal

weight vector $\mathbf{d}$ present in E_2 is problem dependent. For methods in $\mathcal{G}_1$ it is usual to consider $\mathbf{d}^{(1)} = (1, \dots, 1)$ (we denote $E_{21} \equiv E_2(\mathbf{d}^{(1)}, \boldsymbol{\alpha})$), whereas in $\mathcal{G}_3$ it is suggested [15] that one take $\mathbf{d}^{(2)} = (1, 0 \dots, 0)$ (and denote $E_{22} \equiv E_2(\mathbf{d}^{(2)}, \boldsymbol{\alpha})$), since $f_{q+1,1} = \sum_{i=1}^{\bar{m}} \alpha_i^{q+1}$ is the dominant error term for a number of problems.

Here we adopt the following criterion: we look for methods with small values of E_1 which, in addition, have small values of E_{21} for schemes in $\mathcal{G}_1$ and/or E_{22} for schemes in $\mathcal{G}_l$, $l = 2, 3, 4$. Of course, one could propose other vectors $\mathbf{d}^{(j)}$ and thus different objective functions to minimize.

As we mentioned before, the number of coefficients in the composition is usually larger than the number of order conditions, so that there are l ($< m$) free parameters $\alpha_1, \dots, \alpha_l$. Recently, McLachlan [15] has proposed an idea to find (in a relatively simple way) efficient methods in $\mathcal{G}_3$: he takes the free parameters $(\alpha_1, \dots, \alpha_l)$ equal to the first (and last) coefficient α_{l+1} of the symmetric composition with the minimum number of stages necessary to satisfy the order conditions. Then, instead of many multidimensional families of solutions, only isolated solutions are obtained which are very easy to analyze. For methods up to effective order 6 in $\mathcal{G}_3$, the solutions obtained by applying this *rule of thumb* correspond to simultaneous local minima of E_1 and E_{22}. For higher orders, this is not necessarily the case, although the corresponding solutions seem to stay very close to the minima.

Here we propose to extend this rule in the following way: we take $\alpha_1 = \alpha_2 = \cdots = \alpha_l \equiv \alpha$ as free parameter and solve the effective order conditions for different values of α. Notice that $\alpha = \alpha_{l+1}$ corresponds to McLachlan's rule of thumb. A numerical search with values $\alpha_1, \dots, \alpha_l$ chosen at random also has been carried out, but the solutions thus found do not lead to smaller values of E_1 and E_2 (see Appendix B for more details). In fact, there exists a strong correlation between E_1 and E_2, but their absolute minima do not necessarily coincide (E_2 can be zero for some choice of $\mathbf{d}$, whereas E_1 is always nonvanishing).

In view of the number of effective order conditions to be satisfied by symmetric kernels in $\mathcal{G}_l$ and collected in Table 1, it seems reasonable to consider methods up to order 6 in $\mathcal{G}_1$ and $\mathcal{G}_2$, up to order 10 in $\mathcal{G}_3$, and up to order 12 in $\mathcal{G}_4$. The new methods are denoted by $\mathrm{P}_{\bar{m}}q$: a processed method of order q with an $\bar{m}$-stage self-adjoint kernel.

4.2. Kernels in $\mathcal{G}_1$ and $\mathcal{G}_2$ up to effective order 6. Several fourth- and sixth-order schemes in $\mathcal{G}_1$ (also valid in $\mathcal{G}_2$) are available in the literature, both as standard composition methods [5] and as processed algorithms [3]. These integrators, however, have been designed to be efficient only in $\mathcal{G}_1$. In contrast, the methods we present here have been optimized to be used both in $\mathcal{G}_1$ and $\mathcal{G}_2$.

Fourth-order. In [3] symmetric kernels with $m = 3$ and $m = 4$ are designed (with $m = 2$ the order conditions do not admit real solutions). We have built kernels with $m = 5, 6, 7$, obtaining solutions with very small values of E_1 and E_{21} and/or E_{22}. In Table 5 (first column) we present the coefficients of the selected kernel with $\bar{m} = 2m = 12$ stages.

Sixth-order. With $m = 5$ it is possible to solve all the order conditions [3], but the resulting schemes are not very efficient. We have considered kernels with m up to 10. The most efficient methods found have $m = 9$ and $m = 10$ and their coefficients are collected in Table 5 (second and third columns). These methods are discussed in more detail in Appendix B. We should remark that, if $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]}$, these kernels require m evaluations of $\varphi_h^{[a]}$ and $\varphi_h^{[b]}$ (due to the FSAL property).

TABLE 5

Coefficients in (4.1) *for symmetric kernels in* $\mathcal{G}_1$ *and* $\mathcal{G}_2$. *These kernels can be used with the corresponding postprocessors of Table* 8.

P_64	P_96	$P_{10}6$
$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5 = \alpha_6$
$\alpha_4 = 0.1341940158122142$	$\alpha_5 = 0.1106570871853300$	$\alpha_6 = 0.1008383784517379$
$\alpha_5 = -\alpha_4 - 0.18$	$\alpha_6 = -0.2854111127287940$	$\alpha_7 = 0.1008384231345842$
$\alpha_6 = 1/2 - (\alpha_1 + \cdots + \alpha_5)$	$\alpha_7 = 0.2138498496192465$	$\alpha_8 = -0.2387378577407101$
$\alpha_{13-i} = \alpha_i, \quad i = 1, \ldots, 6$	$\alpha_8 = -0.3402583791791715$	$\alpha_9 = -0.2387378757998321$
	$\alpha_9 = 1/2 - (\alpha_1 + \cdots + \alpha_8)$	$\alpha_{10} = 1/2 - (\alpha_1 + \cdots + \alpha_9)$
	$\alpha_{19-i} = \alpha_i, \quad i = i, \ldots, 9$	$\alpha_{21-i} = \alpha_i, \quad i = i, \ldots, 10$

4.3. Kernels in $\mathcal{G}_3$ up to effective order 10. Nonprocessed methods up to order 10 are available in [26, 23, 13, 10, 8] and up to order 6 using processing in [1, 15]. Based on the symmetric $(2m-1)$-stage composition (4.2) we present the following kernels.

Sixth-order. With $m = 3$ (5 stages) the effective order conditions $f_{1,1} = 1$, $f_{3,1} = f_{5,1} = 0$ possess two real solutions, but both of them lead to high values of the noncorrectable error terms $f_{7,1}$ and $f_{7,2}$ [1]. With $l = 1, \ldots, 5$ free parameters we have obtained a number of methods with small values of E_1 and E_{22}. We propose taking the 11-stage kernel given in [15] ($P_{11}6$) and the 13-stage kernel ($P_{13}6$) which, in addition, satisfies $f_{7,2} = 0$ (see Table 6).

Eighth-order. With $m \geq 5$ (at least 9 stages) one has enough variables to achieve effective order 8, although the solutions obtained give poorly efficient schemes. Additional stages are considered by taking up to $l = 5$ free parameters. We choose two solutions, one with $l = 2$ ($P_{13}8$) and the other with $l = 5$ ($P_{19}8$). The coefficients are collected in Table 6. The solution with $l = 2$ does not follow exactly the rule of thumb, but it is quite close to it ($\alpha_1 = \alpha_2 \simeq \alpha_5$).

Tenth-order. If $m \geq 8$ (at least 15 stages) one has enough parameters to achieve effective order 10. Kernels with $l = 0, \ldots, 5$ free parameters have been analyzed, and the schemes $P_{19}10$ and $P_{23}10$ whose coefficients are collected in Table 6 have been selected.

4.4. Kernels in $\mathcal{G}_4$ up to effective order 12. Next we consider the symmetric $(2m-1)$-stage composition (4.3). The numerical search for methods of order q in $\mathcal{G}_3$ and of order $q+2$ in $\mathcal{G}_4$ is closely related because the corresponding equations are very similar. In fact, the number of order conditions is smaller in $\mathcal{G}_4$, and the numerical search is easier. Now the methods chosen exactly follow the rule of thumb.

Eighth-order. If $m \geq 3$ (at least 5 stages) one has enough variables to get effective order 8. With $m = 3$, solutions with large values of the noncorrectable leading error term $f_{9,1}$ are obtained [1]. The coefficients of a 9-stage kernel (P_98) are collected in Table 7. Kernels with more stages have also been analyzed, but their performance is not clearly superior.

Tenth-order. At least 7 stages ($m \geq 4$) are necessary to achieve effective order 10. Kernels with up to $l = 5$ free parameters have been considered, and the kernel $P_{13}10$ with coefficients given in Table 7 is finally selected.

Twelfth-order. If $m \geq 6$ (at least 11 stages) a twelfth-order integrator can be obtained. Kernels with up to $l = 6$ free parameters have been analyzed. The coefficients for the composition with 19 stages, $P_{19}12$, are collected in Table 7.

TABLE 6

Coefficients in (4.2) *for symmetric kernels in* $\mathcal{G}_3$. *They can be used with the respective postprocessors collected in Table* 9.

$P_{11}6$	$P_{13}6$	$P_{19}8$
$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$	$\alpha_1 = \cdots = \alpha_6$
$\alpha_4 = 0.1705768865009222157$	$\alpha_4 = 0.125696288720106$	$\alpha_6 = 0.09155941827296$
$\alpha_5 = -0.423366140892658048$	$\alpha_5 = 0.148070660114965$	$\alpha_7 = 0.36968952549113$
$\alpha_6 = 1 - 2(\alpha_1 + \cdots + \alpha_5)$	$\alpha_6 = -0.350856370823828$	$\alpha_8 = 0.06866857653282$
$\alpha_{12-i} = \alpha_i, \quad i = 1, \ldots, 5$	$\alpha_7 = 1 - 2(\alpha_1 + \cdots + \alpha_6)$	$\alpha_9 = -0.28931413259236$
	$\alpha_{14-i} = \alpha_i, \quad i = 1, \ldots, 6$	$\alpha_{10} = 1 - 2(\alpha_1 + \cdots + \alpha_9)$
		$\alpha_{20-i} = \alpha_i, \quad i = 1, \ldots, 9$
$P_{13}8$	$P_{23}10$	$P_{19}10$
$\alpha_1 = \alpha_2$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5$	$\alpha_1 = \alpha_2 = \alpha_3$
$\alpha_2 = 0.168$	$\alpha_5 = 0.121657748919383$	$\alpha_3 = 0.16176042393895$
$\alpha_3 = 0.585550530805562$	$\alpha_6 = -0.511318780154828$	$\alpha_4 = -0.71963383963697$
$\alpha_4 = -0.460090457516872$	$\alpha_7 = -0.172858614884985$	$\alpha_5 = 0.79594876856276$
$\alpha_5 = 0.172863148729731$	$\alpha_8 = 0.123016258833066$	$\alpha_6 = 0.59733925980951$
$\alpha_6 = 0.179664539695039$	$\alpha_9 = 0.441503951671565$	$\alpha_7 = 0.082347969317011$
$\alpha_7 = 1 - 2(\alpha_1 + \cdots + \alpha_6)$	$\alpha_{10} = -0.327071324165477$	$\alpha_8 = -0.43345109677776$
$\alpha_{14-i} = \alpha_i, \quad i = 1, \ldots, 6$	$\alpha_{11} = 0.070952700957766$	$\alpha_9 = 0.10313406454059$
	$\alpha_{12} = 1 - 2(\alpha_1 + \cdots + \alpha_{11})$	$\alpha_{10} = 1 - 2(\alpha_1 + \cdots + \alpha_9)$
	$\alpha_{24-i} = \alpha_i, \quad i = 1, \ldots, 11$	$\alpha_{20-i} = \alpha_i, \quad i = 1, \ldots, 9$

TABLE 7

Coefficients in (4.3) *for symmetric kernels in* $\mathcal{G}_4$. *The corresponding processed integrator is formed with the postprocessors of Table* 10.

$P_9 8$	$P_{13}10$	$P_{19}12$
$\alpha_1 = \alpha_2 = \alpha_3$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4$	$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = \alpha_5$
$\alpha_3 = 0.2233380451446240$	$\alpha_4 = 0.1578763989460225$	$\alpha_5 = 0.1008183703667023$
$\alpha_4 = -0.3739874130116841$	$\alpha_5 = -0.3010347145730912$	$\alpha_6 = 0.2659998884940344$
$\alpha_5 = 1 - 2(\alpha_1 + \cdots + \alpha_4)$	$\alpha_6 = 0.3577728201055250$	$\alpha_7 = -0.2142408955837595$
$\alpha_{10-i} = \alpha_i, \quad i = 1, \ldots, 4$	$\alpha_7 = 1 - 2(\alpha_1 + \cdots + \alpha_6)$	$\alpha_8 = -0.2968630395353906$
	$\alpha_{14-i} = \alpha_i, \quad i = 1, \ldots, 6$	$\alpha_9 = 0.08741046298860494$
		$\alpha_{10} = 1 - 2(\alpha_1 + \cdots + \alpha_9)$
		$\alpha_{20-i} = \alpha_i, \quad i = 1, \ldots, 9$

4.5. On the accuracy of the coefficients. The coefficients of the previous kernels have been obtained with only 15 digits, even when the order of the scheme is 10 or 12. The error introduced due to the accuracy of the coefficients is propagated along the integration, and thus one could think that a 15-digit precision is not sufficient for carrying out very accurate computations. It turns out, however, that this is not observed in practice. To explain this phenomenon notice that, since the kernel is self-adjoint, the asymptotic expansion of the error contains only odd powers of h: hf_{1,i_1}, $h^3 f_{3,i_3}$, $h^5 f_{5,i_5}, \ldots$, and the step size h has to be small enough to guarantee convergence and sufficiently high accuracy. Now the consistency conditions for f_{1,i_1} are exactly satisfied (up to roundoff), and the error in the lowest-order conditions $N_{3,i}(\boldsymbol{\alpha}) = 0$ and $N_{5,i}(\boldsymbol{\alpha}) = 0$ is approximately $h^3 10^{-15}$ and $h^5 10^{-15}$, respectively, which is sufficiently accurate for most problems. If, for a given method, one is interested in obtaining more precise coefficients, the process illustrated by the following example can be applied.

Example. Suppose that the coefficients of the kernel for the processed method $P_{23}10$ in $\mathcal{G}_3$ are desired with higher accuracy. We show how to increase the actual precision from $h^3 10^{-15}$ to $h^5 10^{-15}$ just by adding some extra digits to one of the coefficients α_j.

First we consider the lowest effective order conditions $f_{1,1} = \sum_i \alpha_i = 1$, $N_{3,1}(\boldsymbol{\alpha}) = f_{3,1} = \sum_i \alpha_i^3 = 0$. Let us fix all values of α_i from Table 6 except one, say α_{11} (as well as α_{12} to ensure that condition $f_{1,1} = 1$ is exactly satisfied). Next, solve $N_{3,1}(\boldsymbol{\alpha}) = 0$ with as many digits as desired. For example, with 25 digits one gets $\alpha_{11} = 0.0709527009577666581583926$, and the error of the method due to the accuracy of the coefficients is $h^3 10^{-25} + h^5 10^{-15} \sim h^5 10^{-15}$. This process can also be used to increase the accuracy to $h^7 10^{-15}$ just by taking two parameters to solve $N_{3,1}(\boldsymbol{\alpha}) = 0$ and $N_{5,1}(\boldsymbol{\alpha}) = f_{5,1} = 0$.

5. Numerical examples. In this section we test the performance of the new processed methods on some numerical examples. A rigorous test is very difficult because the relative performance between the methods usually depend on the particular problem analyzed. If the vector field considered involves some parameters, this relative performance may depend also on them as well as on the particular splitting or basic method used for the integration. It may also depend on the initial conditions and even on the final time at which the comparison is done (although this dependence can be minimized taking an average of the error along the integration). For this reason our experiments involve several problems depending on parameters and a comparison of the processed integrators with other schemes available in the literature for different values of parameters, initial conditions, and splitting or basic methods used. In particular, we compare with the following methods (the order, number of stages, and the class $\mathcal{G}_l$ to which they belong is indicated):

- *Order* 4.
  $\mathcal{G}_1$, $\mathcal{G}_2$ ($\mathrm{BM}_6 4$): the 6-stage method (optimized in $\mathcal{G}_1$) given in [5];
  $\mathcal{G}_3$ ($\mathrm{Y}_3 4$): the well-known 3-stage method of [26] (only used as a reference);
  $\mathcal{G}_3$ ($\mathrm{S}_5 4$): the 5-stage method of [22].
- *Order* 6.
  $\mathcal{G}_1$, $\mathcal{G}_2$ ($\mathrm{BM}_{10} 6$): the 10-stage method (optimized in $\mathcal{G}_1$) given in [5];
  $\mathcal{G}_3$ ($\mathrm{M}_9 6$): the optimized 9-stage method $(S, m = 9)$ of [13].
- *Order* 8.
  $\mathcal{G}_3$ ($\mathrm{M}_{17} 8$): the 17-stage method $(S, m = 17)$ of [13];
  $\mathcal{G}_4$ ($\mathrm{B}_7 8$): the 7-stage method of [1].
- *Order* 10.
  $\mathcal{G}_3$ ($\mathrm{R}_{33} 10$): the 33-stage method (s33odr10c) of [10];
  $\mathcal{G}_3$ ($\mathrm{G}_{33} 10$): the 33-stage method of [8].
  $\mathcal{G}_4$ ($\mathrm{O}_{13} 10$): the 13-stage method of [20].
- *Order* 12.
  $\mathcal{G}_4$ ($\mathrm{O}_{23} 12$): the 23-stage method of [20].

In the comparison, the computational cost (or the number of evaluations) of the kernel is taken as the cost of the processed method. All computations are always started with the most accurate composition preprocessor $(\pi_h^{(c)})^{-1}$ available. For postprocessing we take $\pi_h^{(c)}$, although very similar results are obtained when the alternative cheap postprocessor $\hat{\pi}_h$ is used instead, in agreement with the results exposed in [2].

Example 1. We consider the Lotka–Volterra system

$$u' = u(v-2), \qquad v' = v(1-u), \tag{5.1}$$

which has as first integral $I(u,v) = \ln(uv^2) - (u+v)$. For simplicity, we split the vector field as $f = f_a + f_b$, with $f_a = (u(v-2), 0)$ and $f_b = (0, v(1-u))$, which allows us to consider explicit methods which preserve the Poisson structure. This implies, in particular, that the first integral $I(u,v)$ is approximately preserved (for in that case, splitting

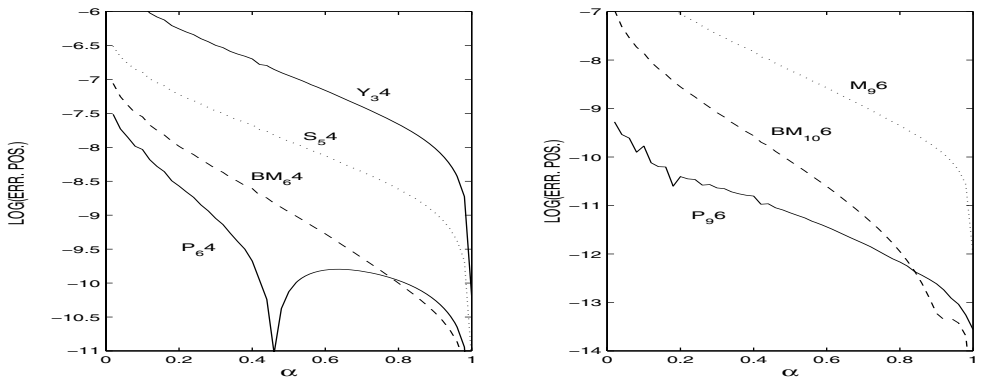


FIG. 1. *Average relative error in position for the methods in $\mathcal{G}_1$ ($\mathcal{G}_2$) and for different initial conditions: $(u_0, v_0) = (\alpha, 2\alpha)$, $0 < \alpha < 1$. The methods* $Y_3 4$, $S_5 4$, *and* $M_9 6$, *belonging to $\mathcal{G}_3$, are included as reference since they are frequently used. Time steps used are $h = m/270$ for the fourth-order methods denoted by $X_m 4$ with X=Y, S, BM, P, so that the four methods require the same number of function evaluations. Similarly, for the sixth-order methods $X_m 6$ with X=M, BM, P, we have taken $h = m/180$.*

methods are, with an appropriate change of variables, symplectic methods applied to a Hamiltonian system with Hamiltonian function $H(p, q) := I(u(p, q), v(p, q)))$. The problem can be integrated by schemes in $\mathcal{G}_1/\mathcal{G}_2$. In the region $0 < u, v$ the system has periodic trajectories around $(u, v) = (1, 2)$. We take as initial conditions $(u_0, v_0) = (\alpha, 2\alpha)$ for different values of $\alpha \in (0, 1)$, and integrate for $t \in [0, 50]$. In this way all periodic trajectories are considered. For each initial condition we compute the error at $t = 1, 2, \ldots, 50$ and measure the average relative error in phase space. The time step is chosen in such a way that (i) all the methods of the same order in the same figure require approximately the same number of function evaluations and (ii) the relative performance between them does not change qualitatively when h is reduced. Figure 1 shows the results obtained for the fourth- and sixth-order methods in $\mathcal{G}_1$ and $\mathcal{G}_2$. We observe that the relative performance between the methods highly depend on the initial conditions but, in general, the new processed methods are more efficient.

To compare the performance of the methods in $\mathcal{G}_3$ we take $\mathcal{S}_h^{[2]} = \varphi_{h/2}^{[a]} \circ \varphi_h^{[b]} \circ \varphi_{h/2}^{[a]}$ as the basic self-adjoint second-order integrator. The corresponding results are shown in Figure 2. Again, the processed methods are clearly superior. When the symmetric second-order implicit midpoint rule is taken as $\mathcal{S}_h^{[2]}$, we have observed that the results of the schemes are more sensitive to initial conditions giving, in general, more accurate results when α is close to 1. Then, depending on the value of α and the qualitative properties which we desire to preserve, one has to choose the most appropriate basic integrator. However, what it is important for our analysis is that the relative performance of the compared methods does not change considerably.

Finally, to compare the behavior of methods of different orders we plot efficiency diagrams. For this test we fix α to a representative value of the performance of the methods: in view of Figures 1 and 2, we take $\alpha = 1/3$. Figure 3 shows the average relative error in phase space as a function of the number of function evaluations for different time steps. The high efficiency achieved by the processed methods $P_9 6$ and $P_{13} 8$ is apparent.

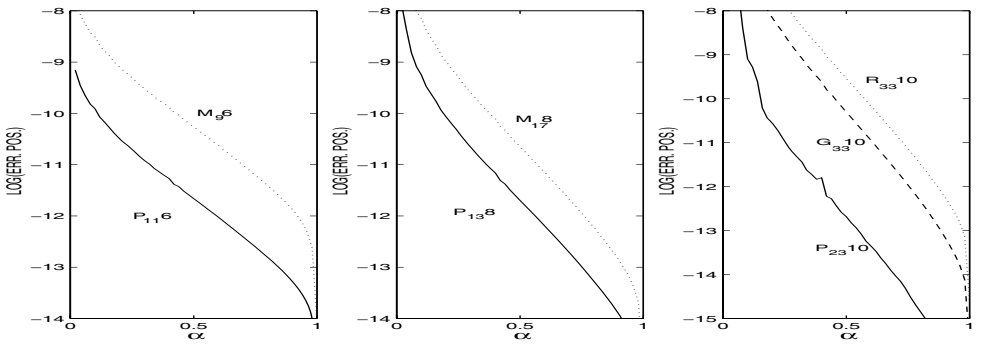


FIG. 2. *Same as Figure* 1 *but for methods in* $\mathcal{G}_3$ *taking* $h = m/396$ *for* X_m6, $h = m/(13\times17)$ *for* X_m8 *(X=M, P),* $h = 1/8$ *for* $G_{33}10$, $R_{33}10$, *and* $h = 1/12$ *for* $P_{23}10$, *so that all methods separately in each plot require, approximately, the same number of function evaluations.*

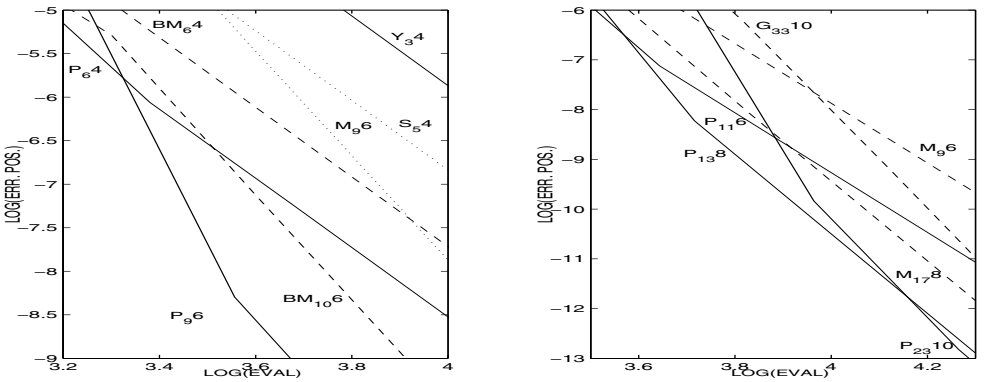


FIG. 3. *Average error in phase space versus the number of evaluations for the methods of Figures* 1 *and* 2 *taking different time steps and* $\alpha = 1/3$.

Example 2. Let us consider now the ABC-flow [8, 16], with equations

$$x' = \mathcal{B}\cos y + \mathcal{C}\sin z, \qquad y' = \mathcal{C}\cos z + \mathcal{A}\sin x, \qquad z' = \mathcal{A}\cos x + \mathcal{B}\sin y, \tag{5.2}$$

which has been studied as a model volume-preserving three-dimensional flow. The vector field f is separable into three solvable parts, namely

$$f = f_a + f_b + f_c = \mathcal{A}(0, \sin x, \cos x) + \mathcal{B}(\cos y, 0, \sin y) + \mathcal{C}(\sin z, \cos z, 0).$$

For $\mathcal{B} = \mathcal{C} = 1$, $\mathcal{A} = \alpha$ we take as initial condition $(x_0, y_0, z_0) = (3.14, 2.77, 0)$ and integrate until $t = 20$ for different values of $\alpha \in [0, 1]$. The α-dependent vector field can be used to test integrators in $\mathcal{G}_l$, $l = 2, 3, 4$. We choose the following explicit basic methods: (i) in $\mathcal{G}_2$, the first-order scheme $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]} \circ \varphi_h^{[c]}$ and its adjoint $\chi_h^* = \varphi_h^{[c]} \circ \varphi_h^{[b]} \circ \varphi_h^{[a]}$; (ii) in $\mathcal{G}_3$, the symmetric composition $\mathcal{S}_h^{[2]} = \chi_{h/2} \circ \chi_{h/2}^*$; and (iii) in $\mathcal{G}_4$, the well-known 3-stage fourth-order scheme $\mathcal{S}_h^{[4]} = \mathcal{S}_{\alpha_1 h}^{[2]} \circ \mathcal{S}_{\alpha_2 h}^{[2]} \circ \mathcal{S}_{\alpha_1 h}^{[2]}$, with $\alpha_1 = 1/(2-2^{1/3})$ and $\alpha_2 = 1-2\alpha_1$. Obviously, more efficient self-adjoint fourth-order basic methods $\mathcal{S}_h^{[4]}$ could improve the accuracy of the results, but at this stage, we are mainly interested in the relative performance of the different integrators, and this does not change considerably with the basic method.

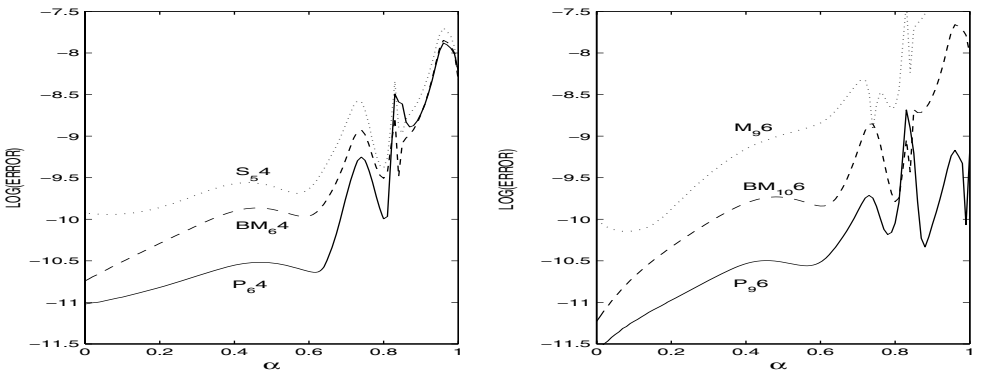


FIG. 4. *Error in position for the ABC-flow with $\mathcal{B} = \mathcal{C} = 1$ at the same computational cost for the fourth- (left) and sixth-order methods (right) in $\mathcal{G}_2$ taking different values of the parameter $\mathcal{A} = \alpha$, $0 \le \alpha \le 1$, and initial condition $(x_0, y_0, z_0) = (3.14, 2.77, 0)$.*

The time step size used for each method is such that all the integrators of the same order appearing in the same figure require approximately the same number of evaluations, and the results have the expected asymptotic behavior corresponding to the respective order. Figure 4 shows the results obtained with the fourth- and sixth-order methods in $\mathcal{G}_2$, whereas Figure 5 corresponds to the sixth-, eighth-, and tenth-order methods in $\mathcal{G}_3$. We have repeated the experiment with different initial conditions, and the relative performance of the methods also changes. However, we have observed a superiority of the processed integrators which, on average, is well represented by Figures 4 and 5.

On the other hand, since we have taken $\mathcal{B} = \mathcal{C}$, the system has a reversing symmetry group with 16 elements [16], not preserved by the preceding splitting. However, if we consider

$$f = f_a + f_b = \mathcal{B}(\cos y + \sin z, 0, 0) + (0, \mathcal{B}\cos z + \mathcal{A}\sin x, \mathcal{A}\cos x + \mathcal{B}\sin y), \tag{5.3}$$

this splitting does preserve the reversing symmetry group. Notice that f_a is explicitly integrable, but f_b is not. We can integrate f_b, $\varphi_h^{[b]}$, with the midpoint rule, $\widetilde{\varphi}_h^{[b]}$, which preserves all the properties [16]. We repeated the experiments shown in Figure 5 using as the basic method $\mathcal{S}_h^{[2]} = \varphi_{h/2}^{[a]} \circ \widetilde{\varphi}_h^{[b]} \circ \varphi_{h/2}^{[a]}$ and the same time steps. The midpoint rule was implemented with the Newton method (three iterations usually gave enough accuracy). The results are shown in Figure 6, where we observe that the relative performance between the methods change, but the processed methods are more accurate.

Finally, Figure 7 shows the results obtained by the methods in $\mathcal{G}_4$ using the explicit methods. Now the processed integrators are always between three and four orders of magnitude more accurate at the same cost, in agreement with the size of the leading order error coefficients.

Example 3. Finally, to illustrate the interest of using methods in $\mathcal{G}_4$, we consider the two-body gravitational problem. The corresponding evolution equation may be written as

$$x_i'' = -\frac{x_i}{(x_1^2 + x_2^2)^{3/2}}, \qquad i = 1, 2, \tag{5.4}$$

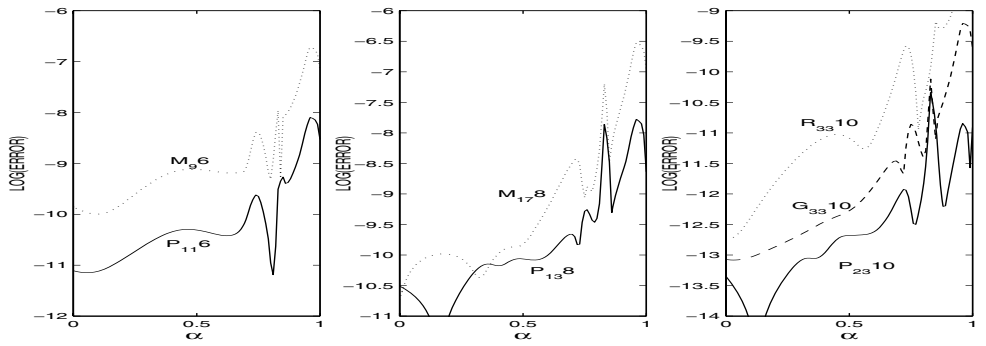


FIG. 5. *Same as Figure* 4 *for the sixth-, eighth-, and tenth-order methods in* $\mathcal{G}_3$ *taking* $h = m/99$ *for* X_m6, $h = 4m/(13 \times 17)$ *for* X_m8 *(X=M, P),* $h = 5/23$ *for* $G_{33}10$, $R_{33}10$, *and* $h = 5/33$ *for* $P_{23}10$.

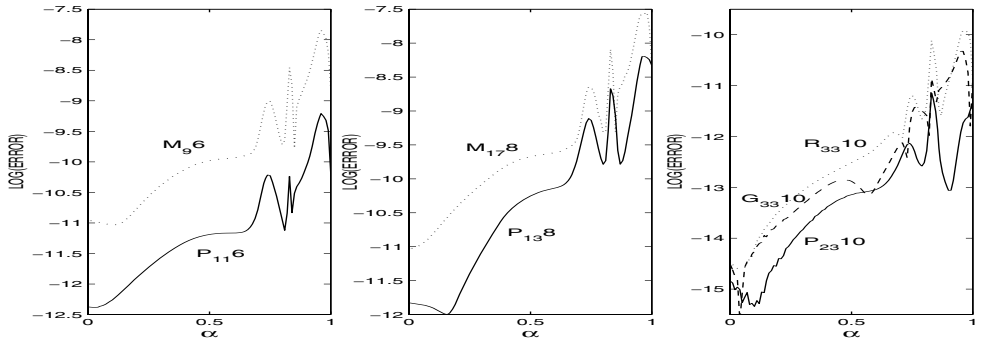


FIG. 6. *Same as Figure* 4 *for the sixth-, eighth-, and tenth-order methods in* $\mathcal{G}_3$ *for the splitting given in* (5.3) *and using the implicit midpoint rule to solve the equation for* f_b.

and we take as initial condition $x_1(0) = 1-e$, $x_2(0) = x_1'(0) = 0$, $x_2'(0) = [(1+e)/(1-e)]^{1/2}$, which produces an orbit with eccentricity e. With the value $e = 1/4$, the orbit is determined numerically for 100 periods and the mean error in positions is computed. This problem is suited to integration by symplectic Runge–Kutta–Nyström methods [21], and moreover, modified potentials can also be used [4, 12]. In particular, Koseleff [11] designed an efficient self-adjoint fourth-order scheme, which can be used as the basic method:

$$\mathcal{S}_h^{[4]} = \varphi_{h/6}^{[b]} \circ \varphi_{h/2}^{[a]} \circ \varphi_{2h/3,h^3/72}^{[b,c]} \circ \varphi_{h/2}^{[a]} \circ \varphi_{h/6}^{[b]}. \tag{5.5}$$

Here

$$\varphi_{\alpha h}^{[a]}(x_i) = x_i + \alpha h x_i', \qquad \varphi_{\beta h,\gamma h^3}^{[b,c]}(x_i') = x_i' - x_i G\,(\beta + \hat{\gamma}\, G)\,, \qquad i = 1, 2, \tag{5.6}$$

with $\hat{\gamma} = 4\gamma h$, $G = h/(x_1^2 + x_2^2)^{3/2}$, and $\varphi_{\beta h}^{[b]}$ corresponds to taking $\gamma = 0$ in the modified flow $\varphi_{\beta h,\gamma h^3}^{[b,c]}$. Notice that the cost of evaluating $\varphi_{\beta h,\gamma h^3}^{[b,c]}$ instead of $\varphi_{\beta h}^{[b]}$ increases only by one addition and one multiplication, so that it can be neglected.

We compare the new processed methods in $\mathcal{G}_4$, taking (5.5) as the basic integrator with the nonprocessed schemes $M_{17}8$ and $G_{33}10$ in $\mathcal{G}_3$ and using as basic second-order symmetric integrator the well-known Störmer–Verlet/leapfrog method. Figure

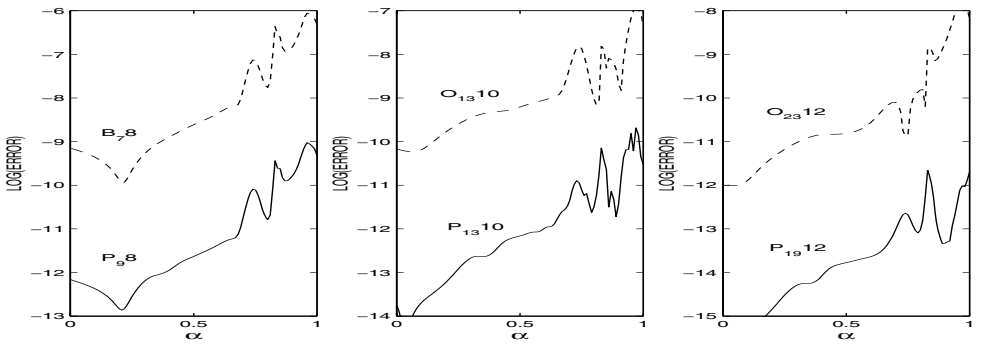


FIG. 7. *Same as Figure* 4 *for the eighth-, tenth-, and twelfth-order methods in* $\mathcal{G}_4$.

8 shows the average relative error in position versus the number of $\varphi_{\beta h}^{[b]}$ evaluations (one evaluation of (5.5) is considered to involve two $\varphi_{\beta h}^{[b]}$ evaluations due to the FSAL property). It is worth noticing the excellent performance of the processed methods $P_9 8$ and $P_{13}10$. In particular, for the step sizes considered in the figure $P_{13}10$ behaves as a twelfth-order integrator: in fact, one has to take a much smaller h to recover the asymptotic behavior expected for a tenth-order method. A similar pattern can be observed in the diagram corresponding to $P_9 8$.

This example clearly shows that in certain circumstances and depending on the particular structure of the system at hand, it can be advantageous to design an efficient fourth-order scheme $\mathcal{S}_h^{[4]}$ and then build by composition higher-order integrators based on it. Obviously, the performance of the methods highly depend on the particular fourth-order basic scheme and a more detailed analysis of this could even improve the performance observed in Figure 8.

We must also remark that after 100 revolutions it is still possible to obtain an accuracy of 18 digits. However, the coefficients for the numerical methods in Table 7 are given with only 15 digits. This is so because the lowest-order condition is multiplied by h^5, which is also a small number. In other words, the comments of section 4.5 are valid here.

6. Conclusions. In this paper we have constructed numerical integrators for differential equations up to order twelve by composition of basic integrators and using the processing technique. Four families of integrators are considered: (i) fourth- and sixth-order composition methods for a system separable into two solvable parts or (ii) using as basic methods a first-order integrator and its adjoint; (iii) sixth-, eighth-, and tenth-order composition methods using second-order self-adjoint methods; and (iv) eighth-, tenth-, and twelfth-order composition methods using fourth-order self-adjoint methods.

Our starting point is a theoretical analysis of the processing technique for the cases considered here, which allows us to obtain in a relatively simple way the number and explicit form of the order conditions for the kernel. The analysis clearly shows that the processing technique is a very promising procedure to obtain efficient methods when implemented with constant time step. In that case, the cost of the postprocessor can be neglected even if the output has to be frequently computed since, if desired, it can be approximated using intermediate stages of the kernel.

Next, we have carried out a numerical search of the coefficients of different com-

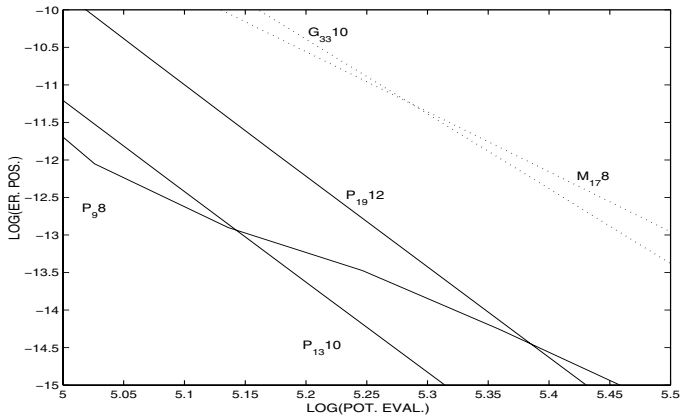


FIG. 8. *Average error in position versus number of evaluations.* $P_9 8$, $P_{13} 10$, *and* $P_{19} 12$ *are processed methods in* $\mathcal{G}_4$ *using* (5.5) *as the basic method (considered as two evaluations of the potential).* $M_{17} 8$ *and* $G_{33} 10$ *are methods in* $\mathcal{G}_3$ *using the Störmer–Verlet leapfrog method (one evaluation of the potential).*

positions to get efficient kernels for the cases considered. Compositions with more stages than strictly required to solve the order conditions are considered for optimization purposes by following, mainly, the rule of thumb proposed in [15]. In this way, the higher accuracy obtained compensates the increase in the computational cost of the methods.

A large number of numerical experiments have been carried out to compare the performance of the new processed methods with other well-established nonprocessed methods, and some representative results are shown. We have considered different initial conditions, values of the parameters, basic methods, and splittings to better appreciate the performance of the methods.

We think that some comments about the choice and applicability of the different methods proposed here could be of interest for the potential user. If a given problem can be integrated using methods in $\mathcal{G}_1$, then it is clear that it can also be integrated with methods in $\mathcal{G}_2, \mathcal{G}_3$, and $\mathcal{G}_4$, but for the same order, methods in $\mathcal{G}_1$ and $\mathcal{G}_2$ are more efficient than schemes in $\mathcal{G}_3$ and $\mathcal{G}_4$. Thus, if one is interested in fourth- or sixth-order methods, it is, in general, preferable to consider methods in $\mathcal{G}_1$ or $\mathcal{G}_2$; for orders 8 or 10 we recommend methods in $\mathcal{G}_3$, and only for higher-order methods or when a very efficient fourth-order scheme is available do we recommend methods in $\mathcal{G}_4$. The same comments are valid for a problem which can be integrated in $\mathcal{G}_2$ but not in $\mathcal{G}_1$.

Although the processing technique has become, during the past few years, a standard technique for constructing efficient high-order composition integrators, there are still some issues which need to be clarified.

- Which is the most appropriate objective function to minimize leading to the most efficient methods for a given class of problems?
- We have considered self-adjoint kernels with more stages than necessary to solve the effective order conditions. On the other hand, for nonsymmetric kernels such a number of stages would contain a larger number of free parameters. The question is, With such assumptions, could other interesting solutions not contemplated in this work exist?

- Could there exist other possible symmetries for the coefficients and different *rules of thumb* which would allow one to obtain more efficient methods?
- How can one build specific fourth-order methods for a given problem that when used as a basic scheme in $\mathcal{G}_4$ would lead to very efficient higher-order composition methods?
- Would it be possible to construct more efficient composition methods for particular problems of physical relevance for which the Lie algebra involved presents a simpler structure than those analyzed in this work?

Appendix A. Construction of specific postprocessors. Next we present several composition and cheap postprocessors for some of the kernels collected in section 4. The resulting integrators provide the most efficient results on different test examples (analyzed in section 5).

With respect to composition postprocessors, in accordance with the comments in section 4.1, we take $\pi_h^{(c)} = w_h^{[i]} \circ w_{-h}^{[i]}$, $i = 2, 3, 4$, for the composition postprocessor, with

$$w_h^{[2]} = \chi_{\gamma_{2r}h} \circ \chi^*_{\gamma_{2r-1}h} \circ \cdots \circ \chi_{\gamma_2 h} \circ \chi^*_{\gamma_1 h}, \tag{A.1}$$

$$w_h^{[3]} = \mathcal{S}^{[2]}_{\gamma_r h} \circ \cdots \circ \mathcal{S}^{[2]}_{\gamma_1 h}, \tag{A.2}$$

$$w_h^{[4]} = \mathcal{S}^{[4]}_{\gamma_r h} \circ \cdots \circ \mathcal{S}^{[4]}_{\gamma_1 h} \tag{A.3}$$

and require that $\pi_h^{(c)} \in \mathbb{P}_{q-1}$, although in some cases a more accurate composition $\pi_h^{(c)} \in \mathbb{P}_q$ is also built, particularly when the number of stages required is not too large. On the other hand, the cheap postprocessors are constructed by following the procedure outlined in section 3.2.

Postprocessors in $\mathcal{G}_1$ and $\mathcal{G}_2$. Condition $p_{1,1} = \sum_i \gamma_i = 0$ produces a composition $\pi_h^{(c)} = w_h^{[2]} \circ w_{-h}^{[2]} \in \mathbb{P}_2$, whereas 4 and 15 order conditions have to be satisfied if $\pi_h^{(c)} \in \mathbb{P}_j$, $j = 4, 6$, respectively. In Table 8 we give coefficients for $\pi_h^{(c)}$ in $\mathbb{P}_4$ ($\mathrm{P}_6 4$). Since the kernel $\mathrm{P}_9 6$ of effective order 6 is the most efficient in practice, we present two possible postprocessors: one in $\mathbb{P}_5$ ($r = 4$) and a more accurate one in $\mathbb{P}_6$ ($r = 9$), which is recommended for use in both $\mathcal{G}_1$ and $\mathcal{G}_2$. Finally, we provide a cheap postprocessor $\hat{\pi}_h \in \mathbb{P}_3$ for $\mathrm{P}_6 4$.

Postprocessors in $\mathcal{G}_3$. For a sixth-order processed integrator it is required that $\pi_h \in \mathbb{P}_5$ and, according to Table 3, this is accomplished if $p_{4,1} = -f_{5,2}$ and $p_{1,1} = p_{3,1} = p_{5,1} = p_{5,2} = 0$. In fact, with $\pi_h^{(c)} = w_h^{[3]} \circ w_{-h}^{[3]}$, $w_h^{[3]}$ given by (A.2) and satisfying $\sum_{i=1}^r \gamma_i = 0$, we need only to enforce $p_{4,1} = -f_{5,2}$. If, in addition, $p_{6,1} = -f_{7,3}$ and $p_{6,2} = -f_{7,4}$ hold, then $\pi_h^{(c)} \in \mathbb{P}_6$, whereas if condition $\sum_{i=1}^r \gamma_i^3 = 0$ is satisfied, then $\pi_h^{(c)} \in \mathbb{P}_7$: this is, in fact, the composition postprocessor collected in Table 9 for $\mathrm{P}_{11} 6$ and $\mathrm{P}_{13} 6$, preserving time-symmetry up to $\mathcal{O}(h^9)$. Taking into account the conditions on $p_{8,i}$, $i = 1, \ldots, 5$, one gets $\pi_h^{(c)} \in \mathbb{P}_8$, and so on. For $\mathrm{P}_{13} 8$ (which works in practice more efficiently than $\mathrm{P}_{19} 8$) we provide a composition postprocessor in $\mathbb{P}_8$ and a 24-stage $\pi_h^{(c)} \in \mathbb{P}_9$ for $\mathrm{P}_{23} 10$. One could get a postprocessor in $\mathbb{P}_{10}$ for the tenth-order case, but this would require 11 new conditions to be satisfied by $w_h^{[3]}$, giving a 46-stage postprocessor. We also present the coefficients of cheap processors $\hat{\pi}_h \in \mathbb{P}_{q-1}$, $q = 6, 8$, valid for $\mathrm{P}_{11} 6$ and $\mathrm{P}_{13} 8$, respectively. A similar $\hat{\pi}_h$ could be obtained for $\mathrm{P}_{23} 10$, but from Table 4, it would need at least 33 coefficients w_i.

TABLE 8
Postprocessor coefficients in (A.1) *and* (3.5) *for some kernels in* $\mathcal{G}_1/\mathcal{G}_2$.

$P_6 4:\ \pi_h^{(c)} \in \mathbb{P}_4$	$P_6 4:\ \hat{\pi}_h \in \mathbb{P}_3$	$P_9 6:\ \pi_h^{(c)} \in \mathbb{P}_5$
$\gamma_1 = 0$	$w_0 = 1 - 2(w_1 + w_5 + w_6 + w_7)$	$\gamma_1 = 0.01012220322650864$
$\gamma_2 = -(\gamma_3 + \cdots + \gamma_6)$	$w_1 = 0.46640472356735$	$\gamma_2 = -0.2840923693556972$
$\gamma_3 = -0.1612056894758833$	$w_5 = -0.02125258839849$	$\gamma_3 = -0.0739474688028612$
$\gamma_4 = -0.2694936673582758$	$w_6 = -0.04899563905006$	$\gamma_4 = 0.00007025760293332$
$\gamma_5 = 0.2736158718483377$	$w_7 = 0.00811211574986$	$\gamma_5 = -0.02385471740106014$
$\gamma_6 = 0.1832420262145362$		$\gamma_6 = 0.08878308313058653$
		$\gamma_7 = -(\gamma_1 + \cdots + \gamma_6)$
		$\gamma_8 = 0$
$P_9 6:\ \pi_h^{(c)} \in \mathbb{P}_6$		
$\gamma_1 = -0.2013206758948216$	$\gamma_2 = -0.1281761283096599$	$\gamma_3 = 0.0570146336015926$
$\gamma_4 = -0.0359500915398769$	$\gamma_5 = 0.0498478378426457$	$\gamma_6 = 0.1725721056066613$
$\gamma_7 = -0.0654901907171583$	$\gamma_8 = -0.0576378685707717$	$\gamma_9 = 0.1817517334193077$
$\gamma_{10} = 0.1672419181837143$	$\gamma_{11} = 0.2111126647112377$	$\gamma_{12} = -0.0733409487114027$
$\gamma_{13} = -0.1420023272628973$	$\gamma_{14} = 0.1215976276874689$	$\gamma_{15} = 0.0329874370062511$
$\gamma_{16} = -0.2361731625865831$	$\gamma_{17} = -(\gamma_1 + \cdots + \gamma_{16})$	$\gamma_{18} = 0$

TABLE 9
Postprocessor coefficients in (A.2) *and* (3.5) *for some kernels in* $\mathcal{G}_3$.

$P_{11} 6:\ \pi_h^{(c)} \in \mathbb{P}_6$	$P_{11} 6:\ \hat{\pi}_h \in \mathbb{P}_5$	$P_{13} 6:\ \pi_h^{(c)} \in \mathbb{P}_6$
$\gamma_6 = -(\gamma_1 + \cdots + \gamma_5)$	$w_0 = 1 - 2(w_1 + w_5 + w_6 + w_7)$	$\gamma_6 = 0.1$
$\gamma_5 = -0.20621953139126$	$w_1 = 0.35601475536028$	$\gamma_5 = 0.225080298761176$
$\gamma_4 = 0.23651387483203$	$w_5 = 0.12246549694690$	$\gamma_4 = 0.191244694511161$
$\gamma_3 = 0.09086982276241$	$w_6 = 0.00415291514453$	$\gamma_3 = -0.212763792194890$
$\gamma_2 = 0.24687306977659$	$w_7 = -0.20658995116781$	$\gamma_2 = -0.09660157306582295$
$\gamma_1 = -0.1$		$\gamma_1 = -(\gamma_2 + \cdots + \gamma_6)$
$P_{13} 8:\ \pi_h^{(c)} \in \mathbb{P}_8$	$P_{13} 8:\ \hat{\pi}_h \in \mathbb{P}_7$	$P_{23} 10:\ \pi_h^{(c)} \in \mathbb{P}_9$
$\gamma_{10} = -(\gamma_1 + \cdots + \gamma_9)$	$w_0 = 1 - 2(w_1 + \cdots + w_{12})$	$\gamma_{12} = -(\gamma_1 + \cdots + \gamma_{11})$
$\gamma_9 = -0.008488123494574411$	$w_1 = -3.6976426586421067$	$\gamma_{11} = 0.4727142080578221$
$\gamma_8 = -0.337188967354338$	$w_2 = 1.0615669344875514$	$\gamma_{10} = 0.01344750613191108$
$\gamma_7 = 0.333987768164597$	$w_3 = 0.040377839731292050$	$\gamma_9 = -0.4637104712987078$
$\gamma_6 = 0.588351189003849$	$w_4 = 0.0830491660507623$	$\gamma_8 = -0.3045590922565247$
$\gamma_5 = 0.162324207599241$	$w_5 = -0.0221811460897851$	$\gamma_7 = -0.5382945821834320$
$\gamma_4 = -0.511744926116413$	$w_6 = -0.1398573630328631$	$\gamma_6 = -0.1899795533199732$
$\gamma_3 = 0.236885952363384$	$w_7 = -0.0074999124845547055$	$\gamma_5 = 0.1548256472553489$
$\gamma_2 = 0.598212975943381$	$w_8 = 0.21992320817724267$	$\gamma_4 = -0.3594148033156072$
$\gamma_1 = -0.543415765371656$	$w_9 = -0.21401705459232256$	$\gamma_3 = 0.3430345669677392$
	$w_{10} = -0.014339878804936956$	$\gamma_2 = 0.5334030283695922$
	$w_{11} = 0.09819025594252939$	$\gamma_1 = 0.009116042043427756$
	$w_{12} = 0.034452779507214946$	

Postprocessors in $\mathcal{G}_4$. In Table 10 we present accurate composition postprocessors $\pi_h^{(c)} = w_h^{[4]} \circ w_{-h}^{[4]}$ belonging to $\mathbb{P}_8$ and $\mathbb{P}_{10}$ for $P_9 8$ and $P_{13} 10$, respectively, whereas the postprocessor for the twelfth-order method belongs to $\mathbb{P}_{11}$.

Appendix B. Here we illustrate the validity of the criteria adopted in section 4 for the choice of the particular kernels collected there. We fix our attention, in particular, on the compositions of effective order 6 in $\mathcal{G}_2$ and effective order 8 in $\mathcal{G}_3$. In the first case we take the symmetric composition (4.1) with the minimum number of maps required to satisfy the effective order conditions ($m = 5$) $\psi(m)$, and then we introduce $l \leq 5$ additional maps with free parameters $\alpha_1, \ldots, \alpha_l$ in the composition.

TABLE 10
Postprocessor coefficients for kernels in $\mathcal{G}_4$.

$P_9 8:\ \pi_h^{(c)} \in \mathbb{P}_8$	$P_{13}10:\ \pi_h^{(c)} \in \mathbb{P}_{10}$	$P_{19}12:\ \pi_h^{(c)} \in \mathbb{P}_{11}$
$\gamma_5 = -(\gamma_1 + \cdots + \gamma_4)$	$\gamma_7 = -(\gamma_1 + \cdots + \gamma_6)$	$\gamma_8 = -(\gamma_1 + \cdots + \gamma_7)$
$\gamma_4 = -0.2196648965658254$	$\gamma_6 = 0.2834714107596056$	$\gamma_7 = 0.2505061593209141$
$\gamma_3 = 0.2405373742563472$	$\gamma_5 = -0.2553426863586816$	$\gamma_6 = -0.01365235094631587$
$\gamma_2 = 0.1406336264566169$	$\gamma_4 = -0.03297486475329144$	$\gamma_5 = 0.1795942654148864$
$\gamma_1 = 0.1$	$\gamma_3 = 0.2298746411002190$	$\gamma_4 = -0.1860801553027685$
	$\gamma_2 = -0.2869384247718548$	$\gamma_3 = -0.2526234033672912$
	$\gamma_1 = -0.2110578773704694$	$\gamma_2 = 0.01093309142620025$
		$\gamma_1 = -0.1873261189973930$

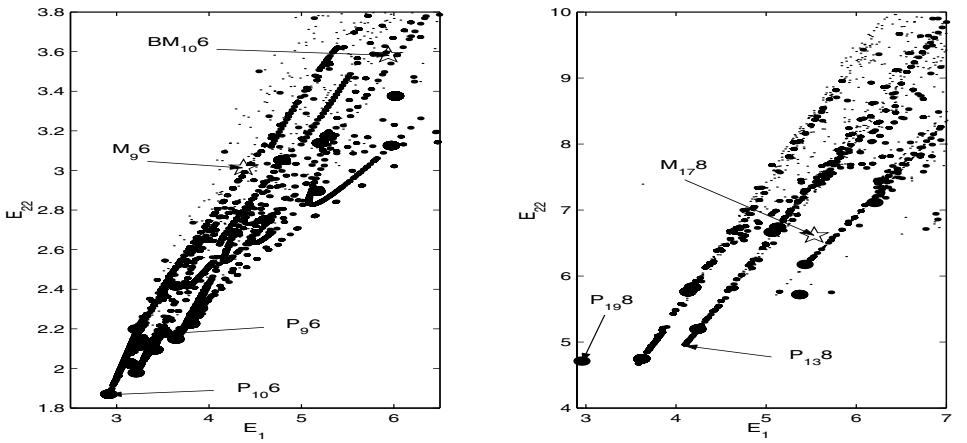


FIG. 9. *Values of E_1 and E_{22} for self-adjoint kernels using l free parameters when they are chosen randomly (small dots), applying the generalized rule of thumb (medium dots) and using McLachlan's rule of thumb (big dots). Only the results for the kernels of effective order 6 in $\mathcal{G}_1/\mathcal{G}_2$ with $l = 3, 4, 5$ (left) and the results for the kernels of effective order 8 in $\mathcal{G}_3$ with $l = 1, 2, 3$ (right) are shown.*

For instance, when $l = 4$ the kernel has the form

$$\psi_h^{[2]} = \chi_{\alpha_1 h} \circ \chi_{\alpha_2 h}^* \circ \chi_{\alpha_3 h} \circ \chi_{\alpha_4 h}^* \circ \psi(m) \circ \chi_{\alpha_4 h} \circ \chi_{\alpha_3 h}^* \circ \chi_{\alpha_2 h} \circ \chi_{\alpha_1 h}^*.$$

Next, for each l and given values of $\alpha_1, \ldots, \alpha_l$, we solve the effective order conditions and compute the objective functions E_1 and E_{22}. In Figure 9 (left) we collect the results obtained when (i) the free parameters $\alpha_1, \ldots, \alpha_l$ are chosen at random (small dots); (ii) $\alpha_1 = \cdots = \alpha_l = \alpha$ for different values of α (medium dots); and (iii) applying McLachlan's rule of thumb, i.e., $\alpha_1 = \cdots = \alpha_l = \alpha_{l+1}$ (big dots). For clarity, only the results with $l = 3, 4, 5$ and a small fraction of points in (i) are shown. It is worth noticing that a random selection of $\alpha_1, \ldots, \alpha_l$ does not lead to smaller values of E_1 and E_{22} than those obtained by applying the rule of thumb or its generalization. In fact, the medium dots lie on curves, whereas the big dots are located in the neighborhood of their minima. For comparison, we also include the values obtained with the nonprocessed 10-stage method $\mathrm{BM}_{10}6$ (optimized as a method in $\mathcal{G}_1$) given in [5] when $\chi_h = \varphi_h^{[a]} \circ \varphi_h^{[b]}$ and the optimized method $\mathrm{M}_9 6$ with $\mathcal{S}_h^{[2]} = \chi_{h/2} \circ \chi_{h/2}^*$ (stars).

This figure should be interpreted with caution: from the obtained values of E_1 and E_{22}, $\mathrm{M}_9 6$ should be more efficient than $\mathrm{BM}_{10}6$, but in practice it is just the opposite.

In fact, the value of E_{21} (using a basis of $\mathcal{L}$ in $\mathcal{G}_1$) is smaller with $\mathrm{BM}_{10}6$. This example indicates some of the difficulties involved in the choice of the best objective functions for methods in $\mathcal{G}_1$ and $\mathcal{G}_2$. Among the different solutions in the lower left corner of the figure we have selected two of them: $\mathrm{P}_{10}6$ (providing the minimum of E_1 and E_{22}) and $\mathrm{P}_9 6$ (with the smallest value of E_{21}). In practice, the kernel $\mathrm{P}_9 6$ shows the best performance.

In Figure 9 (right) we show the results obtained with the kernel of effective order 8 in $\mathcal{G}_3$. Now $l = 1, 2, 3$ free parameters are considered. We also include the values given by the 17-stage method $\mathrm{M}_{17}8$ and the 19-stage kernel $\mathrm{P}_{19}8$ of Table 6.

Acknowledgment. The authors would like to thank the referees for their careful reading of the manuscript and their comments, which helped to improve this paper a great deal.

REFERENCES

[1] S. Blanes, *High order numerical integrators for differential equations using composition and processing of low order methods*, Appl. Numer. Math., 37 (2001), pp. 289–306.
[2] S. Blanes, F. Casas, and A. Murua, *On the numerical integration of ordinary differential equations by processed methods*, SIAM J. Numer. Anal., 42 (2004), pp. 531–552.
[3] S. Blanes, F. Casas, and J. Ros, *Symplectic integration with processing: A general study*, SIAM J. Sci. Comput., 21 (1999), pp. 711–727.
[4] S. Blanes, F. Casas, and J. Ros, *High-order Runge–Kutta–Nyström geometric integrators with processing*, Appl. Numer. Math., 39 (2001), pp. 245–259.
[5] S. Blanes and P. C. Moan, *Practical symplectic partitioned Runge–Kutta and Runge–Kutta–Nyström methods*, J. Comput. Appl. Math., 142 (2002), pp. 313–330.
[6] J. Butcher, *The effective order of Runge-Kutta methods*, in Conference on the Numerical Solution of Differential Equations, Lecture Notes in Math. 109, Springer-Verlag, Berlin, 1969, pp. 133–139.
[7] J. Butcher and J. M. Sanz-Serna, *The number of conditions for a Runge-Kutta method to have effective order p*, Appl. Numer. Math., 22 (1996), pp. 103–111.
[8] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration. Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Ser. Comput. Math. 31, Springer-Verlag, Berlin, 2002.
[9] S.-J. Kang and M.-H. Kim, *Free Lie algebras, generalized Witt formula, and the denominator identity*, J. Algebra, 183 (1996), pp. 560–594.
[10] W. Kahan and R. C. Li, *Composition constants for raising the order of unconventional schemes for ordinary differential equations*, Math. Comp., 66 (1997), pp. 1089–1099.
[11] P. V. Koseleff, *Formal Calculus for Lie Methods in Hamiltonian Mechanics*, Ph.D. thesis, Lawrence Berkeley Laboratory LBID-2030, Tech. report UC-405, Berkeley, CA, 1994.
[12] M. A. López-Marcos, J. M. Sanz-Serna, and R. D. Skeel, *Explicit symplectic integrators using Hessian-vector products*, SIAM J. Sci. Comput., 18 (1997), pp. 223–238.
[13] R. I. McLachlan, *On the numerical integration of ordinary differential equations by symmetric composition methods*, SIAM J. Sci. Comput., 16 (1995), pp. 151–168.
[14] R. I. McLachlan, *More on symplectic correctors*, in Integration Algorithms and Classical Mechanics, Fields Inst. Commun. 10, J. E. Marsden, G. W. Patrick, and W. F. Shadwick, eds., AMS, Providence, RI, 1996, pp. 141–149.
[15] R. I. McLachlan, *Families of high order composition methods*, Numer. Algorithms, 31 (2002), pp. 233–246.
[16] R. I. McLachlan and R. Quispel, *Splitting methods*, Acta Numer., 11 (2002), pp. 341–434.
[17] H. Munthe-Kaas and B. Owren, *Computations in a free Lie algebra*, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., 357 (1999), pp. 957–981.
[18] A. Murua and J. M. Sanz-Serna, *Order conditions for numerical integrators obtained by composing simpler integrators*, R. Soc. Lond. Philos. Trans. Ser. A Math. Phys. Eng. Sci., 357 (1999), pp. 1079–1100.
[19] P. J. Olver, *Applications of Lie Groups to Differential Equations*, 2nd ed., Springer-Verlag, New York, 1993.

[20] I. P. Omelyan, I. M. Mryglod, and R. Folk, *Construction of high-order force-gradient algorithms for integration of motion in classical and quantum systems*, Phys. Rev. E, 66 (2002), 026701.

[21] J. M. Sanz-Serna and M. P. Calvo, *Numerical Hamiltonian Problems*, Chapman and Hall, London, 1994.

[22] M. Suzuki, *General theory of fractal path integrals with applications to many-body and statistical physics*, J. Math. Phys., 32 (1991), pp. 400–407.

[23] M. Suzuki, *Quantum Monte Carlo methods and general decomposition theory of exponential operators and symplectic integrators*, Phys. A, 205 (1994), pp. 65–79.

[24] V. S. Varadarajan, *Lie Algebras and Their Representations*, Springer-Verlag, New York, 1984.

[25] J. Wisdom, M. Holman, and J. Touma, *Symplectic correctors*, in Integration Algorithms and Classical Mechanics, Fields Inst. Commun. 10, J. E. Marsden, G. W. Patrick, and W. F. Shadwick, eds., AMS, Providence, RI, 1996, pp. 217–244.

[26] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A, 150 (1990), pp. 262–268.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1844–1866
© 2006 Society for Industrial and Applied Mathematics

ISOPERIMETRIC PARTITIONING: A NEW ALGORITHM FOR GRAPH PARTITIONING*

LEO GRADY† AND ERIC L. SCHWARTZ‡

Abstract. We present a new algorithm for graph partitioning based on optimization of the combinatorial isoperimetric constant. It is shown empirically that this algorithm is competitive with other global partitioning algorithms in terms of partition quality. The isoperimetric algorithm is easy to parallelize, does not require coordinate information, and handles nonplanar graphs, weighted graphs, and families of graphs which are known to cause problems for other methods. Compared to spectral partitioning, the isoperimetric algorithm is faster and more stable. An exact circuit analogy to the algorithm is also developed with a natural random walks interpretation. The isoperimetric algorithm for graph partitioning is implemented in our publicly available Graph Analysis Toolbox [L. Grady, Ph.D. thesis, Boston University, Boston, MA, 2004], [L. Grady and E. L. Schwartz, Technical report TR-03-021, Boston University, Boston, MA, 2003] for MATLAB obtainable at http://eslab.bu.edu/software/graphanalysis/. This toolbox was used to generate all of the results compiled in the tables of this paper.

Key words. graph partitioning, spectral partitioning, isoperimetric problem, graph theory, linear equations

AMS subject classifications. 91B28, 91B30, 60J75, 30A36

DOI. 10.1137/040609008

1. Introduction.

In the graph partitioning problem, one chooses subsets of the vertex set of a graph such that the sets share a minimal number of spanning edges while satisfying a specified cardinality constraint. Applications of graph partitioning include parallel processing [63], solving sparse linear systems [58], VLSI circuit design [4], and image segmentation [61, 68, 32].

Methods of graph partitioning take different forms, depending on the number of partitions required—whether or not the nodes have coordinates—and the cardinality constraints of the sets. In this paper, we use the term *partition* to refer to the assignment of each node in the vertex set into two (not necessarily equal) parts. We propose an algorithm termed *isoperimetric partitioning*, since it is derived and motivated by the isoperimetric constant defined for continuous manifolds [15]. The isoperimetric algorithm most closely resembles spectral partitioning in its use and ability to create hybrids with other algorithms (e.g., multilevel spectral partitioning [40] and geometric-spectral partitioning [14]). However, it requires the solution to a large, sparse system of equations rather than solving the eigenvector problem for a large, sparse matrix. This difference leads to improved speed and numerical stability.

The paper is organized as follows: we begin by deriving the isoperimetric algorithm from the isoperimetric constant of a graph in section 2, followed in section 3 by two physical analogies for the isoperimetric algorithm, and in section 4 by proving a few formal properties of the algorithm. In section 5, we review the most popular and

*Received by the editors May 25, 2004; accepted for publication (in revised form) February 25, 2005; published electronically February 3, 2006. This work was supported in part by the Office of Naval Research (ONR N00011401-1-0624).

http://www.siam.org/journals/sisc/27-6/60900.html

†Department of Imaging and Visualization, Siemens Corporate Research, 755 College Road East, Princeton, NJ 08540 (leo.grady@siemens.com).

‡Departments of Cognitive and Neural Systems and Electrical and Computer Engineering, Boston University, Boston, MA 02215 (eric@bu.edu).

effective graph partitioning algorithms and the relation of the present work. Section 6 provides examples intended to build intuition about the algorithm behavior. Section 7 validates the present method on various general types of graphs and several specific graphs, followed by the conclusion.

2. Isoperimetric algorithm. A *graph* is a pair $G = (V, E)$ with vertices $v \in V$ and edges $e \in E \subseteq V \times V$. An edge, e, spanning two vertices, v_i and v_j, is denoted by e_{ij}. Let $n = |V|$ and $m = |E|$, where $|\cdot|$ denotes cardinality. A *weighted graph* has a value (here assumed to be nonnegative and real) assigned to each edge called a *weight*. The weight of edge e_{ij} is denoted by $w(e_{ij})$ or w_{ij}. Since weighted graphs are more general than unweighted graphs (i.e., $w(e_{ij}) = 1$ for all $e_{ij} \in E$ in the unweighted case), we develop all our results for weighted graphs.

Graph partitioning has been strongly influenced by properties of a combinatorial formulation of the classic isoperimetric problem: *For a fixed area, find the shape with minimum perimeter* [16]. The approach to graph partitioning presented here is a polynomial time heuristic for the NP-hard [55] problem of finding a graph with minimum perimeter for a fixed area.

Cheeger defined [15] the *isoperimetric constant* h of a manifold as

$$h = \inf_S \frac{|\partial S|}{\mathrm{Vol}_S}, \tag{2.1}$$

where S is a region in the manifold, Vol_S denotes the volume of region S, $|\partial S|$ is the area of the boundary of region S, and h is the infimum of the ratio over all possible S. For a compact manifold, $\mathrm{Vol}_S \leq \frac{1}{2}\mathrm{Vol}_{\mathrm{Total}}$, and for a noncompact manifold, $\mathrm{Vol}_S < \infty$ (see [55, 54]).

For a graph, G, the *isoperimetric number* [55], h_G, is

$$h_G = \inf_S \frac{|\partial S|}{\mathrm{Vol}_S}, \tag{2.2}$$

where $S \subset V$ and

$$\mathrm{Vol}_S \leq \frac{1}{2}\mathrm{Vol}_V. \tag{2.3}$$

In finite graphs, the infimum in (2.2) becomes a minimum. The boundary of a set, S, is defined as $\partial S = \{e_{ij} | i \in S, j \in \overline{S}\}$ and on a weighted graph

$$|\partial S| = \sum_{e_{ij} \in \partial S} w(e_{ij}). \tag{2.4}$$

In the context of graph partitioning, combinatorial volume is typically taken as

$$\mathrm{Vol}_S = |S|. \tag{2.5}$$

For a given set of nodes, S, we term the ratio of its boundary to its volume as the *isoperimetric ratio* and denote it by $h(S)$. The *isoperimetric sets* for a graph, G, are any S and $\overline{S}$ for which $h(S) = h_G$. The specification of a set satisfying (2.3), together with its complement, may be considered as a *partition*, and therefore we use the term interchangeably with the specification of a set satisfying (2.3). A good partition is defined to be one with a low isoperimetric ratio (i.e., the optimal partition consists of the isoperimetric sets themselves). Therefore, our goal is to maximize Vol_S

while minimizing $|\partial S|$. Finding isoperimetric sets is an NP-hard problem [55], so our algorithm is a heuristic for finding a set with a low isoperimetric ratio that runs in polynomial time.

Define an indicator vector that takes a binary value at each node

$$x_i = \begin{cases} 0 & \text{if } v_i \in S, \\ 1 & \text{if } v_i \in \overline{S}. \end{cases} \tag{2.6}$$

A specification of x also defines a partition. Define the $n \times n$ matrix L of a graph as

$$L_{v_i v_j} = \begin{cases} d_i & \text{if } i = j, \\ -w(e_{ij}) & \text{if } e_{ij} \in E, \\ 0 & \text{otherwise,} \end{cases} \tag{2.7}$$

where d_i denotes the weighted *degree* of vertex v_i

$$d_i = \sum_{e_{ij}} w(e_{ij}) \ \ \forall\, e_{ij} \in E. \tag{2.8}$$

The notation $L_{v_i v_j}$ is used to indicate that the matrix L is being indexed by vertices v_i and v_j. This matrix is also known as the *admittance matrix* in the context of circuit theory or the *Laplacian matrix* (see [51] for a review).

By definition of L

$$|\partial S| = x^T L x, \tag{2.9}$$

and $\mathrm{Vol}_S = x^T r$, where r denotes the vector of all ones. Maximizing the volume of S subject to $\mathrm{Vol}_S = k$ for some constant $k \leq \frac{1}{2}\mathrm{Vol}_V$ may be done by asserting the constraint

$$x^T r = k. \tag{2.10}$$

In terms of the indicator vector, the isoperimetric number of a graph (2.2) is given by

$$h_G = \min_x \frac{x^T L x}{x^T r}. \tag{2.11}$$

Given an indicator vector, x, then $h(x)$ is used to represent the isoperimetric ratio associated with that partition.

The constrained optimization of the isoperimetric ratio is made into a free variation via the introduction of a Lagrange multiplier [6], Λ, and relaxation of the binary definition of x to take nonnegative real values. Therefore, solving for an optimal partition may be accomplished by minimizing the function

$$Q(x) = x^T L x - \Lambda(x^T r - k). \tag{2.12}$$

Since L is positive semidefinite (see [8, 26]) and $x^T r$ is nonnegative, $Q(x)$ will be at a minimum at its critical points. Differentiating $Q(x)$ with respect to x yields

$$\frac{dQ(x)}{dx} = 2Lx - \Lambda r. \tag{2.13}$$

Thus, the problem of finding the x that minimizes (2.12) (i.e., the minimal partition) reduces to solving the linear system

$$2Lx = \Lambda r. \tag{2.14}$$

Henceforth, the scalar multiplier 2 and the scalar Λ are ignored since, as will be seen later, we are only concerned with the relative values of the solution.

The matrix L is singular: all rows and columns sum to zero (i.e., the vector r spans its nullspace), so finding a unique solution to (2.14) requires an additional constraint.

The graph is assumed to be connected, since the optimal partitions are clearly each connected component (i.e., $h(x) = h_G = 0$) if the graph is disconnected. A linear time breadth-first search may be performed to check for connectivity of the graph. Note that, in general, a graph with c connected components will correspond to a matrix L with rank $(n - c)$ [8]. If a node, v_g, is arbitrarily designated to include in S (i.e., fix $x_g = 0$), this is reflected in (2.14) by removing the gth row and column of L, and the gth row of x and r such that

$$L_0 x_0 = r_0, \tag{2.15}$$

where L_0 indicates the Laplacian with a row/column removed, x_0 is the vector x with the corresponding removed entry, and r_0 is r with the removed row. Note that (2.15) is a nonsingular system of equations.

Solving (2.15) for x_0 yields a nonnegative, real-valued solution that may be converted into a partition by setting a threshold. Nodes with an x_i below the threshold are placed in S and nodes with an x_j above the threshold are placed in $\overline{S}$. Setting a threshold is referred to as a *cut* since it divides the nodes into S and $\overline{S}$. We use x to collectively refer to the x_0 and the designated $x_g = 0$ value. As in spectral partitioning, common methods [64] of defining a threshold are the *median cut*, which chooses the median value of x as the threshold (thereby guaranteeing $|S| = |\overline{S}|$), the *jump cut*, which chooses a threshold that separates nodes on either side of the largest "jump" in a sorted x, and the *criterion cut*, which chooses the threshold that gives the lowest value of $h(x)$ (called the "ratio cut" in [64]).

2.1. Algorithmic details. The isoperimetric algorithm for partitioning a graph may be summarized as follows:

1. Choose a ground node, v_g.
2. Solve (2.15).
3. Cut x based on the method of choice to obtain S and $\overline{S}$.

There are several possible strategies for choosing the ground node. Anderson and Morley [5] proved that the spectral radius of L, $\rho(L)$, satisfies $\rho(L) \leq 2d_{\max}$, suggesting that grounding the node of maximum degree may have the most beneficial effect on the conditioning of (2.15). In the comparison section of this paper, we employ two grounding strategies (grounding the maximum degree node and grounding a random node).

The main computational hurdle of the isoperimetric algorithm is the solution of (2.15). An advantage of using the method of conjugate gradients is that an efficient parallelization of this technique is known [20, 33], suggesting that the majority of computation in the isoperimetric algorithm may be parallelized. In fact, cheap parallel machines in the form of commodity graphics hardware have already proven to be effective in executing the conjugate gradients method [9].

3. Physical analogies. The solution to (2.15) may be interpreted as the solution to a set of electrical potentials of a particular circuit and, with a slight change, in the context of a random walk. These analogies are primarily introduced in order to guide intuition about the behavior of the algorithm and will be relied upon in section 6 to provide understanding of the examples contained there. However, an added benefit of the circuit analogy is the adoption of terminology used to describe the node fixed to zero and the solution to (2.15).

3.1. Circuit analogy. Equation (2.14) occurs in circuit theory when solving for the electrical potentials of an ungrounded circuit in the presence of current sources [11, 65]. After grounding a node in the circuit (i.e., fixing its potential to zero), determination of the remaining potentials requires a solution of (2.15). Therefore, we refer to the node, v_g, for which we set $x_g = 0$ as the *ground node*. Likewise, the solution, x_i, obtained from (2.15) at node v_i, will be referred to as the *potential* for node v_i. The need for fixing an $x_g = 0$ to constrain (2.14) may be seen not only from the necessity of grounding a circuit powered only by current sources in order to find unique potentials, but also from the need to provide a boundary condition in order to find a solution to Poisson's equation, of which (2.14) is a combinatorial analogue. In our case, the "boundary condition" is that the grounded node is fixed to zero.

Define the $m \times n$ edge-node *incidence matrix* as

$$A_{e_{ij}v_k} = \begin{cases} +1 & \text{if } i = k, \\ -1 & \text{if } j = k, \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

for every vertex v_k and edge e_{ij}, where e_{ij} has been arbitrarily assigned an orientation. As with the Laplacian matrix, $A_{e_{ij}v_k}$ is used to indicate that the incidence matrix is indexed by edge e_{ij} and node v_k. As an operator, A may be interpreted as a combinatorial gradient operator and A^T as a combinatorial divergence [10, 65]. The $m \times m$ *constitutive matrix*, C, is the diagonal matrix with the weights of each edge along the diagonal.

As in the familiar continuous setting, the combinatorial Laplacian is equal to the composition of the combinatorial divergence operator with the combinatorial gradient operator, $L = A^T A$. The constitutive matrix defines a weighted inner product of edge values, i.e., $\langle y, Cy \rangle$ for a vector of edge values, y [65, 11]. Therefore, the combinatorial Laplacian operator generalizes to the combinatorial Laplace–Beltrami operator via $L = A^T C A$. The case of a uniform (unit) metric (i.e., equally weighted edges) reduces to $C = I$ and $L = A^T A$. Removing a column of the incidence matrix produces what is known as the *reduced incidence matrix*, A_0 [27].

With this interpretation of the notation used above, the three fundamental equations of circuit theory (Kirchhoff's current and voltage laws and Ohm's law) may be written for a grounded circuit as

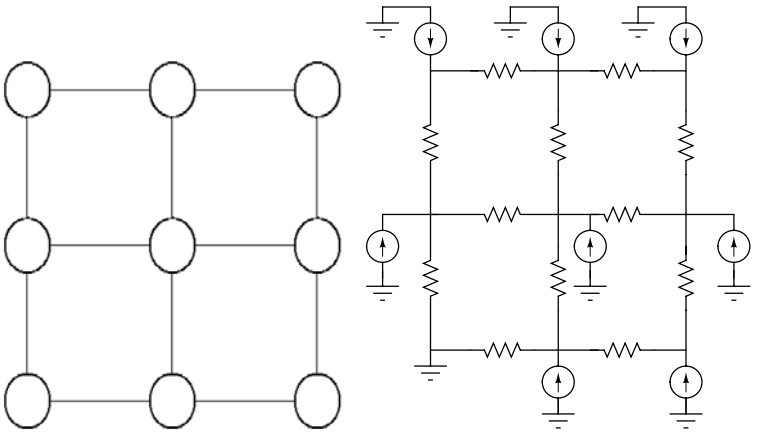
$$A_0^T y = f \quad \text{(Kirchhoff's current law)}, \tag{3.2}$$

$$Cp = y \quad \text{(Ohm's law)}, \tag{3.3}$$

$$p = A_0 x \quad \text{(Kirchhoff's voltage law)} \tag{3.4}$$

for a vector of branch currents, y, current sources, f, and potential drops (voltages), p [11]. Note that there are no voltage sources present in this formulation. These three equations may be combined into the linear system

$$A_0^T C A_0 x = L_0 x = f, \tag{3.5}$$



(a) (b)

FIG. 3.1. *An example of a simple graph* (a)*, and its equivalent circuit* (b)*. Solving* (2.15) (*using the node in the lower left as ground*) *for the graph depicted in* (a) *is equivalent to connecting* (b) *and measuring the potential values at each node.*

since $A^T CA = L$ [8].

In summary, the solution to (2.15) in the isoperimetric algorithm is provided by the steady state of a circuit where each edge has a conductance equal to the edge weight and each node is attached to a current source of magnitude equal to the degree (i.e., the sum of the conductances of incident edges) of the node. The potentials that are established on the nodes of this circuit are exactly those which are being solved for in (2.15). An example of this equivalent circuit is displayed in Figure 3.1.

One final remark on the circuit analogy to (2.15) follows from Maxwell's principle of least dissipation of power: a circuit with minimal power dissipation provides a solution to Kirchhoff's current and voltage laws [50]. Explicitly, solving (2.15) for x is equivalent to solving the dual equation for $y = CAx$. The power of the equivalent circuit is $P = I^2 R = y^T C^{-1} y$ subject to the constraint from Kirchhoff's law that $A^T y = f$. Therefore, the y found via $y = CAx$ also minimizes the above expression [65, 7]. Thus, our approach to minimizing the combinatorial isoperimetric ratio is identical to minimizing the power of the equivalent electrical circuit with the specified current sources and ground [65].

3.2. Random walk interpretation. In section 2, the combinatorial "volume" of a node set was defined by (2.5) as the cardinality of the set. However, this definition of combinatorial volume can lead to strange conclusions when trying to develop the general theory of Riemannian manifolds on a combinatorial space [17, 18, 54]. The alternate definition proposed by [18] defines volume as

$$\text{Vol}_S = \sum_{v_i \in S} d_i. \tag{3.6}$$

Using this notion of volume in the formulation of the isoperimetric constant in (2.2) yields the problem studied under the name of the *minimum quotient cut* [48].

Substituting this notion of volume (which is equivalent up to a scale factor on a

regular graph) into the above development leads to the solution of

$$L_0 x_0 = d_0, \tag{3.7}$$

where d_0 is the reduced vector of node degrees.

The solution, x_i, of (3.7) may be interpreted [67, 21] as the expected number of steps taken by a random walker leaving node v_i before it reaches the ground node, where the transition probability between nodes v_i and v_j is derived from the weights as $p_{ij} = \frac{w_{ij}}{d_i}$.

We note that, in the formulation of their Laplacian matrix, this same notion of "volume" was effectively used in the domain of image segmentation by Shi and Malik's "normalized cuts" algorithm [61].

4. Some formal properties of the algorithm. In this section, we prove one property of the algorithm and examine the behavior of the solution on two classes of graphs: trees and fully connected graphs.

4.1. Connectivity. It is known that at least one of the isoperimetric sets (if not unique) is such that both partitions are connected [55]. Therefore, it is of interest to examine the connectivity properties of a partitioning algorithm. Fiedler proved that partitioning a graph by thresholding the values in the Fiedler vector is guaranteed to produce connected partitions [25]. In this section, we examine the connectivity properties of the partitions obtained by thresholding the potentials solved for in (2.15) (or (3.7)). We prove that the partition containing the grounded node (i.e., the set S) must be connected, regardless of how a threshold (i.e., cut) is chosen. The strategy for establishing this will be to show that every node has a path to ground with a monotonically decreasing potential. We note that the partition not containing the ground may or may not be connected (cf. Figures 6.1 and 6.2).

PROPOSITION 4.1. *If the set of vertices, V, is connected then, for any α, the subgraph with vertex set $N \subseteq V$ defined by $N = \{v_i \in V | x_i < \alpha\}$ is connected when x_0 satisfies $L_0 x_0 = f_0$ for any $f_0 \geq 0$.*

This proposition follows directly from the proof of the following lemma.

LEMMA 4.2. *For every node, v_i, there exists a path to the ground node, v_g, defined by $P_i = \{v_i, v^1, v^2, \ldots, v_g\}$ such that $x_i \geq x^1 \geq x^2 \geq \cdots \geq 0$, when $L_0 x_0 = f_0$ for any $f_0 \geq 0$.*

Proof. By (2.15) each nongrounded node assumes a potential

$$x_i = \frac{1}{d_i} \sum_{e_{ij} \in E} w_{ij} x_j + \frac{f_i}{d_i}, \tag{4.1}$$

i.e., the potential of each nongrounded node is equal to a nonnegative constant added to the (weighted) average potential of its neighbors. Note that (4.1) is a combinatorial formulation of the mean value theorem [1] in the presence of sources. For any connected subset, $S \subseteq V$, $v_g \notin S$, denote the set of nodes on the boundary of S as $S_b \subset V$, such that $S_b = \{v_i |\, e_{ij} \in E,\ \exists\, v_j \in S,\ v_i \notin S\}$.

Now, either

1. $v_g \in S_b$, or
2. $\exists\, v_i \in S_b$, such that $x_i \leq \min x_j$ for all $v_j \in S$ by (4.1), since the graph is connected.

By induction, every node has a path to ground with a monotonically decreasing potential (i.e., start with $S = \{v_i\}$; add nodes with a nonincreasing potential until ground is reached). ∎

4.2. Spanning trees. Since a tree will have a unique path from any node to ground, Lemma 4.2 guarantees that the nodes in this path will have a nonincreasing potential. However, since a tree is a special case of a graph, and its reduced incidence matrix is square, there is an alternate derivation of this result. A theorem by Branin [10] shows that for node v_k, edge e_{ij}, and ground v_g, the inverse of the reduced incidence matrix for a spanning tree, $B = A_0^{-1}$, is

$$B_{v_k e_{ij}} = \begin{cases} +1 & \text{if } e_{ij} \text{ is positively traversed in the path from } v_k \text{ to } v_g, \\ -1 & \text{if } e_{ij} \text{ is negatively traversed in the path from } v_k \text{ to } v_g, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Therefore,

$$L_0^{-1} = (A_0^T C A_0)^{-1} = B^T C^{-1} B. \tag{4.3}$$

Each value of L_0^{-1} may therefore be interpreted as the sum of the reciprocal weights (i.e., the *resistances*) of shared edges along the unique path to v_g between nodes v_i and v_j, i.e., the shared *distance* of the unique paths from v_i and v_j to v_g in the metric interpretation.

It follows that the potential values taken by x_0 in $x_0 = L_0^{-1} f_0$ are monotonically increasing along the path from v_g to any other node for nonnegative f_0 and C.

4.3. Fully connected graphs. The isoperimetric algorithm will produce a solution to (2.15) that prefers each node equally when applied to fully connected graphs with uniform weights. Any set with cardinality equal to half the cardinality of the vertex set is a solution to the isoperimetric problem for a fully connected graph with uniform weights. For a uniform edge weight, $w(e_{ij}) = \kappa$ for all $e_{ij} \in E$, the solution, x_0, to (2.15) will be $x_i = 1/\kappa$ for all $v_i \in V$. The use of the median or criterion cut method will choose half of the nodes arbitrarily. Although it should be pointed out that using a median or criterion cut to partition a vector of randomly assigned potentials will also produce equal-sized (in the case optimal) partitions, the solution to (2.15) is unique for a specified ground (in contrast to spectral partitioning, which has $n-1$ solutions) and explicitly gives no preference to a node by returning all potentials as equal.

5. Review of previous work. Many approaches have been proposed for the graph partitioning problem and its related guises (e.g., circuit placement) in the past 50 years. Consistently, however, the most popular algorithms for graph partitioning are spectral partitioning [35] and the (multilevel) Kernighan–Lin partitioning method [41, 22].

5.1. Algebraic methods. Converting the partitioning problem into a system of linear equations is not new. In fact, Kodres' work [45, 44] specifies two subsets of nodes as being in different partitions. This specification converts the "minimal cut" minimization of (2.9) into a tractable problem with nontrivial (i.e., not the all-zeros) solution. The solution to the system of equations obtained by Kodres is also a real-valued solution that must be converted into a "hard" partitioning with a threshold. Similarly, the GORDIAN algorithm [42] imposes constraints on multiple nodes to obtain a linear system and is later modified [62] to include a modified cost function (weighted by a linear factor). These approaches were combined in the PARABOLI method [59], with the addition that the fixed nodes are those that obtain extremal

values from the solution to the eigenvector problem. However, as indicated by Hall [35], it is often unclear how to designate the nodes that will belong to different partitions. In contrast, our approach assigns only a single node to a partition, which would eventually be assigned to a partition anyway. In general, the multiple-node constraint (MNC) approach differs from the present approach in the following ways:

1. An MNC approach designates some set of nodes to be in opposite partitions, a priori, instead of allowing that decision to be dictated by the graph structure.
2. The isoperimetric algorithm permits one of the partitions (that which does not contain the ground) to be disconnected, while an MNC approach forces both partitions to be connected (for two constrained nodes).
3. On a regular lattice, the isoperimetric algorithm naturally imitates the known solution to the isoperimetric problem on the plane (see Figure 6.1). However, an MNC approach requires a contrived constraint placement to achieve the same.

The *skewed graph partitioning* approach pursued in [38] permits specification of affinities for each node to belong to a particular partition. Viewed in the context of skewed graph partitioning, one might interpret the isoperimetric algorithm as asking the question: given definite knowledge of the partition membership of a single node (i.e., the ground is in set S), compute a good partitioning in which the remaining nodes are biased to be in the opposite partition (i.e., in $\overline{S}$). Under this interpretation, each remaining node is given either a unity bias to belong to $\overline{S}$ or a bias equal to the node degree, depending on the notion of volume employed. Viewed in this context, the approach taken by the isoperimetric algorithm appears reasonable—since the grounded node must fall into one of the two partitions, the problem is to determine which of the remaining nodes fall into the opposite partition.

Finally, we note that grounding node v_i and performing the matrix inversion of (2.15) may also be interpreted algebraically as computing the ith term of the generalized inverse of the Laplacian, termed the *Eichinger matrix* by Kunz [46].

5.2. Spectral partitioning. Building on the early work of Fiedler [24, 25, 23], Alon [3, 2], and Cheeger [15] who demonstrated the relationship between the second smallest eigenvalue of the Laplacian matrix (the *Fiedler value*) for a graph and its isoperimetric number, spectral partitioning was one of the first graph partitioning algorithms to be successful [35, 19, 57]. The algorithm partitions a graph by finding the eigenvector corresponding to the Fiedler value, termed the *Fiedler vector*, and cutting the graph based on the value in the Fiedler vector associated with each node. A physical analogy for the Fiedler vector is the second harmonic of a vibrating surface. Like isoperimetric partitioning, the output of spectral partitioning is a set of values assigned to each node, which allows a cut to be a perfect bisection by choosing a zero threshold (the median cut) or by choosing the threshold that generates a partition with the best isoperimetric ratio (the criterion cut). The flexibility of spectral partitioning allows it to be used as a part of hybridized graph partitioning algorithms, such as geometric-spectral partitioning [14] and multilevel approaches [37, 12].

Spectral partitioning attempts to minimize the isoperimetric ratio of a partition by solving

$$Lz = \lambda z, \tag{5.1}$$

with L defined as above and λ representing the Fiedler value. Since the vector of all ones, r, is an eigenvector corresponding to the smallest eigenvalue (zero) of L, the goal is to find the eigenvector associated with the second smallest eigenvalue of L.

Requiring $z^T r = 0$ and $z^T z = n$ may be viewed as additional constraints employed in the derivation of spectral partitioning to circumvent the singularity of L (for an explicit formulation of spectral partitioning from this viewpoint, see [39]). Therefore, one way of viewing the difference between the isoperimetric and the spectral methods is in the choice of the additional constraint that regularizes the singular nature of the Laplacian L.

In the context of spectral partitioning, the indicator vector z is usually defined as

$$z_i = \begin{cases} -1 & \text{if } v_i \in \overline{S}, \\ +1 & \text{if } v_i \in S, \end{cases} \tag{5.2}$$

such that z is orthogonal to r for an equal-sized partition. The two definitions of the indicator vector ((2.6) and (5.2)) are related through $x = \frac{1}{2}(z + r)$. Since r is in the nullspace of L, these definitions are equivalent up to a scaling.

Simple, unhybridized, unilevel spectral partitioning lags behind modern multilevel algorithms (e.g., [37]) in terms of partition quality. However, compared with other global graph partitioning algorithms, spectral partitioning still performs well. Furthermore, spectral partitioning is one of the only approaches that does not require prior specification of node coordinates or the partition cardinalities, instead offering the flexibility to automatically choose the partition cardinalities that offer the best cut.

The major drawbacks of spectral partitioning are its speed and numerical stability. Even using the Lanczos algorithm [30] to find the Fiedler vector for a sparse matrix, spectral partitioning is still much slower than many other partitioning algorithms. Furthermore, the Lanczos algorithm becomes unstable as the Fiedler value approaches its neighboring eigenvalues (see [36, 30] for discussion of this problem). In fact, the eigenvector problem becomes fully degenerate if the Fiedler value assumes algebraic multiplicity greater than 1. For example, consider finding the Fiedler vector of a fully connected graph, for which the Fiedler value has algebraic multiplicity equal to $n-1$. This situation could allow the Lanczos algorithm to converge to any vector in the subspace spanned by the eigenvectors corresponding to the Fiedler value.

In Figure 5.1, we compare the speed of the isoperimetric and spectral algorithms. Both partitions were computed using MATLAB, where the sparse, Lanczos-based, method for finding eigenvectors employs ARPACK [47] and solution to the system of linear equations is computed using the sparse linear solvers detailed in [28]. These experiments were done on a machine with a 2.39 GHz Intel Xeon chip and 3 GB of RAM. In order to illustrate the convergence problems for graphs having a Fiedler value with algebraic multiplicity greater than 1, we compared the two algorithms on a simple, square, unweighted, 4-connected lattice, for which the algebraic multiplicity of the Fiedler value is known to be 2. Graphs were tested with a number of nodes between $9 \leq N \leq 50,000$. Each algorithm is exhibited to have nearly linear behavior, although the constant is about an order of magnitude less for the isoperimetric algorithm when the algebraic multiplicity of the Fiedler value is greater than 1. Figure 5.1 also shows the experiments run with a graph composed of points placed randomly in the two-dimensional (2D) unit square with a uniform distribution and connected via a Delaunay triangulation. Here we still see an improvement for the speed of the isoperimetric algorithm (with a speed increase of about three times), but the behavior of the spectral runtimes illustrates the degeneracy problem—for those random graphs in which the second and third smallest eigenvalues are equal, or nearly equal, the running time of the spectral algorithm jumps to the plot obtained for the lattice. In other words, our experiments indicate that the isoperimetric algorithm operates

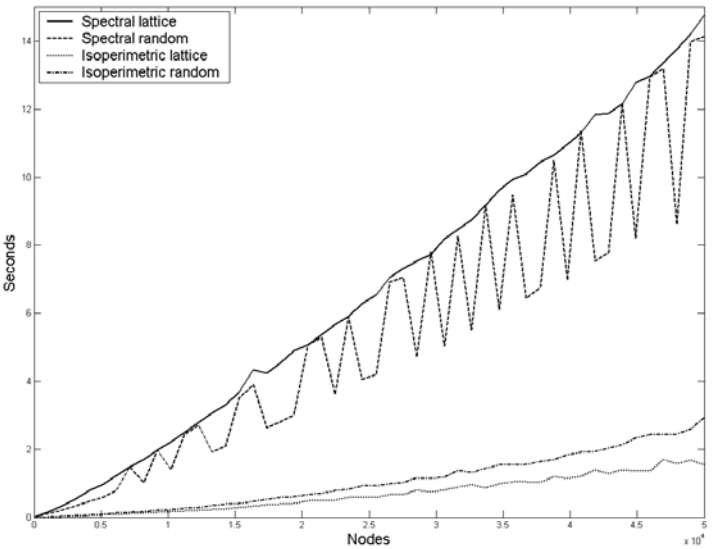


FIG. 5.1. *Speed comparison of the spectral method to the isoperimetric algorithm for two classes of graphs: a square, 4-connected lattice, and a randomly distributed, planar, point set connected via Delaunay. See the text for details.*

between three and ten times faster than the spectral algorithm, depending on the separation of the second and third eigenvalues. Additionally, the second experiment was run with just 30 randomly placed graphs with Delaunay connectivity. Therefore, the situations for which the degeneracy of the spectral approach is a concern do not appear to be rare. We note that the degeneracy problem of the spectral approach also precludes prediction of which runtime plot the spectral algorithm will employ (unless the spectrum is known a priori), while the isoperimetric algorithm has no such concerns. Finally, we note that the runtimes for the isoperimetric algorithm for each type of graph appear in Figure 5.1 to have a slightly different constant. We presume that this discrepancy is because the banded structure of the lattice Laplacian permits a simpler decomposition.

Finally, it has been pointed out [34] that a class of graphs exists for which spectral partitioning will produce consistently poor partitions.

5.3. Geometric partitioning. Geometric partitioning [29] is only defined for graphs with nodal coordinates specified (e.g., finite-element meshes). Furthermore, the geometric partitioning algorithm assumes that the nodes are locally connected and computes a good *spatial* separator (i.e., ignoring topology altogether). Building on theoretical results of [53, 52], geometric partitioning works by stereographically projecting nodes from the plane to the Riemann sphere, conformally mapping the nodes on the sphere such that a special point (called the *centerpoint*) is at the origin, and then randomly choosing any great circle on the Riemann sphere to divide the points into two equal halves. In practice, several great circles are randomly chosen and the best one is used as the output.

Although geometric partitioning is computationally inexpensive (albeit using multiple trials) and produces good partitions, the major drawback of the geometric partitioning algorithm is its inapplicability to graphs without coordinates or to graphs that are not locally connected (e.g., nonplanar graphs).

5.4. Geometric-spectral partitioning. Geometric-spectral partitioning [14] combines elements of both spectral partitioning and geometric partitioning. By finding the eigenvectors corresponding to both the second and third smallest eigenvalues, one may treat the nodes as having *spectral coordinates* by viewing the values associated with each node in the two eigenvectors as 2D coordinates in the plane. Applying geometric partitioning to the spectral coordinates of the nodes instead of actual coordinates (if available) heuristically gives better partitions than either spectral or geometric partitioning alone.

Geometric-spectral partitioning represents a more general algorithm than straightforward geometric partitioning since it applies to more general graphs (i.e., graphs without coordinates) and because it takes advantage of topological information in computing the spectral coordinates. However, geometric-spectral partitioning is more computationally expensive than spectral partitioning since it requires the computation of two eigenvectors instead of one. Numerically, the algorithm is also prone to more numerical problems than spectral partitioning, since the Lanczos algorithm has increased error as more "interior" eigenvectors are computed [30], and because the spectral coordinates are not unique if either the second or the third eigenvalue of L has an algebraic multiplicity greater than one.

5.5. Multilevel Kernighan–Lin. The Kernighan–Lin algorithm [41] employs a greedy approach for graph partitioning. Although fast, the final solution is highly dependent on the initial (random) partition used to speed the algorithm. Due to the speed, this algorithm may be run multiple times in order to achieve an acceptable partition.

By far the most successful use of the Kernighan–Lin algorithm is as a *refinement* technique. A multilevel representation of a graph may be obtained through the use of a maximal independent set [66]. If the coarsest level of the graph is partitioned through some means (e.g., spectral, geometric), then the quality of the partition may be refined to the next level by employing Kernighan–Lin. This multilevel Kernighan–Lin approach is the backbone for state-of-the-art graph partitioning software packages such as Chaco [36] and Metis [40]. We include the Metis [40] package in the comparison section.

One expects these algorithms to perform the best since they take into account multiple levels of graph representation. However, multilevel Kernighan–Lin may be easily combined with other graph partitioning techniques by employing the secondary technique to perform the partitioning at the coarsest level. Independent of the multilevel scheme, Kernighan–Lin may be used to refine a partition given by another algorithm.

One problem with the Kernighan–Lin approach is that the size of the partitions must be specified. This is not an issue for the graph bisection problem, where the partition size is required to be half of the node set. However, if one is interested in finding partitions close to the isoperimetric sets, then the algorithm must be flexible enough to find the best partitions of arbitrary size.

6. Algorithm behavior. As with spectral methods, the solution of (2.15) (or (3.7)) yields a continuous-valued result that must then be converted into a bipartition by thresholding. In this section we introduce two examples that illustrate the distribution of the potentials and their relationship to the ground node.

Figure 6.1 shows the solution to the spectral problem and (3.7) on a simple, square, 4-connected grid. Note that the square grid causes the degeneracy of the Fiedler vector required by the spectral approach, since the algebraic multiplicity of the Fiedler value is 2. Consequently, even for this simple graph, the Lanczos method

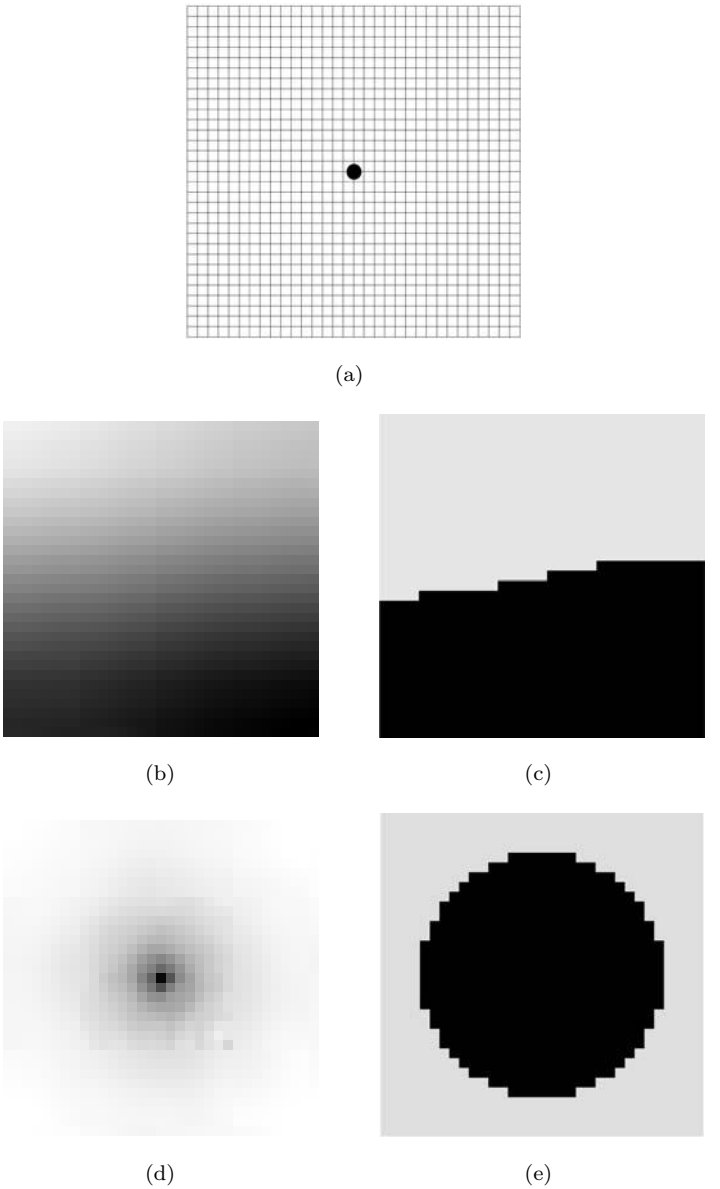


FIG. 6.1. *Comparison of spectral and isoperimetric solutions on a square, unweighted, 4-connected lattice.* (a) *Original lattice. The black dot represents the location of the ground point for the isoperimetric algorithm.* (b) *Spectral solution. Note that the algebraic multiplicity of the Fiedler vector results in a degenerate solution—any vector in the subspace spanned by the Fiedler vectors is a valid output of the Lanczos algorithm.* (c) *Cut obtained from the spectral solution.* (d) *Solution of the isoperimetric algorithm, given a ground point in the lattice center.* (e) *Cut obtained from the isoperimetric solution. Despite the Manhattan distance imposed by 4-connectivity, note the similarity with the well-known solution to the isoperimetric problem in the Euclidean plane—a circle.*

takes longer to converge and returns a vector in the subspace spanned by the multiple Fielder vectors (as seen in Figure 5.1). In contrast, given a ground node in the center of the lattice, the isoperimetric algorithm returns a unique solution. The fact that our algorithm returns a "circle" on this graph lends justification to the name

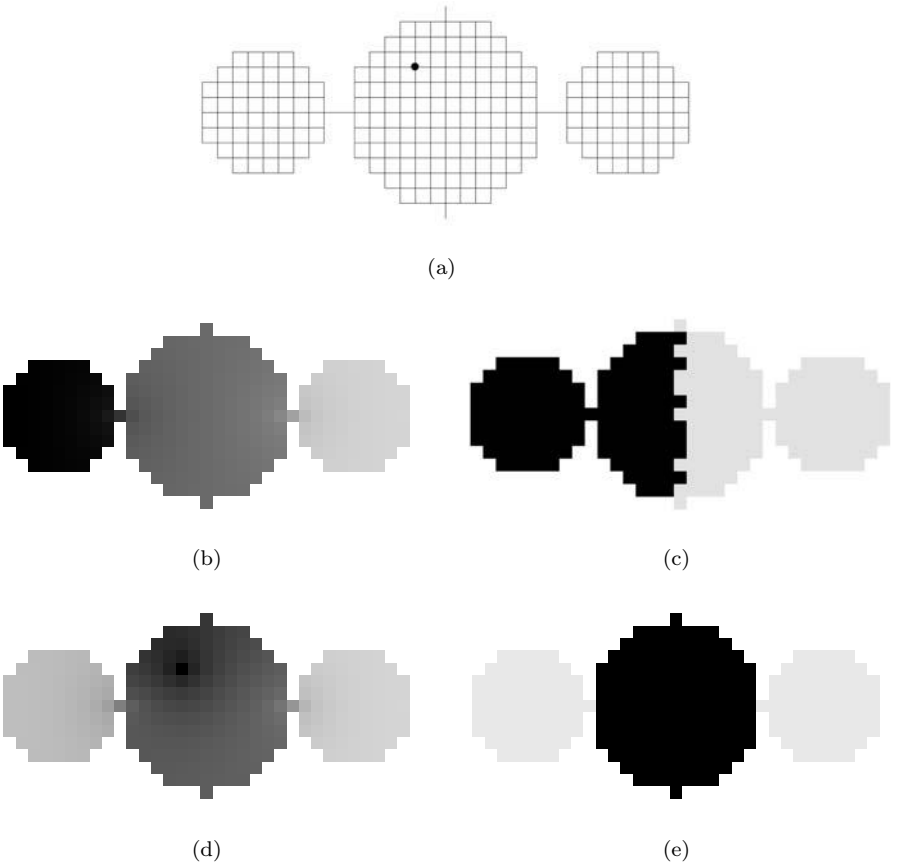


FIG. 6.2. *Comparison of spectral and isoperimetric solutions on a graph comprises a large, central, "circle" with smaller, side "circles." This graph is intended to build the reader's intuition about the solution.* (a) *Original graph.* (b) *Spectral. The symmetry of the graph results in a symmetric solution over the central axis of symmetry.* (c) *Cut obtained from the spectral solution. Since the spectral solution is symmetric over the central axis of symmetry, the equal partition cut must also lie along this axis. For graphs of this type, such a solution may be arbitrarily poor.* (d) *Solution of the isoperimetric algorithm, given a ground point in the central circle, designated by the black dot. Note that the ground is offset from the circle center.* (e) *Cut obtained from the isoperimetric solution. Note that the cut is not simply a circle (relative to the graph distance), as otherwise the partition would tilt toward the side circle closer to the ground, which it does not.*

"isoperimetric" algorithm, since the circle has been known since antiquity to be the solution to the isoperimetric problem in the Euclidean plane [56] and a lattice is widely used as a discrete approximation to the plane when solving PDEs, etc. This appearance of the circle should be no surprise, given the motivation of the algorithm or the random walks interpretation, since it is well known in the Euclidean plane that one is expected to stand on a circle with radius $\sqrt{n}$ after taking n random steps from a central point [49] (given here by the grounded node). However, we stress that the "circle" partition in Figure 6.1 is *not* simply the locus of points within a certain distance from the ground, since the 4-connectivity of the lattice would induce a diamond-shaped locus.

Figure 6.2 shows another simple graph, constructed by windowing a 4-connected lattice in the shape of a large, central "circle," weakly connected to two side "circles,"

such that the cardinality of the central circle equals the sum of the side circles. The symmetry of this graph across an unfavorable cut axis (i.e., the center of the large circle) leads to problems for the spectral algorithm, as well illustrated in Guattery and Miller's "roach" graph [34]. In contrast, the isoperimetric algorithm cuts the side circles off the central circle, even though the ground is offset from the center of the circle. Again, this should be no surprise given the electrical circuit analogy—all of the "current" entering nodes in the side circles must flow through the connecting branch, yielding a huge voltage drop across this edge. Consequently, the center circle is widely separated from the side circles. These experiments demonstrate that the "background" partition (i.e., $\overline{S}$) may or may not be connected. As with the above experiment, the partition is not simply the locus of all points equidistant from the ground. The ground point was offset from the center of the large circle in Figure 6.2, but the partition boundary does not "leak" into the closer side circle.

7. Comparison of algorithms. The graph partition quality produced by the algorithms reviewed above was examined on weighted and unweighted planar, nonplanar and three-dimensional (3D) graphs, as well as some specialized graphs. All algorithms were compared by considering the number of edges spanning the two equally sized partitions produced by each algorithm (i.e., the median cut, for the isoperimetric and spectral algorithms). The spectral and isoperimetric algorithms were additionally compared in terms of the isoperimetric ratio by employing the criterion cut to find the best partition (i.e., not necessarily partitions of equal size). Note that the "uniform" volume definition, i.e., (2.5), was used to evaluate the isoperimetric ratio for these algorithms, since this is the definition most commonly desired in graph partitioning problems. As was noted in [60] we found that the distribution of spanning edges and $h(x)$ resembled Gaussian distributions. For this reason, we report the mean and variance of the number of edges spanning the partitions (or $h(x)$) for the 1,000 graphs generated of each type.

Four versions of the isoperimetric algorithm are compared with the algorithms reviewed above on the graph partitioning problem. The four versions are obtained from the two notions of volume corresponding to (2.15) and (3.7) and two grounding strategies: grounding the maximum degree node and grounding a random node. In the following tables, grounding strategies are denoted by "MG" for maximum degree ground and "RG" for random ground. Volume definitions are designed by "Uni" for the cardinality definition of volume of (2.15) or "Deg" for the degree-based definition of volume given by solution to (3.7). For each random ground partition, three randomly chosen grounds were tried and the best one chosen. The geometric, and geometric-spectral, partitions were all obtained through the MESHPART software package for MATLAB [29] and Metis was executed using the Metis package of [40].

Relative runtimes of the algorithms were considered for a 10,000-node, randomly connected, planar graph of the type described in section 7.1. From fastest to slowest were Metis (0.031 s), Isoperimetric (0.329 s), Geometric (0.500 s), Spectral (2.125 s), and Geo-Spectral (6.531 s). However, as was mentioned above, the code to execute each of these algorithms was written by different groups, with varying dependence on MATLAB.

7.1. Planar graphs. Planar graphs were generated by uniformly sampling 1,000 points from a 2D unit square and connecting them with a Delaunay triangulation. One thousand such graphs were generated for both the weighted and unweighted trials. In the weighted trial, weights were randomly assigned to each edge from a uniform distribution on the interval $[0, 1 \times 10^4]$. Results for the median cut comparison are

TABLE 7.1

Comparison of the algorithms on 1,000 *randomly generated planar graphs produced by uniformly sampling* 1,000 2D *points in the unit square and connecting via a Delaunay triangulation* (*see section* 7.1). *The two leftmost columns represent the mean and variance of* $|\partial S|$ *for equal-sized partitions produced by each algorithm when applied to unweighted planar graphs. The two rightmost columns represent the same quantities for weighted planar graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean cut	Variance cut	Mean cut	Variance cut
Iso MG Uni	80.8	116	3.76×10^5	3.69×10^9
Iso MG Deg	80.7	117	3.75×10^5	3.61×10^9
Iso RG Uni	77.9	44.9	3.51×10^5	1.5×10^9
Iso RG Deg	78.1	43	3.5×10^5	1.27×10^9
Spectral	77.1	57.7	3.47×10^5	1.79×10^9
Geometric	70.6	12.1	3.54×10^5	7.94×10^8
Spectral geometric	66.8	9.81	3.34×10^5	7.97×10^8
Metis	66.9	23.5	3.35×10^5	1.15×10^9

TABLE 7.2

Comparison of the isoperimetric and spectral algorithms on 1,000 *randomly generated planar graphs produced by uniformly sampling* 1,000 2D *points in the unit square and connecting via a Delaunay triangulation* (*see section* 7.1). *The two leftmost columns represent the mean and variance of* $h(x)$ *obtained by applying the criterion cut to the output of each algorithm when applied to unweighted planar graphs. The two rightmost columns represent the same quantities for weighted planar graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Iso MG Uni	0.154	0.0004	683	1.15×10^4
Iso MG Deg	0.154	0.000398	683	1.17×10^4
Iso RG Uni	0.148	0.000131	635	3.75×10^3
Iso RG Deg	0.148	0.000129	636	3.7×10^3
Spectral	0.147	0.000201	625	4.65×10^3

found in Table 7.1 and for the criterion cut comparison are found in Table 7.2.

7.2. Nonplanar graphs. Purely random graphs (typically nonplanar) were generated by connecting each pair of 1,000 nodes with a 1% probability. One thousand such graphs were generated for both the weighted and unweighted trials. In the weighted trial, weights were randomly assigned to each edge from a uniform distribution on the interval $[0, 1 \times 10^4]$. Since coordinate information was meaningless, only those algorithms which made use of topological information were compared (i.e., the isoperimetric, spectral, geometric-spectral, and Metis approaches). Results for the median cut comparison are found in Table 7.3 and for the criterion cut comparison in Table 7.4.

7.3. 3D graphs. Modeling and computer graphics applications frequently require the use of locally connected points in three dimensions. One thousand graphs were generated by uniformly sampling 1,000 points in the unit cube and connecting them via the 3D Delaunay. Both weighted and unweighted graphs were generated. In the weighted trial, weights were randomly assigned to each edge from a uniform distribution on the interval $[0, 1 \times 10^4]$. Results for the median cut comparison are found in Table 7.5 and for the criterion cut comparison in Table 7.6.

TABLE 7.3

Comparison of the algorithms on 1,000 *random graphs* (*symmetric*) *produced by connecting each edge with a* 1% *probability* (*see section* 7.2). *Partitioning algorithms that rely on coordinate information were not included, since they are not applicable to this problem. The two leftmost columns represent the mean and variance of* $|\partial S|$ *for an equal-sized bipartition produced by each algorithm for unweighted random graphs. The two rightmost columns represent the same quantities for weighted random graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean cut	Variance cut	Mean cut	Variance cut
Iso MG Uni	4.47×10^3	7.06×10^3	2.14×10^7	1.71×10^{11}
Iso MG Deg	4.46×10^3	5.62×10^3	2.1×10^7	1.83×10^{11}
Iso RG Uni	4.14×10^3	5.75×10^3	2.11×10^7	1.32×10^{11}
Iso RG Deg	4.11×10^3	6.46×10^3	2.06×10^7	1.41×10^{11}
Spectral	4.06×10^3	5.99×10^3	2.05×10^7	6.62×10^{11}
Spectral geometric	3.98×10^3	3.85×10^3	1.99×10^7	1.54×10^{11}
Metis	3.43×10^3	2.4×10^3	1.72×10^7	1.03×10^{11}

TABLE 7.4

Comparison of the algorithms on 1,000 *random graphs* (*symmetric*) *produced by connecting each edge with a* 1% *probability* (*see section* 7.2). *The two leftmost columns represent the mean and variance of* $h(x)$ *for a partition generated using the criterion cut for random graphs. The two rightmost columns represent the same quantities for weighted random graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Iso MG Uni	8.84	0.104	4.09×10^4	3.86×10^6
Iso MG Deg	8.92	0.0225	4.22×10^4	9.93×10^5
Iso RG Uni	8.22	0.0486	3.91×10^4	7.73×10^6
Iso RG Deg	8.19	0.0207	4.1×10^4	5.87×10^5
Spectral	8.12	0.0244	3.98×10^4	2.26×10^6

TABLE 7.5

Comparison of the algorithms on 1,000 *randomly generated* 3D *produced by uniformly sampling* 1,000 3D *points in the unit cube and connecting via a* 3D *Delaunay* (*see section* 7.3). *The two leftmost columns represent the mean and variance of* $|\partial S|$ *for an equal-sized bipartition produced by each algorithm for unweighted planar graphs. The two rightmost columns represent the same quantities for weighted planar graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean cut	Variance cut	Mean cut	Variance cut
Iso MG Uni	630	2.14×10^3	3.2×10^6	7.61×10^{10}
Iso MG Deg	628	2.12×10^3	3.17×10^6	7.39×10^{10}
Iso RG Uni	633	1.17×10^3	3.17×10^6	3.27×10^{10}
Iso RG Deg	632	1.16×10^3	3.14×10^6	3.24×10^{10}
Spectral	609	1.31×10^3	3.37×10^6	9.05×10^{11}
Geometric	579	348	2.9×10^6	1.38×10^{10}
Spectral geometric	561	515	2.81×10^6	1.96×10^{10}
Metis	551	614	2.75×10^6	2.16×10^{10}

7.4. Specialized graphs. In addition to the randomly generated graphs used above to benchmark the isoperimetric algorithm, we also applied the set of algorithms to 2D graphs taken from applications (see [29, 14, 13] for other uses of these graphs).

TABLE 7.6

Comparison of the isoperimetric and spectral algorithms on 1,000 *randomly generated* 3D *graphs produced by uniformly sampling* 1,000 3D *points in the unit cube and connecting via a* 3D *Delaunay* (*see section* 7.3). *The two leftmost columns represent the mean and variance of* $h(x)$ *obtained using the criterion cut for each algorithm on unweighted planar graphs. The two rightmost columns represent the same quantities for weighted planar graphs.*

	Unweighted graphs		Weighted graphs	
Algorithm	Mean $h(x)$	Variance $h(x)$	Mean $h(x)$	Variance $h(x)$
Iso MG Uni	1.26	0.0086	6.39×10^3	4.36×10^5
Iso MG Deg	1.25	0.00857	6.54×10^3	4.29×10^5
Iso RG Uni	1.25	0.00413	6.1×10^3	2.01×10^5
Iso RG Deg	1.25	0.00413	6.29×10^3	2.03×10^5
Spectral	1.22	0.00527	7.13×10^3	5.77×10^6

TABLE 7.7

Information about the specialized graphs used to benchmark the algorithms on. All graphs were obtained from the ftp site of John Gilbert and the Xerox Corporation at ftp.parc.xerox.com in the file /pub/gilbert/meshpart.uu.

Mesh Name	Nodes	Edges
Eppstein	547	1556
Tapir	1024	2846
"Crack"	136	354
Airfoil1	4253	12289
Airfoil2	4720	13722
Airfoil3	15606	45878
Spiral	1200	3191
Triangle	5050	14850

TABLE 7.8

The number of edges cut by the equal-sized partitions output by the various algorithms (*see section* 7.4). *Note that since the Eppstein mesh is weighted, the cut cost is not an integer.*

Algorithm	Epp	Tapir	Tri	Air1	Air2	Air3	Crack	Spiral
Iso MG Uni	24.8	51	150	137	111	255	200	9
Iso MG Deg	24.8	51	150	137	111	255	197	9
Iso RG Uni	20.2	33	158	105	123	188	152	9
Iso RG Deg	20.4	40	150	104	125	212	161	9
Spectral	21.6	58	152	132	117	194	157	9
Geometric	22.9	34	148	102	103	152	191	58
Geometric-spectral	22.4	23	144	88	104	159	149	9
Metis	27.3	24	144	85	96	141	142	9

The meshes were obtained through the ftp site of John Gilbert and the Xerox Corporation at ftp.parc.xerox.com from the file /pub/gilbert/meshpart.uu. The list of graphs used is given in Table 7.7. The results using the median cut comparison are found in Table 7.8 and for the criterion cut comparison are found in Table 7.9.

The set of algorithms were also applied to three graph families of theoretical interest. The first of these is the "roach" graph of [34] with the total length of the roach ranging from 10 to 50 nodes long (i.e., 20–100 nodes total). The family of roach graphs is known to result in poor partitions when spectral partitioning is employed with the median cut. For a roach with an equal number of "body" and "antennae" segments, the spectral algorithm will always produce a partition with $|\partial S| = \Theta(n)$ (where $\Theta(\cdot)$ is the function of [43]) instead of the constant cut set of two edges obtained

TABLE 7.9

The isoperimetric ratio obtained by applying the criterion cut method to the output of the isoperimetric and spectral partitioning algorithms (see section 7.4).

Algorithm	Epp	Tapir	Tri	Air1	Air2	Air3	Crack	Spiral
Iso MG Uni	0.0906	0.0474	0.0588	0.0351	0.0425	0.0268	0.0758	0.0108
Iso MG Deg	0.0894	0.0473	0.0587	0.0351	0.0425	0.0268	0.0759	0.0108
Iso RG Uni	0.0714	0.0435	0.0594	0.0329	0.0434	0.0212	0.057	0.0108
Iso RG Deg	0.0839	0.0286	0.0643	0.0329	0.043	0.0209	0.0572	0.0108
Spectral	0.0771	0.0262	0.0594	0.0342	0.0425	0.02	0.059	0.0108

TABLE 7.10

The mean number of edges cut by the equal-sized partitions output by the various algorithms over a parameter range for each family of graphs (see section 7.4). The three graph families here (roach, TreeXPath, and "badmesh") are of theoretical interest in that they are known to produce poor results for different classes of partitioning algorithms.

Algorithm	Mean cut roach	Mean cut TreeXPath	Mean cut "badmesh"
Iso MG Uni	2	27.1	2.98
Iso MG Deg	2	26.7	2.98
Iso RG Uni	2.37	29.3	3.34
Iso RG Deg	2.32	27.8	3.1
Spectral	15.2	38	12.5
Geometric	2	26.4	453
Spectral geometric	2	27	2.98
Metis	2	27.5	2.98

TABLE 7.11

The mean isoperimetric ratio obtained using a criterion cut on the output of the partitioning algorithms when applied to three graph families of theoretical interest (see section 7.4). Means are calculated over a range of parameters defining the graph family (see the text for details).

Algorithm	Mean $h(x)$ roach	Mean $h(x)$ TreeXPath	Mean $h(x)$ "badmesh"
Iso MG Uni	0.0815	0.128	0.0297
Iso MG Deg	0.0815	0.128	0.0297
Iso RG Uni	0.0798	0.129	0.0294
Iso RG Deg	0.0798	0.127	0.0294
Spectral	0.0796	0.128	0.0294

by cutting the antennae from the body. It has been demonstrated [64] that the spectral approach may be made to correctly partition the roach graph if additional processing is performed. For this reason, the partitions obtained through use of the criterion cut are reasonable for the spectral algorithm. The second graph family of theoretical interest, referred to as "TreeXPath" [34], is known to result in poor partitions when spectral partitioning is used with the median cut. For purposes of benchmarking here, the cardinality of the 3D point set was varied between 50 and 1,000 nodes. The final graph family of theoretical interest is the so-called "badmesh" [13] for which no quality straight-line separator exists. Badmeshes were generated with node sets varying between 200 and 4,000, with a constant ratio of $\frac{4}{5}$ between shell sizes. Mean values across the nodal range are reported in Table 7.10 for partitions obtained with the median cut and Table 7.11 using the criterion cut.

8. Conclusion. We have presented a new algorithm for partitioning graphs that requires the solution to a sparse, symmetric, positive-definite system of equations.

Analogous to Fiedler's examination of the connectivity behavior of the spectral algorithm [25], we examined the connectivity of the partitions returned by the isoperimetric algorithm and proved that the partition containing the ground node must be connected. Formal behavior of the algorithm was also examined on fully connected graphs and trees. Empirically, the solution to the system of equations was examined on two graphs and interpreted in the context of two physical interpretations of the equations. The partition returned by application of the isoperimetric algorithm to a 4-connected, 2D lattice was shown to resemble the known solution to the isoperimetric problem on the 2D Euclidean plane—a circle.

Based on the above trials, it appears that a random ground approach generally works better than grounding the node with maximum degree. Therefore, we suggest the strategy of grounding the node with maximum degree and, if possible, additionally trying random grounds in order to choose the best partition. Although the difference in partition quality between the two definitions of combinatorial volume is slight, it seems the definition of (3.6) fared better over that of (2.5) for the equally sized partition, yet favored the cardinality-based volume measure in computing the partition with minimum isoperimetric ratio. However, the latter should not be surprising, since the isoperimetric ratio was computed using the cardinality-based volume measure. In practice, neither notion of volume suggests recommendation over the other for an equally sized cut. However, if trying to compute the partition with minimum isoperimetric ratio (i.e., the quotient cut), one should employ the appropriate notion of volume.

While competitive with the algorithms compared in this work, we acknowledge that the isoperimetric algorithm appears to give slightly higher averages than the other algorithms, most notably the multilevel KL approach. However, the isoperimetric algorithm has the following qualities not shared by all of the comparison algorithms:

1. Coordinate information is not needed, allowing application on abstract graphs.
2. Edge weights are taken into account when computing the partitions.
3. The algorithm returns a family of partitions (corresponding to different threshold choices) that allows a partition with the "best" cardinality to be chosen, instead of prespecifying the partition cardinality.

In fact, the only algorithm included here that shares these three properties is spectral partitioning, which is shown above to produce partitions of similar quality, but more slowly and at the risk of a degenerate solution. Furthermore, the isoperimetric algorithm appears to perform better (relative to the other algorithms) on the specific benchmark graphs than on the experimental random graphs. Therefore, it is possible that the family of random graphs used in the experiments may be unfavorable to the algorithm. In contrast, the experiments shown on the "roach," "TreeXPath," and "badmesh" graph families, which were designed to be problematic for other algorithms, showed little problem for the isoperimetric algorithm. We also note that some of the better-performing algorithms (e.g., spectral geometric) use the spectral approach as a component. Therefore, hybridization with the isoperimetric algorithm instead of the spectral approach might improve speed/stability issues associated with the spectral component of such an approach.

The isoperimetric algorithm for graph partitioning is implemented in the publicly available graph analysis toolbox [31, 32] for MATLAB obtainable at http://eslab.bu.edu/software/graphanalysis/. This toolbox was used to generate all of the results compiled in the tables of this paper. Further work with this algorithm might include

the development of a more principled grounding strategy and hybridization with other partitioning methods.

Acknowledgment. The authors would like to thank Jonathan Polimeni for many fruitful discussions and suggestions.

REFERENCES

[1] L. Ahlfors, *Complex Analysis*, McGraw-Hill, New York, 1966.
[2] N. Alon, *Eigenvalues and expanders*, Combinatorica, 6 (1986), pp. 83–96.
[3] N. Alon and V. Milman, λ_1, *isoperimetric inequalities for graphs and superconcentrators*, J. Combin. Theory Ser. B, 38 (1985), pp. 73–88.
[4] C. J. Alpert and A. B. Kahng, *Recent directions in netlist partitioning: A survey*, Integration: The VLSI J., 19 (1995), pp. 1–81.
[5] W. N. Anderson, Jr. and T. D. Morley, *Eigenvalues of the Laplacian of a graph*, Linear Multilinear Algebra, 18 (1985), pp. 141–145.
[6] G. Arfken and H.-J. Weber, eds., *Mathematical Methods for Physicists*, 3rd ed., Academic Press, New York, 1985.
[7] D. A. V. Baak, *Variational alternatives of Kirchhoff's loop theorem in DC circuits*, Amer. J. Phys., 67 (1998), pp. 36–44.
[8] N. Biggs, *Algebraic Graph Theory*, Cambridge Tracts in Math. 67, Cambridge University Press, Cambridge, UK, 1974.
[9] J. Bolz, I. Farmer, E. Grinspun, and P. Schröder, *Sparse matrix solvers on the GPU: Conjugate gradients and multigrid*, ACM Trans. Graphics, 22 (2003), pp. 917–924.
[10] F. H. Branin, Jr., *The inverse of the incidence matrix of a tree and the formulation of the algebraic-first-order differential equations of an RLC network*, IEEE Trans. Circuit Theory, 10 (1963), pp. 543–544.
[11] F. H. Branin, Jr., *The algebraic-topological basis for network analogies and the vector calculus*, in Proceedings of the Symposium on Generalized Networks, J. Fox, ed., Polytechnic Press, Brooklyn, NY, 1966, pp. 453–491.
[12] T. N. Bui and C. Jones, *A heuristic for reducing fill-in in sparse matrix factorization*, in Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Vol. 1, R. F. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, eds., SIAM, Philadelphia, 1993, pp. 445–452.
[13] F. Cao, J. R. Gilbert, and S.-H. Teng, *Partitioning Meshes with Lines and Planes*, Technical report CSL-96-01, Palo Alto Research Center, Xerox Corporation, 1996.
[14] T. F. Chan, J. R. Gilbert, and S.-H. Teng, *Geometric Spectral Partitioning*, Technical report CSL-94-15, Palo Alto Research Center, Xerox Corporation, 1994.
[15] J. Cheeger, *A lower bound for the smallest eigenvalue of the Laplacian*, in Problems in Analysis, R. Gunning, ed., Princeton University Press, Princeton, NJ, 1970, pp. 195–199.
[16] F. R. K. Chung, *Spectral Graph Theory*, CBMS Reg. Conf. Ser. Math. 92, AMS, Providence, RI, 1997.
[17] J. Dodziuk, *Difference equations, isoperimetric inequality and the transience of certain random walks*, Trans. Am. Math. Soc., 284 (1984), pp. 787–794.
[18] J. Dodziuk and W. S. Kendall, *Combinatorial Laplacians and the isoperimetric inequality*, in From Local Times to Global Geometry, Control and Physics, Pitman Res. Notes Math. Ser. 150, K. D. Ellworthy, ed., Longman Scientific and Technical, London, 1986, pp. 68–74.
[19] W. Donath and A. Hoffman, *Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices*, IBM Tech. Disclosure Bull., 15 (1972), pp. 938–944.
[20] J. J. Dongarra, I. S. Duff, D. C. Sorenson, and H. A. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, 1991.
[21] P. Doyle and L. Snell, *Random Walks and Electric Networks*, Carus Math. Monogr. 22, Mathematical Association of America, Washington, D.C., 1984.
[22] C. M. Fiduccia and R. M. Mattheyes, *A linear-time heuristic for improving network partitions*, in Proceedings of the 19th ACM/IEEE Design Automation Conference, Las Vegas, 1982, IEEE Press, Piscataway, NJ, pp. 175–181.
[23] M. Fiedler, *Algebraic connectivity of graphs*, Czechoslovak. Math. J., 23 (1973), pp. 298–305.
[24] M. Fiedler, *Eigenvectors of acyclic matrices*, Czechoslovak. Math. J., 25 (1975), pp. 607–618.
[25] M. Fiedler, *A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory*, Czechoslovak. Math. J., 25 (1975), pp. 619–633.

[26] M. Fiedler, *Special Matrices and their Applications in Numerical Mathematics*, Martinus Nijhoff Publishers, Dordrecht, The Netherlands, 1986.
[27] L. Foulds, *Graph Theory Applications*, Universitext, Springer-Verlag, New York, 1992.
[28] J. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.
[29] J. R. Gilbert, G. L. Miller, and S.-H. Teng, *Geometric mesh partitioning: Implementation and experiments*, SIAM J. Sci. Comput., 19 (1998), pp. 2091–2110.
[30] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
[31] L. Grady, *Space-Variant Computer Vision: A Graph-Theoretic Approach*, Ph.D. thesis, Boston University, Boston, MA, 2004.
[32] L. Grady and E. L. Schwartz, *The Graph Analysis Toolbox: Image Processing on Arbitrary Graphs*, Technical report TR-03-021, Boston University, Boston, MA, 2003.
[33] K. Gremban, *Combinatorial Preconditioners for Sparse, Symmetric Diagonally Dominant Linear Systems*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1996.
[34] S. Guattery and G. Miller, *On the quality of spectral separators*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 701–719.
[35] K. M. Hall, *An r-dimensional quadratic placement algorithm*, Manag. Sci., 17 (1970), pp. 219–229.
[36] B. Hendrickson and R. Leland, *The Chaco User's Guide*, Technical report SAND95-2344, Sandia National Laboratory, Albuquerque, NM, 1995.
[37] B. Hendrickson and R. Leland, *A multilevel algorithm for partitioning graphs*, in Proceedings of the 1995 ACM/IEEE Conference on Supercomputing, San Diego, CA, S. Karin, ed., ACM Press, New York, 1995.
[38] B. Hendrickson, R. Leland, and R. van Driessche, *Skewed graph partitioning*, in Proceedings of the Eighth SIAM Conference on Parallel Processing, M. Heath, V. Torczon, G. Astfalk, P. E. Bjorstad, A. H. Karp, C. H. Koebel, V. Kumar, R. F. Lucas, L. T. Watson, and D. E. Womble, eds., SIAM, Philadelphia, 1997.
[39] Y. Hu and R. Blake, *Numerical experiences with partitioning of unstructured meshes*, Parallel Comput., 20 (1994), pp. 815–829.
[40] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–393.
[41] B. Kernighan and S. Lin, *An efficient heuristic procedure for partitioning graphs*, Bell Syst. Tech. J., 49 (1970), pp. 291–308.
[42] J. M. Kleinhaus, G. Sigl, F. M. Johannes, and K. J. Antreich, *GORDIAN: VLSI placement by quadratic programming and slicing optimization*, IEEE Trans. Comput. Aided Design, 10 (1991), pp. 356–365.
[43] D. E. Knuth, *Big omicron and big omega and big theta*, SIGACT News, 8 (1976), pp. 18–24.
[44] U. R. Kodres, *Geometrical positioning of circuit elements in a computer*, in Proceedings of the 1959 AIEE Fall General Meeting, CP59-1172, Chicago, IL, AIEE Press, New York, 1959.
[45] U. R. Kodres, *Partitioning and card selection*, in Design Automation of Digital Systems, Vol. 1, M. A. Breuer, ed., Prentice-Hall, Englewood Cliffs, NJ, 1972, Chap. 4, pp. 173–212.
[46] M. Kunz, *A Möbius inversion of the Ulam subgraphs conjecture*, J. Math. Chem., 9 (1992), pp. 297–305.
[47] R. B. Lehoucq, D. C. Sorenson, and C. Yang, *ARPACK User's Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, 1998.
[48] T. Leighton and S. Rao, *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*, J. ACM, 46 (1999), pp. 787–832.
[49] J. D. Logan, *An introduction to nonlinear partial differential equations*, Pure Appl. Math. (N. Y.), John Wiley, New York, 1994.
[50] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, 3rd ed., Vol. 1, Dover, New York, 1991.
[51] R. Merris, *Laplacian matrices of graphs: A survey*, Linear Algebra Appl., 197–198 (1994), pp. 143–176.
[52] G. L. Miller, S.-H. Teng, W. P. Thurston, and S. A. Vavasis, *Automatic mesh partitioning*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., IMA Vol. Math. Appl. 56, Springer-Verlag, Berlin, 1993, pp. 57–84.
[53] G. L. Miller, S.-H. Teng, W. P. Thurston, and S. A. Vavasis, *Geometric separators for finite-element meshes*, SIAM J. Sci. Comput., 19 (1998), pp. 364–386.
[54] B. Mohar, *Isoperimetric inequalities, growth and the spectrum of graphs*, Linear Algebra Appl., 103 (1988), pp. 119–131.

[55] B. Mohar, *Isoperimetric numbers of graphs*, J. Combin. Theory Ser. B, 47 (1989), pp. 274–291.
[56] G. Pólya and G. Szegö, *Isoperimetric Inequalities in Mathematical Physics*, Ann. of Math. Stud. 56, Princeton University Press, Princeton, NJ, 1951.
[57] A. Pothen, H. Simon, and K.-P. Liou, *Partitioning sparse matrices with eigenvectors of graphs*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 430–452.
[58] A. Pothen, H. Simon, and L. Wang, *Spectral Nested Dissection*, Technical report CS-92-01, Pennsylvania State University, 1992.
[59] B. M. Riess, K. Doll, and F. M. Johannes, *Partitioning very large circuits using analytical placement techniques*, in Proceedings of the 31st ACM/IEEE Design Automation Conference, IEEE Press, Piscataway, NJ, 1994, pp. 646–651.
[60] G. R. Schreiber and O. C. Martin, *Cut size statistics of graph bisection heuristics*, SIAM J. Optim., 10 (1999), pp. 231–251.
[61] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Trans. Pattern Anal. Mach. Intelligence, 22 (2000), pp. 888–905.
[62] G. Sigl, K. Doll, and F. M. Johannes, *Analytical placement: A linear of a quadratic objective function?*, in Proceedings of the 28th ACM/IEEE Design Automation Conference, IEEE Press, Piscataway, NJ, 1991, pp. 427–432.
[63] H. D. Simon, *Partitioning of unstructured problems for parallel processing*, Comput. Syst. Engrg., 2 (1991), pp. 135–148.
[64] D. A. Spielman and S.-H. Teng, *Spectral Partitioning Works: Planar Graphs and Finite Element Meshes*, Technical report UCB CSD-96-898, University of California, Berkeley, 1996.
[65] G. Strang, *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, Wellesley, MA, 1986.
[66] S.-H. Teng, *Coarsening, Sampling and Smoothing: Elements of the Multilevel Method*, manuscript, 1997.
[67] P. Tetali, *Random walks and the effective resistance of networks*, J. Theor. Probab., 4 (1991), pp. 101–109.
[68] Z. Wu and R. Leahy, *An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation*, IEEE Pattern Anal. Mach. Intelligence, 11 (1993), pp. 1101–1113.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1867–1880
© 2006 Society for Industrial and Applied Mathematics

A BLOCK PRECONDITIONER FOR HIGH-ORDER MIXED FINITE ELEMENT APPROXIMATIONS TO THE NAVIER–STOKES EQUATIONS*

DAVID KAY† AND EMIL LUNGU‡

Abstract. An iterative solver for the linear system arising from a discretization by high-order mixed finite elements for the linearized steady state Navier–Stokes problem is presented. Two mixed finite elements are considered: $\mathbf{P}_{k+1}P_k$ and $\mathbf{P}_{k+2}P_k$. The iterative solver is based on the GMRES method with right preconditioning where the preconditioner has a block upper triangular form. Computational results show that the convergence rate seems to be independent of the grid size parameter h and the polynomial degree k but is slightly affected by the increasing of the Reynolds number.

Key words. Navier–Stokes, high-order mixed finite elements, preconditioning

AMS subject classifications. 76M10, 65F10

DOI. 10.1137/040609045

1. Introduction.

In this paper we present a numerical study of the mixed finite element method applied to the steady state Navier–Stokes problem:

$$-\nu\Delta u + (u\cdot\nabla)u + \nabla p = 0 \quad \text{in } \Omega, \tag{1.1}$$

$$-\nabla\cdot u = 0 \quad \text{in } \Omega, \tag{1.2}$$

$$\mathcal{B}(u,p) = g \quad \text{on } \Gamma, \tag{1.3}$$

where Ω is a bounded open polygonal domain in $\mathbb{R}^2$ with boundary Γ, $u = (u_1, u_2)$ is the velocity of the fluid, p is the pressure, and ν is the viscosity parameter. The boundary conditions are represented by the operator $\mathcal{B}$, which may be a differential operator to allow for the derivative boundary conditions.

The nonlinear equation (1.1) is linearized and the solution is obtained via a fixed point iteration. For a given iteration (u_n, p_n) we construct the next iteration by solving the following problem, known as the Oseen equation:

$$-\nu\Delta u_{n+1} + (u_n\cdot\nabla)u_{n+1} + \nabla p_{n+1} = 0 \quad \text{in } \Omega, \tag{1.4}$$

$$-\nabla\cdot u_{n+1} = 0 \quad \text{in } \Omega, \tag{1.5}$$

$$\mathcal{B}(u_{n+1}, p_{n+1}) = g \quad \text{on } \Gamma. \tag{1.6}$$

*Received by the editors May 26, 2004; accepted for publication (in revised form) March 2, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-6/60904.html
†Mathematics Department, SciTech, University of Sussex, Falmer, East Sussex, BN1 9QH, UK (d.a.kay@sussex.ac.uk).
‡Mathematics Department, University Valahia of Targoviste, Bd. Unirii 18, Targoviste, Romania (emil.lungu@valahia.ro).

The starting iteration is taken to be the solution of the associated Stokes problem with the same boundary condition.

A discretization by a mixed finite element method produces the linear system of equations

$$\mathcal{L}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{1.7}$$

where

$$\mathcal{L} = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}$$

and $F = \nu A + N$ with A and N block diagonal corresponding to the discrete Laplace operator for each component of the velocity and the discrete velocity convection operator, respectively. B corresponds to the discrete divergence operator for the mixed velocity-pressure formulation. In the case of using stabilized formulation the zero block is replaced by a stabilization term, C. The stabilized matrix C is symmetric and positive semidefinite.

In Murphy, Golub, and Wathen [16], a block diagonal preconditioner is presented. The preconditioner has the form

$$\mathcal{P} = \begin{pmatrix} F & 0 \\ 0 & S \end{pmatrix},$$

where $S = BF^{-1}B^T$ is the Schur complement. The matrix $\mathcal{P}^{-1}\mathcal{L}$ of the preconditioned system has three distinct eigenvalues. This assures the convergence of Krylov subspace methods in at most three iterations; see, e.g., [17]. Calculating the action of the preconditioning matrix, $\mathcal{P}$, requires the solution of two separate linear systems corresponding to the blocks F and S.

Elman and Silvester [11] suggested a triangular preconditioner

$$\mathcal{P} = \begin{pmatrix} F & B^T \\ 0 & S \end{pmatrix}.$$

In the case where S is the exact Schur complement, this leads to a preconditioned system, $\mathcal{L}\mathcal{P}^{-1}$. This system has minimal polynomial of degree two, thus guaranteeing convergence in two iterations.

The practical implementation of these preconditioners relies upon the ability to approximate the inverse of both the F and the S blocks. In Kay, Loghin, and Wathen [14] a practical approximation to the Schur complement is given. It generalizes the idea of Elman and Silvester and is based on the Green's tensor for the Oseen operator. This choice is

$$\tilde{S} = A_p F_p^{-1} M_p,$$

where A_p, M_p, and F_p are the discrete Laplacian, identity, and convection diffusion operator on the pressure space. When using this preconditioner, it was seen that the convergence was independent of h, the mesh size parameter, and it had mild dependence on the viscosity. This paper aims to show that increasing the polynomial degree, k, of the finite element space does not affect the optimality of the preconditioner.

The practical implementation of the preconditioned iterative scheme requires the solution of a system with the matrix $\tilde{S}$, which is equivalent to solving a discrete

Poisson problem followed by a multiplication by F_p and finally a solution of a system with the mass matrix M_p.

In this paper we examine the effect of the above preconditioner on high-order mixed finite element approximations; see [5]. Examples of preconditioning for high-order quadrilateral elements can be seen in [15], [7], [8].

The advantage of high-order piecewise polynomial approximations is that they can exploit the smoothness of a solution and gain exponential rates of convergence away from singularities in the solution; see, e.g., Babuska and Suri [3].

The remainder of the paper is organized in three sections. In section 2 we present the two mixed p-version finite element discretization of the Navier–Stokes equations. In section 3 we introduce the preconditioner. The final section contains numerical experiments which show the efficiency of the preconditioner. We also show a numerical result for an hp-finite element discretization; see [4].

2. The discretization. In this section we consider the Oseen equation

$$-\nu\Delta u + (w \cdot \nabla)u + \nabla p = 0 \quad \text{in } \Omega, \tag{2.1}$$

$$-\nabla \cdot u = 0 \quad \text{in } \Omega, \tag{2.2}$$

where w is a divergence-free vector. We assume that the boundary $\overline{\Gamma} = \overline{\Gamma}_D \bigcup \overline{\Gamma}_N$ with $\Gamma_D \neq \emptyset$ and $\Gamma_D \bigcap \Gamma_N = \emptyset$. The boundary conditions are given by

$$u = g \quad \text{on } \Gamma_D, \tag{2.3}$$

$$\nu\nabla u \cdot n - p \cdot n = 0 \quad \text{on } \Gamma_N. \tag{2.4}$$

The weak formulation of (2.1) and (2.2) is defined in terms of subsets of the standard Sobolev spaces $H^1(\Omega)$ and $L^2(\Omega)$. We define the velocity space

$$X := (H^1_D(\Omega))^2 = \{v \in (H^1(\Omega))^2 : v = g \text{ on } \Gamma_D\}$$

and the pressure space

$$M := L^2_0(\Omega) = \left\{ q \in L^2(\Omega) : \int_\Omega q dx = 0 \right\}$$

if $\Gamma_N = \emptyset$; else $M = L^2(\Omega)$.

We also define

$$X_0 := (H^1_0(\Omega))^2 = \{v \in (H^1(\Omega))^2 : v = 0 \text{ on } \Gamma_D\}.$$

The weak formulation is given by the following: find $(u, p) \in X \times M$ such that

$$a(w; u, v) + b(p, v) = 0 \quad \forall\, v \in X_0, \tag{2.5}$$

$$b(q, u) = 0 \quad \forall\, q \in M, \tag{2.6}$$

where

$$a(w; v_1, v_2) := \nu(\nabla v_1, \nabla v_2) + ((w \cdot \nabla)v_1, v_2) \quad \forall\, v_1 \in X,\ v_2 \in X_0$$

and

$$b(q, v) = -(q, \nabla \cdot v) \quad \forall\, q \in M,\ v \in X_0.$$

Here $(\cdot, \cdot)$ denotes the $L^2(\Omega)$ inner product.

It can be shown that $a(\cdot,\cdot)$ is continuous and coercive and that there exists a $\kappa \in \mathbb{R}$ such that the continuous inf-sup condition

$$\sup_{v \in X} \frac{b(q,v)}{||v||_{H^1(\Omega)}} \geq \kappa ||q||_{L^2(\Omega)} \quad \forall\, q \in M$$

holds. Using these properties it can be shown that this problem has a unique solution; see [13].

For the discrete formulation of the problem (2.5), (2.6) we will use the mixed finite element method.

Let τ_h be a triangulation of the domain Ω and denote $h = \max_{T \in \tau_h} h_T$ ($h_T = diam(T)$). We denote by V_h^k the space of continuous piecewise polynomials of degree no more than k on τ_h, i.e.,

$$V_h^k = \{v \in C^0(\Omega) : v|_T \in \mathbf{P}_k[T] \ \forall\, T \in \tau_h\}. \tag{2.7}$$

Here $\mathbf{P}_k[T]$ is the space of the polynomial functions defined on triangle T which have degree at most k.

A local basis in $\mathbf{P}_k[T]$ is given starting from a basis on the reference triangle (see [1], [2]) via a linear mapping. In our experiments we used the integrated Legendre polynomial.

We now define the finite element spaces $X_D^{h,k+1} \subset X$ for velocity and $M_h^k \subset M$ for pressure as follows:

$$X_D^{h,k+1} = \{v \in V_h^{k+1} \times V_h^{k+1} : v = \bar{g} \text{ on } \Gamma_D\}, \tag{2.8}$$

and if $\Gamma_N = \emptyset$,

$$M_h^k = \left\{ q \in V_h^k : \int_\Omega q dx = 0 \right\} \quad (\text{else} \quad M_h^k = V_h^k). \tag{2.9}$$

Here $\bar{g}$ is a continuous piecewise polynomial function of degree at most $k+1$ which interpolates the function g on Γ_D. We will denote this space by $P_{k+1}P_k := X_D^{h,k+1} \times M_h^k$.

The discrete problem is to find $(u_h, p_h) \in P_{k+1}P_k$ such that

$$\nu(\nabla u_h, \nabla v_h) + ((w \cdot \nabla)u_h, v_h) - (p_h, \nabla v_h) = 0 \quad \forall\, v_h \in X_0^{h,k+1}, \tag{2.10}$$

$$-(\nabla \cdot u_h, q_h) = 0 \quad \forall\, q_h \in M_h^k. \tag{2.11}$$

Let us write the velocity solution as $u_h = u_D + u_0$, where $u_0 \in X_0^{h,k+1}$ and $u_D \in X_D^{h,k+1}$ is zero on the triangles which do not have any points on the boundary Γ_D. The function u_0 satisfies equations similar to (2.10) and (2.11) in which the right-hand sides are not zero.

In order to reduce the weak formulation on the finite element spaces to a linear algebra problem we now consider the finite element basis sets

$$X_0^{h,k+1} = span\{\phi_i\}_{i=1}^n, \qquad M_h^k = span\{\psi_j\}_{j=1}^m$$

and associate with the basis functions u_0 and p_h the vectors $u \in \mathbb{R}^n$ and $p \in \mathbb{R}^m$, that is, $u_0 = \sum_{i=1}^n u_i \phi_i$ and $p_h = \sum_{j=1}^m p_j \psi_j$.

Defining, respectively, the $n \times n$ convection and diffusion matrices

$$N_{ij} = (w_h \cdot \nabla \phi_i, \phi_j)$$

and

$$A_{ij} = (\nabla \phi_i, \nabla \phi_j)$$

and also the $m \times n$ divergence matrix

$$B_{ij} = -(\nabla \cdot \phi_i, \psi_j),$$

(2.10) and (2.11) lead to a linear system of the form

$$\begin{pmatrix} \nu A + N & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \tag{2.12}$$

Classical h finite element refinement leads to the approximation result: Given $w \in H_D^m(\Omega)$ there exists a $\pi_h w \in X_D^{h,k}$ such that, for $0 \le l \le k+1$,

$$||w - \pi_h w||_{H^l(\Omega)} \le C(k) h^\mu ||u||_{H^m(\Omega)},$$

where

$$\mu = \min(m - l, k + 1 - l).$$

The advantage of increasing polynomial degree over standard h finite element refinements is that approximation is restricted only by the smoothness of the solution and not by the degree of approximation. From [3] we have the standard approximation result: Given $u \in H_D^m(\Omega)$ there exists a $\pi_k u \in X_D^{h,k}$ such that, for $0 \le l \le m$,

$$||w - \pi_k w||_{H^l(\Omega)} \le C(h) k^{-(m-l)} ||w||_{H^m(\Omega)}.$$

Combining these approximation properties with standard a priori error estimates for the Navier–Stokes equations (see [13]), we obtain convergence rates of $O(h^\mu)$ and $O(k^{-(m-l)})$ for the h and k refinement strategies, respectively. Hence, if we have a smooth solution, then we expect uniform k refinements to give better approximations than uniform h refinements for a given number of degrees of freedom.

Remark 2.1. The discrete inf-sup condition is given by

$$\sup_{v \in X_D^{h,k+1}} \frac{b(q,v)}{||v||_{H^1(\Omega)}} \ge \gamma ||q||_{L^2(\Omega)} \quad \forall\, q \in M^{h,k}. \tag{2.13}$$

For guaranteed optimal convergence of the finite element approximation with respect to both h and k, it is required that γ not tend to zero as $h \to 0$ and/or $k \to \infty$; see [13].

The element $P_{k+1}P_k$ does not satisfy this condition in general as $k \to \infty$; see pressure solutions in section 4.

Due to lack of inf-sup stability for the $P_{k+1}P_k$ we will also consider the mixed finite element spaces

$$P_{k+2}P_k := X_D^{h,k+2} \times M_h^k.$$

Although stability with respect to k is still an issue, this element is seen to be stable for the numerical results presented. It is known (see [9], [6]) that when using spectral elements a k independent inf-sup constant can be obtained. When using this element we will lose the balance of approximation between the velocity error and the pressure error if the convergence is algebraic. On the other hand, if the solution is smooth, this will not change the overall convergence, and exponential rates are to be expected.

3. Preconditioner. The matrix $\mathcal{L}$ of the linear system (1.7) is nonsymmetric, and the condition number grows rapidly with increasing k and decreasing h. We will consider preconditioned GMRES to solve this system; see Saad and Schultz [17]. For good convergence a preconditioning strategy which produces eigenvalues of the preconditioned system clustered in a region independent of both k and h is desirable.

In what follows we introduce our preconditioner under the assumption that a fast solver is available for the convection diffusion system $Fu = f$. Let us consider the preconditioner to be of block upper triangular form:

$$\mathcal{P}^{-1} = \begin{pmatrix} F^{-1} & R \\ 0 & S^{-1} \end{pmatrix}$$

with the matrices $R \in \mathbb{R}^{n\times m}$ and $S \in \mathbb{R}^{m\times m}$ given by

$$R = F^{-1}B^T S^{-1}$$

and

$$S = BF^{-1}B^T.$$

This choice reduces the preconditioned matrix $\mathcal{L}\mathcal{P}^{-1}$ to a lower triangular form having all diagonal entries equal to 1. Consequently the spectrum $\sigma(\mathcal{L}\mathcal{P}^{-1}) = \{1\}$, and this guarantees the convergence of the preconditioned GMRES in at most two iterations; see [16].

Implementation of a right preconditioner for GMRES requires the solution of a system with matrix $\mathcal{P}$ at every iteration. In an explicit form this means that we have to compute a vector $(v;\ q)^T$ satisfying

$$\begin{pmatrix} v \\ q \end{pmatrix} = \begin{pmatrix} F^{-1} & F^{-1}B^T S^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} r \\ s \end{pmatrix}$$

for given vectors $r \in \mathbb{R}^n$ and $s \in \mathbb{R}^m$. This is equivalent to the following sequence of operations:

$$\begin{aligned} &\text{Solve for } q: && Sq = s, \\ &\text{Solve for } v: && Fv = -r - B^T q. \end{aligned}$$

In order to obtain a practical method, the exact preconditioner described above is replaced by an approximation which consists in the replacing of the operators S and F by S_* and F_*, respectively. These new operators have to be easy to implement and they should produce a preconditioned Oseen operator which has a tightly clustered spectrum. Thus, the operators S_* and F_* are chosen such that $\sigma(SS_*^{-1}) \subset \omega_S$ and $\sigma(FF_*^{-1}) \subset \omega_F$, where ω_S and ω_F represent small convex sets in the right half of the complex plane; ideally, these sets would be independent of the problem parameters ν, h, and k.

Following Kay, Loghin, and Wathen [14], we approximate the Schur complement, S, by the matrix

$$S_p := A_p F_p^{-1} M_p,$$

where A_p, F_p, and M_p are the pressure space diffusion, convection-diffusion, and identity matrices, respectively.

The approximate inversion of the matrices A_p and M_p is produced by using preconditioned conjugate gradients. The preconditioner was chosen to be of block diagonal form according to the decomposition of the basis functions into three categories, vertex, edge, and interior functions; see Babuska et al. [5]. Given a positive definite matrix A, we denote N iterations of preconditioned conjugate gradients by $PCG(A, N)$. This leads to the practical approximation of the Schur complement

$$(S_p^*)^{-1} := PCG(M_p, N)F_pPCG(A_p, N),$$

where N is a mesh- and degree-independent positive integer.

Remark 3.1. In most calculations we found that $N = 40$ was a robust amount of iterations. This is slightly larger than would be desired.

4. Results. The following examples show how the iterative solver is affected by h and k refinements and also the viscosity parameter ν.

Example 4.1. The first application of the mixed finite element method using high-degree polynomials is the Stokes equation applied to the driven cavity flow problem:

$$-\Delta u + \nabla p = 0 \quad \text{on } \Omega, \tag{4.1}$$

$$-\nabla \cdot u = 0 \quad \text{on } \Omega, \tag{4.2}$$

where Ω is the unit square $[0, 1] \times [0, 1]$.

The boundary conditions are given by

$$u(x, y) = \begin{cases} (1, 0) & \text{for } y = 1,\ x \in [0, 1], \\ (0, 0) & \text{otherwise,} \end{cases} \quad (x, y) \in \partial\Omega.$$

Our experiments were done on three different meshes. The first one is depicted in Figure 1, while the second and the third are obtained by two successive uniform h refinements.

Since the system is symmetric, instead of the block triangular preconditioner we used a block diagonal preconditioner of the following form:

$$\mathcal{P}^{-1} = \begin{pmatrix} F^{-1} & 0 \\ 0 & S^{-1} \end{pmatrix}.$$

For the theoretical results concerning this preconditioner the reader may consult the paper of Silvester and Wathen [19]. Applying the preconditioner $\mathcal{P}^{-1}$ is equivalent to solving the system corresponding to the two blocks.

For the approximate preconditioner we use a preconditioned conjugate gradient method for each block. The preconditioner for each velocity component was PCG $(F_{ii}, 40)$, $i = 1, 2$. Since this is the Stokes equation, the approximate Schur complement is given by the pressure mass matrix. Hence, we use $PCG(M_p, 40)$.

The results are summarized in Table 1. The entries of this table represent the iteration counts necessary for solving the linear system with a tolerance established at 10^{-6}. We clearly see that the solver is optimal with respect to the polynomial degree and element mesh size. Furthermore, the fact that this element is not inf-sup stable with respect to k does not affect the solver.

Example 4.2. In the second example we present the results obtained for the Navier–Stokes equation on the same domain and with the same boundary conditions as in the previous example. The convection diffusion blocks for the velocity space are calculated exactly.

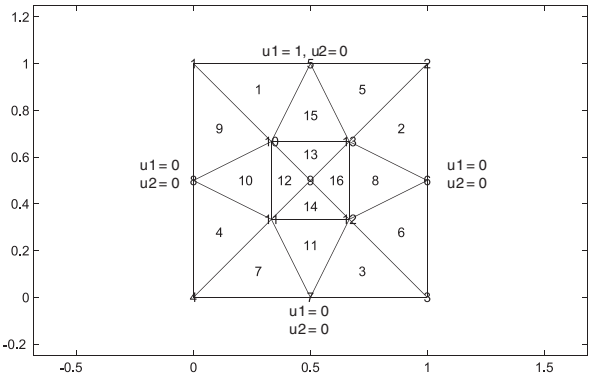


FIG. 1. *Triangulation.*

TABLE 1
GMRES iteration counts for Stokes driven cavity flow discretized by $P_{k+1}P_k$ elements.

k	1	2	3	4	5	6
V_h	10	20	20	20	20	21
$V_{h/2}$	16	20	20	22	20	22
$V_{h/4}$	18	20	20	20	20	22

For $P_{k+1}P_k$ and $P_{k+2}P_k$, respectively, Tables 2 and 3 show the maximum number of iterations required to solve an iteration of the fixed point iteration over all fixed point iterations performed. The number of fixed point iterations required was given by a tolerance set on the convergence of 10^{-6}.

From Tables 2 and 3 we can clearly see the independence of the solver with respect to both h and k refinements. We also can see the dependence of the solver on the viscosity. This dependence seems to be proportional to $\sqrt{\nu}$; see [18], [14].

Example 4.3. The third example treats the Navier–Stokes problem of flow past a backward facing step. The boundary conditions are detailed in Figure 2, and they correspond to an inflow on the edge AB and an outflow on the edge EF. As in the earlier examples the experiments were done on two successive levels of refinement of an initial mesh, which is also depicted in Figure 2.

Tables 4 and 5 show that the solver has no dependence on both the h and k refinements for both $P_{k+1}P_k$ and $P_{k+2}P_k$, respectively.

Figures 3 and 4 show the approximate pressures obtained by discretization with P_7P_6 and P_8P_6, respectively. In Figure 3 one notices some oscillations of the pressure solution near the inflow boundary. These oscillations do not appear in the case of discretization by $P_{k+2}P_k$ elements. The reason for these oscillations is that the mesh given in Figure 2 does not satisfy the stability condition according to the definition given in [10], having triangles with two edges on the boundary of the domain. Experiments made on the mesh given in Figure 5 produced solutions that did not exhibit these oscillations; see Figure 6.

Example 4.4. We now introduce an inexact solver for F using preconditioned GMRES. The preconditioner will take the same block diagonal form as that given earlier for the symmetric matrices A_p and M_p. For the F block the maximum number of GMRES iterations is 30, with a maximum of 3 restarts and a tolerance of 10^{-3} on

TABLE 2
GMRES iteration counts for Navier–Stokes driven cavity flow discretized by $P_{k+1}P_k$ elements.

k		1	2	3	4	5	6	7	8	9
	V_h	12	16	16	16	17	18	18	18	19
$\nu = 0.1$	$V_{h/2}$	14	15	16	17	17	18	18		
	V_h	12	17	18	19	19	20	21	21	21
$\nu = 0.05$	$V_{h/2}$	16	18	19	20	20	20	21		
	V_h	13	21	22	23	24	24	25	25	25
$\nu = 0.01$	$V_{h/2}$	20	23	23	25	25	25	25		
	V_h	13	24	25	26	27	28	29	29	30
$\nu = 0.005$	$V_{h/2}$	22	26	28	29	30	30	30		

TABLE 3
GMRES iteration counts for Navier–Stokes driven cavity flow discretized by $P_{k+2}P_k$ elements.

k		1	2	3	4	5	6	7
	V_h	12	14	16	16	17	17	17
$\nu = 0.1$	$V_{h/2}$	13	15	16	17	17	17	17
	$V_{h/4}$	13	16	16	16	16	16	
	V_h	12	16	17	18	18	19	20
$\nu = 0.05$	$V_{h/2}$	15	18	19	19	20	20	20
	$V_{h/4}$	16	19	19	19	20	20	
	V_h	13	20	20	22	22	23	24
$\nu = 0.01$	$V_{h/2}$	20	23	25	26	26	26	26
	$V_{h/4}$	23	26	27	27	26	27	
	V_h	13	23	25	27	28	29	29
$\nu = 0.005$	$V_{h/2}$	22	27	28	29	30	30	31
	$V_{h/4}$	28	29	30	30	30	31	

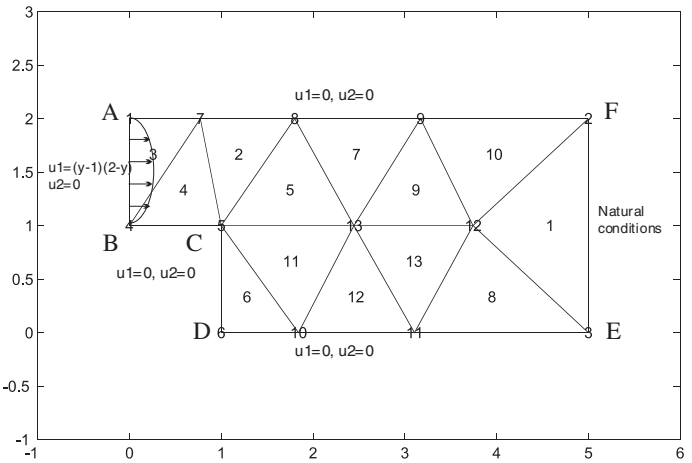


FIG. 2. *Coarsest grid triangulation for the backward facing step.*

the solver. We denote the fully practical GMRES solver by $GMRES^*$. We apply this fully practical solver to the lid driven cavity problem.

Comparing Table 3 with Table 6 we see that the iteration counts increase slightly when using an inexact solver on the F block. This increase is not significant and the key characteristics of the solver are maintained. The inexact solver on the F block is not ideal, since it is not robust with respect to ν, k. In particular, as ν decreases

TABLE 4
GMRES iteration counts for Navier–Stokes flow over a backward facing step discretized by $P_{k+1}P_k$ elements.

k		1	2	3	4	5	6	7	8	9
$\nu = 0.1$	V_h	14	22	27	28	31	32	34	34	30
	$V_{h/2}$	19	25	27	30	31	31	28	27	
	$V_{h/4}$	20	25	27	28	29	26	27		
$\nu = 0.05$	V_h	14	23	28	30	32	33	35	36	34
	$V_{h/2}$	20	25	28	31	32	33	33	29	
	$V_{h/4}$	22	26	28	30	31	30	28		
$\nu = 0.01$	V_h	14	28	35	41	38	39	41	42	43
	$V_{h/2}$	25	29	35	34	36	35	32	33	
	$V_{h/4}$	25	30	32	34	34	33	32		
$\nu = 0.005$	V_h	14	34	44	46	49	49	48	49	50
	$V_{h/2}$	32	39	44	45	44	44	45	44	
	$V_{h/4}$	33	39	42	41	43	43	44		

TABLE 5
GMRES iteration counts for Navier–Stokes flow over a backward facing step discretized by $P_{k+2}P_k$ elements.

k		1	2	3	4	5	6	7	8
$\nu = 0.1$	V_h	13	19	20	22	23	24	25	24
	$V_{h/2}$	18	21	22	23	24	24	24	
	$V_{h/4}$	19	21	22	22	22	22		
$\nu = 0.05$	V_h	14	20	21	23	24	25	25	26
	$V_{h/2}$	18	22	23	24	25	26	26	
	$V_{h/4}$	19	22	22	23	23	23		
$\nu = 0.01$	V_h	14	25	27	31	28	29	29	29
	$V_{h/2}$	22	26	29	28	29	29	28	
	$V_{h/4}$	23	28	29	29	29	29		
$\nu = 0.005$	V_h	14	32	37	36	36	37	35	35
	$V_{h/2}$	31	35	36	37	35	35	35	
	$V_{h/4}$	33	35	37	35	35	35		

the inexact solver fails for large k. This failure can be seen in Table 6, where the GMRES* iteration counts grew too large as ν decreased and k increased.

Example 4.5. In what follows we want to show that our preconditioner is efficient for the *hp* version as well. As a model we consider the backward facing step problem. It is known that the analytical pressure solution has a singularity in the corner C(1,1). This behavior can also be noticed by looking at the graphic representation of the approximate pressure solution. From theory we know that in such cases the convergence rate, with respect to k, is not an exponential one, but it is comparable with the convergence rate of the h-version. Therefore, we are tempted to use high-order elements away from the corner C(1,1) combined with low-order elements near the corner. In our experiments we considered a $\mathbf{P}_2P_1$ discretization for the triangles of the mesh, which lie inside the square centered at C and having the side length equal to 0.5. For the triangles outside of this square we considered a $\mathbf{P}_{k+1}P_k$ discretization. Figure 7 represents the mesh refined near the corner.

The results given in Table 7 clearly show the desired robustness of this solver with respect to nonuniform refinement.

5. Conclusions. The preconditioned iterative solver presented is shown to converge rapidly. This convergence rate is independent of both the mesh size and the

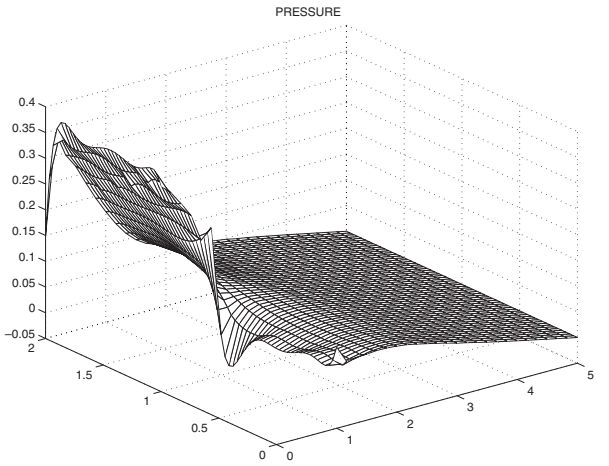


FIG. 3. *Pressure solution for Navier–Stokes problem discretized by* $P_{k+1}P_k$ *(*$\nu = 0.1$*,* $k = 6$*).*

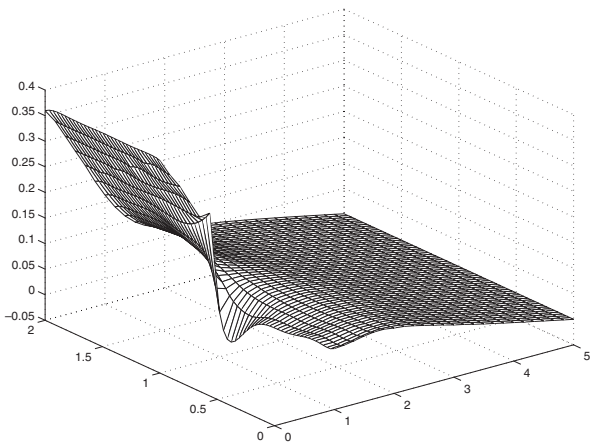


FIG. 4. *Pressure solution for Navier–Stokes problem discretized by* $P_{k+2}P_k$ *(*$\nu = 0.1$*,* $k = 6$*).*

polynomial degree. Furthermore, it seems to be robust to nonuniform mesh and polynomial refinement. This makes the scheme very attractive when trying to efficiently produce accurate approximations to the Navier–Stokes equations, using *hp* finite elements.

We also see that in the case where the inf-sup condition is not satisfied, convergence rates are maintained. This is surprising, since if we consider the symmetric Stokes problem with block diagonal preconditioning, the standard convergence results for MINRES are of the form

$$\frac{||\mathbf{r}_n||^2_{\mathcal{P}^{-1}}}{||\mathbf{r}_0||^2_{\mathcal{P}^{-1}}} \leq \min_{p\in\Pi_n, p(0)=1} \max_{\lambda\in\Lambda(\mathcal{M})} |p(\lambda)|,$$

where $\mathbf{r}_n$ is the residual at the nth iteration, $\mathcal{M} = \mathcal{P}^{-1}\mathcal{L}$ is the preconditioned system, and Π_n is the space of polynomials of degree at most n. Hence, if an eigenvalue, $\lambda(k)$,

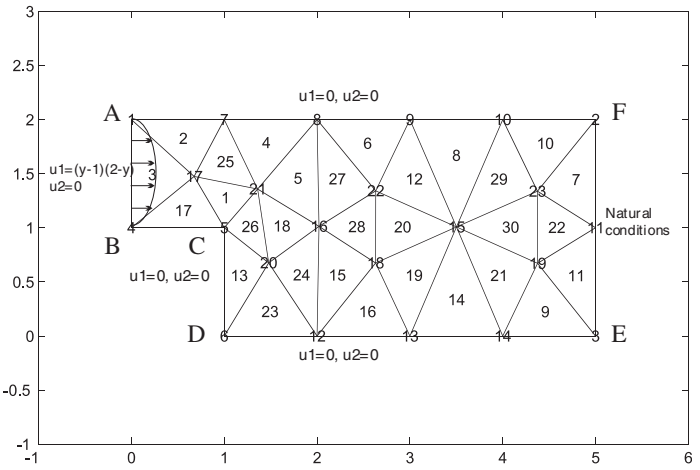


FIG. 5. *Coarsest grid triangulation for the backward facing step.*

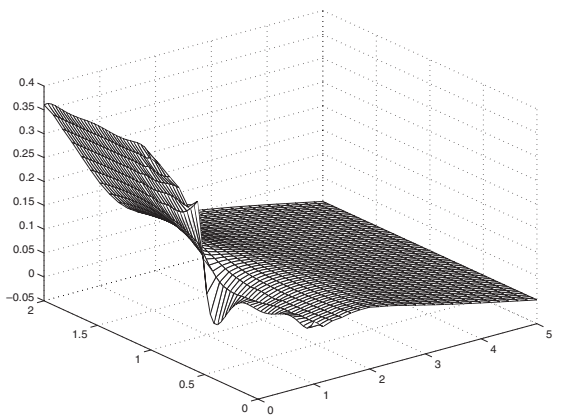


FIG. 6. *Pressure solution for Navier–Stokes problem discretized by $P_{k+1}P_k$, using refinement into the corners ($\nu = 0.1$, $k = 6$).*

TABLE 6

GMRES iteration counts for Navier–Stokes lid driven cavity flow discretized by $P_{k+2}P_k$ elements.*

k		3	4	5	6	7	8
$\nu = 0.1$	V_h	17	17	18	18	18	19
	$V_{h/2}$	16	17	18	18	18	19
$\nu = 0.05$	V_h	18	19	20	20	21	21
	$V_{h/2}$	20	20	20	21	21	>50
$\nu = 0.01$	V_h	22	25	25	26	26	28
	$V_{h/2}$	24	25	26	26	27	>100
$\nu = 0.005$	V_h	31	48	42	32	32	>100
	$V_{h/2}$	28	29	30	31	32	>100

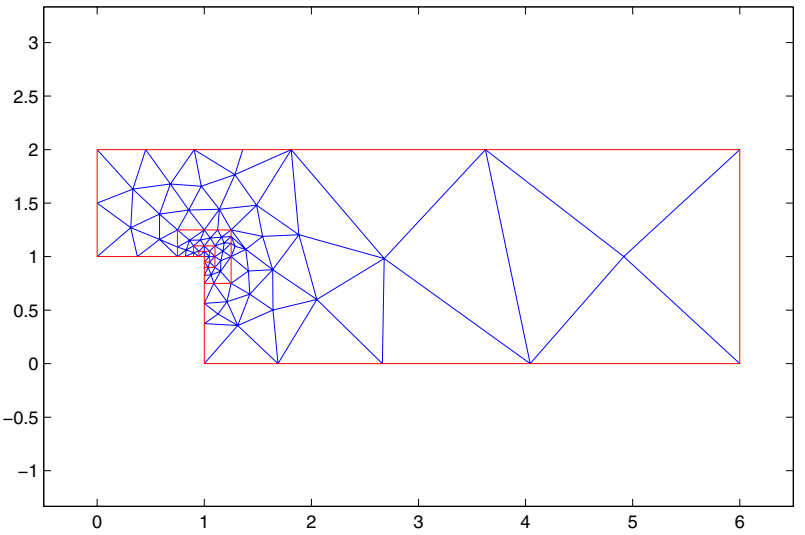


FIG. 7. *Mesh used for hp-refinement.*

TABLE 7
GMRES iteration counts for Navier–Stokes flow over a backward facing step discretized by $P_{k+1}P_k$ hp-finite elements ($k = 6$, $\nu = 0.01$).

k		1	2	3	4	5	6	7	8
ν	0.1	19	21	22	22	23	24	25	25
	0.05	22	23	23	24	25	25	26	26
	0.01	30	30	30	31	31	31	31	32
	0.005	40	41	40	39	40	40	39	40

of the preconditioned system is such that

$$\lambda(k) \to 0 \quad \text{as} \quad k \to \infty,$$

then the convergence rate will be degree dependent.

It can be shown (see Elman, Silvester, and Wathen [12] or Silvester et al. [18]) that the convergence rate of a possible group of eigenvalues to zero is bounded below by the discrete inf-sup value $\gamma(k)$. Hence, if $\gamma(k)$ is converging slowly to zero, or the min max polynomial can capture this set of eigenvalues quickly and all other eigenvalues are tightly clustered together in a region independent of k, then we would not expect to see a dramatic loss in convergence as k increases.

In the case of stable mixed elements, $\gamma(k)$ is bounded below independently of k, and in this case we would expect convergence rates independent of k. For the backward facing step problem this may account for the slightly improved iteration counts when using $P_{k+2}P_k$ over $P_{k+1}P_k$; see Tables 5 and 4, respectively.

Acknowledgment. The second author wishes to thank the Marie Curie fellowship.

REFERENCES

[1] S. ADJERID, M. AIFFA, AND J. E. FLAHERTY, *Hierarchical finite element bases for triangular and tetrahedral elements,* Comput. Methods Appl. Mech. Engrg., 190 (2001) pp. 2925–2941.

[2] I. BABUSKA, B. A. SZABO, AND I. KATZ, *The p-version of the finite element method,* SIAM J. Numer. Anal., 18 (1981), pp. 515–545.

[3] I. Babuška and M. Suri, *The optimal convergence rate of the p-version of the finite element method*, SIAM J. Numer. Anal., 24 (1987), pp. 750–776.
[4] I. Babuska and M. Suri, *The h-p version of the finite element method with quasiuniform meshes*, Math. Mod. Numer. Anal., 21 (1987), pp. 199–238.
[5] I. Babuška, A. Craig, J. Mandel, and J. Pitkäranta, *Efficient preconditioning for the p-version finite element method in two dimensions,* SIAM J. Numer. Anal., 28 (1991), pp. 624–661.
[6] F. Ben Belgacem, C. Bernardi, N. Chorfi, and Y. Maday, *Inf-sup conditions for the mortar spectral discretization of the Stokes problem,* Numer. Math., 85 (2000), pp. 257–281.
[7] S. Beuchler, *A Preconditioner for Solving the Inner Problem of the p-Version of the FEM,* Technical report SFB393 00-25, Technische Universität Chemnitz, Germany, 2000.
[8] S. Beuchler, *Optimal Preconditioners for the p-Version of the FEM,* Technical report SFB393/03-03, Technische Universität Chemnitz, Germany, 2003.
[9] C. Bernardi and Y. Maday, *Uniform inf-sup conditions for the spectral discretization of the Stokes problem,* Math. Models Methods Appl. Sci., 9 (1999), pp. 395–414.
[10] F. Brezzi and R. S. Falk, *Stability of higher-order Hood–Taylor methods,* SIAM J. Numer. Anal., 28 (1991), pp. 581–590.
[11] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier–Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.
[12] H. C. Elman, D. J. Silvester, and A. Wathen, *Iterative methods for problems in computational fluid dynamics*, in Iterative Methods in Scientific Computing, R. Chan, T. Chan, and G. Golub, eds, Springer-Verlag, New York, 1997, pp. 271–327.
[13] V. Girault and P. A. Raviart, *Finite Element Approximation of the Navier-Stokes Equations,* Springer-Verlag, New York, 1979.
[14] D. Kay, D. Loghin, and A. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.
[15] S. A. Korneev and V. G. Korneev, *On the Preconditioning in the Domain Decomposition Technique for the p-Version Finite Element Method, Part* I, II, Technical report SPC 95-35(36), Technische Universität Chemnitz-Zwickau, Germany, 1995.
[16] M. F. Murphy, G. H. Golub, and A. J. Wathen, *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21 (2000), pp. 1969–1972.
[17] Y. Saad and M. Schultz, *GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[18] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations,* J. Comput. Appl. Math., 128 (2001), pp. 261–279.
[19] D. Silvester and A. Wathen, *Fast iterative solution of stabilised Stokes systems part* II: *Using general block preconditioners,* SIAM J. Numer. Anal., 31 (1994), pp. 1352–1367.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1881–1902
© 2006 Society for Industrial and Applied Mathematics

EFFICIENT MINIMIZATION METHODS OF MIXED $\ell 2$-$\ell 1$ AND $\ell 1$-$\ell 1$ NORMS FOR IMAGE RESTORATION*

HAOYING FU†, MICHAEL K. NG‡, MILA NIKOLOVA§, AND JESSE L. BARLOW†

Abstract. Image restoration problems are often solved by finding the minimizer of a suitable objective function. Usually this function consists of a data-fitting term and a regularization term. For the least squares solution, both the data-fitting and the regularization terms are in the $\ell 2$ norm. In this paper, we consider the least absolute deviation (LAD) solution and the least mixed norm (LMN) solution. For the LAD solution, both the data-fitting and the regularization terms are in the $\ell 1$ norm. For the LMN solution, the regularization term is in the $\ell 1$ norm but the data-fitting term is in the $\ell 2$ norm. Since images often have nonnegative intensity values, the proposed algorithms provide the option of taking into account the nonnegativity constraint. The LMN and LAD solutions are formulated as the solution to a linear or quadratic programming problem which is solved by interior point methods. At each iteration of the interior point method, a structured linear system must be solved. The preconditioned conjugate gradient method with factorized sparse inverse preconditioners is employed to solve such structured inner systems. Experimental results are used to demonstrate the effectiveness of our approach. We also show the quality of the restored images, using the minimization of mixed $\ell 2$-$\ell 1$ and $\ell 1$-$\ell 1$ norms, is better than that using only the $\ell 2$ norm.

Key words. least squares, least absolute deviation, interior point method, linear programming, quadratic programming, preconditioner

AMS subject classifications. 65C20, 65F10

DOI. 10.1137/040615079

1. Introduction.

The problem of image restoration is considered. The observed image is the convolution of a shift invariant blurring function with the true image plus some additive noise. Let $f(x, y)$ be the original scene, $g(x, y)$ the observed scene, and $h(x, y)$ the blurring function. The image formation process can be modeled as follows:

$$g(x, y) = h(x, y) \star f(x, y) + w(x, y). \tag{1.1}$$

Here $w(x, y)$ is the additive noise and $\star$ denotes two-dimensional convolution. Let $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{w}$ be the discretized original scene, observed scene, and additive noise, respectively. Let H be the corresponding blurring matrix of appropriate size built according to the discretized point spread function $\mathbf{h}$. Then the discretized image formation process can be put into matrix-vector form:

$$\mathbf{g} = H\mathbf{f} + \mathbf{w}. \tag{1.2}$$

Assuming that the discretized scenes have $m \times n$ pixels, then $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{w}$ are vectors of length mn, and H is a matrix of $mn \times mn$. We remark that H is a matrix of block

*Received by the editors September 14, 2004; accepted for publication (in revised form) March 29, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-6/61507.html

†Department of Computer Science and Engineering, Pennsylvania State University, University Park, PA 16802 (hfu@cse.psu.edu, barlow@cse.psu.edu). Research for this work was done while the first author was visiting the Department of Mathematics, The University of Hong Kong.

‡Department of Mathematics, The University of Hong Kong, Hong Kong (mng@maths.hku.hk). The research of this author was supported in part by Hong Kong Research Grants Council grants HKU 7130/02P, HKU 7046/03P, and HKU 7035/05P.

§Centre de Mathématiques et de Leurs Applications, ENS de Cachan, 61 av. President Wilson, 94235 Cachan Cedex, France (nikolova@cmla.ens-cachan.fr).

Toeplitz with Toeplitz blocks (BTTB) when zero boundary conditions are applied, and block Toeplitz-plus-Hankel with Toeplitz-plus-Hankel blocks (BTHTHB) when Neumann boundary conditions are used [36].

Image restoration problems tend to be very ill-conditioned. Directly solving (1.2) will yield a solution that is extremely sensitive to noise; therefore, regularization methods are needed to stabilize the solution. For example, the linear least squares problem with Tikhonov's regularization [48] takes the following form:

$$\min_{\mathbf{f}} \|\mathbf{g} - H\mathbf{f}\|_2^2 + \alpha \|R\mathbf{f}\|_2^2. \tag{1.3}$$

In this optimization problem, the second term is a regularization term that measures the "irregularity" of the solution. We call R the regularization operator and α the regularization parameter. Very often R is chosen to be the difference operator, for example, the first order or second order finite difference operator.

Assuming the additive noise is white Gaussian and the prior distribution of $\mathbf{f}$ is also Gaussian, and that exact information of the blurring matrix is available, the solution of (1.3) can be interpreted as the "maximum a posteriori (MAP) estimator" of the original scene; see [6], for instance. However, due to the presence of edges, the prior distribution of an image rarely satisfies the Gaussian assumption well. In many cases, the additive noise does not satisfy the Gaussian assumption either; for instance, the noise may follow a Laplace distribution [6]. In the literature, there has been a growing interest in using the $\ell 1$ norm for parameter estimation [4, 6, 5, 26, 9, 10, 44] and for image restoration [3, 12, 21, 28, 35, 38, 39, 25]. The advantage of using the $\ell 1$ norm is that the solution is more robust than when using the $\ell 2$ norm in statistical estimation problems. In particular, a small number of outliers have less influence on the solution; see, for instance, [9, 38].

Since edges in an image lead to outliers in the regularization term, it is natural to consider using the $\ell 1$ norm for this term:

$$\min_{\mathbf{f}} \|\mathbf{g} - H\mathbf{f}\|_2^2 + \alpha \|R\mathbf{f}\|_1. \tag{1.4}$$

We call the solution to (1.4) the least mixed norm (LMN) solution. It is worth pointing out that model (1.4) is closely related to the total variation regularization in image restoration. In the one-dimensional case, if R is chosen to be the first order difference operator, then $\|R\mathbf{f}\|_1$ is the discrete version of the total variation norm. Li and Santosa used a piecewise linear $\ell 1$ function as a measure of total variation in [29]. For details about total variation regularization, we refer the reader to [43, 29, 13, 24, 11, 23].

If the additive noise does not satisfy the Gaussian assumption either, we consider using the $\ell 1$ norm for both the data-fitting and the regularization terms:

$$\min_{\mathbf{f}} \|\mathbf{g} - H\mathbf{f}\|_1 + \alpha \|R\mathbf{f}\|_1. \tag{1.5}$$

We call the solution to (1.5) the least absolute deviation (LAD) solution. Recently, minimizers of cost-functions involving $\ell 1$ data-fidelity have been studied by Nikolova [38, 39], Chan and Esedoglu [12], and Kärkkäinen, Kunisch, and Majava [25]. Applications to the processing of outliers and the removal of impulse noise are considered in denoising problems [38, 39, 25].

Another assumption that must be satisfied in order for (1.3) to yield the MAP estimator and which is often violated is that the exact information of the blurring matrix H is available. Many papers (for instance, [34, 37, 42, 20, 33]) address the issue

of inexact H but still assume white Gaussian noise and Gaussian prior distribution of $\mathbf{f}$. In this work, we assume that the exact information for H is available and focus on the issues of non-Gaussian noise and non-Gaussian prior distribution.

Since the objective functions are nonsmooth, problems (1.4) and (1.5) are difficult to solve [6, 9]. For instance, Mammone and Eichmann [32] solved an $\ell 1$ approximation problem with a nonnegative solution using linear programming techniques, and Abdelmalek and Kasvand [2, 1] solved the problem as a quadratic programming problem whose solution is nonnegative and bounded. However, they studied the algebraic image restoration problem in one dimension only, for example, separable blurring functions. Pruessner and O'Leary [42] considered minimization of total least $\ell 1$ norm by using linear programming for image restoration. However, since they simply use the MATLAB [19] function `linprog.m` to solve the linear programming problem, their algorithm is restricted to very small images. In [15, 16, 17, 18], Dax investigated regularized least norm problems where the identity matrix is used as the regularization operator (i.e., $R = I$) and developed row-relaxation methods for solving such least norm problems. The row-relaxation method is suitable for solving problems in which H is sparse. On the total variation regularization, Li and Santosa [29] employed both the steepest descent and an affine scaling Newton method, Ito and Kunisch [24] presented an active set strategy based on the augmented Lagrangian formulation, and Hintermüller and Kunisch [23] solved it as a bilaterally constrained optimization problem. Also Combettes [14] used an adaptive level set method for nondifferentiable constrained image recovery.

Usually, images satisfy

$$\mathbf{f} \geq 0,$$

that is, pixels have nonnegative values. However, this constraint is often omitted in restoration methods, mainly because of the numerical intricacies it entails. Hanke, Nagy, and Vogel [22] considered and studied three nonnegatively constrained restorations schemes: constrained least squares, maximum likelihood, and maximum entropy. Saunders [46] used primal-dual interior point methods for nonnegative least squares image restoration problems.

The main contribution of this work is to develop efficient algorithms for the minimization of mixed $\ell 2$-$\ell 1$ and $\ell 1$-$\ell 1$ norms for image restoration problems. The novelty comes by taking into account the nonnegativity constraint. We first formulate (1.4) as a linear programming problem and (1.5) as a quadratic programming problem. Then we solve these problems by combining an interior point method and a preconditioned conjugate gradient method. In each interior point iteration, a structured linear system is obtained and solved. The form of the coefficient matrix of this structured linear system is given by $D + H^T D' H + R^T D'' R$, where H is a BTTB matrix, and D, D', and D'' are diagonal matrices. To solve such inner systems quickly, we employ the preconditioned conjugate gradient method with factorized sparse inverse preconditioners. In [30], it is shown that factorized sparse inverse preconditioners are better than circulant preconditioners for certain image restoration problems. In this paper, we further demonstrate that factorized sparse inverse preconditioners are quite effective for minimization of mixed $\ell 2$-$\ell 1$ and $\ell 1$-$\ell 1$ norms in image restoration problems.

The rest of this paper is outlined as follows. In section 2, we formulate (1.5) as a linear programming problem and solve it by an interior point method. The major work of each iteration of the interior point method is to solve a large structured

linear system. We propose to solve this system by iterative methods. In this section, we also discuss finding an initial feasible point and keeping the iterates feasible. In section 3, we formulate (1.4) as a quadratic programming problem and also solve it by an interior point method. In section 4, we discuss how to precondition the structured linear systems encountered in sections 2 and 3. Experimental results are presented in section 5. Conclusions are made in section 6.

2. The LAD solution.

2.1. Formulating the problem as a linear programming problem. Problem (1.5), with the nonnegativity constraint imposed, can be formulated as a linear programming problem as follows. If the nonnegativity constraint is not desired, only slight modifications of this procedure are needed.

Let $\mathbf{u} = H\mathbf{f} - \mathbf{g}$; let $\mathbf{v} = \alpha R\mathbf{f}$. We split $\mathbf{u}$ and $\mathbf{v}$ into their nonnegative and nonpositive parts. That is, $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$ and $\mathbf{v} = \mathbf{v}^+ - \mathbf{v}^-$, where $\mathbf{u}^+ = \max(\mathbf{u}, 0)$, $\mathbf{u}^- = \max(-\mathbf{u}, 0)$, $\mathbf{v}^+ = \max(\mathbf{v}, 0)$, and $\mathbf{v}^- = \max(-\mathbf{v}, 0)$. The problem can now be written as

$$\begin{aligned} \min_{\mathbf{f},\mathbf{u}^+,\mathbf{u}^-,\mathbf{v}^+,\mathbf{v}^-} \quad & \mathbf{1}^T\mathbf{u}^+ + \mathbf{1}^T\mathbf{u}^- + \mathbf{1}^T\mathbf{v}^+ + \mathbf{1}^T\mathbf{v}^- \\ \text{subject to} \quad & H\mathbf{f} - \mathbf{g} = \mathbf{u}^+ - \mathbf{u}^-, \\ & \alpha R\mathbf{f} = \mathbf{v}^+ - \mathbf{v}^-, \\ & \mathbf{u}^+,\ \mathbf{u}^-,\ \mathbf{v}^+,\ \mathbf{v}^-,\ \mathbf{f} \geq 0. \end{aligned} \tag{2.1}$$

We use $\mathbf{1}$ to denote the vector of all ones of appropriate size. This notation is used throughout this paper.

Clearly (2.1) can be written as a linear programming problem in the standard form:

$$\min_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq 0, \tag{2.2}$$

where A, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{x}$ are defined by

$$A = \begin{bmatrix} H & -I & I & 0 & 0 \\ \alpha R & 0 & 0 & -I & I \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{g} \\ 0 \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{f} \\ \mathbf{u}^+ \\ \mathbf{u}^- \\ \mathbf{v}^+ \\ \mathbf{v}^- \end{bmatrix}, \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} 0 \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}.$$

The Lagrangian function for (2.2) is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \mathbf{c}^T\mathbf{x} - \boldsymbol{\lambda}^T(A\mathbf{x} - \mathbf{b}) - \mathbf{s}^T\mathbf{x}. \tag{2.3}$$

Here $\boldsymbol{\lambda}$ and $\mathbf{s}$ are the Lagrange multiplier vectors for the constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$, respectively. For clarity, we partition $\boldsymbol{\lambda}$ as

$$\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_u \\ \boldsymbol{\lambda}_v \end{bmatrix}, \tag{2.4}$$

where $\boldsymbol{\lambda}_u$ is the Lagrange multiplier vector for the constraint $H\mathbf{f} - \mathbf{u}^+ + \mathbf{u}^- = \mathbf{g}$, and $\boldsymbol{\lambda}_v$ is for $\alpha R - \mathbf{v}^+ + \mathbf{v}^- = 0$.

2.2. Interior point methods. Primal-dual interior point methods have become a popular choice for solving large linear programming problems. We briefly outline our adapted interior point method below. A detailed description of interior point methods for linear programming can be found in [49, Chapter 14] or [40, Chapter 1].

The optimality condition for (2.2) is as follows:

$$F(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{bmatrix} A^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} \\ A\mathbf{x} - \mathbf{b} \\ XS\mathbf{1} \end{bmatrix} = 0, \quad \mathbf{x} \geq 0, \quad \mathbf{s} \geq 0, \tag{2.5}$$

where

$$X = \operatorname{diag}(\mathbf{x}) \quad \text{and} \quad S = \operatorname{diag}(\mathbf{s}). \tag{2.6}$$

Primal-dual interior point methods have their origin in Newton's method for the system of nonlinear equations (2.5). Newton's method starts with some initial guess for the solution and calculates a search direction at each iteration by solving a linearized model of the original system. A detailed description of Newton's method for nonlinear systems can be found in [40, Chapter 11]. In the primal-dual interior point algorithm, the basic Newton step is modified such that the search directions are aimed at points on the *central path* $(\mathbf{x}_\tau, \boldsymbol{\lambda}_\tau, \mathbf{s}_\tau)$, which is defined as

$$F(\mathbf{x}_\tau, \boldsymbol{\lambda}_\tau, \mathbf{s}_\tau) = \begin{bmatrix} 0 \\ 0 \\ \tau\mathbf{1} \end{bmatrix}, \quad \mathbf{x}_\tau > 0, \quad \mathbf{s}_\tau > 0. \tag{2.7}$$

Very often τ is written as $\sigma\mu$, where $\sigma \in [0, 1]$ is a *centering parameter*, and μ is the *duality measure* defined by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i s_i = \frac{\mathbf{x}^T\mathbf{s}}{n}. \tag{2.8}$$

The step length at each iteration is chosen such that the new iterate is strictly positive, that is, $\mathbf{x} > 0$ and $\mathbf{s} > 0$.

The Newton search direction, $(\triangle\mathbf{x}, \triangle\boldsymbol{\lambda}, \triangle\mathbf{s})$, is computed by solving the following linear system:

$$\begin{bmatrix} 0 & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle\mathbf{x} \\ \triangle\boldsymbol{\lambda} \\ \triangle\mathbf{s} \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{r}_a \end{bmatrix}, \tag{2.9}$$

where

$$\mathbf{r}_c = A^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c}, \quad \mathbf{r}_b = A\mathbf{x} - \mathbf{b}, \quad \mathbf{r}_a = XS\mathbf{1} - \sigma\mu\mathbf{1}.$$

By eliminating $\triangle\mathbf{s}$ in (2.9), we obtain

$$\begin{bmatrix} -X^{-1}S & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle\mathbf{x} \\ \triangle\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_c \\ -\mathbf{r}_b \end{bmatrix}, \tag{2.10}$$

where $\hat{\mathbf{r}}_c = \mathbf{r}_c - X^{-1}\mathbf{r}_a$. Let $D = S^{-1/2}X^{1/2}$; then (2.10) can be written as

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle\mathbf{x} \\ \triangle\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_c \\ -\mathbf{r}_b \end{bmatrix}. \tag{2.11}$$

Note that since the components of $\mathbf{x}$ and $\mathbf{s}$ are strictly positive, all the diagonal elements of D are well defined.

From here the standard approach is to reduce (2.11) to the following form and solve it for step $\triangle\boldsymbol{\lambda}$:

$$AD^2A^T\triangle\boldsymbol{\lambda} = -AD^2\hat{\mathbf{r}}_c - \mathbf{r}_b. \tag{2.12}$$

The application of the preconditioned MINRES [41] or the preconditioned GMRES [45] to (2.10) may be considered. For example, the preconditioner of Benzi and Golub [7], which is briefly described in section 4, could be used to solve (2.10).

However, after comparing the performance of the different implementations, we found that it is advantageous to proceed as follows. System (2.11) can be written as

$$\begin{bmatrix} -D_1^{-2} & 0 & 0 & 0 & 0 & H^T & \alpha R^T \\ 0 & -D_2^{-2} & 0 & 0 & 0 & -I & 0 \\ 0 & 0 & -D_3^{-2} & 0 & 0 & I & 0 \\ 0 & 0 & 0 & -D_4^{-2} & 0 & 0 & -I \\ 0 & 0 & 0 & 0 & -D_5^{-2} & 0 & I \\ H & -I & I & 0 & 0 & 0 & 0 \\ \alpha R & 0 & 0 & -I & I & 0 & 0 \end{bmatrix} \begin{bmatrix} \triangle\mathbf{f} \\ \triangle\mathbf{u}^+ \\ \triangle\mathbf{u}^- \\ \triangle\mathbf{v}^+ \\ \triangle\mathbf{v}^- \\ \triangle\boldsymbol{\lambda}_u \\ \triangle\boldsymbol{\lambda}_v \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_{c1} \\ -\hat{\mathbf{r}}_{c2} \\ -\hat{\mathbf{r}}_{c3} \\ -\hat{\mathbf{r}}_{c4} \\ -\hat{\mathbf{r}}_{c5} \\ -\mathbf{r}_{b1} \\ -\mathbf{r}_{b2} \end{bmatrix}.$$

Here D_i (for $i = 1, 2, 3, 4, 5$) are submatrices of D of the appropriate size, and $\hat{\mathbf{r}}_{ci}$ (for $i = 1, 2, 3, 4, 5$) and $\mathbf{r}_{bi}$ (for $i = 1, 2$) are subvectors of the appropriate size of $\hat{\mathbf{r}}_c$ and $\mathbf{r}_b$, respectively.

By eliminating $\triangle\mathbf{u}^+$, $\triangle\mathbf{u}^-$, $\triangle\mathbf{v}^+$, and $\triangle\mathbf{v}^-$, we obtain

$$\begin{bmatrix} -D_1^{-2} & H^T & \alpha R^T \\ H & D_2^2 + D_3^2 & 0 \\ \alpha R & 0 & D_4^2 + D_5^2 \end{bmatrix} \begin{bmatrix} \triangle\mathbf{f} \\ \triangle\boldsymbol{\lambda}_u \\ \triangle\boldsymbol{\lambda}_v \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_{c1} \\ -\hat{\mathbf{r}}_{b1} \\ -\hat{\mathbf{r}}_{b2} \end{bmatrix}, \tag{2.13}$$

where

$$\hat{\mathbf{r}}_{b1} = \mathbf{r}_{b1} - D_2^2\hat{\mathbf{r}}_{c2} + D_3^2\hat{\mathbf{r}}_{c3} \quad \text{and} \quad \hat{\mathbf{r}}_{b2} = \mathbf{r}_{b2} - D_4^2\hat{\mathbf{r}}_{c4} + D_5^2\hat{\mathbf{r}}_{c5}.$$

By eliminating $\triangle\boldsymbol{\lambda}_u$ and $\triangle\boldsymbol{\lambda}_v$ from (2.13), we obtain

$$\left[D_1^{-2} + H^T(D_2^2 + D_3^2)^{-1}H + \alpha^2 R^T(D_4^2 + D_5^2)^{-1}R\right]\triangle\mathbf{f} = \tilde{\mathbf{r}}_{c1}, \tag{2.14}$$

where

$$\tilde{\mathbf{r}}_{c1} = \hat{\mathbf{r}}_{c1} - H^T(D_2^2 + D_3^2)^{-1}\hat{\mathbf{r}}_{b1} - \alpha R^T(D_4^2 + D_5^2)^{-1}\hat{\mathbf{r}}_{b2}.$$

System (2.14) can be viewed as the normal equations for a linear least squares problem with the following coefficient matrix:

$$\begin{bmatrix} D_1^{-1} \\ (D_2^2 + D_3^2)^{-1/2}H \\ \alpha(D_4^2 + D_5^2)^{-1/2}R \end{bmatrix} = \begin{bmatrix} D_1^{-1} & & \\ & (D_2^2 + D_3^2)^{-1/2} & \\ & & (D_4^2 + D_5^2)^{-1/2} \end{bmatrix} \begin{bmatrix} I \\ H \\ \alpha R \end{bmatrix}. \tag{2.15}$$

Clearly the second factor in the right-hand side of (2.15) has full column rank and is well-conditioned. Since the matrices D_i are all diagonal matrices with positive diagonal elements, the first factor in the right-hand side of (2.15) has full column

rank. It follows that the coefficient matrix in (2.14) is symmetric positive definite and can be solved by the conjugate gradient method.

In general, when the iterates are close to the solution, if the primal variable is close to zero, then the corresponding dual variable will be far away from zero. This means the corresponding element in D will be very small. On the other hand, if the primal variable is far away from zero, then the corresponding dual variable will be very close to zero, leading to a large entry in D. Therefore, when the iterates are close to the solution, the matrix D will have both huge and tiny elements, causing ill-conditioning. In section 4, we discuss how to precondition system (2.14) efficiently.

Once we have solved (2.14) for $\triangle\mathbf{f}$, the other unknowns in (2.9) can be recovered by the following equations:

$$\begin{cases} \triangle\boldsymbol{\lambda}_u = (D_2^2 + D_3^2)^{-1}(-\hat{\mathbf{r}}_{b1} - H\triangle\mathbf{f}), \\ \triangle\boldsymbol{\lambda}_v = (D_4^2 + D_5^2)^{-1}(-\hat{\mathbf{r}}_{b2} - \alpha R\triangle\mathbf{f}), \\ \triangle\mathbf{u}^+ = D_2^2(\hat{\mathbf{r}}_{c2} - \triangle\boldsymbol{\lambda}_u), \\ \triangle\mathbf{u}^- = D_3^2(\hat{\mathbf{r}}_{c3} + \triangle\boldsymbol{\lambda}_u), \\ \triangle\mathbf{v}^+ = D_4^2(\hat{\mathbf{r}}_{c4} - \triangle\boldsymbol{\lambda}_v), \\ \triangle\mathbf{v}^- = D_5^2(\hat{\mathbf{r}}_{c5} + \triangle\boldsymbol{\lambda}_v), \\ \triangle\mathbf{s} = -\mathbf{r}_c - A^T\triangle\boldsymbol{\lambda}. \end{cases} \tag{2.16}$$

2.3. Finding an initial feasible point and staying feasible. Define the *strictly feasible set* $\mathcal{F}^0$ as

$$\mathcal{F}^0 = \{(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) | A\mathbf{x} = \mathbf{b}, A^T\boldsymbol{\lambda} + \mathbf{s} = \mathbf{c}, \mathbf{x} > 0, \mathbf{s} > 0\}. \tag{2.17}$$

Finding an initial strictly feasible point is not straightforward in many linear programming problems. For our problem, an initial strictly feasible point can be found as follows. Let ϵ be some positive value. Let $\mathbf{f}_{ls}$ be any estimate of the sought-after solution, for example, the least squares solution. We choose $\mathbf{f}^{(0)}$ to be

$$\mathbf{f}^{(0)}[i] = \max\{\mathbf{f}_{ls}[i], \epsilon\}.$$

We let $\mathbf{u}^{+(0)}$ be the sum of ϵ and the nonnegative part of $H\mathbf{f}^{(0)}$ and we let $\mathbf{u}^{-(0)}$ be the sum of ϵ and the nonpositive part of $H\mathbf{f}^{(0)}$. Then the vector

$$\mathbf{x}^{(0)} = \begin{bmatrix} \mathbf{f}^{(0)} \\ \mathbf{u}^{+(0)} \\ \mathbf{u}^{-(0)} \\ \mathbf{v}^{+(0)} \\ \mathbf{v}^{-(0)} \end{bmatrix}$$

satisfies $A\mathbf{x}^{(0)} = \mathbf{b}$. Furthermore, $\mathbf{x}^{(0)} \geq \epsilon$. There are many ways that we can find $\boldsymbol{\lambda}^{(0)}$ and $\mathbf{s}^{(0)}$ such that $\mathbf{s}^{(0)} > 0$ and $A^T\boldsymbol{\lambda}^{(0)} + \mathbf{s}^{(0)} - \mathbf{c} = 0$. Since our focus is on the problem where H corresponds to a shift invariant blurring operator, we can let $\boldsymbol{\lambda}_u^{(0)}$ be a constant vector of -0.5 and let $\boldsymbol{\lambda}_v^{(0)}$ be the zero vector. If Neumann boundary conditions are used, and the blurring function is normalized such that it integrates to 1, then $H^T\boldsymbol{\lambda}_u^{(0)} = 0.5 \cdot \mathbf{1}$. Therefore, if we let $\mathbf{s}^{(0)} = \mathbf{c} - A^T\boldsymbol{\lambda}^{(0)}$, then $\mathbf{s}^{(0)} \geq 0.5 \cdot \mathbf{1}$, and $(\mathbf{x}^{(0)}, \boldsymbol{\lambda}^{(0)}, \mathbf{s}^{(0)})$ is strictly feasible. If zero boundary conditions are used, this procedure would still give an initial strictly feasible point under weak assumptions on the blurring function.

If the current iterate is in the strictly feasible set, and (2.9) is solved exactly, and if a step length is chosen such that the new iterate satisfies $\mathbf{x} > 0$ and $\mathbf{s} > 0$, then the new iterate is strictly feasible.

However, when we use the conjugate gradient method to solve (2.14), only an approximate solution is given. Therefore, even if the current iterate is strictly feasible, the new iterate may not satisfy the conditions

$$A\mathbf{x} = \mathbf{b} \quad \text{and} \quad A^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} = 0$$

to very high accuracy. Thus, if no precaution is taken, the subsequent iterates could move further from the feasible set due to error propagation.

Fortunately, it is easy to show that if (2.16) is used to recover the other unknowns after (2.14) is solved, the conditions $A\mathbf{x} = \mathbf{b}$ and $A^T\boldsymbol{\lambda} + \mathbf{s} - \mathbf{c} = 0$ will be satisfied to machine precision even if (2.14) is solved inexactly. Therefore, if a proper step length is chosen, the next iterate will be strictly feasible.

2.4. Summary of the adapted interior point algorithm. There are many variants of the interior point method. Our algorithm, summarized below, is based on the simplest framework, that is, framework PD in [49, p. 8].

ALGORITHM. INTERIOR POINT FOR THE LAD SOLUTION.

1. Initialize $(\mathbf{x}^{(0)}, \boldsymbol{\lambda}^{(0)}, \mathbf{s}^{(0)}) \in \mathcal{F}^o$.
2. for $k = 0, 1, 2 \ldots$
   - choose $\sigma \in [0, 1]$.
   - solve (2.14) for $\triangle\mathbf{f}$.
   - Use (2.16) to recover $\triangle\mathbf{u}^+$, $\triangle\mathbf{u}^-$, $\triangle\mathbf{v}^+$, $\triangle\mathbf{v}^-$, $\triangle\boldsymbol{\lambda}$, $\triangle\mathbf{s}$.
   - choose β^{pri} and β^{dual}.
   - $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \beta^{pri}\triangle\mathbf{x}$
   - $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \beta^{dual}\triangle\boldsymbol{\lambda}$
   - $\mathbf{s}^{(k+1)} = \mathbf{s}^{(k)} + \beta^{dual}\triangle\mathbf{s}$

   end

Since we use iterative methods to solve the linear systems, no predictor-corrector strategy is involved.

A word needs to be said about the choices of centering parameter σ and step lengths β^{pri} and β^{dual}. It is a common practice to choose the primal and dual step lengths β^{pri} and β^{dual} as follows:

$$\beta^{pri} = \min\left(1, \eta\beta^{pri}_{max}\right), \quad \beta^{dual} = \min\left(1, \eta\beta^{dual}_{max}\right).$$

Here β^{pri}_{max} and β^{dual}_{max} are the maximum steps that can be taken before violating the nonnegativity condition, and $\eta \in [0.9,\ 1.0)$. We simply choose $\eta = 0.95$. Our experiments led us to choose the centering parameter σ to be 0.1 at the first iteration, and

$$\sigma = \max(0.01, (1 - \min(\beta^{pri}, \beta^{dual}))^3)$$

in other iterations. This prevents the iterates from moving too far from the central path and allows significant reduction in duality at each iteration.

3. The LMN solution. Problem (1.4), with the nonnegativity constraint imposed, can be restated as a quadratic programming problem as follows. We let

$\mathbf{v} = \alpha R\mathbf{f}$, and split $\mathbf{v}$ into its nonnegative and nonpositive parts $\mathbf{v}^+$ and $\mathbf{v}^-$. Then (1.4) can be written as

$$\begin{aligned} \min_{\mathbf{f},\mathbf{v}^+,\mathbf{v}^-} \ & \mathbf{1}^T\mathbf{v}^+ + \mathbf{1}^T\mathbf{v}^- + \|H\mathbf{f} - \mathbf{g}\|_2^2 \\ \text{subject to} \ & \alpha R\mathbf{f} = \mathbf{v}^+ - \mathbf{v}^-, \\ & \mathbf{v}^+, \mathbf{v}^-, \mathbf{f} \geq 0. \end{aligned} \tag{3.1}$$

Problem (3.1) can be written as

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{c}^T\mathbf{x} \quad \text{subject to} \quad A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq 0, \tag{3.2}$$

where G, A, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{x}$ are defined by

$$G = \begin{bmatrix} 2H^TH & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad A = \begin{bmatrix} \alpha R & -I & I \end{bmatrix},$$

$$\mathbf{b} = \mathbf{0}, \quad \mathbf{x} = \begin{bmatrix} \mathbf{f} \\ \mathbf{v}^+ \\ \mathbf{v}^- \end{bmatrix}, \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} -2H^T\mathbf{g} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}.$$

We remark that the objective function is convex since G is symmetric positive semidefinite. Problem (3.2) can also be solved by an interior point method. The Lagrangian function for (3.2) is

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{c}^T\mathbf{x} - \boldsymbol{\lambda}^T(A\mathbf{x} - \mathbf{b}) - \mathbf{s}^T\mathbf{x}, \tag{3.3}$$

where $\boldsymbol{\lambda}$ and $\mathbf{s}$ are the generalized Lagrange multiplier vectors for the constraints $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq 0$, respectively.

The optimality condition for (3.2) is

$$F(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{s}) = \begin{bmatrix} G\mathbf{x} + \mathbf{c} - A^T\boldsymbol{\lambda} - \mathbf{s} \\ A\mathbf{x} - \mathbf{b} \\ XS\mathbf{1} \end{bmatrix} = 0, \quad \mathbf{x} \geq 0, \quad \mathbf{s} \geq 0. \tag{3.4}$$

The central path, $(\mathbf{x}_{\sigma\mu}, \boldsymbol{\lambda}_{\sigma\mu}, \mathbf{s}_{\sigma\mu})$, is defined similarly as in the previous section:

$$F(\mathbf{x}_{\sigma\mu}, \boldsymbol{\lambda}_{\sigma\mu}, \mathbf{s}_{\sigma\mu}) = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu\mathbf{1} \end{bmatrix}. \tag{3.5}$$

Recall that $\sigma \in (0, 1)$ and μ is defined as in (2.8). The Newton step, $(\triangle\mathbf{x}, \triangle\boldsymbol{\lambda}, \triangle\mathbf{s})$, is computed by solving the following linear system:

$$\begin{bmatrix} G & -A^T & -I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \triangle\mathbf{x} \\ \triangle\boldsymbol{\lambda} \\ \triangle\mathbf{s} \end{bmatrix} = \begin{bmatrix} -\mathbf{r}_c \\ -\mathbf{r}_b \\ -\mathbf{r}_a \end{bmatrix}, \tag{3.6}$$

where

$$\mathbf{r}_c = G\mathbf{x} + \mathbf{c} - A^T\boldsymbol{\lambda} - \mathbf{s}, \quad \mathbf{r}_b = A\mathbf{x} - \mathbf{b}, \quad \mathbf{r}_a = XS\mathbf{1} - \sigma\mu\mathbf{1}.$$

By eliminating $\triangle \mathbf{s}$, we obtain

$$\begin{bmatrix} G + X^{-1}S & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle \mathbf{x} \\ \triangle \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_c \\ -\mathbf{r}_b \end{bmatrix}, \tag{3.7}$$

where $\hat{\mathbf{r}}_c = \mathbf{r}_c + X^{-1}\mathbf{r}_a$. This system can be made symmetric by a change of variable:

$$\begin{bmatrix} G + X^{-1}S & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle \mathbf{x} \\ -\triangle \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_c \\ -\mathbf{r}_b \end{bmatrix}. \tag{3.8}$$

Let $D = S^{-1/2}X^{1/2}$. Note that the elements of D are well defined. Partition D as

$$D = \ \mathrm{diag}(D_1, D_2, D_3),$$

where D_1, D_2, and D_3 are matrices of appropriate size such that system (3.8) can be put into the following form:

$$\begin{bmatrix} 2H^TH + D_1^{-2} & 0 & 0 & \alpha R^T \\ 0 & D_2^{-2} & 0 & -I \\ 0 & 0 & D_3^{-2} & I \\ \alpha R & -I & I & 0 \end{bmatrix} \begin{bmatrix} \triangle \mathbf{f} \\ \triangle \mathbf{v}^+ \\ \triangle \mathbf{v}^- \\ -\triangle \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_{c1} \\ -\hat{\mathbf{r}}_{c2} \\ -\hat{\mathbf{r}}_{c3} \\ -\mathbf{r}_b \end{bmatrix}. \tag{3.9}$$

Here $\hat{\mathbf{r}}_{ci}$ (for $i = 1,\ 2,\ 3$) are subvectors of the appropriate size of $\hat{\mathbf{r}}_c$. By eliminating $\triangle \mathbf{v}^+$ and $\triangle \mathbf{v}^-$, we get

$$\begin{bmatrix} 2H^TH + D_1^{-2} & \alpha R^T \\ \alpha R & -D_2^2 - D_3^2 \end{bmatrix} \begin{bmatrix} \triangle \mathbf{f} \\ -\triangle \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} -\hat{\mathbf{r}}_{c1} \\ -\hat{\mathbf{r}}_b \end{bmatrix}, \tag{3.10}$$

where

$$\hat{\mathbf{r}}_b = \mathbf{r}_b + D_2^2\hat{\mathbf{r}}_{c2} - D_3^2\hat{\mathbf{r}}_{c3}.$$

Eliminating $\triangle \boldsymbol{\lambda}$, we get

$$\left[2H^TH + D_1^{-2} + \alpha^2 R^T(D_2^2 + D_3^2)^{-1}R\right] \triangle \mathbf{f} = -\tilde{\mathbf{r}}_{c1}, \tag{3.11}$$

where

$$\tilde{\mathbf{r}}_{c1} = \hat{\mathbf{r}}_{c1} + \alpha R^T(D_2^2 + D_3^2)^{-1}\hat{\mathbf{r}}_b.$$

Similar to system (2.14), system (3.11) can be viewed as the normal equations of a linear least squares problem, whose coefficient matrix has full column rank. Therefore, the coefficient matrix in (3.11) is symmetric positive definite, and system (3.11) can be solved by the preconditioned conjugate gradient method.

Once (3.11) has been solved, the other unknowns can be calculated by the following equations:

$$\begin{cases} \triangle \boldsymbol{\lambda} = (D_2^2 + D_3^2)^{-1}(-\hat{\mathbf{r}}_b - \alpha R \triangle \mathbf{f}), \\ \triangle \mathbf{v}^+ = D_2^2(-\hat{\mathbf{r}}_{c2} - \triangle \boldsymbol{\lambda}), \\ \triangle \mathbf{v}^- = D_3^2(-\hat{\mathbf{r}}_{c3} + \triangle \boldsymbol{\lambda}), \\ \triangle \mathbf{s} = G \triangle \mathbf{x} - A^T \triangle \boldsymbol{\lambda} + \mathbf{r}_c. \end{cases} \tag{3.12}$$

An initial feasible point can be found in a similar manner as was done for the LAD solution. Moreover, if (3.12) is used to recover the rest of the unknowns after (3.11) is solved approximately, and a proper step length is chosen, then the iterates will stay feasible.

The interior point algorithm for the LMN solution is quite similar to that for the LAD solution. One difference that is worth pointing out is that β^{pri} and β^{dual} are allowed to take different values in the algorithm for the LAD solution but have to be the same in that for the LMN solution.

4. Preconditioning the structured linear systems. As mentioned earlier, linear systems (2.14) and (3.11) become ill-conditioned as the iterates get close to the solution. Thus preconditioners are needed to accelerate the convergence of the conjugate gradient iterations.

For the nonnegative least squares image restoration problem, Saunders [46] used preconditioned LSQR to solve the linear system at each interior point iteration. He used a diagonal preconditioner. However, the effectiveness of diagonal preconditioners is limited in that preconditioned LSQR needed more than 1000 iterations to converge to a desired level of accuracy at the final stages of the interior point algorithm. Our experimental results indicate that the conditions of systems (2.14) and (3.11) tend to be worse than that of the systems in the nonnegative least squares problem. This should be no surprise since our problems are transformed from nonsmooth problems.

4.1. Saddle point preconditioners. For linear systems (2.10), Benzi and Golub [7] proposed using the preconditioned GMRES to solve the equivalent system

$$\begin{bmatrix} D^{-2} & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \triangle \mathbf{x} \\ \triangle \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{r}}_c \\ -\mathbf{r}_b \end{bmatrix}. \tag{4.1}$$

Their preconditioning strategy is based on the symmetric/skew-symmetric splitting of the coefficient matrix, that is, they propose using the following matrix as the preconditioner:

$$\mathcal{M}_\gamma = \begin{bmatrix} D^{-2} + \gamma I & 0 \\ 0 & \gamma I \end{bmatrix} \begin{bmatrix} \gamma I & -A^T \\ A & \gamma I \end{bmatrix}, \tag{4.2}$$

where γ is some positive constant. Linear systems in the form of $\mathcal{M}_\gamma \mathbf{z} = \mathbf{r}$ can be solved by first solving

$$\begin{bmatrix} D^{-2} + \gamma I & 0 \\ 0 & \gamma I \end{bmatrix} \tilde{\mathbf{z}} = \mathbf{r}, \tag{4.3}$$

followed by

$$\begin{bmatrix} \gamma I & -A^T \\ A & \gamma I \end{bmatrix} \mathbf{z} = \tilde{\mathbf{z}}. \tag{4.4}$$

Solving (4.3) is a simple dotwise division. Since our focus is on the shift invariant blur, system (4.4) can be solved by the discrete cosine transform if the blur is symmetric and Neumann boundary conditions are applied; see [36], for instance. This preconditioning technique can also be applied to (2.13), which yields better results in practice.

4.2. Factorized sparse inverse preconditioner. Our experimental results indicate that the most efficient way to compute the Newton step is to apply the factorized sparse inverse preconditioner (FSIP) [27, 8, 47] to systems (2.14) and (3.11). Let B be a symmetric positive definite matrix, and let its Cholesky factorization be $B = CC^T$. The idea of FSIP is to find the lower triangular matrix L with the sparsity pattern $\mathcal{S}$ such that

$$\|I - CL\|_F$$

is minimized, where $\|\cdot\|_F$ denotes the Frobenius norm. Kolotilina and Yeremin [27] showed that L can be obtained by the following algorithm.

Algorithm. Construction of FSIP.
1. Compute $\hat{L}$ with sparsity pattern $\mathcal{S}$ such that $\left[\hat{L}B\right]_{ij} = \delta_{ij}, \quad (i,j) \in \mathcal{S}$.
2. Let $\hat{D} = (\mathrm{diag}(\hat{L}))^{-1}$ and $L = \hat{D}^{1/2}\hat{L}$.

According to this algorithm, n small linear systems need to be solved, where n is the number of rows in the matrix B. These systems can be solved in parallel. Thus the above algorithm is well suited for modern parallel computing.

A special type of FSIP, the factorized banded inverse preconditioner (FBIP), was proposed by Lin, Ng, and Ching [30] for Toeplitz related matrices. We say that $B \in \mathbb{R}^{m\times n}$ has *lower bandwidth* p if $b_{ij} = 0$ whenever $i > j + p$. The *upper bandwidth* of a matrix is defined similarly. Note that by this definition, a tridiagonal matrix has upper and lower bandwidth of 1.

Lin, Ng, and Ching [30] proved the following theorem.

Theorem 4.1. *Let T be a Hermitian Toeplitz matrix. Denote the kth diagonal of T by t_k. Assume the diagonals of T satisfy*

$$|t_k| \le ce^{-\gamma|k|} \tag{4.5}$$

for some $c > 0$ and $\gamma > 0$, or

$$|t_k| \le c(|k| + 1)^{-s} \tag{4.6}$$

for some $c > 0$ and $s > 3/2$. Then for any given $\epsilon > 0$, there exists $p' > 0$ such that for all $p > p'$,

$$\|L_p - C^{-1}\| \le \epsilon,$$

where L_p is the FBIP of T with the lower bandwidth p, and C is the Cholesky factor of T.

This theorem indicates that if a Toeplitz matrix T has certain off-diagonal decay property, then the FBIP will be a good approximation to T^{-1}. Note that if a Toeplitz matrix is banded, then both (4.5) and (4.6) are satisfied trivially.

Lin, Ng, and Ching [30] also considered Toeplitz-related systems of the form $I + T^T DT$, where D is a positive diagonal matrix, and proved the following corollary.

Corollary 4.2. *Let T be a Toeplitz matrix with its diagonals satisfying* (4.5) *or* (4.6). *Let $D = \mathrm{diag}(d_i)$ with $0 < d_i \le d$, and*

$$B = I + T^T DT.$$

Then for any given $\epsilon > 0$, there exists $p' > 0$ such that for all $p > p'$,

$$\|L_p - C^{-1}\| \leq \epsilon,$$

where L_p is the FBIP of B with the lower bandwidth p, and C is the Cholesky factor of B.

The matrices we are trying to precondition are the coefficient matrices in (2.14) and (3.11), that is, $B = D_1^{-2} + H^T(D_2^2 + D_3^2)^{-1}H + \alpha^2 R^T(D_4^2 + D_5^2)^{-1}R$ and $B = 2H^TH + D_1^{-2} + \alpha^2 R^T(D_2^2 + D_3^2)^{-1}R$. Although the focus of this research is on two-dimensional images, we have the following lemma for the one-dimensional problem.

LEMMA 4.3. *Let T be a Toeplitz matrix with its diagonals satisfying* (4.5) *or* (4.6). *Let D_1, D_2, and D_3 be diagonal matrices with positive diagonal entries. Let*

$$B = T^T D_1 T + D_2 + R^T D_3 R \tag{4.7}$$

be a well-conditioned matrix, where R is the first order or the second order difference operator. Then for any given $\epsilon > 0$, there exists $p' > 0$ such that for all $p > p'$,

$$\|L_p - C^{-1}\| \leq \epsilon,$$

where L_p is the FBIP of B with lower bandwidth p, and C is the Cholesky factor of B.

Proof. The proof is similar to the proof of Corollary 4 in [30]. ∎

Lemma 4.3 indicates that the linear system in the one-dimensional LMN and LAD problems can be efficiently preconditioned by the FBIP.

It is important to note that in order to construct the FBIP for B, it is not necessary to explicitly calculate all the elements of B when p is small. The following lemma indicates that only certain diagonals of B need to be calculated.

LEMMA 4.4. *Let L_p be the FBIP with lower bandwidth p for a symmetric matrix B. Then L_p can be determined by the first p bands of the lower triangular part of B.*

Proof. For $i = 1, 2, \ldots, p-1$, the nonzero elements of the ith row of L_p is $L(i, 1 : i)$. They are determined by $B(1 : i, 1 : i)$.

On the other hand, for $i = p, p+1, \ldots, n$, the nonzero elements of the ith row of L_p is $L(i, i+1-p : i)$. They are determined by $B(i+1-p : i, i+1-p : i)$.

Therefore, L_p can be determined by the first upper and lower p bands of B. Since B is symmetric, the result follows. ∎

For two-dimensional problems, we assume that the blurring matrix H has a block-level off-diagonal decay property, and each block also has off-diagonal decay property. This is true if the blurring function decays in spatial domain, or if the support of the blur is small. In this case, we can set the FSIP to be a triangular block banded matrix with each block being a banded matrix. Let p be the block-level lower bandwidth of the FSIP and let q be the lower and the upper bandwidths of each block. Figure 4.1(a) shows the sparsity pattern of an FSIP with $p = 2$ and $q = 1$. Note that if $p = q = 0$, then the FSIP becomes a diagonal preconditioner.

Similar to the one-dimensional case, when constructing the FSIP, it is not necessary to calculate all the elements of the coefficient matrix in (2.14) or (3.11). However, the situation for the two-dimensional case is a bit more complicated.

LEMMA 4.5. *Let L be the FSIP of the coefficient matrix in* (2.14) *or* (3.11), *and let p be the block-level lower bandwidth and q the upper and the lower bandwidths of the blocks of L. Then L can be determined by those elements that are in the lower*

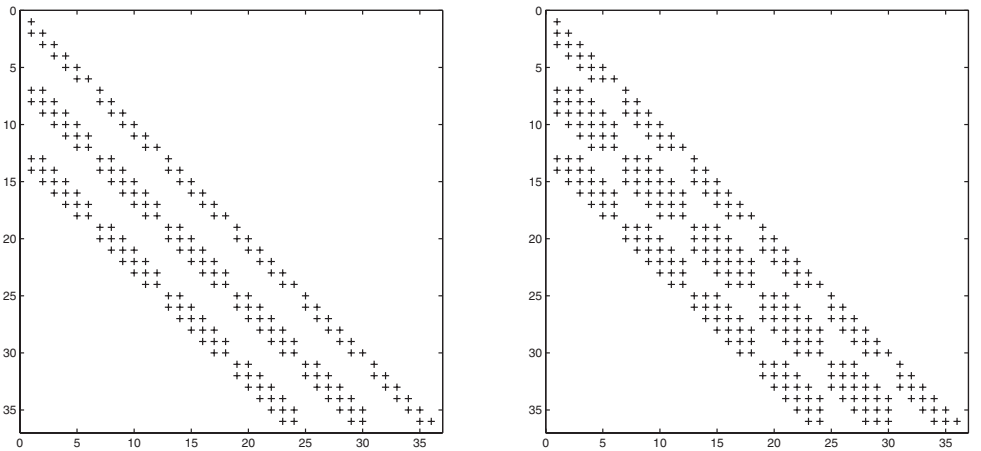


FIG. 4.1. (a) *FSIP with* $p = 2$ *and* $q = 1$. (b) *Sparsity pattern of the elements of the coefficient matrix that needs to be calculated.*

triangular part of the coefficient matrix and have a block-level lower bandwidth of p *with each block having upper and lower bandwidth of* $2q$.

Lemma 4.5 is actually a bit conservative. Figure 4.1(b) shows the sparsity pattern of the elements of the coefficient matrix that needs to be calculated if the FSIP has a sparsity pattern shown in Figure 4.1(a). Note that the blocks in the last block diagonal have lower bandwidth of q. If the original blurring matrix H is sparse enough such that there is no need to use the FFT [31] for matrix multiplications, the cost of calculating these elements is no more than $(p + 1)(2q + 1)$ plain conjugate gradient iterations.

Other than explicitly calculating certain diagonals of the coefficient matrix in (2.14) or (3.11), the cost of constructing the FSIP consists of calculating the lower triangular factor L. Since this step solves mn linear systems of size $(p+1)(2q+1)\times(p+1)(2q+1)$, the complexity of this step is $O(p^3q^3mn)$. Because this cost grows cubically with the increase of p and q, the benefit of the decrease in the number of iterations will be offset by the cost of constructing the preconditioner with the increase of p and q. Therefore, it is important to choose suitable values of p and q in practical applications.

Usually the linear system to be solved is well-conditioned at the first few interior point iterations. Due to the initial cost of constructing the FSIP, it is too expensive to choose a big p or q for early iterates. However, as the iterates get close to the solution, the ill-conditioning of the linear system makes choosing bigger p and q necessary. Therefore, the best strategy is to choose small p and q (say $p = q = 0$) at the beginning, and increase p and q gradually. For details, see experimental results in the next section.

5. Experimental results. Most of our experiments are based on the cameraman image shown in Figure 5.1. The blurring function is chosen to be a two-dimensional Gaussian,

$$h(i, j) = e^{-2(i/3)^2-2(j/3)^2},$$

truncated such that the function has a support of 7×7.

We generate two observed images. For the first one, a Gaussian white noise with standard deviation 0.05 is added. For the second one, we randomly pick 50% of the pixels to be contaminated by noise; the type and standard deviation of the noise is the same as in the first one. The observed images are shown in Figure 5.2. For the



FIG. 5.1. *The original image.*

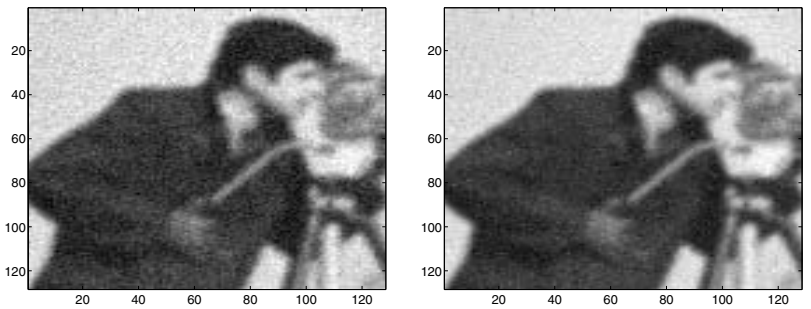


FIG. 5.2. *Observed images:* (a) *all pixels are contaminated by noise;* (b) *only* 50% *of the pixels are contaminated by noise.*

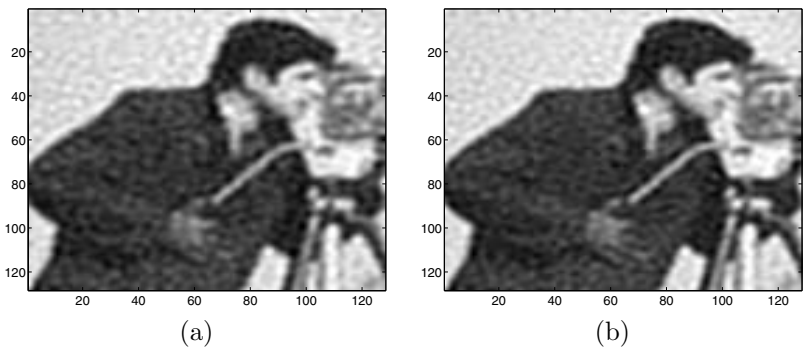


FIG. 5.3. (a) *The least squares solutions for the first observed image, PSNR* = 20.46 *db.* (b) *The least squares solution for the second observed image, PSNR* = 20.87 *db.*

first observed image, since the noise is normal, we consider only the LMN solution. For the second one, we consider both the LMN solution and the LAD solution.

Neumann boundary conditions are used to build the blurring matrix H. Figure 5.3 shows the least squares restoration of the two observed images, respectively. The optimal regularization parameter (the one that yields the highest peak signal-to-noise ratio (PSNR)) is used.

The interior point algorithms are implemented in MATLAB, except the construction of the FSIP, which is written as a MATLAB callable C function. Recall that in order to construct the FSIP, mn small linear systems need to be solved. Doing this

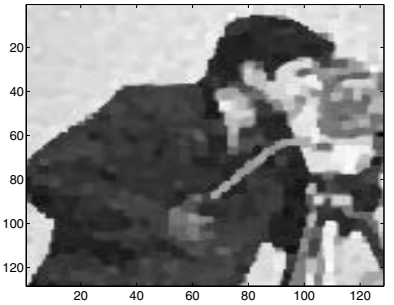


FIG. 5.4. *The LMN solution for the first observed image, PSNR* = 20.78 *db.*

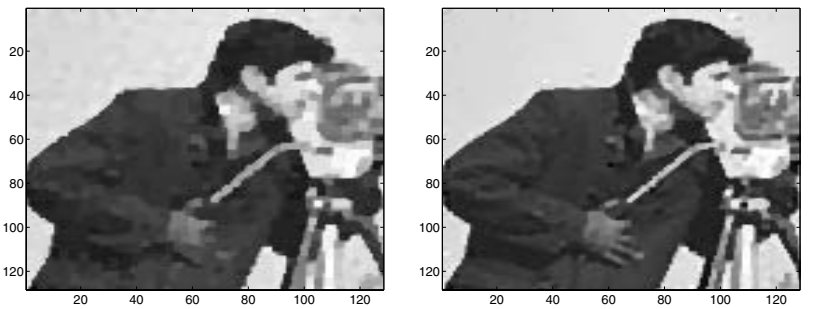


FIG. 5.5. (a) *The LMN solution for the second observed image, PSNR* = 21.30 *db.* (b) *The LAD solution for the second observed image, PSNR* = 22.82 *db.*

in MATLAB would be slow because MATLAB loops are inefficient.

The LMN solution for the first observed image is shown in Figure 5.4. The PSNR of this image indicates it is superior to the least squares solution. It is interesting to note that the LMN solution is quite different from the least squares solution. The least squares solution tends to smooth out the edges but undersmooths flat areas. The LMN solution tends to sharpen the edges and makes flat areas even flatter.

The LAD and LMN solutions for the second observed image is shown in Figure 5.5. As can be seen, both solutions have a higher PSNR than the least squares solution. Furthermore, the LAD solution is superior to the LMN solution.

Similar to the total variation deblurring [43], using the ℓ_1 norm for the regularization term is very effective in recovering "blocky" images. Using the ℓ_1 norm for the residual term yields very good results if only a small portion of the pixels are contaminated by noise. Figure 5.6(a)–(e) show an original "blocky" image, blurred image with 50% of the pixels contaminated by additive noise, least squares restoration, the LMN solution, and the LAD restoration.

Figures 5.7 and 5.8 show the convergence of the primal-dual interior point algorithm for the LMN and LAD solutions, respectively. Theoretically, the interior point iterations should exhibit superlinear convergence. The experimental results do not seem to confirm the theoretical predictions. This can be explained as follows. The linear systems are approximately solved such that the residual is below a certain level. Because the conditions of the linear systems get worse as the iterates get closer to the solution, the error in the calculated search directions becomes bigger, causing poorer convergence. In fact, for small images, if direct methods are used to solve the linear

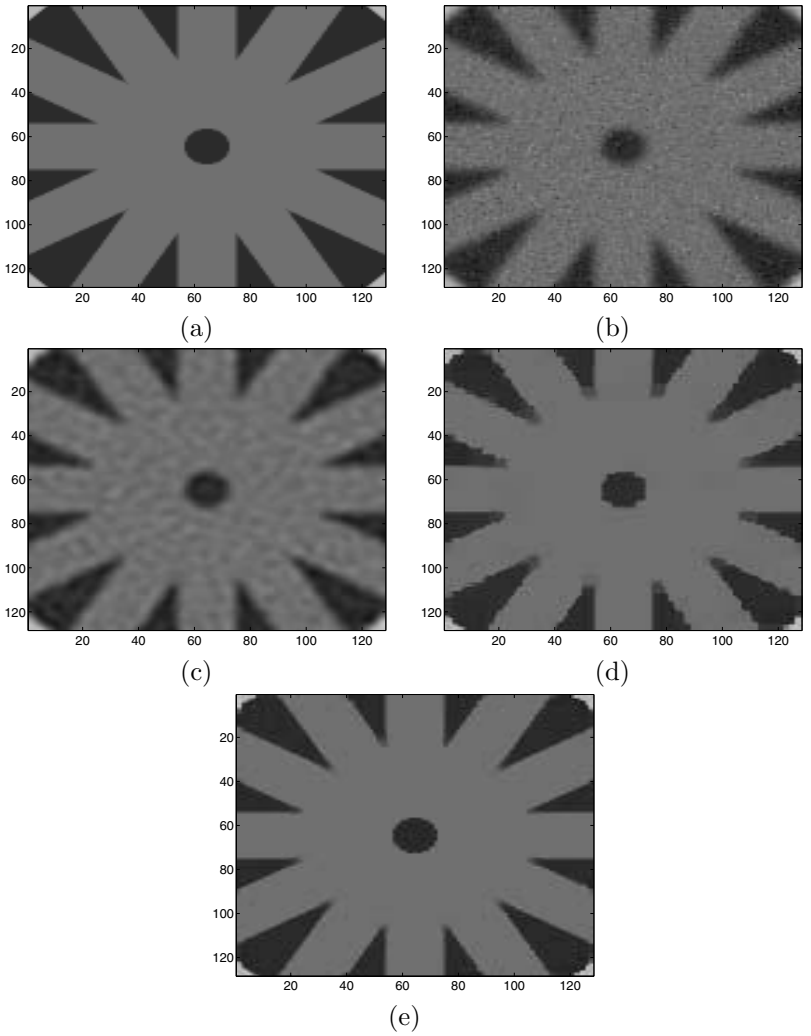


FIG. 5.6. (a) *An original wheel image.* (b) *The observed noisy image with* 50% *of the pixels contaminated by noise.* (c) *The least squares restoration,* $PSNR = 29.25$ *db.* (d) *The LMN solution,* $PSNR = 30.64$ *db.* (e) *The LAD solution,* $PSNR = 35.26$ *db.*

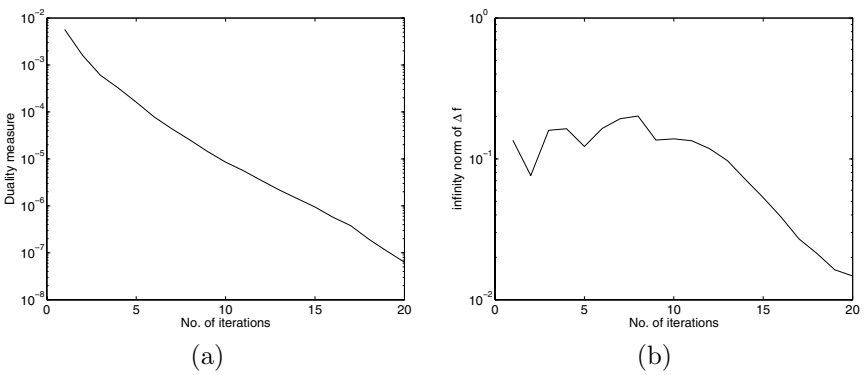
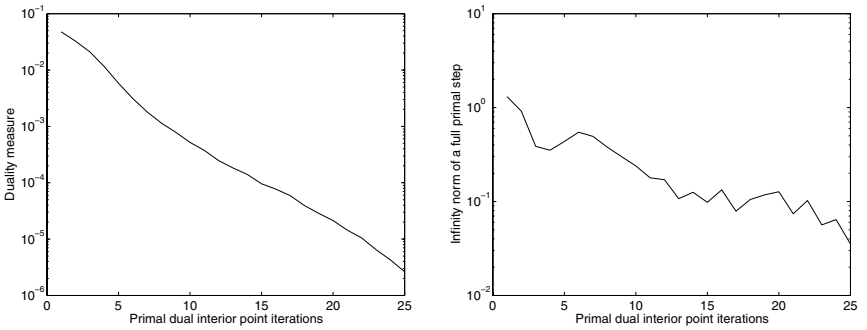


FIG. 5.7. *Convergence of the primal-dual interior point method for the LMN solution:* (a) *the duality measure;* (b) *the infinity norm of* $\Delta \mathbf{f}$.



(a) (b)

FIG. 5.8. *Convergence of the primal-dual interior point method for the LAD solution:* (a) *the duality measure;* (b) *the infinity norm of* $\Delta \mathbf{f}$.

TABLE 5.1

Number of conjugate gradient iterations and the CPU time needed at each interior point iteration for the LAD solution when different preconditioning strategies are used. The preconditioning strategies used include (1) *NOR,* (2.12) *is solved, using a diagonal preconditioner;* (2) *BENZI, the Benzi and Golub preconditioner is applied to* (2.13)*;* (3) *No Pre, plain conjugate gradient is applied to* (2.14)*;* (4) *Diag Pre, diagonal preconditioner is applied to* (2.14)*;* (5) *FSIP*2*, FSIP with* $p = q = 2$ *is applied to* (2.14)*;* (6) *FSIP*3*, FSIP with* $p = q = 3$*;* (7) *FSIP*4*, FSIP with* $p = q = 4$.

PD Itn	NOR	BENZI	No Pre	Diag Pre	FSIP2	FSIP3	FSIP4
10	>3000	295	644	544	48	22	18
	–	161.78	78.68	68.72	15.24	**13.32**	17.08
12	>3000	483	982	804	51	26	21
	–	237.70	125.92	105.53	15.75	**14.14**	18.01
14	>3000	1047	1603	1334	60	31	25
	–	589.68	207.05	175.88	17.42	**15.31**	18.99
16	>3000	1791	2452	1934	77	43	34
	–	969.84	305.14	242.51	20.90	**18.47**	22.14
18	>3000	>2000	>3000	2607	106	54	41
	–	–	–	339.06	25.85	**20.87**	25.11
20	>3000	>2000	>3000	>3000	170	85	58
	–	–	–	–	37.71	**28.18**	28.91

systems, a slightly superlinear convergence is observed.

Table 5.1 shows the number of conjugate gradient iterations and CPU time at each interior point iteration for the LAD solution when different preconditioning strategies are used. The CPU time listed here includes the time for constructing the preconditioners. Table 5.2 shows the same kind of results for the linear systems in the LMN solution. As can be seen, in both cases, the FSIP with proper choices of p and q can significantly speed up the convergence.

We remark that the performance of the FSIP depends on the support of the blurring function and on how fast the function decays in space. The effect of the support of the blurring matrix on the effectiveness of the FSIP is twofold. On one hand, a big support means a relatively dense blurring matrix, therefore, allowing the preconditioner to be relatively dense. On the other hand, a big support usually means a relatively dense preconditioner is needed in order to improve the condition significantly. Table 5.3 shows the performance of the FSIP with a different choice of p and q when the support of the blur is 5×5 (the same Gaussian blur is used). Table 5.4 shows the same kind of information when the support of the blur is 9×9.

TABLE 5.2

Number of conjugate gradient iterations and the CPU time needed at each interior point iteration for the LMN solution when different preconditioners are used. The preconditioning strategies used include (1) *No Pre, plain conjugate gradient is applied to* (3.11)*;* (2) *Diag Pre, diagonal preconditioner is applied to* (3.11)*;* (3) *FSIP*2*, FSIP with* $p = q = 2$ *is applied to* (3.11)*;* (4) *FSIP*3*, FSIP with* $p = q = 3$*;* (5) *FSIP*4*, FSIP with* $p = q = 4$*.*

PD Itn	No Pre	Diag Pre	FSIP2	FSIP3	FSIP4
10	368	180	31	21	15
	45.01	22.94	**6.15**	6.89	10.52
12	618	281	51	32	22
	76.27	35.27	9.71	**9.24**	12.72
14	908	438	85	48	32
	112.37	55.28	16.12	**13.35**	15.86
16	1391	626	139	71	49
	171.36	78.27	25.95	**18.31**	20.84
18	2202	989	225	123	67
	265.25	119.56	40.32	29.99	**26.55**
20	>3000	1814	408	228	120
	–	222.30	72.91	54.15	**41.62**

TABLE 5.3

Number of conjugate gradient iterations and the CPU time needed at each interior point iteration for the LAD solution when different FSIP is used. The support of the blur is 5×5*.*

PD Itn	No Pre	Diag Pre	FSIP2	FSIP3	FSIP4
10	839	669	57	24	14
	79.10	64.95	11.40	**9.39**	11.76
12	1406	1140	96	38	21
	130.74	108.45	17.16	**11.94**	13.40
14	2199	1672	129	52	28
	208.98	164.74	22.31	**14.61**	15.54
16	>3000	2296	175	73	39
	–	222.31	29.55	19.45	**18.64**
18	>3000	>3000	266	106	54
	–	–	43.23	26.57	**22.80**
20	>3000	>3000	378	142	74
	–	–	59.50	32.89	**28.19**

TABLE 5.4

Number of conjugate gradient iterations and the CPU time needed at each interior point iteration for the LAD solution when different FSIP is used. The support of the blur is 9×9*.*

PD Itn	No Pre	Diag Pre	FSIP2	FSIP3	FSIP4
10	1184	578	74	46	30
	198.76	100.03	26.66	**24.41**	25.43
12	1759	940	115	67	46
	286.54	157.77	36.17	**30.23**	31.29
14	2394	1432	175	98	65
	389.36	241.66	49.39	38.63	**38.23**
16	>3000	2093	272	153	97
	–	353.55	71.68	54.44	**48.50**
18	>3000	>3000	409	237	143
	–	–	100.86	76.32	**65.62**
20	>3000	>3000	710	402	236
	–	–	166.37	121.45	**96.83**

TABLE 5.5

Number of conjugate gradient iterations and the CPU time needed at each interior point iteration for the LAD solution when different FSIP is used. The blur is a 7×7 uniform blur.

PD Itn	No Pre	Diag Pre	FSIP2	FSIP3	FSIP4
10	454	346	96	94	88
	57.25	44.83	**23.69**	30.28	38.72
12	731	535	118	113	105
	92.19	69.10	**28.16**	33.76	45.07
14	1241	1022	169	156	141
	158.31	135.20	**37.24**	44.61	55.51
16	1826	1596	250	229	197
	221.76	199.87	**50.20**	61.90	72.77
18	2796	2306	334	287	269
	347.30	295.07	**66.21**	75.45	92.30
20	>3000	>3000	550	464	411
	–	–	**104.73**	117.31	136.25

In general, the faster the blurring function decays in space, the more effective the FSIP is. In that sense, the worst blurring function is a uniform blur. Table 5.5 shows the performance of the FSIP if a 7×7 uniform blur is used.

6. Conclusions. In conclusion, we have proposed efficient algorithms for finding the LAD or LMN solution for the image restoration problem. We formulated these problems as solutions to smooth linear or quadratic programming problems, which are solved by a primal-dual interior point method. The linear system at each interior point iteration is first reduced to a more compact system and then solved by the PCG method. The FSIP is used to speed up the convergence of the conjugate gradient method.

Experimental results indicate that the LAD and LMN solutions have advantages over the least squares solution. We also demonstrated that the FSIP can speed up the convergence considerably if a proper sparsity pattern is chosen.

REFERENCES

[1] N. ABDELMALEK, *Restoration of images with missing high frequency components using quadratic programming*, Appl. Opt., 22 (1983), pp. 2182–2188.

[2] N. ABDELMALEK AND T. KASVAND, *Digital image restoration using quadratic programming*, Appl. Opt., 19 (1980), pp. 3407–3415.

[3] N. ABDELMALEK AND N. OTSU, *Restoration of images with missing high-frequency components by minimizing the $\ell 1$ norm of the solution vector*, Appl. Opt., 24 (1985), pp. 1415–1420.

[4] S. ALLINEY, *Digital filters as absolute norm regularizers*, IEEE Trans. Signal Process., 40 (1992), pp. 1548–1562.

[5] S. ALLINEY, *A property of the minimum vectors of a regularizing functional defined by means of absolute norm*, IEEE Trans. Signal Process., 45 (1997), pp. 913–917.

[6] S. ALLINEY AND S. RUZINSKY, *An algorithm for the minimization of mixed $\ell 1$ and $\ell 2$ norms with application to Bayesian estimation*, IEEE Trans. Signal Process., 42 (1994), pp. 618–627.

[7] M. BENZI AND G. H. GOLUB, *A preconditioner for generalized saddle point problems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 20–41.

[8] M. BENZI, C. MEYER, AND M. TUMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[9] P. BLOOMFIELD AND W. STEIGER, *Least Absolute Deviations: Theory, Applications, and Algorithms*, Birkhäuser, Boston, 1983.

[10] L. CANALES, *L1 Norm Spectral Estimation*, Technical report, Stanford University, Stanford Exploration Project, http://sepwww.stanford.edu/oldreports/sep07/sp07_16.pdf (1975).

[11] A. CHAMBOLLE, *An algorithm for total variation minimization and applications*, J. Math. Imaging Vision, 20 (2004), pp. 89–97.

[12] T. Chan and S. Esedoglu, *Aspects of Total Variation Regularized $\ell 1$ Function Approximation*, Technical report, University of California at Los Angeles, 2004.
[13] G. Chavent and K. Kunisch, *Regularization of linear least squares problems by total bounded variation*, ESAIM: Control Optim. Calc. Var., 2 (1997), pp. 359–376.
[14] P. Combettes, *An adaptive level set method for nondifferentiable constrained image recovery*, IEEE Trans. Image Process., 11 (2002), pp. 1295–1304.
[15] A. Dax, *A row relaxation method for large $\ell 1$ problems*, Linear Algebra Appl., 156 (1991), pp. 793–818.
[16] A. Dax, *On regularized least norm problems*, SIAM J. Optim., 2 (1992), pp. 602–618.
[17] A. Dax, *On row-relaxation methods for large constrained least squares problems*, SIAM J. Sci. Comput., 14 (1993), pp. 570–584.
[18] A. Dax, *A row relaxation method for large ℓ_p least norm problems*, Numer. Linear Algebra Appl., 1 (1994), pp. 247–263.
[19] D. Etter, D. Kuncicky, and D. Hull, *Introduction to Matlab* 6, Prentice-Hall, Upper Saddle River, NJ, 2002.
[20] H. Fu and J. Barlow, *A regularized structured total least squares algorithm for high resolution image reconstruction*, Linear Algebra Appl., 391 (2004), pp. 75–98.
[21] A. Guitton and D. J. Verschuur, *Adaptive subtraction of multiples using the $\ell 1$-norm*, Geophys. Prospecting, 52 (2004), pp. 1–27.
[22] M. Hanke, J. Nagy, and C. Vogel, *Quasi-Newton approach to nonnegative image restorations*, Linear Algebra Appl., 316 (2000), pp. 223–236.
[23] M. Hintermüller and K. Kunisch, *Total bounded variation regularization as a bilaterally constrained optimization problem*, SIAM J. Appl. Math., 64 (2004), pp. 1311–1333.
[24] K. Ito and K. Kunisch, *An active set strategy based on the augmented Lagrangian formulation for image restoration*, ESAIM: Math. Model. Numer. Anal., 33 (1999), pp. 1–21.
[25] T. Kärkkäinen, K. Kunisch, and K. Majava, *Denoising of smooth images using L^1-fitting*, Computing, 74 (2005), pp. 353–376.
[26] Q. Ke and T. Kanade, *Robust Subspace Computation Using $\ell 1$ Norm*, Technical report, School of Computer Science, Carnegie Mellon University, CMU-CS-03-172, 2003.
[27] L. Kolotilina and A. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.
[28] S. Kuo and R. Mammone, *Image restoration by convex projections using adaptive constraints and the $\ell 1$ norm*, IEEE Trans. Signal Process., 40 (1992), pp. 1159–1168.
[29] Y. Li and F. Santosa, *A computational algorithm for minimizing total variation in image restoration*, IEEE Trans. Image Process., 6 (1996), pp. 987–995.
[30] F.-R. Lin, M. K. Ng, and W.-K. Ching, *Factorized banded inverse preconditioners for matrices with Toeplitz structure*, SIAM J. Sci. Comput., 26 (2005), pp. 1852–1870.
[31] C. van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.
[32] R. Mammone and G. Eichmann, *Superresolving image restoration using linear programming*, Appl. Opt., 21 (1983), pp. 496–501.
[33] N. Mastronardi, P. Lemmerling, A. Kalsi, D. O'Leary, and S. van Huffel, *Implementation of regularized structured total least squares algorithms for blind image deblurring*, Linear Algebra Appl., 391 (2004), pp. 203–221.
[34] V. Mesarović, N. Galatsanos, and A. Katsaggelos, *Regularized constrained total least squares image restoration*, IEEE Trans. Image Process., 4 (1995), pp. 1096–1108.
[35] T. Miyashita, *Super-resolved image restoration of holographic images by $\ell 1$-norm minimization with clutter rejection*, Acoustical Imaging, 19 (1992), pp. 77–82.
[36] M. Ng, R. Chan, and W. Tang, *A fast algorithm for deblurring models with Neumann boundary conditions*, SIAM J. Sci. Comput., 21 (1999), pp. 851–866.
[37] M. K. Ng, R. J. Plemmons, and F. Pimentel, *A new approach to constrained total least squares image restoration*, Linear Algebra Appl., 316 (2000), pp. 237–258.
[38] M. Nikolova, *Minimizers of cost-functions involving nonsmooth data-fidelity terms. Application to the processing of outliers*, SIAM J. Numer. Anal., 40 (2002), pp. 965–994.
[39] M. Nikolova, *A variational approach to remove outliers and impulse noise*, J. Math. Imaging Vision, 20 (2004), pp. 99–120.
[40] J. Nocedal and S. Wright, *Numerical Optimization*, Springer, New York, 1999.
[41] C. Paige and M. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.
[42] A. Pruessner and D. O'Leary, *Blind deconvolution using a regularized structured total least norm algorithm*, SIAM J. Matrix Anal. Appl., 24 (2003), pp. 1018–1037.
[43] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.

[44] S. Ruzinsky, *Sequential Least Absolute Deviation Estimation of Autoregressive Parameters*, Ph.D. thesis, Illinois Institute of Technology, Chicago, IL, 1989.
[45] Y. Saad and M. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[46] M. Saunders, *Interior Methods for Optimization with Application to Maximum Entropy Problems*, http://sccm.stanford.edu/csri/Saunders.pdf.
[47] W. Tang, *Toward an effective sparse approximate inverse preconditioner*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 970–986.
[48] A. Tikhonov and V. Arsenin, *Solution of Ill-Posed Problems*, Winston, Washington, DC, 1977.
[49] S. Wright, *Primal-Dual Interior-Point Methods*, SIAM, Philadelphia, 1997.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1903–1928
© 2006 Society for Industrial and Applied Mathematics

FAST ALGORITHMS FOR SPHERICAL HARMONIC EXPANSIONS*

VLADIMIR ROKHLIN† AND MARK TYGERT‡

Abstract. An algorithm is introduced for the rapid evaluation at appropriately chosen nodes on the two-dimensional sphere S^2 in $\mathbb{R}^3$ of functions specified by their spherical harmonic expansions (known as the inverse spherical harmonic transform), and for the evaluation of the coefficients in spherical harmonic expansions of functions specified by their values at appropriately chosen points on S^2 (known as the forward spherical harmonic transform). The procedure is numerically stable and requires an amount of CPU time proportional to $N^2(\log N)\log(1/\varepsilon)$, where N^2 is the number of nodes in the discretization of S^2, and ε is the precision of computations. The performance of the algorithm is illustrated via several numerical examples.

Key words. spherical harmonics, fast algorithms, expansions

AMS subject classification. 65T50

DOI. 10.1137/050623073

1. Introduction. Spherical harmonic expansions are a widely used and well-understood tool of applied mathematics; they are encountered, inter alia, in weather and climate modeling, in the representation of gravitational, topographic, and magnetic data in geophysics, in the numerical solution of certain partial differential equations, etc. The role of spherical harmonic expansions in diagonalizing the Laplacian in three dimensions is similar to the role played by Fourier series expansions in two dimensions.

The spherical harmonic expansion of a function f in $\mathcal{L}^2(S^2)$ is the series of the form

$$f(\theta,\varphi)=\sum_{l=0}^{\infty}\sum_{m=-l}^{l}\alpha_l^m\,P_l^{|m|}(\cos\theta)\,e^{im\varphi}, \tag{1.1}$$

where (θ,φ) are the standard spherical coordinates on the two-dimensional sphere S^2 in $\mathbb{R}^3$, $0\le\theta<\pi$ and $0\le\varphi<2\pi$, and P_l^m is the associated Legendre function of degree l and order m. While the functions $\{P_l^{|m|}(\cos\theta)\,e^{im\theta}\}$ constitute a basis of $\mathcal{L}^2(S^2)$ that is orthogonal, that is,

$$\int_{-1}^{1}P_k^{|m|}(x)\,P_l^{|m|}(x)\,dx=0 \tag{1.2}$$

when $l\ne k$, their norms are not equal to 1; in fact, they are so badly normalized as to be virtually unusable in numerical calculations (see section 2.1 for a detailed discussion of the associated Legendre functions). Therefore, it is customary to replace

*Received by the editors January 21, 2005; accepted for publication (in revised form) April 12, 2005; published electronically February 3, 2006. This work was partially supported by the U.S. DoD under AFOSR grant F49620-03-C-0041 and by a 2001 NDSEG Fellowship. This work was released previously as the technical report [15].

http://www.siam.org/journals/sisc/27-6/62307.html

†Departments of Computer Science and Mathematics, Yale University, New Haven, CT 06511 (vladimir.rokhlin@yale.edu).

‡Department of Applied Mathematics, Yale University, New Haven, CT 06511 (mark.tygert@yale.edu).

expansions of the form (1.1) with expansions of the form

$$f(\theta, \varphi) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} \alpha_l^m \, \overline{P}_l^{|m|}(\cos\theta) \, e^{im\varphi}, \tag{1.3}$$

where $\overline{P}_l^{|m|}$ denotes the normalized version of the associated Legendre function $P_l^{|m|}$, defined on $[-1, 1]$ via the formula

$$\overline{P}_l^{|m|}(x) = (-1)^{|m|} \sqrt{\frac{2l+1}{2} \, \frac{(l-|m|)!}{(l+|m|)!}} \, P_l^{|m|}(x), \tag{1.4}$$

so that

$$\int_{-1}^{1} \left(\overline{P}_l^{|m|}(x) \right)^2 \, dx = 1. \tag{1.5}$$

In numerical practice, the series (1.3) is truncated after a finite number of terms, leading to expressions of the form

$$f(\theta, \varphi) \sim \sum_{l=0}^{N} \sum_{m=-l}^{l} \alpha_l^m \, \overline{P}_l^{|m|}(\cos\theta) \, e^{im\varphi}. \tag{1.6}$$

Formula (1.6) is viewed as an approximation to the function f, and N is called the order of the expansion (1.6). Obviously, the expansion (1.6) contains $(N+1)^2$ terms; the order N required to obtain a prescribed accuracy of the approximation is determined by the complexity of the function f.

Frequently, the need arises to evaluate the coefficients in an expansion of the form (1.6) for a function f given by a table of its values at a collection of appropriately chosen nodes on S^2; conversely, given the coefficients in (1.6), one often needs to evaluate f at a collection of points on S^2. The former is usually called the forward spherical harmonic transform, and the latter is known as the inverse spherical harmonic transform. A standard discretization of S^2 is the "tensor product," consisting of all pairs of the form (θ_k, φ_j), with equispaced nodes $\theta_0, \theta_1, \ldots, \theta_{N-1}, \theta_N$ discretizing the interval $[0, \pi]$, defined by the formula

$$\theta_k = \frac{\pi(k+1/2)}{N+1}, \tag{1.7}$$

and equispaced nodes $\varphi_0, \varphi_1, \ldots, \varphi_{2N-1}, \varphi_{2N}$ discretizing the interval $[0, 2\pi]$, defined by the formula

$$\varphi_j = \frac{2\pi(j+1/2)}{2N+1}. \tag{1.8}$$

This leads immediately to numerical schemes for both the forward and inverse spherical harmonic transforms costing $\mathcal{O}(N^3)$ operations. Indeed, given a function f defined on S^2 by the formula (1.6), one can rewrite (1.6) in the form

$$f(\theta, \varphi) = \sum_{m=-N}^{N} e^{im\varphi} \sum_{l=|m|}^{N} \alpha_l^m \, \overline{P}_l^{|m|}(\cos\theta). \tag{1.9}$$

For a fixed value of θ, each of the inner sums in (1.9) contains no more than $N+1$ terms, and there are $2N+1$ such sums (one for each value of m); since the inverse spherical harmonic transform involves $N+1$ values $\theta_0, \theta_1, \ldots, \theta_{N-1}, \theta_N$, the cost of evaluating all inner sums in (1.9) is $\mathcal{O}(N^3)$. Once all inner sums have been evaluated, evaluation of each outer sum costs $\mathcal{O}(N)$ operations (since each of them contains $2N+1$ terms), and there are $\mathcal{O}(N^2)$ such sums to be evaluated, leading to $\mathcal{O}(N^3)$ CPU time requirements for the evaluation of all outer sums in (1.9). The cost of the evaluation of the whole inverse spherical harmonic transform (in the form (1.9)) is the sum of the costs for the inner and outer sums, and is also $\mathcal{O}(N^3)$; a virtually identical calculation shows that the cost of evaluating the forward spherical harmonic transform is also $\mathcal{O}(N^3)$.

A trivial modification of the scheme described in the preceding paragraph uses the fast Fourier transform to evaluate the outer sums in (1.9), roughly halving the CPU time requirements of the whole procedure. Several other considerations (see, for example, [2], [18]) can be used to reduce the CPU time requirements by a further factor of 4 or so, but there is no simple trick for reducing the asymptotic CPU time requirements of the whole spherical harmonic transform (either forward or inverse) below N^3. In this paper, we introduce algorithms for both forward and inverse spherical harmonic transforms with CPU time requirements proportional to $N^2(\log N)\log(1/\varepsilon)$, where ε is the precision of computations.

The algorithm of this paper is a procedure for the rapid evaluation of the inner sums in expressions of the form (1.9). It is based principally on two observations, as follows.

1. The differential equations defining the functions $\overline{P}_l^m$ with arbitrary positive integer m are very close to the differential equations defining the functions $\overline{P}_l^1$ and $\overline{P}_l^2$.

2. There exist fast algorithms for decomposing functions into and reconstructing functions from sums of the forms

$$f(x) = \sum_{l=0}^{N} p_l^1 \, \overline{P}_l^1(x), \tag{1.10}$$

$$f(x) = \sum_{l=0}^{N} p_l^2 \, \overline{P}_l^2(x). \tag{1.11}$$

We use the connections between the functions $\overline{P}_l^m$ with arbitrary positive integer m and the functions $\overline{P}_l^1$ and $\overline{P}_l^2$ to apply rapidly to arbitrary vectors the matrices converting between expansions of the forms (1.10) and (1.11) and expansions of the form

$$f(x) = \sum_{l=0}^{N} p_l^m \, \overline{P}_l^m(x). \tag{1.12}$$

This step utilizes the observation made in [4] that the $N \times N$ matrix of eigenvectors of the sum of a diagonal matrix and a semiseparable matrix (see section 2.4 for the definition of a semiseparable matrix) can be applied to an arbitrary vector of length N for a cost proportional to $N(\log N)\log(1/\varepsilon)$ operations, where ε is the precision of computations.

During the last several years, the interest in fast transforms has been growing, stimulated by the combination of recent progress in fast algorithms of various kinds

with the importance of the fast Fourier transform in computational mathematics, electrical engineering, etc., and by the success of various types of multilevel computational techniques. In particular, several prior attempts have been made to construct numerically stable fast spherical harmonic transforms.

Schemes have been constructed that are fast, but unstable in floating point arithmetic; [10], [9], [12] elucidate this approach, and the ongoing efforts to stabilize algorithms of this type. Other schemes have been proposed that are instead approximate, performing calculations up to an arbitrary but fixed precision ε. In [13], a procedure is described with the asymptotic CPU time estimate $\mathcal{O}(N^{5/2}(\log N)\log(1/\varepsilon))$ for either the forward or the inverse spherical harmonic transform. Also in [13], an algorithm is proposed whose asymptotic CPU time requirements are conjectured to be $\mathcal{O}(N^2(\log N)^2\log(1/\varepsilon))$; this estimate is not fully proved in [13], and the numerical results presented there do not unequivocally support it. A different approach is proposed in [16], [17]; like the approaches of both [13] and the present paper, the scheme of [17] is based on analytical (rather than algebraic) techniques. The asymptotic CPU time requirements of the procedure described in [17] are $\mathcal{O}(N^2(\log N)\log(1/\varepsilon))$; while [16], [17] report work in progress, we expect this attack to be successful. Finally, it should be observed that some of the papers describing fast spherical transforms compare the actual timings obtained in them to those produced by the straightforward matrix-vector multiplication scheme, while others compare to the "semi-naive" algorithm described in [10], [2]. It should be kept in mind that the "semi-naive" scheme is about twice as fast as the straightforward one.

It should also be observed that the algorithm of this paper is not intended to be used as a component in what are known as "spherical filters" (see, for example, [11]); special purpose schemes of the type originated in [11] tend to be considerably faster than the scheme presented here.

The structure of this paper is as follows. In section 2, we summarize a number of facts (from both mathematical and numerical analysis) to be used in the rest of the paper; all of the content of section 2 is either well known or follows easily from well-known facts. In section 3, we build the analytical apparatus to be used in the construction of the algorithms of this paper. Section 4 contains an informal description of the algorithm, and in section 5 the procedure is described in detail. The performance of the scheme is illustrated with numerical examples in section 6, and section 7 contains a discussion of possible applications of the approach of this paper in other environments.

2. Mathematical and numerical preliminaries. In this section, we summarize several facts from mathematical and numerical analysis. Please note that in this section and throughout this paper, the variable x always takes arbitrary values in $[-1,\,1]$, θ takes values in $[0,\,\pi]$, and φ takes values in $[0,\,2\pi]$. We will always use the term "eigenvector" to mean "normalized eigenvector."

2.1. Spherical harmonics and associated Legendre functions. In this section, we summarize a number of properties of spherical harmonics and associated Legendre functions; all of these can be found, for example, in [1].

The coefficients in the spherical harmonic expansion (1.6) of a function f in $\mathcal{L}^2(S^2)$ are given by the formula

$$\alpha_l^m = \int_0^\pi \int_0^{2\pi} \overline{P}_l^{|m|}(\cos\theta)\, e^{-im\varphi}\, f(\theta,\,\varphi)\, \sin\theta\, d\varphi\, d\theta. \tag{2.1}$$

For the forward spherical harmonic transform of order N, we have to compute the coefficients (2.1) from the values $f(\theta_k, \varphi_j)$, where θ_0, $\theta_1, \ldots, \theta_{N-1}$, θ_N are defined in (1.7), and $\varphi_0, \varphi_1, \ldots, \varphi_{2N-1}, \varphi_{2N}$ are defined in (1.8). For a cost of $\mathcal{O}(N^2 \log N)$, we use the fast Fourier transform to obtain the $2(2N+1)(N+1)$ values $g^m(\theta_k)$ and $h^m(\theta_k)$ ($m = -N, -N+1, \ldots, N-1, N$; $k = 0, 1, \ldots, N-1, N$) of the functions g^m and h^m defined on $[0, \pi]$ by the formulae

$$g^m(\theta) = \sum_{j=0}^{2N} \cos(m\,\varphi_j)\, f(\theta, \varphi_j), \tag{2.2}$$

$$h^m(\theta) = \sum_{j=0}^{2N} \sin(m\,\varphi_j)\, f(\theta, \varphi_j). \tag{2.3}$$

We then evaluate the coefficients (2.1) via the formula

$$\alpha_l^m = \int_0^\pi \overline{P}_l^{|m|}(\cos\theta)\, g^m(\theta)\, \sin\theta\, d\theta - i \int_0^\pi \overline{P}_l^{|m|}(\cos\theta)\, h^m(\theta)\, \sin\theta\, d\theta. \tag{2.4}$$

To evaluate the integrals in (2.4), we have to convert the values $f^m(\cos\theta_k)$ into the coefficients p_m^m, $p_{m+1}^m, \ldots, p_{N-1}^m$, p_N^m in expansions of functions f^m on $[-1, 1]$ of the form

$$f^m(x) = \sum_{l=m}^{N} p_l^m\, \overline{P}_l^m(x), \tag{2.5}$$

$$p_l^m = \int_{-1}^{1} \overline{P}_l^m(x)\, f^m(x)\, dx, \tag{2.6}$$

where m is any integer with $0 \le m \le N$.

The principal purpose of this paper is the construction of a "fast" scheme for computing the coefficients (2.6) from the values $f^m(\cos\theta_k)$, and for computing the inverse of this transformation, that is, for computing the values $f^m(\cos\theta_k)$ of the function f^m defined in (2.5) from the coefficients (2.6).

For any nonnegative integers l and m with $m \le l$, the associated Legendre function P_l^m on $[-1, 1]$ is defined by the formula

$$P_l^m(x) = (-1)^m \sqrt{1-x^2}^m\, \frac{d^m}{dx^m} P_l(x), \tag{2.7}$$

where P_l is the Legendre polynomial of degree l. Obviously, P_l^m is a polynomial when m is even and a polynomial multiplied by $\sqrt{1-x^2}$ when m is odd.

For any nonnegative integer m, we define the differential operator L_m by the formula

$$L_m(f)(x) = -\frac{d}{dx}\left((1-x^2)\,\frac{d}{dx} f(x)\right) + \frac{m^2}{1-x^2}\, f(x) \tag{2.8}$$

for any function f on $[-1, 1]$ with a continuous second derivative. For any nonnegative integers l and m with $m \le l$, the function $\overline{P}_l^m$ satisfies the differential equation

$$L_m\left(\overline{P}_l^m\right)(x) = l(l+1)\, \overline{P}_l^m(x), \tag{2.9}$$

where the differential operator L_m is defined in (2.8).

For any integers l and m with $0 \le m \le l$ and $l > 0$,

$$(2l+1)\, x\, P_l^m(x) = (l+m)\, P_{l-1}^m(x) + (l-m+1)\, P_{l+1}^m(x). \tag{2.10}$$

For any nonnegative integers l and m with $1 \le m \le l$,

$$\int_{-1}^{1} \overline{P}_l^m(x)\, \frac{1}{1-x^2}\, \overline{P}_l^m(x)\, dx = \frac{2l+1}{2m}. \tag{2.11}$$

LEMMA 2.1. *Suppose that l and m are even integers such that $2 \le m \le l$. Then, there exist $l/2$ real numbers ξ_0, $\xi_1, \ldots, \xi_{l/2-2}$, $\xi_{l/2-1}$ such that*

$$\overline{P}_l^m(x) = \sum_{k=0}^{l/2-1} \xi_k\, \overline{P}_{2k+2}^2(x). \tag{2.12}$$

LEMMA 2.2. *Suppose that l and m are integers such that m is even, l is odd, and $2 \le m < l$. Then, there exist $(l-1)/2$ real numbers ξ_0, $\xi_1, \ldots, \xi_{(l-1)/2-2}$, $\xi_{(l-1)/2-1}$ such that*

$$\overline{P}_l^m(x) = \sum_{k=0}^{(l-1)/2-1} \xi_k\, \overline{P}_{2k+3}^2(x). \tag{2.13}$$

LEMMA 2.3. *Suppose that l and m are integers such that m is odd, l is even, and $1 \le m < l$. Then, there exist $l/2$ real numbers ξ_0, $\xi_1, \ldots, \xi_{l/2-2}$, $\xi_{l/2-1}$ such that*

$$\overline{P}_l^m(x) = \sum_{k=0}^{l/2-1} \xi_k\, \overline{P}_{2k+2}^1(x). \tag{2.14}$$

LEMMA 2.4. *Suppose that l and m are odd integers such that $1 \le m \le l$. Then, there exist $(l+1)/2$ real numbers ξ_0, $\xi_1, \ldots, \xi_{(l+1)/2-2}$, $\xi_{(l+1)/2-1}$ such that*

$$\overline{P}_l^m(x) = \sum_{k=0}^{(l+1)/2-1} \xi_k\, \overline{P}_{2k+1}^1(x). \tag{2.15}$$

2.2. Chebyshev polynomials. In this section, we cite the existence of a fast algorithm for computing with Chebyshev polynomials.

For any nonnegative integer k, we define T_k to be the Chebyshev polynomial of degree k of the first kind, defined by the formula

$$T_k(\cos\theta) = \cos(k\theta) \tag{2.16}$$

for any real θ, and U_k to be the Chebyshev polynomial of degree k of the second kind, defined by the formula

$$U_k(\cos\theta) = \frac{\sin((k+1)\theta)}{\sin\theta} \tag{2.17}$$

for any real θ.

The following observation cites the relationship between the fast Fourier transform and expansions in series of Chebyshev polynomials.

Observation 2.5. Suppose that $N \geq 0$ is an integer, $c_0,\ c_1, \ldots, c_{N-1},\ c_N$ and $u_0,\ u_1, \ldots, u_{N-1},\ u_N$ are real numbers, and f and g are the functions on $[-1,\ 1]$ defined by the formulae

$$f(x) = \sum_{k=0}^{N} c_k\, T_k(x), \tag{2.18}$$

$$g(x) = \sum_{k=0}^{N} u_k \sqrt{1-x^2}\, U_k(x). \tag{2.19}$$

Then, there exists an algorithm which uses $\mathcal{O}(N \log N)$ operations to convert the coefficients $c_0,\ c_1, \ldots, c_{N-1},\ c_N$ into the values $f(x_0),\ f(x_1), \ldots, f(x_{N-1}),\ f(x_N)$, and to convert the coefficients $u_0, u_1, \ldots, u_{N-1}, u_N$ into the values $g(x_0), g(x_1), \ldots, g(x_{N-1}),$ $g(x_N)$, where the sampling locations $x_0, x_1, \ldots, x_{N-1}, x_N$ are defined by the formula

$$x_k = \cos\left(\frac{\pi(k+1/2)}{N+1}\right). \tag{2.20}$$

Moreover, there exists an algorithm which uses $\mathcal{O}(N \log N)$ operations to convert the values $f(x_0),\ f(x_1), \ldots, f(x_{N-1}),\ f(x_N)$ into the coefficients $c_0,\ c_1, \ldots, c_{N-1},\ c_N$, and to convert the values $g(x_0),\ g(x_1), \ldots, g(x_{N-1}),\ g(x_N)$ into the coefficients $u_0,\ u_1, \ldots,$ u_{N-1}, u_N (see, for example, [14]).

2.3. Associated Legendre functions of low orders. In this section, we summarize certain simple relationships between Chebyshev polynomials and associated Legendre functions of orders 1 and 2. These relationships are a straightforward consequence of formulae 7.112.1, 8.339.1, 8.339.2, 8.700.1, 8.752.1, 8.826.1, 8.828.1, 8.832.2, and 8.911.4 of [7].

We define the function Λ on $[0,\ \infty)$ by the formula

$$\Lambda(z) = \frac{\Gamma(z+1/2)}{\Gamma(z+1)}, \tag{2.21}$$

where Γ is the Euler gamma function.

For any integer $n \geq 1$ and $l, k = 0,\ 1, \ldots, n-2,\ n-1$, we define the entry $A^{n,1,+}_{l,k}$ of the $n \times n$ matrix $A^{n,1,+}$ by the formulae

$$A^{n,1,+}_{l,k} = -\frac{4l+3}{2(2k+2l+3)(2k-2l-1)}\, \Lambda\,(k-l)\ \Lambda\left(\frac{2k+2l+1}{2}\right) \times \sqrt{\frac{4(l+1)(2l+1)}{4l+3}} \tag{2.22}$$

when $k \geq l$, and

$$A^{n,1,+}_{l,k} = 0 \tag{2.23}$$

otherwise (when $k < l$).

For any integer $n \geq 1$ and $l, k = 0,\ 1, \ldots, n-2,\ n-1$, we define the entry $A^{n,1,-}_{l,k}$ of the $n \times n$ matrix $A^{n,1,-}$ by the formulae

$$A^{n,1,-}_{l,k} = -\frac{4l+5}{2(2k+2l+5)(2k-2l-1)}\, \Lambda\,(k-l)\ \Lambda\left(\frac{2k+2l+3}{2}\right) \times \sqrt{\frac{4(l+1)(2l+3)}{4l+5}} \tag{2.24}$$

when $k \geq l$, and

$$A^{n,1,-}_{l,k} = 0 \tag{2.25}$$

otherwise (when $k < l$).

For any integer $n \geq 1$ and $k, l = 0, 1, \ldots, n-2, n-1$, we define the entry $B^{n,1,+}_{k,l}$ of the $n \times n$ matrix $B^{n,1,+}$ by the formulae

$$B^{n,1,+}_{k,l} = 2 \, \frac{2k+1}{\pi} \, \Lambda\,(l-k) \, \Lambda\,(l+k+1) \, \sqrt{\frac{4l+3}{4(l+1)(2l+1)}} \tag{2.26}$$

when $k \leq l$, and

$$B^{n,1,+}_{k,l} = 0 \tag{2.27}$$

otherwise (when $k > l$).

For any integer $n \geq 1$ and $k, l = 0, 1, \ldots, n-2, n-1$, we define the entry $B^{n,1,-}_{k,l}$ of the $n \times n$ matrix $B^{n,1,-}$ by the formulae

$$B^{n,1,-}_{k,l} = 4 \, \frac{k+1}{\pi} \, \Lambda\,(l-k) \, \Lambda\,(l+k+2) \, \sqrt{\frac{4l+5}{4(l+1)(2l+3)}} \tag{2.28}$$

when $k \leq l$, and

$$B^{n,1,-}_{k,l} = 0 \tag{2.29}$$

otherwise (when $k > l$).

For any integer $n \geq 1$, $l = 0, 1, \ldots, n-2, n-1$, and $k = 0, 1, \ldots, n-1, n$, we define the entry $A^{n,2,+}_{l,k}$ of the $n \times (n+1)$ matrix $A^{n,2,+}$ by the formulae

$$\begin{aligned} A^{n,2,+}_{l,k} = \Bigg(& 4 + \frac{2k \, (6(l+1)(2l+3) - 2(2k-1)(2k+1))}{(2k+2l+3)(2k-2l-3)} \, \Lambda\,(k-l-1) \\ & \times \Lambda\left(\frac{2k+2l+1}{2}\right) \Bigg) \sqrt{\frac{4l+5}{8(l+1)(l+2)(2l+1)(2l+3)}} \end{aligned} \tag{2.30}$$

when $k \geq l+1$, and

$$A^{n,2,+}_{l,k} = 4 \sqrt{\frac{4l+5}{8(l+1)(l+2)(2l+1)(2l+3)}} \tag{2.31}$$

otherwise (when $k < l+1$).

For any integer $n \geq 1$, $l = 0, 1, \ldots, n-2, n-1$, and $k = 0, 1, \ldots, n-1, n$, we define the entry $A^{n,2,-}_{l,k}$ of the $n \times (n+1)$ matrix $A^{n,2,-}$ by the formulae

$$\begin{aligned} A^{n,2,-}_{l,k} = \Bigg(& 4 + \frac{(2k+1) \, (6(l+2)(2l+3) - 8k(k+1))}{(2k+2l+5)(2k-2l-3)} \, \Lambda\,(k-l-1) \\ & \times \Lambda\left(\frac{2k+2l+3}{2}\right) \Bigg) \sqrt{\frac{4l+7}{8(l+1)(l+2)(2l+3)(2l+5)}} \end{aligned} \tag{2.32}$$

when $k \geq l+1$, and

$$A_{l,k}^{n,2,-} = 4\sqrt{\frac{4l+7}{8(l+1)(l+2)(2l+3)(2l+5)}} \tag{2.33}$$

otherwise (when $k < l+1$).

For any integer $n \geq 1$, $k = 0, 1, \ldots, n-1, n$, and $l = 0, 1, \ldots, n-2, n-1$, we define the entry $B_{k,l}^{n,2,+}$ of the $(n+1) \times n$ matrix $B^{n,2,+}$ by the formulae

$$B_{k,l}^{n,2,+} = \frac{2(l+1)(2l+3) - 8k^2}{\pi} \Lambda(l-k+1)\, \Lambda(l+k+1) \times \sqrt{\frac{4l+5}{8(l+1)(l+2)(2l+1)(2l+3)}} \tag{2.34}$$

when $k = 0$,

$$B_{k,l}^{n,2,+} = 2\,\frac{2(l+1)(2l+3) - 8k^2}{\pi} \Lambda(l-k+1)\, \Lambda(l+k+1) \times \sqrt{\frac{4l+5}{8(l+1)(l+2)(2l+1)(2l+3)}} \tag{2.35}$$

when $0 < k \leq l+1$, and

$$B_{k,l}^{n,2,+} = 0 \tag{2.36}$$

otherwise (when $k > l+1$).

For any integer $n \geq 1$, $k = 0, 1, \ldots, n-1, n$, and $l = 0, 1, \ldots, n-2, n-1$, we define the entry $B_{k,l}^{n,2,-}$ of the $(n+1) \times n$ matrix $B^{n,2,-}$ by the formulae

$$B_{k,l}^{n,2,-} = \frac{4(l+2)(2l+3) - 4(2k+1)^2}{\pi} \Lambda(l-k+1)\, \Lambda(l+k+2) \times \sqrt{\frac{4l+7}{8(l+1)(l+2)(2l+3)(2l+5)}} \tag{2.37}$$

when $k \leq l+1$, and

$$B_{k,l}^{n,2,-} = 0 \tag{2.38}$$

otherwise (when $k > l+1$).

The following four lemmas are proven via mechanical but rather tedious manipulations of formulae 7.112.1, 8.339.1, 8.339.2, 8.700.1, 8.752.1, 8.826.1, 8.828.1, 8.832.2, and 8.911.4 of [7].

The following lemma provides explicit expressions for the matrix $B^{n,1,+}$ converting coefficients in linear combinations of associated Legendre functions of order 1 of odd degrees into coefficients in linear combinations of even Chebyshev polynomials of the second kind, scaled by $\sqrt{1-x^2}$, and for the matrix $A^{n,1,+}$ converting the latter into the former.

LEMMA 2.6. *Suppose that $n \geq 1$ is an integer, $q = (q_0, q_1, \ldots, q_{n-2}, q_{n-1})^{\mathrm{T}}$ is a real vector, and f is the function on $[-1, 1]$ defined by the formula*

$$f(x) = \sum_{l=0}^{n-1} q_l\, \overline{P}_{2l+1}^{1}(x). \tag{2.39}$$

Then,

$$f(x) = \sum_{k=0}^{n-1} u_k \sqrt{1-x^2}\, U_{2k}(x), \tag{2.40}$$

where $u = (u_0, u_1, \ldots, u_{n-2}, u_{n-1})^{\mathrm{T}}$ *is the real vector defined by the formula*

$$u = B^{n,1,+}\, q, \tag{2.41}$$

and $B^{n,1,+}$ *is defined in* (2.21), (2.26), *and* (2.27). *Furthermore,*

$$q = A^{n,1,+}\, u, \tag{2.42}$$

where $A^{n,1,+}$ *is defined in* (2.21), (2.22), *and* (2.23).

The following lemma provides explicit expressions for the matrix $B^{n,1,-}$ converting coefficients in linear combinations of associated Legendre functions of order 1 of even degrees into coefficients in linear combinations of odd Chebyshev polynomials of the second kind, scaled by $\sqrt{1-x^2}$, and for the matrix $A^{n,1,-}$ converting the latter into the former.

LEMMA 2.7. *Suppose that* $n \geq 1$ *is an integer,* $q = (q_0, q_1, \ldots, q_{n-2}, q_{n-1})^{\mathrm{T}}$ *is a real vector, and* f *is the function on* $[-1, 1]$ *defined by the formula*

$$f(x) = \sum_{l=0}^{n-1} q_l\, \overline{P}^{\,1}_{2l+2}(x). \tag{2.43}$$

Then,

$$f(x) = \sum_{k=0}^{n-1} u_k \sqrt{1-x^2}\, U_{2k+1}(x), \tag{2.44}$$

where $u = (u_0, u_1, \ldots, u_{n-2}, u_{n-1})^{\mathrm{T}}$ *is the real vector defined by the formula*

$$u = B^{n,1,-}\, q, \tag{2.45}$$

and $B^{n,1,-}$ *is defined in* (2.21), (2.28), *and* (2.29). *Furthermore,*

$$q = A^{n,1,-}\, u, \tag{2.46}$$

where $A^{n,1,-}$ *is defined in* (2.21), (2.24), *and* (2.25).

The following lemma provides explicit expressions for the matrix $B^{n,2,+}$ converting coefficients in linear combinations of associated Legendre functions of order 2 of even degrees into coefficients in linear combinations of even Chebyshev polynomials of the first kind, and for the matrix $A^{n,2,+}$ converting the latter into the former.

LEMMA 2.8. *Suppose that* $n \geq 1$ *is an integer,* $p = (p_0, p_1, \ldots, p_{n-2}, p_{n-1})^{\mathrm{T}}$ *is a real vector, and* f *is the function on* $[-1, 1]$ *defined by the formula*

$$f(x) = \sum_{l=0}^{n-1} p_l\, \overline{P}^{\,2}_{2l+2}(x). \tag{2.47}$$

Then,

$$f(x) = \sum_{k=0}^{n} c_k\, T_{2k}(x), \tag{2.48}$$

where $c = (c_0, c_1, \ldots, c_{n-1}, c_n)^{\mathrm{T}}$ *is the real vector defined by the formula*

$$c = B^{n,2,+}\, p, \tag{2.49}$$

and $B^{n,2,+}$ *is defined in* (2.21), (2.34), (2.35), *and* (2.36). *Furthermore,*

$$p = A^{n,2,+}\, c, \tag{2.50}$$

where $A^{n,2,+}$ *is defined in* (2.21), (2.30), *and* (2.31).

The following lemma provides explicit expressions for the matrix $B^{n,2,-}$ converting coefficients in linear combinations of associated Legendre functions of order 2 of odd degrees into coefficients in linear combinations of odd Chebyshev polynomials of the first kind, and for the matrix $A^{n,2,-}$ converting the latter into the former.

LEMMA 2.9. *Suppose that* $n \geq 1$ *is an integer,* $p = (p_0, p_1, \ldots, p_{n-2}, p_{n-1})^{\mathrm{T}}$ *is a real vector, and* f *is the function on* $[-1, 1]$ *defined by the formula*

$$f(x) = \sum_{l=0}^{n-1} p_l\, \overline{P}^2_{2l+3}(x). \tag{2.51}$$

Then,

$$f(x) = \sum_{k=0}^{n} c_k\, T_{2k+1}(x), \tag{2.52}$$

where $c = (c_0, c_1, \ldots, c_{n-1}, c_n)^{\mathrm{T}}$ *is the real vector defined by the formula*

$$c = B^{n,2,-}\, p, \tag{2.53}$$

and $B^{n,2,-}$ *is defined in* (2.21), (2.37), *and* (2.38). *Furthermore,*

$$p = A^{n,2,-}\, c, \tag{2.54}$$

where $A^{n,2,-}$ *is defined in* (2.21), (2.32), *and* (2.33).

Observation 2.10. Suppose that $n \geq 1$ is an integer. Then, there exists an algorithm which uses $\mathcal{O}(n \log(n/\varepsilon))$ operations to apply to an arbitrary vector any of the matrices $A^{n,1,+}$, $B^{n,1,+}$, $A^{n,1,-}$, $B^{n,1,-}$, $A^{n,2,+}$, $B^{n,2,+}$, $A^{n,2,-}$, and $B^{n,2,-}$ defined in (2.21)–(2.38), where ε is the precision of computations (see [3]).

2.4. Semiseparable matrices. For any integer $n > 0$, a semiseparable real $n \times n$ matrix S is a matrix whose entry $S_{j,k}$ is given by the formulae

$$S_{j,k} = a_j\, b_k \tag{2.55}$$

when $j \leq k$, and

$$S_{j,k} = a_k\, b_j \tag{2.56}$$

when $j > k$, where $a = (a_0, a_1, \ldots, a_{n-2}, a_{n-1})^{\mathrm{T}}$ and $b = (b_0, b_1, \ldots, b_{n-2}, b_{n-1})^{\mathrm{T}}$ are real vectors.

Matrices of the form

$$G = D + S, \tag{2.57}$$

where D is a diagonal real matrix and S is a semiseparable real matrix, will be encountered repeatedly throughout this paper. The matrix U of eigenvectors of the matrix G in (2.57) will be particularly important; U is orthogonal and diagonalizes G, so that

$$U^{\mathrm{T}} G U = \Lambda, \tag{2.58}$$

where Λ is a diagonal real matrix.

The principal numerical tool of this paper is the following observation, made in [8] and [4].

Observation 2.11. The matrices U and U^{T} in (2.58) can be applied to an arbitrary vector of length N for a cost of $\mathcal{O}(N(\log N)\log(1/\varepsilon))$ operations, where ε is the precision of computations. More precisely, there exists a constant C independent of N, of ε, and of the particular matrix G in (2.58) such that the matrices U and U^{T} can be applied for a cost of at most $C\,N(\log N)\log(1/\varepsilon)$ operations.

Remark 2.12. Strictly speaking, only the numerical apparatus behind Observation 2.11 is constructed in [4]. However, the observation itself is stated explicitly in a very similar environment in [8]. In our implementation, we used a minor modification of the apparatus in [4], to be reported at a later date.

3. Analytical apparatus. In this section, we construct the principal analytical tools used in this paper.

In section 3.1, we observe that when the function $\overline{P}_l^m$ is represented as a linear combination of functions $\overline{P}_j^1$ or $\overline{P}_j^2$ (depending on whether m is even or odd), the Sturm–Liouville problem (2.9) becomes an eigenvector problem for the matrix G in (2.57). Thus, according to Observation 2.11, there exists an algorithm that uses $\mathcal{O}\left(N(\log N)\log(1/\varepsilon)\right)$ operations to apply the matrices U and U^{T} in (2.58) to arbitrary vectors of length N, where ε is the precision of computations.

In section 3.2, we observe that the problem of evaluating expansions of the form (1.12) can be reduced to the problem of evaluating expansions of the forms (1.10) and (1.11), via the matrices U and U^{T} in (2.58). These matrices can be applied to arbitrary vectors efficiently, due to Observation 2.11.

3.1. Associated Legendre differential equations in terms of associated Legendre functions of low orders. For any even integers n and m with $2 \le m \le n$, and for $j, k = 0,\, 1, \ldots, n/2-2,\, n/2-1$, we define the entry $G^{n,m}_{j,k}$ of the $n/2 \times n/2$ matrix $G^{n,m}$ by the formula

$$G^{n,m}_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+2}(x)\;\; L_m\left(\overline{P}^2_{2k+2}\right)(x)\,dx, \tag{3.1}$$

where the differential operator L_m is defined in (2.8).

For any odd integer n and even integer m with $2 \le m < n$, and for $j, k = 0,\, 1, \ldots, (n-1)/2-2,\, (n-1)/2-1$, we define the entry $G^{n,m}_{j,k}$ of the $(n-1)/2 \times (n-1)/2$ matrix $G^{n,m}$ by the formula

$$G^{n,m}_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+3}(x)\;\; L_m\left(\overline{P}^2_{2k+3}\right)(x)\,dx, \tag{3.2}$$

where the differential operator L_m is defined in (2.8).

For any even integer n and odd integer m with $1 \le m < n$, and for $j,k = 0, 1, \ldots, n/2-2, n/2-1$, we define the entry $G^{n,m}_{j,k}$ of the $n/2 \times n/2$ matrix $G^{n,m}$ by the formula

$$G^{n,m}_{j,k} = \int_{-1}^{1} \overline{P}^1_{2j+2}(x)\ L_m\left(\overline{P}^1_{2k+2}\right)(x)\,dx, \tag{3.3}$$

where the differential operator L_m is defined in (2.8).

For any odd integers n and m with $1 \le m \le n$, and for $j,k = 0, 1, \ldots, (n+1)/2-2, (n+1)/2-1$, we define the entry $G^{n,m}_{j,k}$ of the $(n+1)/2 \times (n+1)/2$ matrix $G^{n,m}$ by the formula

$$G^{n,m}_{j,k} = \int_{-1}^{1} \overline{P}^1_{2j+1}(x)\ L_m\left(\overline{P}^1_{2k+1}\right)(x)\,dx, \tag{3.4}$$

where the differential operator L_m is defined in (2.8).

The following lemma states that the coefficients in the expansion of the function $\overline{P}^m_l$ in terms of either the functions $\overline{P}^1_j$ or the functions $\overline{P}^2_j$ (depending on whether m is even or odd) are the entries in an eigenvector of the matrix $G^{n,m}$.

LEMMA 3.1. *Suppose that m and n are integers such that $1 \le m \le n$.*

Then, when m and n are both even, $l(l+1)$ is an eigenvalue of the matrix $G^{n,m}$ defined in (3.1)*, for any even integer l with $m \le l \le n$, and the coordinates $\xi_0, \xi_1, \ldots, \xi_{n/2-2}$, $\xi_{n/2-1}$ of the corresponding eigenvector are the coefficients in the expansion*

$$\overline{P}^m_l(x) = \sum_{k=0}^{n/2-1} \xi_k\, \overline{P}^2_{2k+2}(x). \tag{3.5}$$

When m is even and n is odd, $l(l+1)$ is an eigenvalue of the matrix $G^{n,m}$ defined in (3.2)*, for any odd integer l with $m < l \le n$, and the coordinates ξ_0, $\xi_1, \ldots, \xi_{(n-1)/2-2}$, $\xi_{(n-1)/2-1}$ of the corresponding eigenvector are the coefficients in the expansion*

$$\overline{P}^m_l(x) = \sum_{k=0}^{(n-1)/2-1} \xi_k\, \overline{P}^2_{2k+3}(x). \tag{3.6}$$

When m is odd and n is even, $l(l+1)$ is an eigenvalue of the matrix $G^{n,m}$ defined in (3.3)*, for any even integer l with $m < l \le n$, and the coordinates ξ_0, $\xi_1, \ldots, \xi_{n/2-2}$, $\xi_{n/2-1}$ of the corresponding eigenvector are the coefficients in the expansion*

$$\overline{P}^m_l(x) = \sum_{k=0}^{n/2-1} \xi_k\, \overline{P}^1_{2k+2}(x). \tag{3.7}$$

When m and n are both odd, $l(l+1)$ is an eigenvalue of the matrix $G^{n,m}$ defined in (3.4)*, for any odd integer l with $m \le l \le n$, and the coordinates ξ_0, $\xi_1, \ldots, \xi_{(n+1)/2-2}$, $\xi_{(n+1)/2-1}$ of the corresponding eigenvector are the coefficients in the expansion*

$$\overline{P}^m_l(x) = \sum_{k=0}^{(n+1)/2-1} \xi_k\, \overline{P}^1_{2k+1}(x). \tag{3.8}$$

Proof. We outline the proof in the case that m and n are both even; the proofs in the other three cases are similar.

Substituting (2.12) into (2.9) and using (1.2) and (1.5), we obtain from (3.1) that the numbers $\xi_0, \xi_1, \ldots, \xi_{l/2-2}, \xi_{l/2-1}$ from (2.12), along with the numbers $\xi_{l/2} = 0$, $\xi_{l/2+1} = 0, \ldots, \xi_{n/2-2} = 0$, $\xi_{n/2-1} = 0$ when $l < n$, are the coordinates of the eigenvector of $G^{n,m}$ with corresponding eigenvalue $l(l+1)$, giving (3.5). ∎

The following lemma states that $G^{n,m}$ is the sum of a diagonal matrix and a semiseparable matrix and provides expressions for the entries of $G^{n,m}$.

LEMMA 3.2. *Suppose that m and n are integers such that $1 \leq m \leq n$.*

Then, the matrix $G^{n,m}$ defined in (3.1)–(3.4) *has the form*

$$G^{n,m} = D + S, \tag{3.9}$$

where D is a diagonal matrix and S is a semiseparable matrix.

When m and n are both even, D is the diagonal $n/2 \times n/2$ matrix with the diagonal entries $D_{0,0}$, $D_{1,1}, \ldots, D_{n/2-2,n/2-2}$, $D_{n/2-1,n/2-1}$ defined by the formula

$$D_{k,k} = (2k+2)(2k+3), \tag{3.10}$$

and S is the semiseparable $n/2 \times n/2$ matrix with the entry $S_{j,k}$ defined by the formulae

$$S_{j,k} = a_j \, b_k \tag{3.11}$$

when $j \leq k$, and

$$S_{j,k} = a_k \, b_j \tag{3.12}$$

otherwise (when $j > k$), where the numbers a_0, $a_1, \ldots, a_{n/2-2}$, $a_{n/2-1}$ and b_0, $b_1, \ldots, b_{n/2-2}, b_{n/2-1}$ are defined by the formulae

$$a_k = \sqrt{\frac{(2k+1)(2k+2)(2k+3)(2k+4)(4k+5)}{8 \cdot 15}}, \tag{3.13}$$

$$b_k = (m^2 - 4) \sqrt{\frac{15(4k+5)}{2(2k+1)(2k+2)(2k+3)(2k+4)}}. \tag{3.14}$$

When m is even and n is odd, D is the diagonal $(n-1)/2 \times (n-1)/2$ matrix with the diagonal entries $D_{0,0}, D_{1,1}, \ldots, D_{(n-1)/2-2,(n-1)/2-2}$, $D_{(n-1)/2-1,(n-1)/2-1}$ defined by the formula

$$D_{k,k} = (2k+3)(2k+4), \tag{3.15}$$

and S is the semiseparable $(n-1)/2 \times (n-1)/2$ matrix with the entry $S_{j,k}$ defined by the formulae

$$S_{j,k} = a_j \, b_k \tag{3.16}$$

when $j \leq k$, and

$$S_{j,k} = a_k \, b_j \tag{3.17}$$

otherwise (when $j > k$), where the numbers a_0, $a_1, \ldots, a_{(n-1)/2-2}$, $a_{(n-1)/2-1}$ and b_0, $b_1, \ldots, b_{(n-1)/2-2}, b_{(n-1)/2-1}$ are defined by the formulae

$$a_k = \sqrt{\frac{(2k+2)(2k+3)(2k+4)(2k+5)(4k+7)}{14 \cdot 15}}, \tag{3.18}$$

$$b_k = (m^2-4)\sqrt{\frac{7 \cdot 15(4k+7)}{8(2k+2)(2k+3)(2k+4)(2k+5)}}. \tag{3.19}$$

When m is odd and n is even, D is the diagonal $n/2 \times n/2$ matrix with the diagonal entries $D_{0,0}, D_{1,1}, \ldots, D_{n/2-2,n/2-2}, D_{n/2-1,n/2-1}$ defined by the formula

$$D_{k,k} = (2k+2)(2k+3), \tag{3.20}$$

and S is the semiseparable $n/2 \times n/2$ matrix with the entry $S_{j,k}$ defined by the formulae

$$S_{j,k} = a_j\, b_k \tag{3.21}$$

when $j \leq k$, and

$$S_{j,k} = a_k\, b_j \tag{3.22}$$

otherwise (when $j > k$), where the numbers a_0, $a_1, \ldots, a_{n/2-2}, a_{n/2-1}$ and b_0, $b_1, \ldots, b_{n/2-2}, b_{n/2-1}$ are defined by the formulae

$$a_k = \sqrt{\frac{(2k+2)(2k+3)(4k+5)}{30}}, \tag{3.23}$$

$$b_k = (m^2-1)\sqrt{\frac{15(4k+5)}{2(2k+2)(2k+3)}}. \tag{3.24}$$

When m and n are both odd, D is the diagonal $(n+1)/2 \times (n+1)/2$ matrix with the diagonal entries $D_{0,0}$, $D_{1,1}, \ldots, D_{(n+1)/2-2,(n+1)/2-2}$, $D_{(n+1)/2-1,(n+1)/2-1}$ defined by the formula

$$D_{k,k} = (2k+1)(2k+2), \tag{3.25}$$

and S is the semiseparable $(n+1)/2 \times (n+1)/2$ matrix with the entry $S_{j,k}$ defined by the formulae

$$S_{j,k} = a_j\, b_k \tag{3.26}$$

when $j \leq k$, and

$$S_{j,k} = a_k\, b_j \tag{3.27}$$

otherwise (when $j > k$), where the numbers a_0, $a_1, \ldots, a_{(n+1)/2-2}$, $a_{(n+1)/2-1}$ and b_0, $b_1, \ldots, b_{(n+1)/2-2}$, $b_{(n+1)/2-1}$ are defined by the formulae

$$a_k = \sqrt{\frac{(2k+1)(2k+2)(4k+3)}{6}}, \tag{3.28}$$

$$b_k = (m^2 - 1) \sqrt{\frac{3(4k+3)}{2(2k+1)(2k+2)}}. \tag{3.29}$$

Proof. We outline the proof in the case that m and n are both even; the proofs in the other three cases are similar.

We define the entry $D_{j,k}$ of the $n/2 \times n/2$ matrix D by the formula

$$D_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+2}(x) \;\; L_2 \left(\overline{P}^2_{2k+2} \right)(x) \, dx, \tag{3.30}$$

where the differential operator L_2 is defined in (2.8), and we define the entry $S_{j,k}$ of the $n/2 \times n/2$ matrix S by the formula

$$S_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+2}(x) \, \frac{m^2 - 4}{1 - x^2} \, \overline{P}^2_{2k+2}(x) \, dx. \tag{3.31}$$

We now show that $G^{n,m} = D + S$, D is diagonal, and S is semiseparable.

Combining (3.1), (3.30), (3.31), and (2.8), we obtain the decomposition (3.9).

Substituting (2.9) into (3.30), and using (1.2) and (1.5), we observe that the matrix D is diagonal with the diagonal entries given by (3.10).

In order to obtain the formulae (3.11)–(3.14), we define the entry $M_{j,k}$ of the infinite-dimensional matrix M, for $j, k = 0, 1, 2, \ldots$, by the formula

$$M_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+2}(x) \, \frac{m^2 - 4}{1 - x^2} \, \overline{P}^2_{2k+2}(x) \, dx \tag{3.32}$$

and observe that the entry $(M^{-1})_{j,k}$ of the inverse M^{-1} of the matrix M is given by the formula

$$(M^{-1})_{j,k} = \int_{-1}^{1} \overline{P}^2_{2j+2}(x) \, \frac{1 - x^2}{m^2 - 4} \, \overline{P}^2_{2k+2}(x) \, dx, \tag{3.33}$$

since M represents the operator acting on functions on $[-1, 1]$ by multiplication by the factor

$$\frac{m^2 - 4}{1 - x^2}, \tag{3.34}$$

whereas M^{-1} represents the operator acting on functions on $[-1, 1]$ by multiplication by the inverse factor

$$\frac{1 - x^2}{m^2 - 4}. \tag{3.35}$$

Using (1.2), (1.5), (2.10), and (3.33), we observe that M^{-1} is tridiagonal. So, M is the inverse of a tridiagonal matrix, and, as such, M is semiseparable (see, for example, [6]). But, for $j, k = 0, 1, \ldots, n/2 - 2, n/2 - 1$, (3.31) and (3.32) show that $S_{j,k} = M_{j,k}$, so that S is also semiseparable.

Integrating by parts a few times, while using (1.2), (1.5), and (2.7), together with the definition (3.31), yields explicit expressions for the uppermost entries $S_{0,0}, S_{0,1}, \ldots, S_{0,n/2-2}, S_{0,n/2-1}$ of S. Combining (1.2), (1.5), (2.11), and (3.31) yields explicit expressions for the diagonal entries $S_{0,0}, S_{1,1}, \ldots, S_{n/2-2,n/2-2}, S_{n/2-1,n/2-1}$ of S. Combining all of these explicit expressions with the fact that S is semiseparable, we obtain the formulae (3.11)–(3.14). ∎

3.2. Associated Legendre expansions of arbitrary orders and associated Legendre expansions of low orders. Lemmas 3.3, 3.4, 3.5, and 3.6 of this section follow immediately from Lemma 3.1.

Lemma 3.3 states that the matrix of eigenvectors of the matrix $G^{n,m}$ in (3.9) converts the coefficients in the expansion of a function f in terms of the functions $\overline{P}_m^m, \overline{P}_{m+2}^m, \ldots, \overline{P}_{n-2}^m, \overline{P}_n^m$ into the coefficients in the expansion of the function f in terms of the functions $\overline{P}_2^2, \overline{P}_4^2, \ldots, \overline{P}_{n-2}^2, \overline{P}_n^2$. Lemma 3.3 states, moreover, that the adjoint of the matrix of eigenvectors converts the latter coefficients into the former coefficients.

LEMMA 3.3. *Suppose that n and m are even integers such that $2 \le m \le n$, $p^m = (p_0^m, p_1^m, \ldots, p_{n/2-2}^m, p_{n/2-1}^m)^{\mathrm{T}}$ is a real column vector such that $p_{(n-m)/2+1}^m = 0, p_{(n-m)/2+2}^m = 0, \ldots, p_{n/2-2}^m = 0$, $p_{n/2-1}^m = 0$, and f is the function defined on $[-1, 1]$ by the formula*

$$f(x) = \sum_{j=0}^{(n-m)/2} p_j^m \, \overline{P}_{2j+m}^m(x). \tag{3.36}$$

Then,

$$f(x) = \sum_{l=0}^{n/2-1} p_l^2 \, \overline{P}_{2l+2}^2(x), \tag{3.37}$$

where $p^2 = (p_0^2, p_1^2, \ldots, p_{n/2-2}^2, p_{n/2-1}^2)^{\mathrm{T}}$ is the real vector defined by the formula

$$p^2 = U \, p^m, \tag{3.38}$$

and U is an $n/2 \times n/2$ matrix of eigenvectors of the symmetric matrix $G^{n,m}$ in (3.9) with

$$\overline{P}_{2j+m}^m(x) = \sum_{l=0}^{n/2-1} U_{l,j} \, \overline{P}_{2l+2}^2(x) \tag{3.39}$$

for $j = 0, 1, \ldots, (n-m)/2 - 1, (n-m)/2$. Moreover,

$$p^m = U^{\mathrm{T}} \, p^2. \tag{3.40}$$

Lemmas 3.4, 3.5, and 3.6 are analogues of Lemma 3.3 for different conditions on m and n.

LEMMA 3.4. *Suppose that n and m are integers such that m is even, n is odd, $2 \le m < n$, $p^m = (p_0^m, p_1^m, \ldots, p_{(n-5)/2}^m, p_{(n-3)/2}^m)^{\mathrm{T}}$ is a real column vector such that $p_{(n-m+1)/2}^m = 0$, $p_{(n-m+3)/2}^m = 0, \ldots, p_{(n-5)/2}^m = 0$, $p_{(n-3)/2}^m = 0$, and f is the function defined on $[-1, 1]$ by the formula*

$$f(x) = \sum_{j=0}^{(n-m-1)/2} p_j^m \, \overline{P}_{2j+m+1}^m(x). \tag{3.41}$$

Then,

$$f(x) = \sum_{l=0}^{(n-3)/2} p_l^2 \, \overline{P}_{2l+3}^2(x), \tag{3.42}$$

where $p^2 = (p^2_0, p^2_1, \ldots, p^2_{(n-5)/2}, p^2_{(n-3)/2})^{\mathrm{T}}$ *is the real vector defined by the formula*

$$p^2 = U\, p^m, \tag{3.43}$$

and U *is an* $(n-1)/2 \times (n-1)/2$ *matrix of eigenvectors of the symmetric matrix* $G^{n,m}$ *in* (3.9) *with*

$$\overline{P}^m_{2j+m+1}(x) = \sum_{l=0}^{(n-3)/2} U_{l,j}\, \overline{P}^2_{2l+3}(x) \tag{3.44}$$

for $j = 0, 1, \ldots, (n-m-3)/2, (n-m-1)/2$. *Moreover,*

$$p^m = U^{\mathrm{T}}\, p^2. \tag{3.45}$$

LEMMA 3.5. *Suppose that* n *and* m *are integers such that* m *is odd,* n *is even,* $1 \le m < n$, $p^m = (p^m_0, p^m_1, \ldots, p^m_{n/2-2}, p^m_{n/2-1})^{\mathrm{T}}$ *is a real column vector such that* $p^m_{(n-m+1)/2} = 0$, $p^m_{(n-m+3)/2} = 0, \ldots, p^m_{n/2-2} = 0$, $p^m_{n/2-1} = 0$, *and* f *is the function defined on* $[-1, 1]$ *by the formula*

$$f(x) = \sum_{j=0}^{(n-m-1)/2} p^m_j\, \overline{P}^m_{2j+m+1}(x). \tag{3.46}$$

Then,

$$f(x) = \sum_{l=0}^{n/2-1} p^1_l\, \overline{P}^1_{2l+2}(x), \tag{3.47}$$

where $p^1 = (p^1_0, p^1_1, \ldots, p^1_{n/2-2}, p^1_{n/2-1})^{\mathrm{T}}$ *is the real vector defined by the formula*

$$p^1 = U\, p^m, \tag{3.48}$$

and U *is an* $n/2 \times n/2$ *matrix of eigenvectors of the symmetric matrix* $G^{n,m}$ *in* (3.9) *with*

$$\overline{P}^m_{2j+m+1}(x) = \sum_{l=0}^{n/2-1} U_{l,j}\, \overline{P}^1_{2l+2}(x) \tag{3.49}$$

for $j = 0, 1, \ldots, (n-m-3)/2, (n-m-1)/2$. *Moreover,*

$$p^m = U^{\mathrm{T}}\, p^1. \tag{3.50}$$

LEMMA 3.6. *Suppose that* n *and* m *are odd integers such that* $1 \le m \le n$, $p^m = (p^m_0, p^m_1, \ldots, p^m_{(n-3)/2}, p^m_{(n-1)/2})^{\mathrm{T}}$ *is a real column vector such that* $p^m_{(n-m)/2+1} = 0$, $p^m_{(n-m)/2+2} = 0, \ldots, p^m_{(n-3)/2} = 0$, $p^m_{(n-1)/2} = 0$, *and* f *is the function defined on* $[-1, 1]$ *by the formula*

$$f(x) = \sum_{j=0}^{(n-m)/2} p^m_j\, \overline{P}^m_{2j+m}(x). \tag{3.51}$$

Then,

$$f(x) = \sum_{l=0}^{(n-1)/2} p_l^1 \, \overline{P}_{2l+1}^1(x), \tag{3.52}$$

where $p^1 = (p_0^1, p_1^1, \ldots, p_{(n-3)/2}^1, p_{(n-1)/2}^1)^{\mathrm{T}}$ *is the real vector defined by the formula*

$$p^1 = U \, p^m, \tag{3.53}$$

and U *is an* $(n+1)/2 \times (n+1)/2$ *matrix of eigenvectors of the symmetric matrix* $G^{n,m}$ *in* (3.9) *with*

$$\overline{P}_{2j+m}^m(x) = \sum_{l=0}^{(n-1)/2} U_{l,j} \, \overline{P}_{2l+1}^1(x) \tag{3.54}$$

for $j = 0, 1, \ldots, (n-m)/2 - 1, (n-m)/2$. *Moreover,*

$$p^m = U^{\mathrm{T}} \, p^1. \tag{3.55}$$

4. Informal description of the algorithm. In this section, we outline a "fast" algorithm for the conversion of the values $f_0, f_1, \ldots, f_{N-1}, f_N$ of the function f tabulated at the nodes $x_0, x_1, \ldots, x_{N-1}, x_N$, defined by the formula

$$x_k = \cos\left(\frac{\pi(k+1/2)}{N+1}\right), \tag{4.1}$$

into the coefficients $p_0, p_1, \ldots, p_{N-m-1}, p_{N-m}$ in the expansion

$$f(x) = \sum_{l=0}^{N-m} p_l \, \overline{P}_{l+m}^m(x). \tag{4.2}$$

(The inverse procedure of converting the coefficients $p_0, p_1, \ldots, p_{N-m-1}, p_{N-m}$ into the values $f_0, f_1, \ldots, f_{N-1}, f_N$ is quite similar, so we omit its description.)

The procedure consists of four steps, described briefly below. Since the procedure when m is odd is virtually the same as when m is even, we describe the procedure only for the case that m is even. For definitiveness, we assume also that N is even.

Step 1. We separate f into its even and odd parts f^+ and f^-, defined by the formulae

$$f^+(x) = \frac{f(x) + f(-x)}{\sqrt{2}}, \tag{4.3}$$

$$f^-(x) = \frac{f(x) - f(-x)}{\sqrt{2}}. \tag{4.4}$$

All subsequent processing is performed separately for f^+ and f^-. Since the procedures for f^+ and f^- are virtually identical, we describe only the procedure for the even part f^+.

Step 2. Using the fast Fourier transform as in Observation 2.5, we convert the values $f_0^+, f_1^+, \ldots, f_{N-1}^+, f_N^+$ of the function f^+, defined in (4.3), at the points $x_0, x_1,$

$\ldots$, x_{N-1}, x_N, defined in (4.1), into the coefficients c_0, $c_1, \ldots, c_{N/2-1}$, $c_{N/2}$ in the expansion

$$f^+(x) = \sum_{k=0}^{N/2} c_k \, T_{2k}(x). \tag{4.5}$$

Step 3. Using Lemma 2.8 and Observation 2.10, we convert the coefficients c_0, $c_1, \ldots, c_{N/2-1}$, $c_{N/2}$ in the expansion (4.5) into the coefficients p_0^2, $p_1^2, \ldots, p_{N/2-2}^2$, $p_{N/2-1}^2$ in the expansion

$$f^+(x) = \sum_{l=0}^{N/2-1} p_l^2 \, \overline{P}_{2l+2}^2(x). \tag{4.6}$$

Step 4. Using Lemmas 3.2 and 3.3 and Observation 2.11, we convert the coefficients p_0^2, p_1^2, $\ldots$, $p_{N/2-2}^2$, $p_{N/2-1}^2$ in the expansion (4.6) into the coefficients p_0^m, p_1^m, $\ldots$, $p_{(N-m)/2-1}^m$, $p_{(N-m)/2}^m$ in the expansion

$$f^+(x) = \sum_{l=0}^{(N-m)/2} p_l^m \, \overline{P}_{2l+m}^m(x). \tag{4.7}$$

Remark 4.1. Clearly, Step 1 takes $\mathcal{O}(N)$ operations. As per Observation 2.5, Step 2 takes $\mathcal{O}(N \log N)$ operations. As per Observation 2.10, Step 3 takes $\mathcal{O}(N \log(N/\varepsilon))$ operations, where ε is the precision of computations. As per Observation 2.11, Step 4 takes $\mathcal{O}(N(\log N)\log(1/\varepsilon))$ operations, where ε is the precision of computations used in this step. All together, Steps 1–4 take $\mathcal{O}(N(\log N)\log(1/\varepsilon))$ operations.

Remark 4.2. To handle f^- when m is even, we substitute Lemma 2.9 for Lemma 2.8 in Step 3 and Lemma 3.4 for Lemma 3.3 in Step 4. For f^- when m is even, we find that $p_{N/2-1}^2 = 0$ and $c_{N/2} = 0$, since, when m is even, f^- is a linear combination of only $N/2-1$ functions $\overline{P}_3^2$, $\overline{P}_5^2, \ldots, \overline{P}_{N-3}^2$, $\overline{P}_{N-1}^2$ (or, equivalently, of the $N/2$ Chebyshev polynomials of the first kind T_1, $T_3, \ldots, T_{N-3}$, T_{N-1}).

To handle f^+ when m is odd, we substitute Lemma 2.6 for Lemma 2.8 in Step 3 and Lemma 3.6 for Lemma 3.3 in Step 4.

To handle f^- when m is odd, we substitute Lemma 2.7 for Lemma 2.8 in Step 3 and Lemma 3.5 for Lemma 3.3 in Step 4.

When computing the coefficients (2.1) in the spherical harmonic expansion (1.6), we tabulate the function f on $[-1, 1]$ at the same $N+1$ sampling locations x_0, $x_1, \ldots,$ x_{N-1}, x_N when m is odd as when m is even, even though when m is odd, f is a linear combination of only N functions $\overline{P}_1^1$, $\overline{P}_2^1, \ldots,$ $\overline{P}_{N-1}^1$, $\overline{P}_N^1$ (or, equivalently, of the N Chebyshev polynomials of the second kind $\sqrt{1-x^2}\,U_0$, $\sqrt{1-x^2}\,U_1, \ldots,$ $\sqrt{1-x^2}\,U_{N-2}$, $\sqrt{1-x^2}\,U_{N-1}$, which are scaled by $\sqrt{1-x^2}$).

5. Detailed description of the algorithm. In this section, we describe in detail the algorithm described informally in section 4.

PRECOMPUTATIONS.

COMMENT [Compute all the data that Steps 3 and 4 require that do not depend on f.]

Compute all the data that Step 3 requires to apply fast the matrix $A^{N/2,2,+}$ from Lemma 2.8 (i.e., "compress" the matrix $A^{N/2,2,+}$).

Compute all the data that Step 4 requires to apply fast the adjoint of the matrix U from Lemmas 3.2 and 3.3 (i.e., "compress" the matrix U^{T}).

STEP 1.

COMMENT [Convert the values f_0, $f_1, \ldots, f_{N-1}$, f_N into the values f_0^+, $f_1^+, \ldots,$ f_{N-1}^+, f_N^+.]
do $n = 0, \ldots, N$
Set $f_n^+ = (f_n + f_{N-n})/\sqrt{2}$.
enddo

STEP 2.

COMMENT [Convert the values $f_0^+, f_1^+, \ldots, f_{N-1}^+, f_N^+$ into the coefficients $c_0, c_1, \ldots,$ $c_{N/2-1}, c_{N/2}$ of the Chebyshev polynomials $T_0, T_2, \ldots, T_{N-2}, T_N$.]
Use the discrete cosine transform (see, for example, [14]) to convert the values $f_0^+, f_1^+, \ldots, f_{N-1}^+, f_N^+$ into the coefficients $c_0, c_1, \ldots, c_{N/2-1}, c_{N/2}$.

STEP 3.

COMMENT [Convert the coefficients $c_0, c_1, \ldots, c_{N/2-1}, c_{N/2}$ of the Chebyshev polynomials $T_0, T_2, \ldots, T_{N-2}, T_N$ into the coefficients $p_0^2, p_1^2, \ldots, p_{N/2-2}^2, p_{N/2-1}^2$ of the associated Legendre functions $\overline{P}_2^2, \overline{P}_4^2, \ldots, \overline{P}_{N-2}^2, \overline{P}_N^2$.]
Use Observation 2.10 to apply the matrix $A^{N/2,2,+}$ from Lemma 2.8 to the vector $c = (c_0, c_1, \ldots, c_{N/2-1}, c_{N/2})^{\mathrm{T}}$, in order to obtain the vector $p^2 = (p_0^2, p_1^2, \ldots, p_{N/2-2}^2, p_{N/2-1}^2)^{\mathrm{T}}$.

STEP 4.

COMMENT [Convert the coefficients $p_0^2, p_1^2, \ldots, p_{N/2-2}^2, p_{N/2-1}^2$ of the associated Legendre functions $\overline{P}_2^2, \overline{P}_4^2, \ldots, \overline{P}_{N-2}^2, \overline{P}_N^2$ of order 2 into the coefficients $p_0^m, p_1^m, \ldots, p_{(N-m)/2-1}^m, p_{(N-m)/2}^m$ of the associated Legendre functions $\overline{P}_m^m, \overline{P}_{m+2}^m, \ldots, \overline{P}_{N-2}^m, \overline{P}_N^m$.]
Use Observation 2.11 to apply the adjoint of the matrix U from Lemmas 3.2 and 3.3 to the vector $p^2 = (p_0^2, p_1^2, \ldots, p_{N/2-2}^2, p_{N/2-1}^2)^{\mathrm{T}}$, in order to obtain the vector $p^m = (p_0^m, p_1^m, \ldots, p_{N/2-2}^m, p_{N/2-1}^m)^{\mathrm{T}}$.

COMMENT [Only the first $(N-m)/2+1$ entries of p^m interest us; the other entries vanish: $p_{(N-m)/2+1}^m = 0$, $p_{(N-m)/2+2}^m = 0, \ldots, p_{N/2-2}^m = 0$, $p_{N/2-1}^m = 0$.]

6. Numerical results. The algorithms described in this paper have been implemented in Fortran. Tables 1–8 report the results of applying the algorithm to functions f defined on $[-1, 1]$ by the formulae

$$f(x) = \sum_{l=0}^{(N-m)/2} \varepsilon_l \, \overline{P}_{2l+m}^m(x) \tag{6.1}$$

when m is even and degrees are even,

$$f(x) = \sum_{l=0}^{(N-m-2)/2} \varepsilon_l \, \overline{P}_{2l+m+1}^m(x) \tag{6.2}$$

when m is even and degrees are odd,

$$f(x) = \sum_{l=0}^{(N-m-1)/2} \varepsilon_l \, \overline{P}_{2l+m+1}^m(x) \tag{6.3}$$

TABLE 1
Times in seconds and errors for $m = \frac{N+6}{2}$, even degrees, reconstruction.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	515	.20E−02	.91E−03	.20E+01	.64E−07
2048	1027	.52E−02	.36E−02	.80E+01	.58E−07
4096	2051	.23E−01	.15E−01	.33E+02	.17E−06
8192	4099	.48E−01	.58E−01	.16E+03	.39E−06
16384	8195	.11E+00	.23E+00	.69E+03	.72E−06
32768	16387	.24E+00	.96E+00	.36E+04	.14E−05
65536	32771	.51E+00	(.37E+01)	.14E+05	.52E−06
131072	65539	.11E+01	(.15E+02)	.63E+05	.70E−05

TABLE 2
Times in seconds and errors for $m = \frac{N+6}{2}$, even degrees, decomposition.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	515	.19E−02	.91E−03	.30E+01	.73E−08
2048	1027	.49E−02	.36E−02	.80E+01	.70E−08
4096	2051	.22E−01	.15E−01	.32E+02	.76E−08
8192	4099	.47E−01	.58E−01	.16E+03	.11E−07
16384	8195	.11E+00	.23E+00	.68E+03	.13E−07
32768	16387	.23E+00	.96E+00	.35E+04	.15E−07
65536	32771	.50E+00	(.37E+01)	.14E+05	.20E−07
131072	65539	.12E+01	(.15E+02)	.62E+05	.35E−07

TABLE 3
Times in seconds and errors for $m = \frac{N-4}{2}$, even degrees, reconstruction.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	510	.20E−02	.91E−03	.20E+01	.77E−07
2048	1022	.53E−02	.36E−02	.90E+01	.72E−07
4096	2046	.23E−01	.15E−01	.38E+02	.65E−07
8192	4094	.49E−01	.58E−01	.17E+03	.15E−06
16384	8190	.11E+00	.23E+00	.78E+03	.41E−06
32768	16382	.24E+00	.96E+00	.34E+04	.54E−06
65536	32766	.52E+00	(.37E+01)	.15E+05	.31E−06
131072	65534	.11E+01	(.15E+02)	.64E+05	.41E−05

TABLE 4
Times in seconds and errors for $m = \frac{N-4}{2}$, even degrees, decomposition.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	510	.19E−02	.91E−03	.20E+01	.37E−08
2048	1022	.51E−02	.36E−02	.90E+01	.79E−08
4096	2046	.22E−01	.15E−01	.38E+02	.95E−08
8192	4094	.48E−01	.58E−01	.16E+03	.11E−07
16384	8190	.11E+00	.23E+00	.77E+03	.16E−07
32768	16382	.24E+00	.96E+00	.33E+04	.15E−07
65536	32766	.51E+00	(.37E+01)	.16E+05	.17E−07
131072	65534	.11E+01	(.15E+02)	.62E+05	.21E−07

TABLE 5
Times in seconds and errors for $m = \frac{N-32}{32}$, odd degrees, reconstruction.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	31	.16E−02	.91E−03	.20E+01	.86E−08
2048	63	.43E−02	.36E−02	.70E+01	.20E−07
4096	127	.20E−01	.15E−01	.28E+02	.16E−06
8192	255	.43E−01	.58E−01	.11E+03	.41E−06
16384	511	.94E−01	.23E+00	.56E+03	.60E−06
32768	1023	.21E+00	.96E+00	.26E+04	.10E−05
65536	2047	.46E+00	(.37E+01)	.14E+05	.14E−05
131072	4095	.10E+01	(.15E+02)	.47E+05	.14E−05

TABLE 6
Times in seconds and errors for $m = \frac{N-32}{32}$, odd degrees, decomposition.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	31	.15E−02	.91E−03	.20E+01	.71E−08
2048	63	.41E−02	.36E−02	.70E+01	.59E−08
4096	127	.20E−01	.15E−01	.27E+02	.82E−08
8192	255	.42E−01	.58E−01	.12E+03	.76E−08
16384	511	.92E−01	.23E+00	.55E+03	.12E−07
32768	1023	.21E+00	.96E+00	.25E+04	.13E−07
65536	2047	.45E+00	(.37E+01)	.11E+05	.11E−07
131072	4095	.98E+00	(.15E+02)	.46E+05	.20E−07

TABLE 7
Times in seconds and errors for $m = \frac{15N}{16}$, odd degrees, reconstruction.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	960	.21E−02	.91E−03	.30E+01	.12E−07
2048	1920	.55E−02	.36E−02	.11E+02	.20E−07
4096	3840	.23E−01	.15E−01	.39E+02	.48E−07
8192	7680	.49E−01	.58E−01	.16E+03	.14E−06
16384	15360	.11E+00	.23E+00	.83E+03	.15E−06
32768	30720	.24E+00	.96E+00	.36E+04	.23E−06
65536	61440	.53E+00	(.37E+01)	.21E+05	.12E−05
131072	122880	.11E+01	(.15E+02)	.62E+05	.15E−05

TABLE 8
Times in seconds and errors for $m = \frac{15N}{16}$, odd degrees, decomposition.

N	m	"Fast" transformation	Third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix	Precomputation	Relative r.m.s. error
1024	960	.20E−02	.91E−03	.30E+01	.48E−08
2048	1920	.52E−02	.36E−02	.10E+02	.74E−08
4096	3840	.23E−01	.15E−01	.38E+02	.10E−07
8192	7680	.49E−01	.58E−01	.16E+03	.11E−07
16384	15360	.11E+00	.23E+00	.81E+03	.13E−07
32768	30720	.24E+00	.96E+00	.35E+04	.18E−07
65536	61440	.51E+00	(.37E+01)	.16E+05	.16E−07
131072	122880	.11E+01	(.15E+02)	.60E+05	.27E−07

when m is odd and degrees are even, and

$$f(x) = \sum_{l=0}^{(N-m-1)/2} \varepsilon_l \, \overline{P}_{2l+m}^{m}(x) \tag{6.4}$$

when m is odd and degrees are odd; ε_0, ε_1, ε_2, ... are randomly generated numbers in the interval $[-1, 1]$. The decomposition algorithm computes from the sample values of f the coefficients in the representation of f as a linear combination of the functions $\overline{P}_m^m$, $\overline{P}_{m+1}^m$, ..., $\overline{P}_{N-1}^m$, $\overline{P}_N^m$; the reconstruction algorithm computes the sample values of f from the coefficients in the representation of f as a linear combination of the functions $\overline{P}_m^m$, $\overline{P}_{m+1}^m$, ..., $\overline{P}_{N-1}^m$, $\overline{P}_N^m$. The CPU times are in seconds. The errors are relative root-mean-square errors.

The columns labeled "'fast' transformation" list the times taken by the algorithm to evaluate one sum of the form (6.1), (6.2), (6.3), or (6.4).

The columns labeled "third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix" list the times, divided by 3, taken to apply an $\frac{N}{2} \times \frac{N}{2}$ matrix once to a vector of length $\frac{N}{2}$. These times scale quadratically with N; the figures in parentheses are estimates used when the memory required by direct calculations would be excessive.

Remark 6.1. The columns labeled "third of applying an $\frac{N}{2} \times \frac{N}{2}$ matrix" give some indication of how much time current implementations of the decompositions and reconstructions take to run; these times are believed to be reasonable estimates of how a standard package like Spherepack (see [2], [18]) would perform, using the "semi-naive" algorithm described in [10], for example. Clearly, for sufficiently small N, the "slow" implementation would actually run faster than the "fast" implementation.

The code was compiled with the Lahey–Fujitsu compiler, with optimization flag `--o2`, and run on a 2.8 GHz Intel Pentium Xeon microprocessor with 512 KB of L2 cache.

No effort was made to optimize the precomputations. To simplify the implementation, precomputations that take $\mathcal{O}(N^2)$ operations were used, even though the techniques described in [4] lead naturally to precomputations that would take only $\mathcal{O}(N \log N)$ operations. The precomputations were run to yield approximately 6 digits of accuracy, running all computations (including the precomputations) in double precision arithmetic.

Observation 6.2. Asymptotically, the algorithm should require a number of operations proportional to $N(\log N) \log(1/\varepsilon)$ to evaluate one sum of the form (6.1), (6.2), (6.3), or (6.4), where $\varepsilon \approx 10^{-6}$. The times in the columns labeled "'fast' transformation" appear to be consistent with this estimate. The algorithm appears to break even with standard matrix multiplication methods for computing spherical harmonic expansions between $N = 4096$ and $N = 8192$; however, no serious effort has been made to optimize our implementation.

7. Generalizations. The algorithm of this paper admits a number of generalizations and extensions. The following list is not intended to be exhaustive; subjects in it are under investigation, and these investigations will be reported at a later date.

1. *Different discretizations of* S^2. Throughout this paper, we have assumed that the meridians on S^2 are discretized in an equispaced manner, i.e., that the points θ_0, $\theta_1, \ldots, \theta_{N-1}$, θ_N subdivide the interval $[0, \pi]$ into equal subintervals. This limitation is easily removed via techniques described in [11], [19], and [5]; however, to maintain numerical stability, the nodes have to satisfy certain quite restrictive criteria. One important collection of nodes that does in fact lead to stable algorithms is given

by the formula

$$\theta_k = \cos^{-1}(x_k), \tag{7.1}$$

where x_0, $x_1, \ldots, x_{N-1}$, x_N are Gaussian nodes on the interval $[-1, 1]$ (see, for example, [2]).

Furthermore, the nodes φ_0, $\varphi_1, \ldots, \varphi_{2N-1}$, φ_{2N} in the discretizations of the parallels on S^2 do not have to be equispaced, provided some form of nonequispaced fast Fourier transform is used; again, numerical stability requires that φ_0, $\varphi_1, \ldots, \varphi_{2N-1}, \varphi_{2N}$ be fairly close to being equispaced.

2. *Associated Laguerre functions.* Associated Laguerre functions are defined on the entire half-line $[0, \infty)$, but are very small outside of a finite interval. The Sturm–Liouville problem that generates the associated Laguerre functions becomes an eigenvector problem for the sum of a diagonal matrix and a semiseparable matrix when discretized using associated Laguerre functions of low orders (very much like the associated Legendre functions).

3. *Prolate spheroidal wave functions.* The Sturm–Liouville problem that generates the prolate spheroidal wave functions becomes an eigenvector problem for a tridiagonal matrix when discretized using Legendre polynomials.

4. *Associated prolate spheroidal wave functions.* The Sturm–Liouville problem that generates the associated prolate spheroidal wave functions becomes an eigenvector problem for the sum of a tridiagonal matrix and a semiseparable matrix when discretized using associated Legendre functions of low orders.

Acknowledgments. The authors would like to thank R. R. Coifman for many useful and inspiring discussions, and P. N. Swarztrauber, M. J. Mohlenkamp, and the referees for several suggestions that have improved the exposition.

REFERENCES

[1] M. Abramowitz and I. Stegun, eds., *Handbook of Mathematical Functions,* Dover, New York, 1972.

[2] J. C. Adams and P. N. Swarztrauber, *SPHEREPACK* 3.0*: A model development facility,* Mon. Wea. Rev., 127 (1999), pp. 1872–1878.

[3] B. Alpert and V. Rokhlin, *A fast algorithm for the evaluation of Legendre expansions,* SIAM J. Sci. Statist. Comput., 12 (1991), pp. 158–179.

[4] S. Chandrasekaran and M. Gu, *A divide-and-conquer algorithm for the eigendecomposition of symmetric block-diagonal plus semiseparable matrices,* Numer. Math., 96 (2004), pp. 723–731.

[5] A. Dutt, M. Gu, and V. Rokhlin, *Fast algorithms for polynomial interpolation, integration, and differentiation,* J. Numer. Anal., 33 (1996), pp. 1689–1711.

[6] F. R. Gantmacher and M. G. Krein, *Oscillation Matrices and Small Vibrations of Mechanical Systems,* revised English ed., AMS Chelsea Publishing, Providence, RI, 2002.

[7] I. S. Gradshteyn and I. M. Ryzhik, *Tables of Integrals, Series, and Products,* 6th ed., Academic Press, San Diego, CA, 2000.

[8] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem,* SIAM J. Matrix Anal. Appl., 16 (1995), pp. 172–191.

[9] D. M. Healy, P. J. Kostelec, and D. Rockmore, *Towards safe and effective high-order Legendre transforms with applications to FFTs for the* 2*-sphere,* Adv. Comput. Math., 21 (2004), pp. 59–105.

[10] D. M. Healy, P. J. Kostelec, D. Rockmore, and S. S. B. More, *FFTs for the* 2*-sphere—Improvements and variations,* J. Fourier Anal. Appl., 9 (2003), pp. 341–385.

[11] R. Jacob-Chien and B. Alpert, *A fast spherical filter with uniform resolution,* J. Comput. Phys., 136 (1997), pp. 580–584.

[12] S. Kunis and D. Potts, *Fast spherical Fourier algorithms,* J. Comput. Appl. Math., 161 (2003), pp. 75–98.

[13] M. J. Mohlenkamp, *A fast transform for spherical harmonics,* J. Fourier Anal. Appl., 5 (1999), pp. 159–184.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran* 77*,* 2nd ed., Cambridge University Press, Cambridge, UK, 1992.

[15] V. Rokhlin and M. Tygert, *Fast Algorithms for Spherical Harmonic Expansions,* Technical report 1309, Department of Computer Science, Yale University, New Haven, CT, 2004.

[16] R. Suda, *Stability analysis of the fast Legendre transform algorithm based on the fast multipole method,* Proc. Estonian Acad. Sci. Phys. Math., 53 (2004), pp. 107–115.

[17] R. Suda and M. Takami, *A fast spherical harmonics transform algorithm,* Math. Comp., 71 (2002), pp. 703–715.

[18] P. N. Swarztrauber and W. F. Spotz, *Generalized discrete spherical harmonic transforms,* J. Comput. Phys., 159 (2000), pp. 213–230.

[19] N. Yarvin and V. Rokhlin, *A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics,* J. Comput. Phys., 147 (1998), pp. 594–609.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1929–1947
© 2006 Society for Industrial and Applied Mathematics

SENSITIVITY ANALYSIS OF ODEs/DAEs USING THE TAYLOR SERIES METHOD*

ROBERTO BARRIO†

Abstract. This paper studies the applicability of the Taylor method for the sensibility analysis of ODEs and DAEs. Extended automatic differentiation rules are introduced for the calculus of partial derivatives of Taylor series. The numerical method is implemented using an efficient variable-step variable-order scheme. Finally, some numerical tests are presented showing the benefits of the formulation.

Key words. Taylor method, sensitivity analysis, automatic differentiation, partial derivatives

AMS subject classifications. 65L05, 65L80, 41A58

DOI. 10.1137/030601892

1. Introduction.

In recent years the sensitivity analysis of ordinary differential equations (ODEs) [16, 27, 42] and differential-algebraic equations (DAEs) [8, 9, 11, 17, 31] has attracted much research. The principal motivation comes from mathematical models used to investigate physical phenomena where some parameters may not be accurately known, and, therefore, sensitivity analysis becomes a useful tool. One important need for sensitivities is in parameter identification where we would like to use a Newton-type method to adjust parameters so that a function fits some data set.

Another important field where sensitivity analysis plays a very important role is in dynamical systems [12, 21, 29]. For example, periodic orbits, the Lyapunov exponents, chaos indicators, and bifurcation analysis are fundamental objects for the complete study of a dynamical system, and they require computation of the sensitivities with respect to the initial conditions of the problem. In [12, 21, 29] the numerical integration of the dynamical systems is done by using the Taylor method, but the sensitivities are obtained by means of the numerical integration of the variational equations or by a link with an external software for automatic differentiation (AD). In this paper we extend the Taylor method in such a way that the explicit construction of the variational equations is not needed. On the other hand, Taylor methods are also used in the validated solutions of ODEs/DAEs by means of interval analysis [26, 35], and a validated solution of first-order sensitivity equations (variational equations) has been recently studied in [43]. The approach presented in this paper, which directly uses Taylor methods, may simplify the validated solution of the sensitivity equations of any order.

The plan of the paper is as follows. Section 2 reviews the Taylor method for ODEs and DAEs. Section 3 introduces the sensitivity analysis problem. Section 4 describes the generation of the Taylor coefficients of the partial derivatives of a function. Finally, section 5 presents several numerical tests and an application to a chaos indicator that confirm the promise of the present approach.

*Received by the editors November 19, 2003; accepted for publication (in revised form) April 20, 2005; published electronically February 3, 2006. This work was supported by the Spanish Research grant DGYCT BFM2003-02137.

http://www.siam.org/journals/sisc/27-6/60189.html

†GME, Depto. Matemática Aplicada, Edificio de Matemáticas, University of Zaragoza, E-50009 Zaragoza, Spain (rbarrio@unizar.es).

2. Taylor methods for ODEs/DAEs. The Taylor method is one of the oldest numerical methods for solving ODEs. Let us consider the initial value problem

$$\frac{d\boldsymbol{y}(t)}{dt} = \boldsymbol{f}(t, \boldsymbol{y}(t)), \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{y} \in \mathbb{R}^s,\ t \in \mathbb{R}.$$

The value of the solution at t_i (that is, $\boldsymbol{y}(t_i)$) is approximated by $\boldsymbol{y}_i$ from the nth degree Taylor polynomial of $\boldsymbol{y}(t)$ at $t = t_i$. In this paper we consider that $\boldsymbol{f}$ is analytic. Denoting $h_i = t_i - t_{i-1}$,

$$\begin{aligned}
&\boldsymbol{y}(t_0) =: \boldsymbol{y}_0,\\
&\boldsymbol{y}(t_i) \simeq \boldsymbol{y}(t_{i-1}) + \frac{d\boldsymbol{y}(t_{i-1})}{dt}\, h_i + \frac{1}{2!}\, \frac{d^2\boldsymbol{y}(t_{i-1})}{dt^2}\, h_i^2 + \cdots + \frac{1}{n!}\, \frac{d^n\boldsymbol{y}(t_{i-1})}{dt^n}\, h_i^n\\
&\qquad \simeq \boldsymbol{y}_{i-1} + \boldsymbol{f}(t_{i-1}, \boldsymbol{y}_{i-1})\, h_i + \frac{1}{2!}\, \frac{d\boldsymbol{f}(t_{i-1}, \boldsymbol{y}_{i-1})}{dt}\, h_i^2 + \cdots\\
&\qquad\quad + \frac{1}{n!}\, \frac{d^{n-1}\boldsymbol{f}(t_{i-1}, \boldsymbol{y}_{i-1})}{dt^{n-1}}\, h_i^n =: \boldsymbol{y}_i.
\end{aligned} \tag{1}$$

We must determine the Taylor coefficients $\{1/j!\, d^j\boldsymbol{y}(t_{i-1})/dt^j\}$, which may be obtained directly by differentiating the right-hand side of the differential equation $\boldsymbol{f}(t, \boldsymbol{y}(t))$. This approach is useful only in theoretical studies because when n increases, the complexity of the explicit computation of the derivatives becomes too cumbersome. Another approach is the use of AD techniques [19, 41] as in [11, 13, 28, 30]. In the literature different preprocessing codes have been designed to get the recurrence formulae of the Taylor coefficients via AD [11, 13, 28]. In this paper we have used a preprocessor called MARSTIN [3, 30], a *Mathematica* package, to obtain the Taylor series solution of an ODE problem. The output is the `FORTRAN77` program that computes the recurrence formulae to integrate the differential system by recurrent power series, that is, the coefficients of the Taylor method.

The Taylor method presents several advantages. It gives directly a dense output in the form of a power series, and it can be formulated as an interval method [34, 35] giving guaranteed integration methods. In addition, Taylor methods may directly manage high-order differential equations taking into account only the fact that the Taylor coefficients for the solution and its derivatives $\boldsymbol{y}, \boldsymbol{y}', \ldots$ are evidently related ($d^n\boldsymbol{y}^{(k)}/dt^n = (n+1)_k\, d^{n+k}\boldsymbol{y}/dt^{n+k}$, $(a)_k = a\,(a+1)\cdots(a+k-1)$ being the Pochhammer symbol). Also, as Taylor methods of degree n are also of order n, the use of Taylor methods of a high degree gives us numerical methods of a high order. Therefore, they are very useful for a high-precision solution of ODEs [3].

It is well known that Taylor methods are not A-stable (they are only in the infinite case, taking the infinite series development). In fact, the stability function for the Taylor method of degree n is $R_n(z) = \sum_{i=0}^{n} z^i/i!$, the same as for explicit Runge–Kutta (RK) methods with the same order as the number of internal stages for order lower than 5 (no explicit RK method exists of order n with n stages for $n \geq 5$ [22, 23]). When the order increases, the stability domain approaches a semicircle in the negative complex plane with its center at the origin. In [3] the radius of the semicircle is approximated (using numerical data up to degree 60) by a linear L_2 fit, obtaining $r(n) \simeq 1.3614 + 0.3725\, n$. Hence, high-order Taylor methods can be used with moderately stiff equations because for large n the stability domain is large enough. Obviously, Taylor methods cannot be used for highly stiff systems, where it is necessary to use implicit A-stable methods. Implicit versions of Taylor methods that are A-stable have been developed in [6, 15].

In the practical implementation of a numerical method for the solution of ODEs, the use of variable stepsizes is a crucial point because it permits an automatic control of the error. Several formulations of variable-stepsize Taylor methods can be found in [13, 28, 30]. Here we use the approach given in [1, 3, 30]. In Taylor methods there is no rejected step as occurs in any variable-stepsize formulation for RK or multistep methods because we choose the stepsize once the series are generated to obtain a required precision level. Furthermore, we use a variable-order formulation of the Taylor method developed in [3] to obtain a more adaptive method. To reach this goal it is necessary to know a priori an estimation of both the computational time (we use the fact that the computational complexity of each step of the Taylor method is $\mathcal{O}(n^2)$ [19, 28], although a better estimation may be obtained by recording series generation time vs. order to give a second-order curve fit $a_0 + a_1\, n + a_2\, n^2$ of the computational time) and the stepsize for a fixed error tolerance for different orders (for more details see [3]).

In the case of smooth DAEs the Taylor method was first applied by Chang and Corliss [11, 13] and a systematic study was done by Pryce in [36, 39, 40]. Consider a DAE system

$$\boldsymbol{F}\boldsymbol{y} := \boldsymbol{f}(t,\, \boldsymbol{y},\, \boldsymbol{y}', \dots) = 0, \tag{2}$$

$$\boldsymbol{\Phi}\, \boldsymbol{y}_0 := \boldsymbol{\phi}(t_0,\, \boldsymbol{y}_0,\, \boldsymbol{y}_0', \dots) = 0 \tag{3}$$

with $\boldsymbol{y}, \boldsymbol{y}', \dots \in \mathbb{R}^s$. Consistent initial conditions are given by (3). The Taylor series method for DAEs involves a preanalysis stage where one solves an assignment problem (the Σ-*method* [40]). The solvability of a DAE system by means of the Taylor series method is given by the following theorem.

THEOREM 1 (Pryce [39]). *Let the DAE* $\boldsymbol{F}\boldsymbol{y} = 0$ *be analytic in a neighborhood of a consistent point. Then the Taylor series method applied at this point succeeds iff the* $s \times s$ *matrix* J *is nonsingular at this point, where*

$$J_{ij} = \begin{cases} \dfrac{\partial f_i}{\partial y_j^{(d_j - c_i)}} & \text{if this derivative is present,} \\ 0 & \text{otherwise.} \end{cases}$$

The indices d_j and c_i give, respectively, the appropriate derivatives of the system variables y_j and the number of differentiations of the ith equation needed to convert to an explicit ODE system (see [39, 40]).

For a practical implementation of Taylor methods for DAEs first we enter into the preanalysis stage [36, 39, 40], which gives us the new equations used in the Taylor scheme $\widetilde{\boldsymbol{F}}\boldsymbol{y} := \widetilde{\boldsymbol{f}}(t,\, \boldsymbol{y}, \boldsymbol{y}', \dots, \boldsymbol{y}^{(\mathbf{d})}) = 0$, where we denote $\boldsymbol{y}^{(\mathbf{d})} = (y_1^{(d_1)}, \dots, y_s^{(d_s)})^{\mathrm{T}}$. Then, we use AD in the generation of the Taylor coefficients of $\widetilde{\boldsymbol{f}}$ in a way similar to that in the case of ODEs, and afterwards we solve for each Taylor coefficient a linear system with the same coefficient matrix J (defined in Theorem 1),

$$-J \cdot (\boldsymbol{y}^{(\mathbf{d})})_i = (\widetilde{\boldsymbol{F}}\boldsymbol{y})_{i0} := (\widetilde{\boldsymbol{f}}(t,\, \boldsymbol{y}, \boldsymbol{y}', \dots, \boldsymbol{y}^{(\mathbf{d})}))_{i0},$$

with $(\boldsymbol{y}^{(\mathbf{d})})_i = ((y_1^{(d_1)})_i, \dots, (y_s^{(d_s)})_i)^{\mathrm{T}}$, where $(g)_i$ stands for the ith Taylor coefficient of g and $(\widetilde{\boldsymbol{f}}(t,\, \boldsymbol{y}, \boldsymbol{y}', \dots, \boldsymbol{y}^{(\mathbf{d})}))_{i0}$ is the ith Taylor coefficient of $\widetilde{\boldsymbol{f}}$ considering $(\boldsymbol{y}^{(\mathbf{d})})_i = 0$. We may use the relation among the Taylor coefficients of a function and its derivatives to obtain the series of the solution $\boldsymbol{y}$. On each step we must also obtain the first Taylor coefficients in such a way that the solution is consistent with

the DAE at the point of the development of the Taylor polynomial. This involves solving nonlinear equations (3) with the constraints of the DAEs. Apart from the initial conditions, the convergence of, for example, the Newton methods on this problem with the numerical values of the Taylor polynomial as an initial guess is quite fast (in general, one iteration is enough) because these values are very good approximations of a solution. This step is essentially a projection onto all the constraint manifolds (the given constraints (3) and the hidden constraints obtained after the preanalysis stage [36, 39, 40]), but this is done at the beginning of the step (to obtain the first Taylor coefficients) and not at the end, as a standard projection method does (so we may call it a "preprojection" [1]). If we do not project onto all the constraints and we use only (3), we retain consistency with the DAEs but we solve the DAEs less well because we are forgetting the hidden constraints.

For more details about the use of the Taylor method in the numerical solution of DAEs, see [1, 36, 39, 40].

3. Sensitivity analysis. Consider the ODE system

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}; \boldsymbol{p}), \quad \boldsymbol{y}(0) = \boldsymbol{y}_0$$

with $\boldsymbol{y} \in \mathbb{R}^s$ and a set of parameters $\boldsymbol{p} \in \mathbb{R}^p$. Sensitivity analysis entails finding the derivative of the above system with respect to each parameter. This produces an additional $p \cdot s$ sensitivity equations

$$\boldsymbol{s}_k' = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\, \boldsymbol{s}_k + \frac{\partial \boldsymbol{f}}{\partial p_k}, \quad k = 1, \ldots, p,$$

where $\boldsymbol{s}_k = \partial \boldsymbol{y}/\partial p_k$. A common situation is $\boldsymbol{p} = \boldsymbol{y}_0$.

In some applications we are also interested in higher-order partial derivatives. For example, they are necessary for nonlinear equations and when we need parameter–parameter interactions. Therefore, we also need the second- (or higher-) order sensitivity equations

$$(\boldsymbol{s}_{kj})_m' = \frac{\partial f_m}{\partial \boldsymbol{y}}\, \boldsymbol{s}_{kj} + (\boldsymbol{s}_j)^{\mathrm{T}}\, \frac{\partial^2 f_m}{\partial \boldsymbol{y}^2}\, \boldsymbol{s}_k + \frac{\partial^2 f_m}{\partial p_j\, \partial \boldsymbol{y}}\, \boldsymbol{s}_k + \frac{\partial^2 f_m}{\partial p_k\, \partial \boldsymbol{y}}\, \boldsymbol{s}_j + \frac{\partial^2 f_m}{\partial p_k\, \partial p_j}$$

for $m = 1, \ldots, s$ and where $(\boldsymbol{s}_{kj})_m = \partial^2 y_m / \partial p_k\, \partial p_j$.

In the sensitivity analysis several methods have been used in the literature: "brute force," in which the differential equation is solved, and the sensitivity values are computed by means of finite differences [42]; numerical integration of the differential and the variational equations [16, 42]; the use of AD techniques on the numerical integration of the differential equations [16, 27], that is, to use an AD library (e.g., ADIFOR [7]) to create a code that generates the sensitivity values given the code for generating the solution of the ODE. Each of these methods usually computes only first-order sensitivities.

Instead, we propose to use extended AD rules to obtain both the solution of the ODE and the sensitivity values: "the Taylor series method." This method permits computing any order of sensitivities. Also, once the differential system is introduced we do not need to perform any additional precompilation process to generate any sensitivity. Taylor polynomials with partial derivatives have already been used by Berz [4, 5, 33], computing the partial derivatives using rules for the composition of polynomials considering the whole Taylor polynomials (in fact the Taylor polynomial

and the remainder in [33], the "Taylor model") at the same time or computing multivariate Taylor polynomials. On the contrary, in this paper, we give rules for the direct computation of Taylor coefficients of the partial derivatives that gives the univariate Taylor polynomials of the sensitivities. Our approach gives easier methods to program because we only have to do the precompilation of the original system and the sensitivities are obtained just by the extended differentiation rules.

The Taylor series method computes the Taylor polynomials of the solution of the differential equation (1) and the Taylor polynomials of the partial derivatives of the solution

$$\boldsymbol{s}_k(t_i) \simeq \boldsymbol{s}_k(t_{i-1}) + \frac{d\boldsymbol{s}_k(t_{i-1})}{dt}\, h_i + \frac{1}{2!}\, \frac{d^2\boldsymbol{s}_k(t_{i-1})}{dt^2}\, h_i^2 + \cdots + \frac{1}{n!}\, \frac{d^n\boldsymbol{s}_k(t_{i-1})}{dt^n}\, h_i^n,$$

$$\boldsymbol{s}_{kj}(t_i) \simeq \boldsymbol{s}_{kj}(t_{i-1}) + \frac{d\boldsymbol{s}_{kj}(t_{i-1})}{dt}\, h_i + \frac{1}{2!}\, \frac{d^2\boldsymbol{s}_{kj}(t_{i-1})}{dt^2}\, h_i^2 + \cdots + \frac{1}{n!}\, \frac{d^n\boldsymbol{s}_{kj}(t_{i-1})}{dt^n}\, h_i^n.$$

We compute the coefficients $d^l\boldsymbol{s}_k(t_{i-1})/dt^l$, $d^l\boldsymbol{s}_{kj}(t_{i-1})/dt^l$ by partial derivation of the coefficients of the Taylor polynomials

$$\frac{d^l\boldsymbol{s}_k(t_{i-1})}{dt^l} = \frac{\partial}{\partial p_k}\, \frac{d^l\boldsymbol{y}(t_{i-1})}{dt^l}, \quad \frac{d^l\boldsymbol{s}_{kj}(t_{i-1})}{dt^l} = \frac{\partial^2}{\partial p_k\, \partial p_j}\, \frac{d^l\boldsymbol{y}(t_{i-1})}{dt^l}.$$

Hence, we need extended AD rules for the computation of the coefficients of the Taylor expansions of partial derivatives of any order of any elementary function.

In the case of DAE systems, the first-order sensitivity analysis produces additional sensitivity equations (to simplify the notation, we have considered the case $\boldsymbol{F}\boldsymbol{y} := \boldsymbol{f}(t, \boldsymbol{y}, \boldsymbol{y}'; \boldsymbol{p}) = 0$)

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}\, \boldsymbol{s}_k + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}'}\, \boldsymbol{s}'_k + \frac{\partial \boldsymbol{f}}{\partial p_k} = 0, \tag{4}$$

$$\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}}\, \boldsymbol{s}_k(t_0) + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{y}'}\, \boldsymbol{s}'_k(t_0) + \frac{\partial \boldsymbol{\phi}}{\partial p_k} = 0, \tag{5}$$

where $\boldsymbol{s}_k = \partial\boldsymbol{y}/\partial p_k$. Consistent initial conditions are given by (5). For a second-order sensitivity analysis, the additional equations are ($m = 1, \ldots, s$)

$$\begin{aligned}
&\frac{\partial f_m}{\partial \boldsymbol{y}}\, \boldsymbol{s}_{kj} + \frac{\partial f_m}{\partial \boldsymbol{y}'}\, (\boldsymbol{s}_{kj})' + (\boldsymbol{s}_j)^{\mathrm{T}}\, \frac{\partial^2 f_m}{\partial \boldsymbol{y}^2}\, \boldsymbol{s}_k + (\boldsymbol{s}'_k)^{\mathrm{T}}\, \frac{\partial^2 f_m}{\partial \boldsymbol{y}'^2}\, \boldsymbol{s}'_j \\
&\quad + (\boldsymbol{s}'_k)^{\mathrm{T}}\, \frac{\partial^2 f_m}{\partial \boldsymbol{y}'\, \partial \boldsymbol{y}}\, \boldsymbol{s}_j + (\boldsymbol{s}_k)^{\mathrm{T}}\, \frac{\partial^2 f_m}{\partial \boldsymbol{y}\, \partial \boldsymbol{y}'}\, \boldsymbol{s}'_j + \frac{\partial^2 f_m}{\partial p_j\, \partial \boldsymbol{y}}\, \boldsymbol{s}_k \\
&\quad + \frac{\partial^2 f_m}{\partial p_k\, \partial \boldsymbol{y}}\, \boldsymbol{s}_j + \frac{\partial^2 f_m}{\partial p_j\, \partial \boldsymbol{y}'}\, \boldsymbol{s}'_k + \frac{\partial^2 f_m}{\partial p_k\, \partial \boldsymbol{y}'}\, \boldsymbol{s}'_j + \frac{\partial^2 f_m}{\partial p_k\, \partial p_j} = 0.
\end{aligned}$$

This problem (in the case of first-order sensitivities) has been extensively treated by Petzold et al. [8, 9, 17, 31] using the simultaneous corrector method (to solve simultaneously the DAE system and the sensitivity equations), the staggered direct method, and the adjoint sensitivity analysis. Again, we propose using the Taylor series method. For each Taylor coefficient and partial derivative, we solve a linear system with the same coefficient matrix J. For instance, up to second-order sensitivity analysis we have

$$-J \cdot (\boldsymbol{y}')_i = (\boldsymbol{F}\boldsymbol{y})_{i0}, \quad -J \cdot (\boldsymbol{s}'_k)_i = \left(\frac{\partial}{\partial p_k}\boldsymbol{F}\boldsymbol{y}\right)_{i0}, \quad -J \cdot (\boldsymbol{s}'_{kj})_i = \left(\frac{\partial^2}{\partial p_k\, \partial p_j}\boldsymbol{F}\boldsymbol{y}\right)_{i0},$$

where the second members of the linear system are obtained by using the extended AD rules of the next section and where in $(\partial \boldsymbol{F}\boldsymbol{y}/\partial p_k)_{i0}$ and $(\partial^2 \boldsymbol{F}\boldsymbol{y}/\partial p_k \partial p_j)_{i0}$ we consider $(\boldsymbol{s}'_k)_i = 0$ and $(\boldsymbol{s}''_{kj})_i = 0$, respectively.

The first Taylor coefficients of the sensitivities have to satisfy the constraint equations (5) for the sensitivities, but now the constraint equations are linear with respect to the sensitivities. Therefore, for the sensitivities we do not have to solve any nonlinear equation (in the case of obtaining the solution of the DAEs system we have to solve the nonlinear constraint equations for the first coefficients).

Another important question for the development of a good scientific software package is the storage requirement, since the efficiency of the Taylor method depends deeply on the precompilation process where we minimize the polynomial operations and additional polynomial storage. For the DAE problem a very detailed analysis of the Taylor method was recently done in [36].

4. Automatic differentiation: Partial derivatives. In the practical determination of the Taylor coefficients of a function we use the classical rules for AD of the elementary operations and standard functions ($\pm$, $\times$, /, ln, sin, ...) developed by, among others, Moore [34, 41]. AD gives a recursive procedure to obtain the numerical value of the reiterated derivatives of the elementary functions at a given point [19, 41]. For example, for the real power function (for a complete list, see [19, 34, 41]), if $f : t \in \mathbb{R} \mapsto \mathbb{R}$ is a function of class $\mathcal{C}^n$ and denoting

$$f^{[j]}(t) = \frac{1}{j!} f^{(j)}(t),$$

we have, if $h(t) = f(t)^\alpha$ ($\alpha \in \mathbb{R}$),

$$h^{[0]}(t) = (f^{[0]}(t))^\alpha, \quad h^{[n]}(t) = \frac{1}{n\, f^{[0]}(t)} \sum_{i=0}^{n-1} (n\,\alpha - i(\alpha+1))\; f^{[n-i]}(t)\, h^{[i]}(t). \tag{6}$$

In this paper, as we focus our attention on partial derivatives, we have extended the classical formulae of the Taylor coefficients of a function to a set of formulae that gives us the Taylor coefficients of a function and/or its partial derivatives of any order. In the following we denote by $\mathbb{N}_0$ the set of the natural numbers with zero ($\mathbb{N}_0 := \mathbb{N} \cup \{0\}$).

PROPOSITION 2. *If $f(t, \boldsymbol{y}(t)), g(t, \boldsymbol{y}(t)) : (t, \boldsymbol{y}) \in \mathbb{R}^{s+1} \mapsto \mathbb{R}$ are functions of class $\mathcal{C}^n$, $\mathbf{i} = (i_1, i_2, \ldots, i_s) \in \mathbb{N}_0^s$ a vector with the partial derivative orders, $\mathbf{i}^* = \mathbf{i} - (0, \ldots, 0, 1, 0, \ldots, 0) = (i_1, i_2, \ldots, i_k - 1, 0, \ldots, 0)$ (that is, we subtract 1 from the last nonzero element i_k of $\mathbf{i}$ if $\mathbf{i} \neq \mathbf{0}$), and $\|\mathbf{i}\| = \sum_{j=1}^s i_j$ the total order of derivation, we denote*

$$f^{[j,\,\mathbf{i}]} := \frac{1}{j!} \frac{\partial^{\|\mathbf{i}\|} f^{(j)}(t)}{\partial y_1^{i_1}\, \partial y_2^{i_2} \cdots \partial y_s^{i_s}}, \quad f^{[j,\,\mathbf{0}]} := f^{[j]} = \frac{1}{j!} \frac{d^j f(t)}{dt^j},$$

that is, the jth Taylor coefficient of the partial derivative of $f(t, \boldsymbol{y}(t))$ with respect to $\mathbf{i}$ and

$$\widetilde{h}^{[j,\,\mathbf{v}]}_{n,\,\mathbf{i}} = h^{[j,\,\mathbf{v}]} \quad (j \neq n \text{ or } \mathbf{v} \neq \mathbf{i}), \quad \widetilde{h}^{[n,\,\mathbf{i}]}_{n,\,\mathbf{i}} = 0.$$

In addition, given $\mathbf{v} = (v_1, \ldots, v_s) \in \mathbb{N}_0^s$ we define the multicombinatorial number $\binom{\mathbf{i}}{\mathbf{v}} = \binom{i_1}{v_1} \cdot \binom{i_2}{v_2} \cdots \binom{i_s}{v_s}$, and we consider the classical partial order in $\mathbb{N}_0^s$. Then we have the following:

(i) *If* $h(t) = f(t) \pm g(t)$, *then* $h^{[n,\,\mathbf{i}]} = f^{[n,\,\mathbf{i}]} \pm g^{[n,\,\mathbf{i}]}$.
(ii) *If* $h(t) = \alpha\, f(t)$ *with* $\alpha \in \mathbb{R}$, *then* $h^{[n,\,\mathbf{i}]} = \alpha\, f^{[n,\,\mathbf{i}]}$.
(iii) *If* $h(t) = f(t) \cdot g(t)$, *then*

$$h^{[n,\,\mathbf{i}]} = \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} \left\{ \sum_{j=0}^{n} f^{[n-j,\,\mathbf{v}]} \cdot g^{[j,\,\mathbf{i}-\mathbf{v}]} \right\}.$$

(iv) *If* $h(t) = f(t)/g(t)$, *then*

$$h^{[n,\,\mathbf{i}]} = \frac{1}{g^{[0]}} \left(f^{[n,\,\mathbf{i}]} - \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} \left\{ \sum_{j=0}^{n} \widetilde{h}_{n,\,\mathbf{i}}^{[n-j,\,\mathbf{v}]} \cdot g^{[j,\,\mathbf{i}-\mathbf{v}]} \right\} \right).$$

(v) *If* $h(t) = f(t)^{\alpha}$ *with* $\alpha \in \mathbb{R}$, *then*

$$h^{[0,\,\mathbf{0}]} = (f^{[0]}(t))^{\alpha},$$

$$h^{[0,\,\mathbf{i}]} = \frac{1}{f^{[0]}} \sum_{\mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} \left\{ \alpha\, h^{[0,\,\mathbf{v}]} \cdot f^{[0,\,\mathbf{i}-\mathbf{v}]} - \widetilde{h}_{0,\,\mathbf{i}}^{[0,\,\mathbf{i}-\mathbf{v}]} \cdot f^{[0,\,\mathbf{v}]} \right\}, \qquad \mathbf{i} > \mathbf{0},$$

$$h^{[n,\,\mathbf{i}]} = \frac{1}{n\, f^{[0]}} \sum_{j=0}^{n} \big(n\,\alpha - j(\alpha+1) \big) \left\{ \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} \widetilde{h}_{n,\,\mathbf{i}}^{[j,\,\mathbf{v}]} \cdot f^{[n-j,\,\mathbf{i}-\mathbf{v}]} \right\}, \quad n > 0,\ \mathbf{i} > \mathbf{0}.$$

(vi) *If* $h(t) = \exp(f(t))$, *then*

$$h^{[0,\,\mathbf{0}]} = \exp(f^{[0]}(t)),$$

$$h^{[0,\,\mathbf{i}]} = \sum_{\mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} h^{[0,\,\mathbf{v}]} \cdot f^{[0,\,\mathbf{i}-\mathbf{v}]}, \qquad \mathbf{i} > \mathbf{0},$$

$$h^{[n,\,\mathbf{i}]} = \frac{1}{n} \sum_{j=0}^{n-1} (n-j) \left\{ \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} h^{[j,\,\mathbf{v}]} \cdot f^{[n-j,\,\mathbf{i}-\mathbf{v}]} \right\}, \qquad n > 0,\ \mathbf{i} > \mathbf{0}.$$

(vii) *If* $h(t) = \ln(f(t))$, *then*

$$h^{[0,\,\mathbf{0}]} = \ln(f^{[0]}(t)),$$

$$h^{[0,\,\mathbf{i}]} = \frac{1}{f^{[0]}} \left(f^{[0,\,\mathbf{i}]} - \sum_{\mathbf{0} < \mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} h^{[0,\,\mathbf{i}-\mathbf{v}]} \cdot f^{[0,\,\mathbf{v}]} \right), \qquad \mathbf{i} > \mathbf{0},$$

$$h^{[n,\,\mathbf{i}]} = \frac{1}{f^{[0]}} \left(f^{[n,\,\mathbf{i}]} - \frac{1}{n} \sum_{j=0}^{n-1} (n-j) \left\{ \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} \widetilde{h}_{n,\,\mathbf{i}}^{[n-j,\,\mathbf{i}-\mathbf{v}]} \cdot f^{[j,\,\mathbf{v}]} \right\} \right), \quad n > 0,\ \mathbf{i} > \mathbf{0}.$$

(viii) *If* $g(t) = \cos(f(t))$ *and* $h(t) = \sin(f(t))$, *then*

$$\begin{cases} g^{[0,\,\mathbf{0}]} = \cos(f^{[0]}(t)), \\ g^{[0,\,\mathbf{i}]} = -\displaystyle\sum_{\mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} h^{[0,\,\mathbf{v}]} \cdot f^{[0,\,\mathbf{i}-\mathbf{v}]}, \qquad \mathbf{i} > \mathbf{0}, \\ g^{[n,\,\mathbf{i}]} = -\dfrac{1}{n} \displaystyle\sum_{j=1}^{n} j \left\{ \sum_{\mathbf{v} \le \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} h^{[n-j,\,\mathbf{v}]} \cdot f^{[j,\,\mathbf{i}-\mathbf{v}]} \right\}, \qquad n > 0,\ \mathbf{i} > \mathbf{0}, \end{cases}$$

$$\begin{cases} h^{[0,\,\mathbf{0}]} = \sin(f^{[0]}(t)), \\ \\ h^{[0,\,\mathbf{i}]} = \displaystyle\sum_{\mathbf{v}\leq \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}}\, g^{[0,\,\mathbf{v}]}\cdot f^{[0,\,\mathbf{i}-\mathbf{v}]}, \qquad \mathbf{i}>\mathbf{0}, \\ \\ h^{[n,\,\mathbf{i}]} = \dfrac{1}{n}\displaystyle\sum_{j=1}^{n} j\left\{\sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, g^{[n-j,\,\mathbf{v}]}\cdot f^{[j,\,\mathbf{i}-\mathbf{v}]}\right\}, \qquad n>0,\ \mathbf{i}>\mathbf{0}. \end{cases}$$

(ix) *If* $h(t)=\arctan(f(t))$, *then*

$$f_2^{[n,\,\mathbf{i}]} = \sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}} \left\{\sum_{j=0}^{n} f^{[n-j,\,\mathbf{v}]}\cdot f^{[j,\,\mathbf{i}-\mathbf{v}]}\right\}, \qquad n\geq 0,\ \mathbf{i}\geq\mathbf{0},$$

$$h^{[0,\,\mathbf{0}]} = \arctan(f^{[0]}(t)),$$

$$h^{[0,\,\mathbf{i}]} = \frac{1}{1+(f^{[0]})^2}\left\{f^{[0,\,\mathbf{i}]} - \sum_{\mathbf{0}<\mathbf{v}\leq \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}}\, h^{[0,\,\mathbf{i}-\mathbf{v}]}\cdot f_2^{[0,\,\mathbf{v}]}\right\}, \qquad \mathbf{i}>\mathbf{0},$$

$$h^{[n,\,\mathbf{i}]} = \frac{1}{1+(f^{[0]})^2}\left(f^{[n,\,\mathbf{i}]} - \frac{1}{n}\sum_{j=0}^{n-1}(n-j)\left\{\sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, \widetilde{h}_{n,\,\mathbf{i}}^{[n-j,\,\mathbf{i}-\mathbf{v}]}\cdot f_2^{[j,\,\mathbf{v}]}\right\}\right), \qquad n>0,\ \mathbf{i}>\mathbf{0}.$$

Proof. As the proof of all the statements is technical, we present only the proof of two cases.

If $h(t)=f(t)^\alpha$ (case (v)), then formula (6) gives

$$h^{[n]}\, f^{[0]} = \frac{1}{n}\sum_{j=0}^{n-1}\left(n\,\alpha - j(\alpha+1)\right) h^{[j]}\, f^{[n-j]}.$$

By applying Leibniz's formula for partial derivatives, we obtain

$$\begin{aligned} \frac{\partial^{\|\mathbf{i}\|}\, h^{[n]}\, f^{[0]}}{\partial y_1^{i_1}\,\partial y_2^{i_2}\cdots\partial y_s^{i_s}} &= \sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, h^{[n,\,\mathbf{v}]}\cdot f^{[0,\,\mathbf{i}-\mathbf{v}]} \\ &= \frac{\partial^{\|\mathbf{i}\|}}{\partial y_1^{i_1}\,\partial y_2^{i_2}\cdots\partial y_s^{i_s}}\left\{\frac{1}{n}\sum_{j=0}^{n-1}\left(n\,\alpha - j(\alpha+1)\right) h^{[j]}\cdot f^{[n-j]}\right\} \\ &= \frac{1}{n}\sum_{j=0}^{n-1}\left(n\,\alpha - j(\alpha+1)\right)\left\{\sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, h^{[j,\,\mathbf{v}]}\cdot f^{[n-j,\,\mathbf{i}-\mathbf{v}]}\right\}. \end{aligned}$$

Therefore,

$$\begin{aligned} h^{[n,\,\mathbf{v}]} = \frac{1}{f^{[0]}}\Bigg\{&\frac{1}{n}\sum_{j=0}^{n-1}\left(n\,\alpha - j(\alpha+1)\right)\sum_{\mathbf{v}\leq \mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, h^{[j,\,\mathbf{v}]}\cdot f^{[n-j,\,\mathbf{i}-\mathbf{v}]} \\ &- \sum_{\mathbf{v}<\mathbf{i}} \binom{\mathbf{i}}{\mathbf{v}}\, h^{[n,\,\mathbf{v}]}\cdot f^{[0,\,\mathbf{i}-\mathbf{v}]}\Bigg\}. \end{aligned}$$

That expression, taking into account the definition of $\widetilde{h}_{n,\,\mathbf{i}}^{[j,\,\mathbf{v}]}$, gives us the formula for $n>0$ in (v). The case $h^{[0,\,\mathbf{i}]}$ is obtained by taking first a partial derivative of the

equation $h^{[0,\,\mathbf{0}]} := h^{[0]}(t) = (f^{[0]}(t))^{\alpha}$ with respect to $\mathbf{e}_k = (0, \ldots, 0, 1, 0, \ldots, 0)$, giving

$$h^{[0,\,\mathbf{e}_k]} = \alpha (f^{[0]})^{\alpha-1}\, f^{[0,\,\mathbf{e}_k]} = \alpha\, h^{[0]}\, f^{[0,\,\mathbf{e}_k]} / f^{[0]}.$$

Therefore, $h^{[0,\,\mathbf{e}_k]}\, f^{[0]} = \alpha\, h^{[0]}\, f^{[0,\,\mathbf{e}_k]}$, and from partial derivation with respect to $\mathbf{i}^* = \mathbf{i} - \mathbf{e}_k$, we obtain by applying Leibniz's rule

$$\sum_{\mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} h^{[0,\,\mathbf{i}-\mathbf{v}]} \cdot f^{[0,\,\mathbf{v}]} = \alpha \sum_{\mathbf{v} \le \mathbf{i}^*} \binom{\mathbf{i}^*}{\mathbf{v}} h^{[0,\,\mathbf{v}]} \cdot f^{[0,\,\mathbf{i}-\mathbf{v}]}.$$

And, after manipulation, we obtain the result.

If $h(t) = \arctan(f(t))$ (case (ix)), then, taking derivatives, we obtain $h'(t)\,(1 + f^2(t)) = f'(t)$. Therefore, the coefficients of the partial derivatives are obtained as in the previous case by partial derivation of the Taylor coefficients of the arctan function [19, 34] and of the function $(1 + f^2(t))$. □

In the above formulae one can easily add more elementary functions such as tan, arccos, arcsin, cosh, sinh, etc., just by using similar arguments.

Proposition 2 is also valid for the computation of the partial derivatives of a function (in fact $f^{[0,\,\mathbf{i}]}(t) := \partial^{\|\mathbf{i}\|} f(t) / \partial y_1^{i_1}\, \partial y_2^{i_2} \cdots \partial y_s^{i_s}$) and for just the determination of the Taylor coefficients of an elementary function. In this case the vector of partial derivatives is the null vector $\mathbf{i} = \mathbf{0}$, and we denote $f^{[0,\,\mathbf{0}]} := f^{[0]}(t)$. Therefore, in the generation of the Taylor polynomials of a function and its partial derivatives we need only the formulae of the above proposition. This fact is useful in practical implementation of the algorithm because we need only one set of subroutines.

The problem of evaluating partial derivatives was recently studied in [20, 37] but not in the context of obtaining the Taylor polynomials that give us the sensitivity values of an ODE or DAE differential system.

Once we have the formulae for the Taylor coefficients, we are interested in the computational cost of such formulae. In the case of the Taylor coefficients the complexity is well known [28, 34].

PROPOSITION 3. *If the evaluation of $f(t, \boldsymbol{y}(t))$ involves k elementary functions $(\times, /, \ln, \exp, \sin, \cos, \ldots)$, then the computational complexity of the evaluation of $f^{[0]}, f^{[1]}, \ldots, f^{[n-1]}$ is $\mathcal{O}(k\, n^2)$.*

We notice that the above result is based on counting floating point operations (FLOPS), and currently, on modern architectures, this measure is not related linearly to the CPU time (the real measure of interest). Therefore, such a complexity result is just a theoretical analysis.

In the case of obtaining the coefficients of the partial derivatives we only have to determine, given a vector with the indices of the partial derivatives $\mathbf{i} = (i_1, i_2, \ldots, i_s)$, the number of vectors $\mathbf{v} \in \mathbb{N}_0^s$ such that $\mathbf{v} \le \mathbf{i}$ because in all the elementary functions $(\times, /, \ln, \exp, \sin, \cos, \ldots)$ there appears a sum of the type $\sum_{\mathbf{v} \le \mathbf{i}}$. This number is $\prod_{j=1}^{s} (i_j + 1)$, a result that proves the following proposition.

PROPOSITION 4. *If the evaluation of $f(t, \boldsymbol{y}(t))$ involves k elementary functions $(\times, /, \ln, \exp, \sin, \cos, \ldots)$, and given $\mathbf{i} = (i_1, i_2, \ldots, i_s) \in \mathbb{N}_0^s$, then the computational complexity of the evaluation of $f^{[0,\,\mathbf{i}]}, f^{[1,\,\mathbf{i}]}, \ldots, f^{[n-1,\,\mathbf{i}]}$, supposing all the derivatives of index $\mathbf{v} < \mathbf{i}$ are already known, is*

$$\mathcal{O}\left(\prod_{j=1}^{s} (i_j + 1) \cdot k\, n^2 \right).$$

As before, the above result is a theoretical result. On modern computers the CPU time may evolve in a slightly different way.

If the function $f(t,\, \boldsymbol{y}(t))$ is a linear function (involves only additions/subtractions and multiplication by scalars), then the computational complexity is evidently $\mathcal{O}(n)$.

COROLLARY 5. *The computational complexity of evaluating the Taylor coefficients of a partial derivative of f is twice the complexity of evaluating the Taylor coefficients of f, and the computational complexity of evaluating the Taylor coefficients of a second-order partial derivative of f is, once the coefficients of the first-order partial derivatives are known, four times the complexity of evaluating the Taylor coefficients of f in the case of $\partial^2 f/\partial y_i\, \partial y_j\ (i \neq j)$ and three times in the case $\partial^2 f/\partial y_i^2$.*

In the case of sensitivities with respect to a parameter different from the initial conditions we use the above formulae, but we also have to consider the partial derivatives with respect to the parameters. This can be done in an automatic way by using another set of extended AD rules, in which we use an extended vector of variables with the variables and the parameters, but where the length of the Taylor polynomials of the parameters depends on the function with the parameter (in some cases just one coefficient). Therefore, we will have to consider the Taylor coefficients of elementary functions with polynomials of different length (in some cases with one constant argument). We do not detail the formulae because the essence is Proposition 2. On the other hand, in most of the applications related with dynamical systems, as periodic orbits, Lyapunov exponents, chaos indicators, bifurcation analysis, and so on, the partial derivatives are calculated with respect to the variables (we are interested in the Jacobian) or with respect to the initial conditions (we are interested in the solution of the variational equations), and we use Proposition 2 with the initial conditions of the identity matrix on each integration step or just at the first step, respectively.

TABLE 4.1

Left column: Pseudocode of the program to compute Taylor coefficients of the solution of the Kepler problem. Right column: Pseudocode of the program to compute Taylor coefficients of the solution and the sensitivity values of the Kepler problem.

Kepler problem	Kepler problem & sensitivity values
`for` $m = 0$ `to` $n - 2$ `do`	`for` $m = 0$ `to` $n - 2$ `do`
$c = (1+m)(2+m)$	$c = (1+m)(2+m)$
	`for` $\mathbf{v} = \mathbf{0}$ `to` $\mathbf{i}$ `do`
$s_1^{[m]} = \boxed{x \times x}^{[m]} + \boxed{y \times y}^{[m]}$	$s_1^{[m,\mathbf{v}]} = \boxed{x \times x}^{[m,\mathbf{v}]} + \boxed{y \times y}^{[m,\mathbf{v}]}$
$s_2^{[m]} = \boxed{(s_1)^{-3/2}}^{[m]}$	$s_2^{[m,\mathbf{v}]} = \boxed{(s_1)^{-3/2}}^{[m,\mathbf{v}]}$
$x^{[m+2]} = -\boxed{x \times s_2}^{[m]}/c$	$x^{[m+2,\mathbf{v}]} = -\boxed{x \times s_2}^{[m,\mathbf{v}]}/c$
$y^{[m+2]} = -\boxed{y \times s_2}^{[m]}/c$	$y^{[m+2,\mathbf{v}]} = -\boxed{y \times s_2}^{[m,\mathbf{v}]}/c$
	`end`
`end`	`end`

At this point it is interesting to note that the programs to integrate an ODE/DAE or to integrate an ODE/DAE and any order sensitivity value are the "same." That is, once we have obtained the scheme for the series development we do not have to modify this scheme to introduce the calculus of different sensitivity orders. On the contrary, other methods to obtain sensitivity values need to obtain the additional sensitivity equations, and this process may be highly cumbersome. In Table 4.1 we present the kernel of the series development program for the classical Kepler problem (see the next section for a description) in the case of obtaining just the solution and

in the case of obtaining both the solution and sensitivity values up to any order with respect to the initial conditions. Any operation inside a box stands for an operation among Taylor polynomials. In the example we need only the formulae for the Taylor coefficients of the partial derivatives of the product and the real power function of Taylor polynomials (formulae (iii) and (v) of Proposition 2).

5. Numerical tests. We have performed several numerical tests on a Windows PC Pentium III-866 MHz using `g77` (GNU `FORTRAN77`). We perform comparisons with the well-established software `dop853` developed by Hairer, Nørsett, and Wanner [22] (based on an explicit RK of order $8(5,3)$ given by Prince and Dormand [38] with stepsize control and dense output) for nonstiff problems, and `radau5` and `radau` developed by Hairer and Wanner [23, 24] (based on implicit RK schemes, Radau IIA, of order 5 or variable order, 5, 9, or 13, respectively, and equipped with stepsize control and dense output) for the stiff case. In these comparisons we use for the determination of the sensitivity values the simultaneous corrector method [17] in the `dop853`, `radau5`, and `radau` codes for the DAE problems. The reference values have been obtained using quadruple precision and the Taylor series method with an error tolerance of 10^{-30}.

5.1. ODE problems. We have taken three classical ODE problems as test problems.

• A *Galactic dynamics model* [22]. This problem is a Hamiltonian ODE problem with coordinates q_1, q_2, q_3 and moments p_1, p_2, p_3. The Hamiltonian function and parameter values (the initial conditions have been fixed to obtain $\mathcal{H} = 2$) are

$$\mathcal{H} = \frac{1}{2}\,(p_1^2 + p_2^2 + p_3^2) + \Omega\,(p_1\,q_2 - p_2\,q_1) + \ln\left(1 + \frac{q_1^2}{(1.25)^2} + q_2^2 + \frac{q_3^2}{(0.75)^2}\right),$$

$$q_1(0) = 2.5, \quad q_2(0) = q_3(0) = p_1(0) = 0, \quad p_2(0) = \frac{25 + \sqrt{6961 - 3200\,\ln 5}}{40}, \quad p_3(0) = 0.2.$$

In the tests we have considered $\Omega = 0.25$ and the integration interval is $[0,\, 1000]$.

• The *Lorenz model* [22] with Saltzman's values ($b = 8/3$, $\sigma = 10$, $r = 28$). This problem is a classic example of chaotic behavior:

$$x' = -\sigma\,(x - y), \quad y' = -x\,z + r\,x - y, \quad z' = x\,y - b\,z, \quad (x_0,\, y_0,\, z_0) = (-8,\, 8,\, r - 1).$$

The integration interval is $[0,\, 16]$.

• The *Kepler problem* describes planar two-body motion:

$$x'' = -x/(x^2 + y^2)^{3/2}, \quad y'' = -y/(x^2 + y^2)^{3/2},$$

$$x_0 = 1 - e, \quad x_0' = 0, \quad y_0 = 0, \quad y_0' = \sqrt{(1 + e)/(1 - e)}.$$

The integration interval is 20 orbital periods, that is, $[0,\, 40\pi]$.

In Figure 5.1 we show precision-computer time (in seconds) diagrams in logarithmic scale for the `dop853` code and the Taylor method for the computation of first- and second-order sensitivities for all the above three problems. These graphs show the relative error and computer time, and each point corresponds to a different requested accuracy. Because the algorithms are adaptive, neither relative error nor computer time depends continuously on the requested accuracy, although we present the graphs connected with continuous lines to help the visualization. In particular, we show first-order sensitivities with respect to the parameter Ω (graphs (g1) and (g2)) and

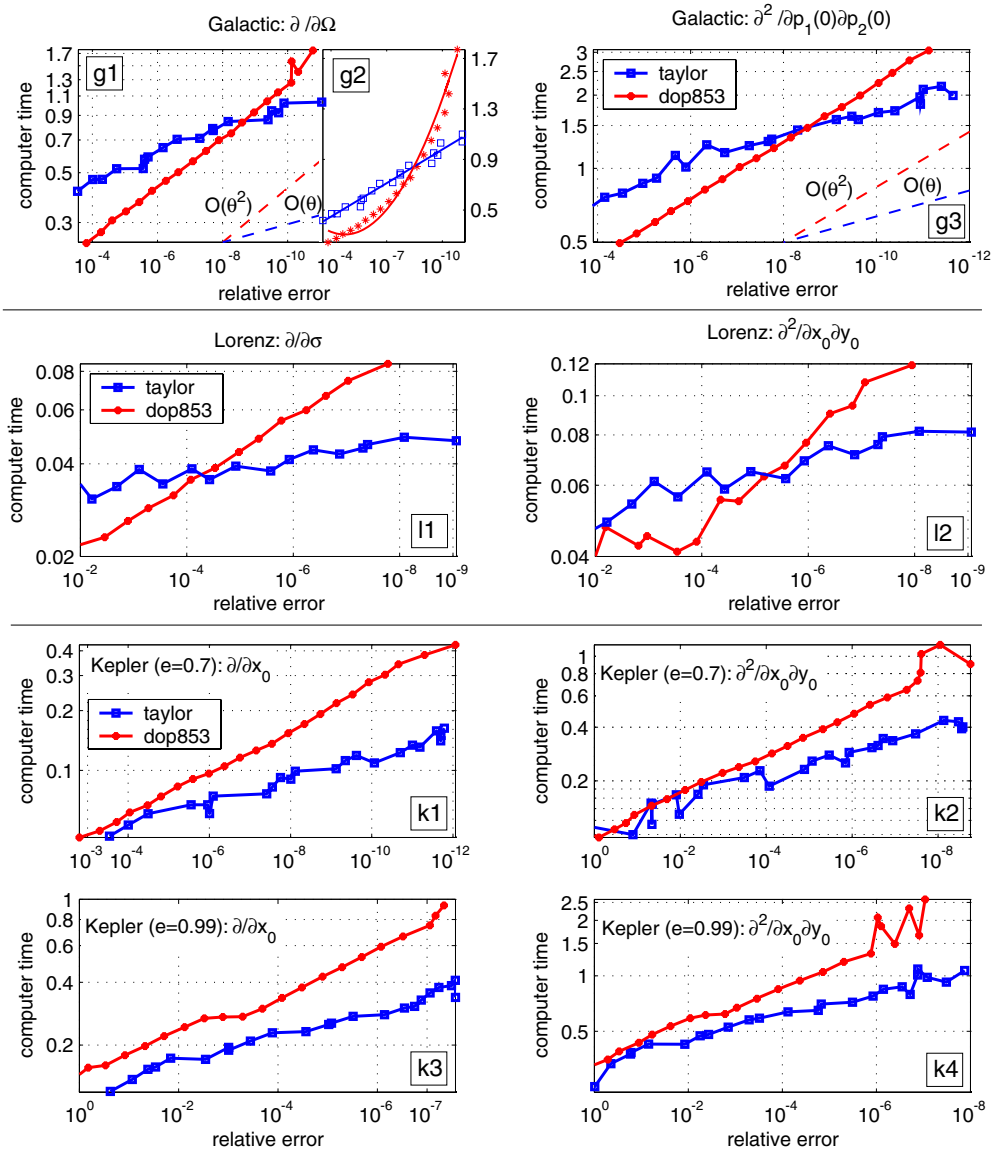


FIG. 5.1. *Precision-computer time diagrams for the Galactic, Lorenz, and Kepler problems in the first- and second-order sensitivities.*

second-order sensitivities with respect to the initial conditions $p_1(0)$ and $p_2(0)$ (graph (g3)) for the Galactic problem, first-order sensitivities with respect to the parameter σ (graph (l1)) and second-order sensitivities with respect to the initial conditions x_0 and y_0 (graph (l2)) for the Lorenz model, and first-order sensitivities with respect to the initial condition x_0 (graphs (k1) and (k3)) and second-order sensitivities with respect to the initial conditions x_0 and y_0 (graphs (k2) and (k4)) for the Kepler problem for two values of the eccentricity ($e = 0.7$ and $e = 0.99$). In all the simulations the relative error is computed at the end of the integration time using the infinity norm.

From the graphs we observe that for low-precision demands the RK code is faster,

but for high-precision demands the Taylor method becomes faster. This means that, increasing the accuracy request, the computer time increases much more rapidly for the RK method than for the Taylor method (due mainly to the variable order formulation of the Taylor method that gives a more adaptive method) and, therefore, the slopes of both curves are different. Thus, the Taylor method is especially useful for very high precision demands [3] (as they occur if we use quadruple precision or multiple precision libraries). In graphs (g1) and (g2) of Figure 5.1 we show the diagrams for the computation of first-order sensitivities with respect to the parameter Ω for the Galactic problem, (g1) in the logarithmic scale for both axis (`loglog`) and (g2) in the logarithmic scale only the relative error axis. From the (g2) graph we may observe that the evolution of the computer time for the RK code is $\mathcal{O}(\theta^2)$, while for the Taylor method it is $\mathcal{O}(\theta)$ with $\theta = -\log_{10} \|\texttt{error}\|$ (we also present a quadratic L_2-fit for the RK and a linear L_2-fit for the Taylor method in the continuous lines; the data are represented by symbols). Therefore, on a `loglog` scale (g1) the RK graph will be a straight line with a slope double the slope of the graph of the Taylor method. At the bottom right of graphs (g1) and (g3) we show with dashed lines the evolution of two straight lines in the `loglog` figures, one with a slope double that of the other.

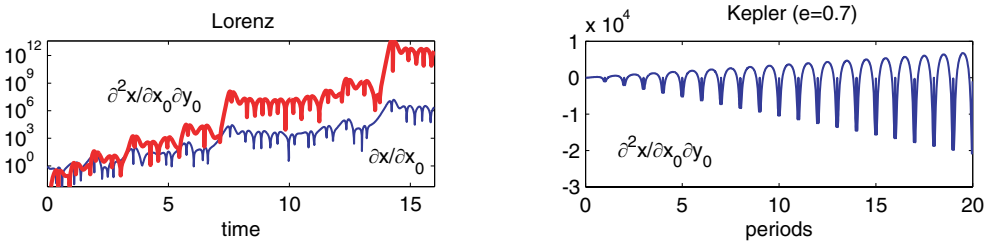


FIG. 5.2. *Evolution of the sensitivity values $\partial x/\partial x_0$ and $\partial^2 x/\partial x_0\,\partial y_0$ for the Lorenz model and $\partial^2 x/\partial x_0\,\partial y_0$ for the Kepler problem.*

Finally, in Figure 5.2, where we give the evolution of some sensitivity values, we observe the exponential dependence in the initial conditions of the Lorenz model but not in the Kepler problem, giving an indicator of chaos (see section 5.3 for a brief comment about chaos indicators).

5.2. DAE problems. We have taken three classical DAE problems (two of them are slightly stiff) as test problems.

• *Nonlinear simple pendulum.* A DAE system of index 3 (we consider the values $L = g = 1$):

$$\left\{\begin{array}{r} x'' + x\,\lambda = 0, \\ y'' + y\,\lambda - g = 0, \\ x^2 + y^2 - L^2 = 0, \end{array}\right. \quad \left|\begin{array}{l} x_0 = 1/2, \quad x_0' = 1, \\ y_0 = -\sqrt{1 - x(0)^2}, \quad y_0' = -x_0\,x_0'/y_0, \\ \lambda_0 = y_0 + (x_0')^2 + (y_0')^2 = \frac{1}{6}\,(8 - 3\,\sqrt{3}). \end{array}\right.$$

The integration interval is [0, 100].

• *Transistor–amplifier problem* (a simplified version given in [23] of Problem 6 of [32]). This problem is an example of a stiff DAE system of index 1:

$$M\,\frac{d\boldsymbol{U}}{dt} = \boldsymbol{f}(\boldsymbol{U}), \qquad \boldsymbol{U}(0) = \left(0,\, \frac{U_b\,R_1}{R_1 + R_2},\, \frac{U_b\,R_1}{R_1 + R_2},\, U_b,\, 0\right)^{\mathrm{T}}$$

with

$$M = \begin{pmatrix} -C_1 & C_1 & 0 & 0 & 0 \\ C_1 & -C_1 & 0 & 0 & 0 \\ 0 & 0 & -C_2 & 0 & 0 \\ 0 & 0 & 0 & -C_3 & C_3 \\ 0 & 0 & 0 & C_3 & -C_3 \end{pmatrix},$$

$$\boldsymbol{f}(\boldsymbol{U}) = \begin{pmatrix} -\dfrac{U_e(t)}{R_0} + \dfrac{U_1}{R_0} \\ -\dfrac{U_b}{R_2} + U_2\left(\dfrac{1}{R_1} + \dfrac{1}{R_2}\right) + (1-\alpha)\, g(U_2 - U_3) \\ \dfrac{U_3}{R_3} - g(U_2 - U_3) \\ -\dfrac{U_b}{R_4} + \dfrac{U_4}{R_4} + \alpha\, g(U_2 - U_3) \\ \dfrac{U_5}{R_5} \end{pmatrix},$$

$$g(x) = 10^{-6}\left(\exp\left(\frac{x}{0.026}\right) - 1\right), \quad U_e(t) = 0.4\sin(200\,\pi\, t), \quad \alpha = 0.99,$$

$$R_0 = 1000, \quad R_1 = \cdots = R_5 = 9000, \quad U_b = 6, \quad C_1 = 10^{-6}, \quad C_k = k\, C_1\ (k = 2, 3).$$

The integration interval is [0, 0.5].

• *The chemical Akzo problem* (Problem 1 of [32]). This problem is an example of a stiff DAEs system of index 1:

$$M\,\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(\boldsymbol{y}), \quad \boldsymbol{y}(0) = (0.444,\, 0.00123,\, 0,\, 0.0017,\, 0,\, K_s \cdot y_1(0) \cdot y_4(0))^{\mathrm{T}}$$

with

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{f}(\boldsymbol{y}) = \begin{pmatrix} -2r_1 + r_2 - r_3 - r_4 \\ -\frac{1}{2}\, r_1 - r_4 - \frac{1}{2}\, r_5 + F_{in} \\ r_1 - r_2 + r_3 \\ -2_2 + r_3 - 2r_4 \\ r_2 - r_3 + r_5 \\ K_s \cdot y_1 \cdot y_4 - y_6, \end{pmatrix},$$

$$r_1 = k_1 \cdot y_1^4 \cdot y_2^{\frac{1}{2}}, \qquad r_2 = k_2 \cdot y_3 \cdot y_4, \qquad r_3 = \frac{k_2}{K} \cdot y_1 \cdot y_5,$$

$$r_4 = k_3 \cdot y_1 \cdot y_4^2, \qquad r_5 = k_4 \cdot y_6^2 \cdot y_2^{\frac{1}{2}}, \qquad F_{in} = klA \cdot \left(\frac{p(\mathrm{CO}_2)}{H} - y_2\right).$$

The value of the parameters k_1, k_2, k_3, k_4, K, klA, $p(\mathrm{CO}_2)$, and H are $k_1 = 18.7$, $k_2 = 0.58$, $k_3 = 0.09$, $k_4 = 0.42$, $K_s = 115.83$, $K = 34.4$, $p(\mathrm{CO}_2) = 0.9$, $klA = 3.3$, and $H = 737$. The integration interval is [0, 180].

In Figure 5.3 we show precision-computer time diagrams in logarithmic scale for the `dop853` (only for the pendulum problem), `radau5` and `radau` codes, and the Taylor

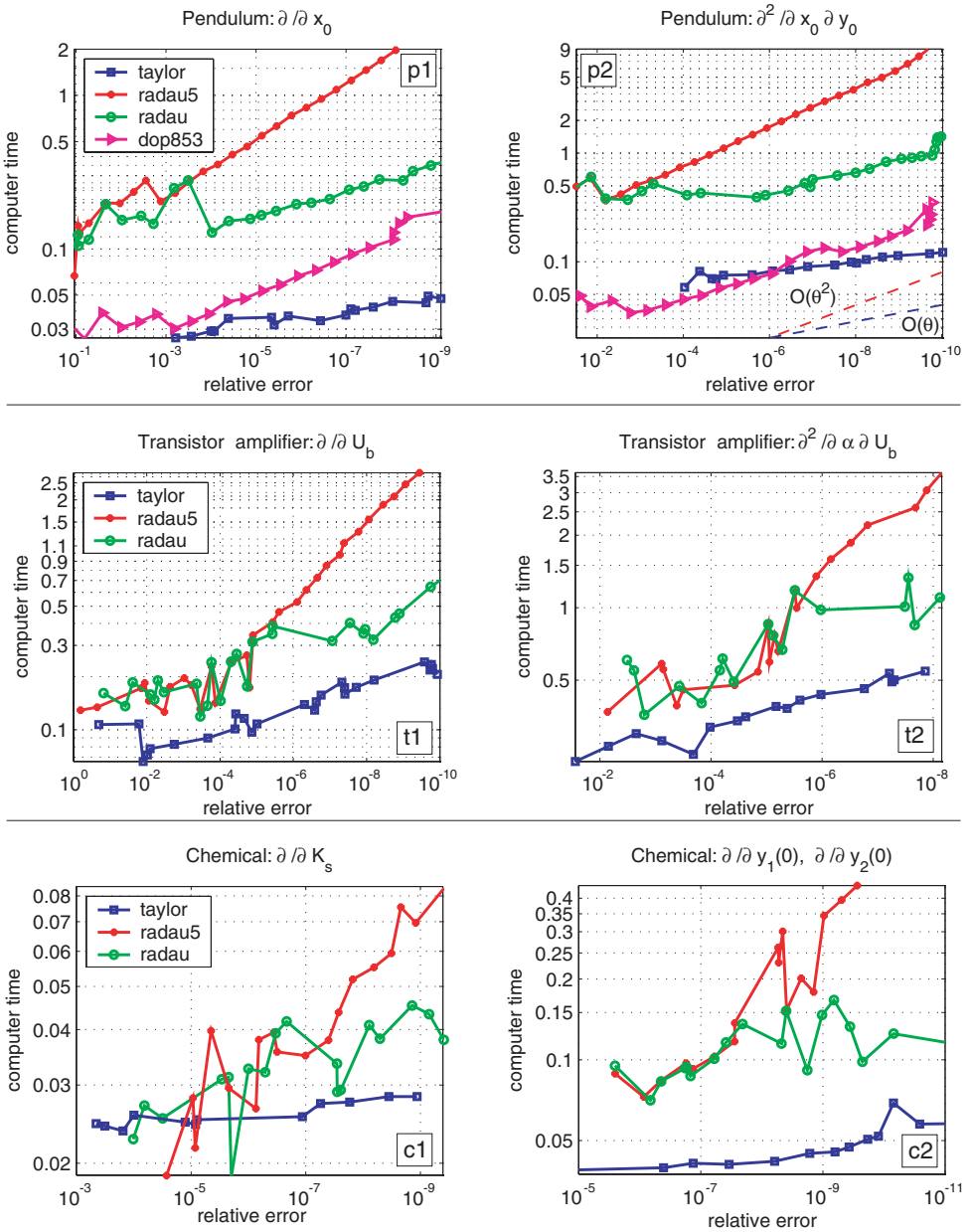


FIG. 5.3. *Precision-computer time diagrams for the pendulum, transistor–amplifier, and chemical problems in first- and second-order sensitivities.*

method for the computation of first- and second-order sensitivities for all the above three problems. Again, these graphs show the relative error and the computer time, and each point corresponds to a different requested accuracy. Because the algorithms are adaptive, neither the relative error nor the computer time depends continuously on the requested accuracy, which is more clear for DAEs than for ODEs, although we present the graphs connected with continuous lines to help the visualization. In

particular, we show first-order sensitivities with respect to the initial condition x_0 (graph (p1)) and second-order sensitivities with respect to the initial conditions x_0 and y_0 (graph (p2)) for the pendulum problem, first-order sensitivities with respect to the parameter U_b (graph (t1)) and second-order sensitivities with respect to the parameters α and U_b (graph (t2)) for the transistor–amplifier problem, and first-order sensitivities with respect to the parameter K_s (graph (c1)) and first-order sensitivities with respect to the initial conditions $y_1(0)$ and $y_2(0)$ (graph (c2)) for the chemical Akzo problem. In all the simulations the relative error is computed at the end of the integration time using the infinity norm.

From the graphs we observe that for low-precision demands the explicit methods (`dop853` for the pendulum problem and the Taylor method) are the most efficient, with the Taylor method the fastest for high-precision demands. Therefore, as occurs on the ODE tests, the slopes of the curves, for fixed-order RK (`dop853` and `radau5`) on one hand, and variable-order RK (`radau`) and the Taylor method on the other hand, are different due to the variable-order formulation that gives a more adaptive method. For very high precision demands the Taylor method is the method of choice (see [3] for numerical tests). At the bottom-right of graph (p2) we show with dashed lines the evolution of two straight lines in the `loglog` figures, one with a slope double that of the other.

In the DAE problems the initial conditions for the sensitivity values have to be consistent with the problem. For example, on the nonlinear pendulum we have taken as consistent initial conditions for first-order sensitivities:

$$\left(\frac{\partial x}{\partial x_0}(0),\, \frac{\partial x'}{\partial x_0}(0),\, \frac{\partial y}{\partial x_0}(0),\, \frac{\partial y'}{\partial x_0}(0),\, \frac{\partial \lambda}{\partial x_0}(0)\right) = \left(1,\, 0,\, \frac{1}{\sqrt{3}},\, \frac{8}{3\sqrt{3}},\, \frac{8+3\sqrt{3}}{9}\right).$$

In graphs (t1), (t2), (c1), and (c2) the code `dop853` does not work as the problems are stiff, but the Taylor method of high order works, and it is the fastest method. Although the Taylor method may be used for these moderately stiff problems, this is not true for highly stiff systems, where we need implicit methods.

5.3. Dynamical systems: Application to a chaos indicator. The above tests show that the Taylor method may be competitive in the numerical computation of sensitivities for some problems. Now, we present an application to dynamical systems (a discipline where Taylor methods may be quite useful [21, 29]), in particular to the determination of the $\mathrm{OFLI}^2_{\mathrm{TT}}$ chaos indicator [2]. These chaos indicator techniques try to analyze if a given initial condition generates a chaotic orbit or the orbit lies on an invariant torus. In the literature there exist several numerical tools as the Poincaré surfaces of section, the maximum Lyapunov exponent, the heliticity and twist angles, frequency map analysis, the mean exponential growth factor of nearby orbits (MEGNO), the fast Lyapunov indicator (FLI) [18], and the smaller alignment index (SALI). Most of them need the computation of first-order variational equations. A new chaos indicator called $\mathrm{OFLI}^2_{\mathrm{TT}}$, based on the FLI indicator, was recently introduced in [2]. The $\mathrm{OFLI}^2_{\mathrm{TT}}$ indicator at the final time t_f is defined as

$$\mathrm{OFLI}^2_{\mathrm{TT}} := \sup_{0<t<t_f} \log \left\| \left\{ \delta \boldsymbol{y}(t) + \frac{1}{2}\, \delta^2 \boldsymbol{y}(t) \right\}^{\perp} \right\|,$$

where $\delta \boldsymbol{y}$ and $\delta^2 \boldsymbol{y}$ are first- and second-order sensitivities with respect to carefully chosen initial vectors. In this case the variational equations are up to second-order

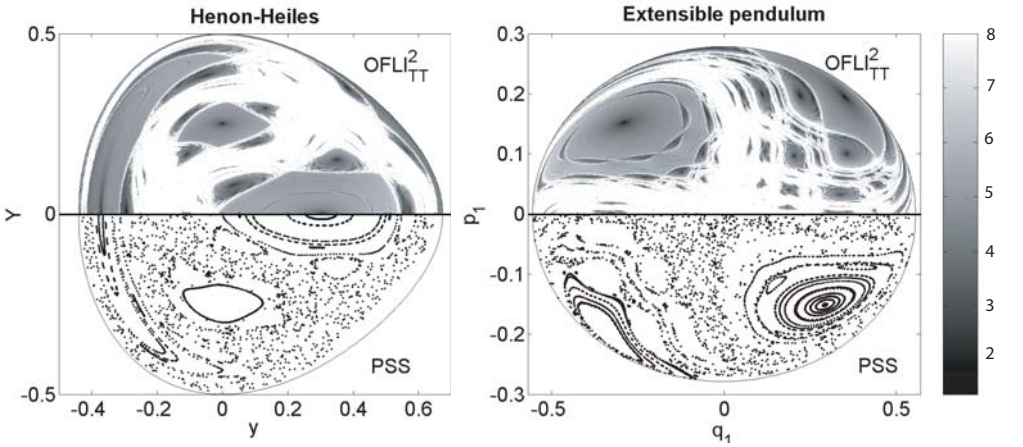


FIG. 5.4. $\mathrm{OFLI}^2_{\mathrm{TT}}$ *and Poincaré surfaces of section (PSS) of the Hénon–Heiles and the extensible-pendulum problems.*

and the initial conditions are

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{f}(t,\boldsymbol{y}), \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0,$$

$$\frac{d\,\delta\boldsymbol{y}}{dt} = \frac{\partial \boldsymbol{f}(t,\boldsymbol{y})}{\partial \boldsymbol{y}}\,\delta\boldsymbol{y}, \qquad \delta\boldsymbol{y}(0) = \boldsymbol{T}_0 := \frac{\boldsymbol{f}(0,\boldsymbol{y}_0)}{\|\boldsymbol{f}(0,\boldsymbol{y}_0)\|},$$

$$\frac{d\,\delta^2\boldsymbol{y}}{dt} = \frac{\partial \mathbf{f}}{\partial \boldsymbol{y}}\,\delta^2\boldsymbol{y} + \frac{\partial^2 \mathbf{f}}{\partial \boldsymbol{y}^2}\,(\delta\boldsymbol{y})^2, \quad \delta^2\boldsymbol{y}(0) = \mathbf{0}.$$

In the definition $\boldsymbol{v}^{\perp}$ stands for the component of $\boldsymbol{v}$ orthogonal to the flow. The $\mathrm{OFLI}^2_{\mathrm{TT}}$ requires the computation of second-order sensitivities with respect to a vector, and therefore the Taylor method may be used, requiring only the first- and second-order directional derivatives and not all the partial derivatives. We notice that the problems usually studied in dynamical systems are of low dimension and without a very complicated right-hand side. In this kind of problem the Taylor method gives a very good performance [3].

In Figure 5.4 we show the $\mathrm{OFLI}^2_{\mathrm{TT}}$ and the classical Poincaré sections of two classical problems in dynamical systems: the Hénon–Heiles and the extensible-pendulum problems. The Hénon–Heiles system [25] is given by the Hamiltonian

$$\mathcal{H}(x,\,y,\,X,\,Y) = \frac{1}{2}\,(X^2 + Y^2 + x^2 + y^2) + x^2\,y - \frac{1}{3}y^3,$$

and the extensible-pendulum problem [10] is given by the Hamiltonian

$$\mathcal{H}(q_1,\,q_2,\,p_1,\,p_2) = \frac{1}{2}\,(p_1^2 + p_2^2) + \frac{1}{2}\big((1-c)\,q_1^2 + q_2^2 - c\,q_1^2\,q_2\big).$$

On the tests we have considered $c = 0.75$. Due to the symmetry of the problems we show in the upper part of Figure 5.4 the $\mathrm{OFLI}^2_{\mathrm{TT}}$ and in the bottom part the Poincaré sections for the value of the energy $E = 1/8$ and $E = 31/800$ for the Hénon–Heiles and the extensible pendulum, respectively. The new technique permits us to obtain much more information than the Poincaré sections. The Taylor method has been

employed with a local error tolerance level of 10^{-12} for both techniques. For the Poincaré sections we use a *G-stop* criteria [14] (when $x = 0$ for the Hénon–Heiles and $q_2 = 0$ for the extensible pendulum) taking advantage of the continuous output as a power series. For the $\mathrm{OFLI}^2_{\mathrm{TT}}$ we use the Taylor method for computing sensitivities that permits a fast and easy computation of the values of the indicator (easy because the code is the "same" for just the integration and the integration and computation of the sensitivities). In the figure a high value of the chaos indicator (white) implies a chaotic behavior, whereas the lowest value (black) is associated with periodic orbits and the intermediate values (gray) are in the KAM tori. For more details about chaos indicators, the $\mathrm{OFLI}^2_{\mathrm{TT}}$, and its dynamical implications, see [2].

6. Conclusions. We have introduced the Taylor series method for the sensitivity analysis of ODE/DAE systems. In order to use this method we need a set of AD rules for the determination of the Taylor coefficients of the partial derivatives. From the numerical tests we observe that this method may be quite useful for low-dimensional problems such as in many interesting problems in dynamical systems [12, 21], where we are also interested in high precision. For this kind of problem the use of the Taylor series method gives an attractive alternative to other schemes. Although the numerical tests presented in this paper are not an exhaustive comparison among the different methods for computing sensitivity values, our numerical tests show that the Taylor series method is useful and provides good performance, at least for some kinds of problems.

Acknowledgments. The author thanks an anonymous referee for the numerous suggestions, remarks, and corrections that have contributed to improving the paper. The author thanks Professor J. D. Pryce for sending him a preprint of reference [36].

REFERENCES

[1] R. Barrio, *Performance of the Taylor series method for ODEs/DAEs*, Appl. Math. Comput., 163 (2005), pp. 525–545.
[2] R. Barrio, *Sensitivity tools vs. Poincaré sections*, Chaos Solitons Fractals, 25 (2005), pp. 711–726.
[3] R. Barrio, F. Blesa, and M. Lara, *VSVO formulation of the Taylor method for the numerical solution of ODEs*, Comput. Math. Appl., 50 (2005), pp. 93–111.
[4] M. Berz, *The method of power series tracking for the mathematical description of beam dynamics*, Nuclear Inst. Methods, A258 (1987), pp. 431–436.
[5] M. Berz, *Differential algebraic description of beam dynamics to very high orders*, Particle Accelerators, 24 (1989), pp. 109–124.
[6] D. Barton, *On Taylor series and stiff equations*, ACM Trans. Math. Software, 6 (1980), pp. 280–294.
[7] C. Bischof, A. Carle, P. Khademi, and A. Mauer, *ADIFOR* 2.0: *Automatic differentiation of Fortran* 77 *programs*, IEEE Comput. Sci. Engrg., 3 (1996), pp. 18–32.
[8] Y. Cao, S. Li, and L. Petzold, *Adjoint sensitivity analysis for differential-algebraic equations: Algorithms and software*, J. Comput. Appl. Math., 125 (2002), pp. 131–145.
[9] Y. Cao, S. Li, L. Petzold, and R. Serban, *Adjoint sensitivity analysis for differential-algebraic equations: The adjoint DAE system and its numerical solution*, SIAM J. Sci. Comput., 24 (2003), pp. 1076–1089.
[10] R. Carretero-González, H. N. Núñez-Yépez, and A. L. Salas-Brito, *Regular and chaotic behaviour in extensible pendulum*, European J. Phys., 15 (1994), pp. 139–148.
[11] Y. F. Chang and G. F. Corliss, *ATOMFT: Solving ODEs and DAEs using Taylor series*, Comput. Math. Appl., 28 (1994), pp. 209–233.
[12] W. G. Choe and J. Guckenheimer, *Computing periodic orbits with high accuracy*, Comput. Methods Appl. Mech. Engrg., 170 (1999), pp. 331–341.
[13] G. F. Corliss and Y. F. Chang, *Solving ordinary differential equations using Taylor series*, ACM Trans. Math. Software, 8 (1982), pp. 114–144.

[14] G. F. Corliss and Y. F. Chang, *G-stop facility in ATOMFT, a Taylor series ordinary differential equation solver*, in Computational Ordinary Differential Equations, S. O. Fatunla, ed., University Press, Champaign, IL, 1992, pp. 37–77.
[15] G. F. Corliss and G. Kirlinger, *On implicit Taylor series methods for stiff ODEs*, in Computer Arithmetic and Enclosure Methods (Oldenburg, 1991), North-Holland, Amsterdam, 1992, pp. 371–379.
[16] P. Eberhard and C. Bischof, *Automatic differentiation of numerical integration algorithms*, Math. Comp., 68 (1999), pp. 717–731.
[17] W. F. Feehery, J. E. Tolsma, and P. I. Barton, *Efficient sensitivity analysis of large-scale differential-algebraic systems*, Appl. Numer. Math., 25 (1997), pp. 41–54.
[18] M. Fouchard, E. Lega, C. Froeschlé, and C. Froeschlé, *On the relationship between fast Lyapunov indicator and periodic orbits for continuous flows*, Celestial Mech. Dynam. Astronom., 83 (2002), pp. 205–222.
[19] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers Appl. Math. 19, SIAM, Philadelphia, PA, 2000.
[20] A. Griewank, J. Utke, and A. Walther, *Evaluating higher derivative tensors by forward propagation of univariate Taylor series*, Math. Comp., 69 (2000), pp. 1117–1130.
[21] J. Guckenheimer and B. Meloon, *Computing periodic orbits and their bifurcations with automatic differentiation*, SIAM J. Sci. Comput., 22 (2000), pp. 951–985.
[22] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations* I. *Nonstiff Problems*, 2nd ed., Springer Ser. Comput. Math. 8, Springer-Verlag, New York, 1993.
[23] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II. *Stiff and Differential Algebraic Problems*, Springer Ser. Comput. Math. 14, Springer-Verlag, New York, 1991.
[24] E. Hairer and G. Wanner, *Stiff differential equations solved by Radau methods*, J. Comput. Appl. Math., 111 (1999), pp. 93–111.
[25] M. Hénon and C. Heiles, *The applicability of the third integral of motion: Some numerical experiments*, Astron. J., 1 (1964), pp. 73–79.
[26] J. Hoefkens, M. Berz, and K. Makino, *Computing validated solutions of implicit differential equations*, Adv. Comput. Math., 19 (2003), pp. 231–253.
[27] P. Hovland, C. Bischof, D. Spiegelman, and M. Casella, *Efficient derivative codes through automatic differentiation and interface contraction: An application in biostatistics*, SIAM J. Sci. Comput., 18 (1997), pp. 1056–1066.
[28] A. Jorba and M. Zou, *A software package for the numerical integration of ODE by means of high-order Taylor methods*, Experiment. Math., 14 (2005), pp. 99–117.
[29] M. Lara, A. Deprit, and A. Elipe, *Numerical continuation of families of frozen orbits in the zonal problem of artificial satellite theory*, Celestial Mech. Dynam. Astronom., 62 (1995), pp. 167–181.
[30] M. Lara, A. Elipe, and M. Palacios, *Automatic programming of recurrent power series*, Math. Comput. Simulation, 49 (1999), pp. 351–362.
[31] S. Li, L. Petzold, and W. Zhu, *Sensitivity analysis of differential-algebraic equations: A comparison of methods on a special problem*, Appl. Numer. Math., 32 (2000), pp. 161–174.
[32] W. M. Lioen and J. J. B. de Swart, *Test Set for Initial Value Problem Solvers*, Release 2.1, 1999; available online from http://ftp.cwi.nl/IVPtestset/report/testset.pdf.
[33] K. Makino and M. Berz, *Remainder differential algebras and their applications*, in Computational Differentiation (Santa Fe, NM, 1996), SIAM, Philadelphia, PA, 1996, pp. 63–74.
[34] R. E. Moore, *Interval Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1966.
[35] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, *Validated solutions of initial value problems for ordinary differential equations*, Appl. Math. Comput., 105 (1999), pp. 21–68.
[36] N. S. Nedialkov and J. D. Pryce, *Solving differential-algebraic equations by Taylor series* (I)*: Computing Taylor coefficients*, BIT, 45 (2005), pp. 561–591.
[37] R. Neidinger, *An efficient method for the numerical evaluation of partial derivatives of arbitrary order*, ACM Trans. Math. Software, 18 (1992), pp. 159–173.
[38] P. J. Prince and J. R. Dormand, *High order embedded Runge-Kutta formulae*, J. Comput. Appl. Math., 7 (1981), pp. 67–75.
[39] J. D. Pryce, *Solving high-index DAEs by Taylor series*, Numer. Algorithms, 19 (1998), pp. 195–211.
[40] J. D. Pryce, *A simple structural analysis method for DAEs*, BIT, 41 (2001), pp. 364–394.
[41] L. B. Rall, *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Comput. Sci. 120, Springer-Verlag, Berlin, 1981.
[42] A. Sandu, G. R. Carmichael, and F. A. Potra, *Sensitivity analysis for atmospheric chemistry models via automatic differentiation*, Atmos. Environ., 31 (1997), pp. 475–489.
[43] P. Zgliczynski, *C^1-Lohner algorithm*, Found. Comput. Math., 2 (2002), pp. 429–465.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1948–1980
© 2006 Society for Industrial and Applied Mathematics

SYSTEMATIC DERIVATION OF JUMP CONDITIONS FOR THE IMMERSED INTERFACE METHOD IN THREE-DIMENSIONAL FLOW SIMULATION*

SHENG XU† AND Z. JANE WANG†

Abstract. In this paper, we systematically derive jump conditions for the immersed interface method [*SIAM J. Numer. Anal.*, 31 (1994), pp. 1019–1044; *SIAM J. Sci. Comput.*, 18 (1997), pp. 709–735] to simulate three-dimensional incompressible viscous flows subject to moving surfaces. The surfaces are represented as singular forces in the Navier–Stokes equations, which give rise to discontinuities of flow quantities. The principal jump conditions across a closed surface of the velocity, the pressure, and their normal derivatives have been derived by Lai and Li [*Appl. Math. Lett.*, 14 (2001), pp. 149–154]. In this paper, we first extend their derivation to generalized surface parametrization. Starting from the principal jump conditions, we then derive the jump conditions of all first-, second-, and third-order spatial derivatives of the velocity and the pressure. We also derive the jump conditions of first- and second-order temporal derivatives of the velocity. Using these jump conditions, the immersed interface method is applicable to the simulation of three-dimensional incompressible viscous flows subject to moving surfaces, where near the surfaces the first- and second-order spatial derivatives of the velocity and the pressure can be discretized with, respectively, third- and second-order accuracy, and the first-order temporal derivatives of the velocity can be discretized with second-order accuracy.

Key words. immersed interface method, immersed boundary method, Cartesian grid methods, jump conditions, three-dimensional Navier–Stokes equations, singular force

AMS subject classifications. 76D05, 76M20, 65M06

DOI. 10.1137/040604960

1. Introduction.

Blood flow in the heart [23, 24, 25], aquatic animal locomotion [10, 15], bird and insect flight [31, 32, 3, 8, 9, 13], and flow passing a compliant wall [35, 6, 14, 4] are examples of fluid dynamics problems where it is essential to understand the coupling between moving surfaces and fluids. A main difficulty in direct numerical simulation of these problems is to accurately and efficiently resolve the moving surfaces and their effects on the fluids.

Cartesian grid methods, for example [28, 5, 30, 22, 21], avoid mesh regeneration and allow for fast flow solvers, and thus have the advantage of simplicity and efficiency for this type of problem. The immersed boundary method is a robust Cartesian grid method. It was originally proposed by Peskin [23, 24] and later further developed in [29, 26, 27, 16]. In this method, the moving surface of an immersed object is parametrized by a set of Lagrangian points comoving with a fluid. The relative positions of the Lagrangian points determine a singular force distribution on the surface based on the solid model of the object. The communication between the surface and the fluid is achieved through the spreading of the singular force and the interpolation of the surface velocity using discrete Dirac δ functions. The method has been applied to a wide variety of problems [10, 11, 25, 1, 37].

The initial implementations of the immersed boundary method were only first-order accurate in space due to the use of grid-dependent discrete Dirac δ functions.

*Received by the editors March 9, 2004; accepted for publication (in revised form) April 28, 2005; published electronically February 3, 2006.

http://www.siam.org/journals/sisc/27-6/60496.html

†Department of Theoretical and Applied Mechanics, Cornell University, Ithaca, NY 14853 (sx12@cornell.edu, jane.wang@cornell.edu).

Beyer and LeVeque [2] examined the accuracy of the method for the one-dimensional diffusion equation and found that additional terms for the discrete approximation of the Dirac δ function are sometimes necessary in order to achieve second-order accuracy, but it is unclear how to maintain the second-order accuracy by incorporating additional terms in fluid dynamics problems of higher dimensions. Although a formally second-order immersed boundary method was proposed [16], it is second-order accurate only if the Dirac δ function is replaced by a grid-independent smooth function; in practice, it is still first-order accurate. Realizing that only the divergence-free portion of the singular force contributes to the temporal evolution of the velocity, and that the projection of discrete Dirac δ functions onto a divergence-free space may be computed analytically, Cortez and Minion [7] devised the blob projection immersed boundary method, which displays the formally fourth-order convergence rate of their background flow solver. However, the analytical form of the projection depends on the velocity boundary conditions imposed on a computational domain. Thus the method may be limited to particular boundary conditions in simple geometries. It is also unclear how accurately the pressure can be recovered.

Motivated by the goal of eventually obtaining second-order accuracy in Peskin's immersed boundary method, LeVeque and Li [18, 19] have developed the immersed interface method (IIM). The IIM was originally proposed for elliptic equations [18] and Stokes equations [19]. Later, it was extended to one-dimensional nonlinear parabolic equations [33], Poisson equations with Neumann boundary conditions [34, 12], and two-dimensional incompressible Navier–Stokes equations [20]. The key idea of the IIM, which is also its main difference from the immersed boundary method, is to incorporate the known jump conditions of a solution and its derivatives into finite difference schemes in the neighborhood of the discontinuities arising from the Dirac δ function. For fluid dynamics problems with moving surfaces, the coupling between the moving surfaces and the fluids is now translated into the incorporation of the jump conditions.

If necessary jump conditions are known, the IIM can achieve second-order or even higher-order accuracy. The two-dimensional Navier–Stokes simulations by Li and Lai [20] using the IIM have indicated fully second-order accuracy for the velocity and nearly second-order accuracy for the pressure in the infinity norm. Solutions computed by the IIM are sharper across surfaces than those computed by the immersed boundary method. Furthermore, the IIM shows better conservation of the mass enclosed by a no-penetration surface.

Like any other method, the IIM has its limitations. For instance, the current IIM applies only to flows with closed smooth surfaces, as seen in its presentation later in this paper. Both the IIM and the immersed boundary method also inherit the shortcomings of fixed Cartesian grid methods. For example, thin boundary layers developed along a moving boundary and fine geometric details can be adequately resolved only if the uniform computational mesh is fine enough. It should also be noted that, for many bio-fluid dynamics problems, the computation of the singular force distribution is a modeling process, and improvement in the accuracy of the IIM or the immersed boundary method cannot eliminate modeling errors.

The applicability of the IIM depends on whether the necessary jump conditions are all known. The principal jump conditions across a closed surface of the velocity, the pressure, and their normal derivatives have been derived by Lai and Li [17] for three-dimensional incompressible viscous flows. The main contribution of our paper is to derive for the IIM the necessary jump conditions of flow variables and their

derivatives to achieve a given-order discretization accuracy in three-dimensional flow simulation. Other contributions include generalized Taylor expansion, which is the basis for devising finite difference schemes for the IIM; a generalized Gauss theorem, which serves as the basic tool in the derivation of the principal jump conditions; the principal jump conditions in generalized surface parametrization, which bring the flexibility to parametrize a singular surface in practical applications; and the jump conditions of temporal derivatives, which are required to achieve first-order or higher-order temporal discretization accuracy.

The content of the paper is organized as follows. In section 2, the governing equations are described; they are the starting point for the derivation of the principal jump conditions. In section 3, the principal jump conditions are derived. In section 4, finite difference schemes with jump conditions incorporated are presented. In sections 5 and 6, the required spatial and temporal jump conditions are derived. A simple example is also provided in section 6 to address the proper discretization of temporal derivatives. In section 7, the possibility of improving the IIM to arbitrarily high-order discretization accuracy is discussed.

Since the original submission of this paper, we have implemented and tested the IIM in two-dimensional flow simulations with jump conditions obtained from our theoretical derivation below. Please refer to Xu and Wang [36] for the full numerical implementation and the test results. The test results serve in part to verify our derivation in the current paper. We have also progressed on the development of a three-dimensional code. We hope to provide the three-dimensional results soon.

2. Governing equations. Incompressible Navier–Stokes equations subject to singular force are

$$\rho\left(\frac{\partial u^i}{\partial t}+u^j\frac{\partial u^i}{\partial x^j}\right)=-\frac{\partial p}{\partial x^i}+\mu\frac{\partial^2 u^i}{\partial x^j\partial x^j}+F^i, \tag{2.1}$$

$$\frac{\partial u^i}{\partial x^i}=0, \tag{2.2}$$

where $x^i(i=1,2,3)$ is in Cartesian coordinates, t is time, ρ is fluid density, u^i is velocity, p is pressure, μ is dynamic fluid viscosity, and F^i is the singular force. Taking the divergence of momentum equation (2.1) and applying continuity condition (2.2) gives the Poisson equation for pressure p as

$$\frac{\partial^2 p}{\partial x^i\partial x^i}=\frac{\partial F^i}{\partial x^i}-\frac{\partial}{\partial x^i}\left(\rho\frac{\partial u^i}{\partial t}+\rho u^j\frac{\partial u^i}{\partial x^j}\right). \tag{2.3}$$

We consider the situation that the singular force is applied on the closed surface of an immersed object, and we call the surface singular surface $\mathcal{S}$. Referring to Figure 2.1, singular force F^i is given by

$$F^i=\int_{\mathcal{S}} f^i(\alpha^1,\alpha^2,t)\delta(\mathbf{x}-\mathbf{X}(\alpha^1,\alpha^2,t))\mathrm{d}\alpha^1\mathrm{d}\alpha^2, \tag{2.4}$$

in which $\mathbf{x}:=(x^1,x^2,x^3)$ is the Cartesian coordinates, $\mathbf{X}(\alpha^1,\alpha^2,t):=(X^1,X^2,X^3)$ is the coordinates of the singular surface, $\delta(\mathbf{x}-\mathbf{X}(\alpha^1,\alpha^2,t))$ is the three-dimensional Dirac δ function, $f^i(\alpha^1,\alpha^2,t)$ is force density, and α^1 and α^2 are two Lagrangian parameters which parametrize the singular surface at a reference time. We assume

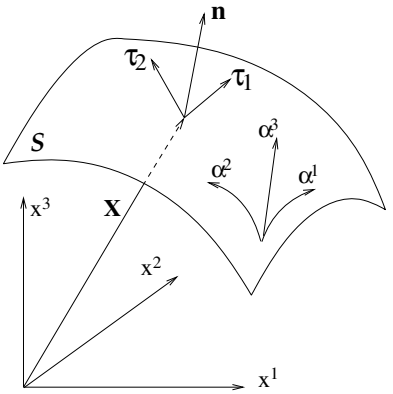


FIG. 2.1. *Cartesian coordinates* $x^i (i = 1, 2, 3)$ *and curvilinear coordinates* $\alpha^i (i = 1, 2, 3)$. α^1 *and* α^2 *are two Lagrangian parameters which parametrize the singular surface* $\mathcal{S}$ *at a reference time.* $\mathbf{X}$ *is the coordinates of the singular surface.* $\boldsymbol{\tau_1}$, $\boldsymbol{\tau_2}$, *and* $\mathbf{n}$ *are respectively the two tangential vectors and the normal vector at* $\mathbf{X}$.

that $f^i(\alpha^1, \alpha^2, t)$ is a smooth function of α^1, α^2, and t. Mathematically, F^i can be regarded as a distribution function with the property

$$\int_\Omega F^i \mathrm{d}\Omega = \int_{\Omega \cap \mathcal{S}} f^i(\alpha^1, \alpha^2, t) \mathrm{d}(\Omega \cap \mathcal{S}),$$

where Ω can be a volume or a surface.

3. Principal jump conditions. We now derive the principal jump conditions of the velocity, u^i, the pressure, p, and their normal derivatives, $\frac{\partial u^i}{\partial \mathbf{n}}$ and $\frac{\partial p}{\partial \mathbf{n}}$, across the singular surface, $\mathcal{S}$.

We assume that the velocity, the pressure, and their derivatives are piecewise smooth with discontinuities only at the singular surface. The singular surface, $\mathcal{S}$: $\mathbf{X} = (X^1, X^2, X^3)$, is assumed geometrically regular and orientable. For every point on the surface, the parametrization by α^1 and α^2 generates a rank-two matrix

$$\begin{pmatrix} \dfrac{\partial X^1}{\partial \alpha^1} & \dfrac{\partial X^2}{\partial \alpha^1} & \dfrac{\partial X^3}{\partial \alpha^1} \\ \dfrac{\partial X^1}{\partial \alpha^2} & \dfrac{\partial X^2}{\partial \alpha^2} & \dfrac{\partial X^3}{\partial \alpha^2} \end{pmatrix}.$$

The row vectors in the above matrix are two independent tangential vectors at the point (see Figure 2.1)

$$\boldsymbol{\tau_1} := (\tau_1^1, \tau_1^2, \tau_1^3) = \left(\frac{\partial X^1}{\partial \alpha^1}, \frac{\partial X^2}{\partial \alpha^1}, \frac{\partial X^3}{\partial \alpha^1} \right),$$
$$\boldsymbol{\tau_2} := (\tau_2^1, \tau_2^2, \tau_2^3) = \left(\frac{\partial X^1}{\partial \alpha^2}, \frac{\partial X^2}{\partial \alpha^2}, \frac{\partial X^3}{\partial \alpha^2} \right).$$

A unit normal vector can be expressed by

$$\mathbf{n} := (n^1, n^2, n^3) = \frac{\boldsymbol{\tau_1} \times \boldsymbol{\tau_2}}{\|\boldsymbol{\tau_1} \times \boldsymbol{\tau_2}\|} = \frac{\boldsymbol{\tau_1} \times \boldsymbol{\tau_2}}{\mathcal{J}},$$

where $\|\cdot\|$ denotes the length of a vector and $\mathcal{J} := \|\tau_{\mathbf{1}} \times \tau_{\mathbf{2}}\|$; see Figure 2.1. In Cartesian coordinates, $\mathbf{n} = (n^1, n^2, n^3) = (n_1, n_2, n_3)$, where $n^i (i = 1, 2, 3)$ is a contravariant component and $n_i (i = 1, 2, 3)$ a covariant component.

PROPOSITION 3.1. *For a flow defined by* (2.1) *and* (2.2)*, the velocity,* u^i*, is finite and continuous at the immersed singular surface,* $\mathcal{S}$*; i.e.,*

$$[u^i] = 0, \tag{3.1}$$

where $[\cdot]$ *denotes a jump as* $[\cdot] := (\cdot)_{\mathbf{X}^+} - (\cdot)_{\mathbf{X}^-}$*, with* $\mathbf{X}^+$ *representing the point at the side of* $\mathcal{S}$ *in the direction of normal* $\mathbf{n}$ *and* $\mathbf{X}^-$ *the point at the other side.*

Continuity equation (2.2) assures only the continuity of the normal velocity component across the singular surface. In our applications, the singular surface is a physical boundary immersed in an incompressible viscous fluid. No-slip and no-penetration conditions on this physical boundary require the velocity to be continuous across the singular surface and the singular surface to move with the local flow velocity, as expressed mathematically by Proposition 3.1.

COROLLARY 3.2.

$$\left[\frac{\partial u^i}{\partial t} + u^j \frac{\partial u^i}{\partial x^j}\right] = \left[\frac{\partial u^i}{\partial t}\right] + u^j \left[\frac{\partial u^i}{\partial x^j}\right] = 0. \tag{3.2}$$

Proof. A jump condition is a function of the time t and the surface coordinates $\mathbf{X}(\alpha^1, \alpha^2, t)$, i.e., $[\cdot] = [\cdot](\mathbf{X}, t)$. Differentiating (3.1) with respect to time t gives

$$\begin{aligned}
\frac{\mathrm{d}[u^i](\mathbf{X}, t)}{\mathrm{d}t} &= \frac{\mathrm{d}u^i(\mathbf{X}^+, t)}{\mathrm{d}t} - \frac{\mathrm{d}u^i(\mathbf{X}^-, t)}{\mathrm{d}t} \\
&= \frac{\partial u^i(\mathbf{X}^+, t)}{\partial t} + \frac{\partial u^i(\mathbf{X}^+, t)}{\partial x^j}\frac{\mathrm{d}X^{j+}}{\mathrm{d}t} \\
&\quad - \left(\frac{\partial u^i(\mathbf{X}^-, t)}{\partial t} + \frac{\partial u^i(\mathbf{X}^-, t)}{\partial x^j}\frac{\mathrm{d}X^{j-}}{\mathrm{d}t}\right) \\
&= \left[\frac{\partial u^i}{\partial t} + \frac{\partial u^i}{\partial x^j}\frac{\mathrm{d}X^j}{\mathrm{d}t}\right] = 0.
\end{aligned}$$

From Proposition 3.1, we have $\frac{\mathrm{d}X^{j+}}{\mathrm{d}t} = \frac{\mathrm{d}X^{j-}}{\mathrm{d}t} = u^j$. Thus, the result follows. ∎

The notion of a jump condition, $[\cdot]$, commutes with a differentiation, and we can therefore write the function form of a jump condition as $[\cdot(\mathbf{X}, t)]$ to carry out the differentiation hereafter.

To derive the other principal jump conditions, we need to generalize Gauss's theorem. Gauss's theorem in the usual form reads

$$\int_{\mathcal{V}} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} = \oint_{\mathcal{A}} G^i N_i \mathrm{d}\mathcal{A},$$

where $\mathcal{A}$ is a regular and positively oriented closed surface, $\mathcal{V}$ is the region enclosed by $\mathcal{A}$, and N_i is a normal to $\mathcal{A}$. It is required that the function G^i and its divergence $\frac{\partial G^i}{\partial x^i}$ be continuous over $\mathcal{A}$ and $\mathcal{V}$. We generalize Gauss's theorem to relax these continuity restrictions.

THEOREM 3.3 (generalized Gauss theorem). *If* G^i *and* $\frac{\partial G^i}{\partial x^i}$ *are continuous over* $\mathcal{A}$ *and* $\mathcal{V}$ *except that* G^i *has finite jumps or a singularity of the Dirac* δ *function type at the singular surface* $\mathcal{S}$ *enclosed in* $\mathcal{V}$*, then*

$$\int_{\mathcal{V}} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} = \oint_{\mathcal{A}} G^i N_i \mathrm{d}\mathcal{A}. \tag{3.3}$$

The proof of Theorem 3.3 is given in Appendix A.

LEMMA 3.4. *For a flow defined by* (2.1) *and* (2.2)*, the jump conditions across singular surface* $\mathcal{S}$ *of pressure* p *and normal velocity derivative* $\frac{\partial u^i}{\partial \mathbf{n}}$ *satisfy*

$$[p] = \frac{f^k n_k}{\mathcal{J}}, \tag{3.4}$$

$$\left[\frac{\partial u^i}{\partial \mathbf{n}}\right] = \frac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}}. \tag{3.5}$$

Equation (3.4) states that the normal force on the singular surface is balanced by the difference of the pressure force across the singular surface, and (3.5) states that the tangential force on the singular surface is balanced by the difference of the shear force across the singular surface. Below, we prove Lemma 3.4 through the force balance on a control volume. The same results were obtained by Lai and Li [17] by using the test function method.

Proof. Take an infinitesimal area, $\delta\mathcal{S} = \mathcal{J}\delta\alpha^1\delta\alpha^2$, on singular surface $\mathcal{S}$, which corresponds to an infinitesimal area, $\delta\alpha^1\delta\alpha^2$, in the parameter space. Translate $\delta\mathcal{S}$ in the directions of $\mathbf{n}$ and $-\mathbf{n}$ by $\epsilon/2$, and denote the swept region $\delta\mathcal{V}$. Integrating (2.1) over $\delta\mathcal{V}$ and letting $\epsilon \to 0$ yields

$$\lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} \rho\left(\frac{\partial u^i}{\partial t} + u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V} = \lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} \frac{\partial}{\partial x^j}\left(-p\delta^{ij} + \mu\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V} + \lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} F^i \mathrm{d}\mathcal{V}, \tag{3.6}$$

where δ^{ij} is the Kronecker symbol. Applying (2.2) and the Reynolds transport theorem, the left-hand side of (3.6) becomes

$$\lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} \rho\left(\frac{\partial u^i}{\partial t} + u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V} = \lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} \left(\frac{\partial \rho u^i}{\partial t} + \frac{\partial \rho u^i u^j}{\partial x^j}\right)\mathrm{d}\mathcal{V} = \lim_{\epsilon\to 0}\frac{\mathrm{d}}{\mathrm{d}t}(\rho u^i \delta\mathcal{V}) = 0.$$

Applying our generalized Gauss theorem to the first term on the right-hand side of (3.6) yields

$$\begin{aligned}\lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} \frac{\partial}{\partial x^j}\left(-p\delta^{ij} + \mu\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V} &= \lim_{\epsilon\to 0}\int_{\delta\mathcal{S}} n_j\left[-p\delta^{ij} + \mu\frac{\partial u^i}{\partial x^j}\right]\mathrm{d}\mathcal{S} \\ &= \left(-[p]n^i + \mu\left[\frac{\partial u^i}{\partial x^k}\right]n^k\right)\delta\mathcal{S}.\end{aligned}$$

The last term in (3.6) is

$$\begin{aligned}\lim_{\epsilon\to 0}\int_{\delta\mathcal{V}} F^i \mathrm{d}\mathcal{V} &= \lim_{\epsilon\to 0}\int_{\mathcal{S}}\int_{\delta\mathcal{V}} f^i(\alpha^1,\alpha^2,t)\delta(\mathbf{x} - \mathbf{X}(\alpha^1,\alpha^2,t))\mathrm{d}\mathcal{V}\mathrm{d}\alpha^1\mathrm{d}\alpha^2 \\ &= f^i(\alpha^1,\alpha^2,t)\delta\alpha^1\delta\alpha^2.\end{aligned}$$

Thus,

$$\begin{aligned}\left(-[p]n^i + \mu\left[\frac{\partial u^i}{\partial x^k}\right]n^k\right)\delta\mathcal{S} + f^i(\alpha^1,\alpha^2,t)\delta\alpha^1\delta\alpha^2 &= 0 \\ \Rightarrow -[p]n^i + \mu\left[\frac{\partial u^i}{\partial x^k}\right]n^k &= -\frac{f^i(\alpha^1,\alpha^2,t)}{\mathcal{J}}.\end{aligned}$$

Multiplying n_i above and applying the facts that $n^k n_i + \tau^k \tau_i + b^k b_i = \delta_i^k$ (where δ_i^k is the Kronecker symbol, and $\mathbf{n}$, τ, and $\mathbf{b}$ are mutually orthogonal unit vectors), $\frac{\partial u^i}{\partial x^i} = 0$, and $[\frac{\partial u^i}{\partial \tau}] = [\frac{\partial u^i}{\partial \mathbf{b}}] = 0$ (from $[u^i] = 0$), we obtain

$$[p] = \frac{f^i n_i}{\mathcal{J}},$$
$$\mu \left[\frac{\partial u^i}{\partial x^k}\right] n^k = \frac{-f^i}{\mathcal{J}} + [p] n^i \Rightarrow \left[\frac{\partial u^i}{\partial \mathbf{n}}\right] = \frac{(f^k n_k) n^i - f^i}{\mu \mathcal{J}}. \qquad \square$$

Now we use the test function method to derive the jump condition of the normal pressure derivative, $\frac{\partial p}{\partial \mathbf{n}}$, across the singular surface with generalized surface parametrization. To prepare for the derivation, we first introduce a coordinate transformation (see Figure 2.1) between Cartesian coordinates $x^i (i = 1, 2, 3)$ and curvilinear coordinates $\alpha^i (i = 1, 2, 3)$ as

$$x^i = x^i(\alpha^1, \alpha^2, \alpha^3),$$
$$\alpha^i = \alpha^i(x^1, x^2, x^3),$$

where α^3 is a new coordinate with $\mathbf{x}(\alpha^1, \alpha^2, \alpha^3 = 0, t) = \mathbf{X}(\alpha^1, \alpha^2, t)$ corresponding to the singular surface, $\mathcal{S}$, and α^3 is chosen to satisfy

$$\frac{\partial \mathbf{x}(\alpha^1, \alpha^2, \alpha^3 = 0, t)}{\partial \alpha^3} = \mathbf{n}.$$

It can be shown that

$$\nabla \alpha^3 = \mathbf{n},$$

where ∇ is the gradient operator in the Cartesian coordinate system. Let $\tilde{f}^i$ be a contravariant component of the forcing density vector in the curvilinear coordinate system. It is related to f^i through

$$\tilde{f}^i = \frac{\partial \alpha^i}{\partial x^j} f^j,$$

where $\frac{\partial \alpha^i}{\partial x^j}$ satisfies

$$\frac{\partial x^k}{\partial \alpha^i} \frac{\partial \alpha^i}{\partial x^j} = \delta_j^k,$$

in which δ_j^k is the Kronecker symbol and $\frac{\partial x^k}{\partial \alpha^i}$ can be written in a matrix form on the singular surface $\mathcal{S}$ as

$$C_1^T := \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 \\ n^1 & n^2 & n^3 \end{pmatrix}^T, \tag{3.7}$$

where superscript T denotes the transposition of a matrix. As the determinant of C_1 is $|C_1| = \mathbf{n} \cdot (\tau_\mathbf{1} \times \tau_\mathbf{2}) \neq 0$, C_1 is nonsingular.

LEMMA 3.5. *The jump condition of normal pressure derivative* $\frac{\partial p}{\partial \mathbf{n}}$ *across the singular surface is*

$$\left[\frac{\partial p}{\partial \mathbf{n}}\right] = \frac{1}{\mathcal{J}} \left(\frac{\partial \tilde{f}^1}{\partial \alpha^1} + \frac{\partial \tilde{f}^2}{\partial \alpha^2}\right). \tag{3.8}$$

Proof. Take control volume $\mathcal{V}_s$ so that $\mathcal{V}_s$ is a layer with thickness ϵ containing singular surface $\mathcal{S}$. Denote the surface of $\mathcal{V}_s$ by $\mathcal{A}$. Multiplying a smooth test function, $\phi(\mathbf{x})$, by (2.3) and then integrating over $\mathcal{V}_s$ with $\epsilon \to 0$ yields

$$\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial^2 p}{\partial x^i\partial x^i}\mathrm{d}\mathcal{V}=\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial F^i}{\partial x^i}\mathrm{d}\mathcal{V}-\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial}{\partial x^i}\left(\rho\frac{\partial u^i}{\partial t}+\rho u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V}. \tag{3.9}$$

With Theorem 3.3, the term on the left-hand side of (3.9) can be written as

$$\begin{aligned}\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial^2 p}{\partial x^i\partial x^i}\mathrm{d}\mathcal{V}&=\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\left(\frac{\partial}{\partial x^i}\left(\phi\frac{\partial p}{\partial x^i}\right)+p\frac{\partial^2\phi}{\partial x^i\partial x^i}-\frac{\partial}{\partial x^i}\left(p\frac{\partial\phi}{\partial x^i}\right)\right)\mathrm{d}\mathcal{V}\\&=\int_{\mathcal{S}}\left(\phi\left[\frac{\partial p}{\partial x^i}\right]n^i-[p]\frac{\partial\phi}{\partial x^i}n^i\right)\mathrm{d}\mathcal{S}\\&=\int_{\mathcal{S}}\left(\phi\left[\frac{\partial p}{\partial \mathbf{n}}\right]-[p]\frac{\partial\phi}{\partial\mathbf{n}}\right)\mathcal{J}\mathrm{d}\alpha^1\mathrm{d}\alpha^2.\end{aligned}$$

Due to Theorem 3.3, the first term on the right-hand side of (3.9) becomes

$$\begin{aligned}\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial F^i}{\partial x^i}\mathrm{d}\mathcal{V}&=\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\frac{\partial(\phi F^i)}{\partial x^i}\mathrm{d}\mathcal{V}-\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}F^i\frac{\partial\phi}{\partial x^i}\mathrm{d}\mathcal{V}\\&=-\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}F^i\frac{\partial\phi}{\partial x^i}\mathrm{d}\mathcal{V}=-\int_{\mathcal{S}}f^i(\alpha^1,\alpha^2,t)\frac{\partial\phi}{\partial x^i}\mathrm{d}\alpha^1\mathrm{d}\alpha^2\\&=-\int_{\mathcal{S}}\left(\tilde{f}^1\frac{\partial\phi}{\partial\alpha^1}+\tilde{f}^2\frac{\partial\phi}{\partial\alpha^2}+\mathbf{f}\cdot\mathbf{n}\frac{\partial\phi}{\partial\mathbf{n}}\right)\mathrm{d}\alpha^1\mathrm{d}\alpha^2\\&=\int_{\mathcal{S}}\phi\left(\frac{\partial\tilde{f}^1}{\partial\alpha^1}+\frac{\partial\tilde{f}^2}{\partial\alpha^2}\right)\mathrm{d}\alpha^1\mathrm{d}\alpha^2-\int_{\mathcal{S}}\mathbf{f}\cdot\mathbf{n}\frac{\partial\phi}{\partial\mathbf{n}}\mathrm{d}\alpha^1\mathrm{d}\alpha^2.\end{aligned}$$

In the last step above, we used the fact that the singular surface is closed. With (3.2), the last term on the right-hand side of (3.9) becomes

$$\begin{aligned}&\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\phi\frac{\partial}{\partial x^i}\left(\rho\frac{\partial u^i}{\partial t}+\rho u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V}\\&=\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\frac{\partial}{\partial x^i}\left(\phi\rho\frac{\partial u^i}{\partial t}+\phi\rho u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V}-\lim_{\epsilon\to 0}\int_{\mathcal{V}_s}\frac{\partial\phi}{\partial x^i}\left(\rho\frac{\partial u^i}{\partial t}+\rho u^j\frac{\partial u^i}{\partial x^j}\right)\mathrm{d}\mathcal{V}\\&=\int_{\mathcal{S}}\phi\rho\left(\left[\frac{\partial u^i}{\partial t}\right]+u^j\left[\frac{\partial u^i}{\partial x^j}\right]\right)n^i\mathrm{d}\mathcal{S}-\lim_{\epsilon\to 0}\frac{\mathrm{d}}{\mathrm{d}t}\int_{\mathcal{V}_s}\left(\frac{\partial\phi}{\partial x^i}\rho u^i\right)\mathrm{d}\mathcal{V}=0.\end{aligned}$$

Plugging the equalities for the three terms back into (3.9) and applying (3.4) gives

$$\int_{\mathcal{S}}\phi\left(\left[\frac{\partial p}{\partial\mathbf{n}}\right]\mathcal{J}-\frac{\partial\tilde{f}^1}{\partial\alpha^1}-\frac{\partial\tilde{f}^2}{\partial\alpha^2}\right)\mathrm{d}\alpha^1\mathrm{d}\alpha^2=0.$$

Because ϕ is arbitrary, we have

$$\left[\frac{\partial p}{\partial\mathbf{n}}\right]\mathcal{J}-\frac{\partial\tilde{f}^1}{\partial\alpha^1}-\frac{\partial\tilde{f}^2}{\partial\alpha^2}=0,$$

which ends the proof. ∎

We have by now derived all the principal jump conditions. They are

$$[u^i] = 0, \tag{3.10}$$

$$[p] = \frac{f^k n_k}{\mathcal{J}}, \tag{3.11}$$

$$\left[\frac{\partial u^i}{\partial \mathbf{n}}\right] = \frac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}}, \tag{3.12}$$

$$\left[\frac{\partial p}{\partial \mathbf{n}}\right] = \frac{1}{\mathcal{J}}\left(\frac{\partial \tilde{f}^1}{\partial \alpha^1} + \frac{\partial \tilde{f}^2}{\partial \alpha^2}\right). \tag{3.13}$$

4. Finite differencing in the IIM. The fundamental idea of the IIM in a flow simulation is to incorporate jump conditions in finite differencing at discontinuities caused by the singular force. A finite difference scheme has its usual form if its stencil does not cross the singular surface. If its stencil crosses the singular surface, it contains additional terms. The additional terms are composed of the jump conditions and are referred to the jump contribution to the finite difference scheme hereafter. To determine the form of the jump contribution in a finite difference scheme, we follow generalized Taylor expansion for a piecewise smooth function, which is given below as a lemma.

LEMMA 4.1 (generalized Taylor expansion). *Assume function $g(z)$ has discontinuity points of the first kind at $z_1, z_2, \ldots, z_m$ in $(z_0, z_{m+1}), z_0 < z_1 < z_2 < \cdots < z_m < z_{m+1}$, and $g(z) \in \mathcal{C}^\infty(z_0, z_1) \cup (z_1, z_2) \cup \cdots \cup (z_m, z_{m+1})$. $g(z)$ can be either continuous or discontinuous at z_0 and z_{m+1}. Let $[g^{(n)}(z_l)] = g^{(n)}(z_l^+) - g^{(n)}(z_l^-)(n = 1, 2, \ldots; l = 1, 2, \ldots, m)$. Then*

$$g(z_{m+1}^-) = \sum_{n=0}^{\infty} \frac{g^{(n)}(z_0^+)}{n!}(z_{m+1} - z_0)^n + \sum_{l=1}^{m}\sum_{n=0}^{\infty} \frac{[g^{(n)}(z_l)]}{n!}(z_{m+1} - z_l)^n. \tag{4.1}$$

Proof. Taylor expansion for $g^{(n)}(z_l^-)$ about z_{l-1}^+ yields

$$g^{(n)}(z_l^-) = \sum_{\beta=0}^{\infty} \frac{g^{(n+\beta)}(z_{l-1}^+)}{\beta!}(z_l - z_{l-1})^\beta.$$

With the use of the binomial theorem, we thus find that

$$\begin{aligned}
\sum_{n=0}^{\infty} \frac{g^{(n)}(z_l^-)}{n!}(z_{m+1} - z_l)^n &= \sum_{n=0}^{\infty}\sum_{\beta=0}^{\infty} \frac{g^{(n+\beta)}(z_{l-1}^+)(z_l - z_{l-1})^\beta}{n!\beta!}(z_{m+1} - z_l)^n \\
&= \sum_{\gamma=0}^{\infty}\sum_{\beta=0}^{\gamma} g^{(\gamma)}(z_{l-1}^+)\frac{(z_l - z_{l-1})^\beta (z_{m+1} - z_l)^{(\gamma-\beta)}}{\beta!(\gamma-\beta)!} \\
&= \sum_{\gamma=0}^{\infty} \frac{g^{(\gamma)}(z_{l-1}^+)}{\gamma!}(z_{m+1} - z_{l-1})^\gamma.
\end{aligned}$$

With $g^{(n)}(z_{l-1}^+) = g^{(n)}(z_{l-1}^-) + [g^{(n)}(z_{l-1})]$, thus

$$\begin{aligned}
\sum_{n=0}^{\infty} \frac{g^{(n)}(z_l^-)}{n!}(z_{m+1} - z_l)^n &= \sum_{n=0}^{\infty} \frac{g^{(n)}(z_{l-1}^-)}{n!}(z_{m+1} - z_{l-1})^n \\
&\quad + \sum_{n=0}^{\infty} \frac{[g^{(n)}(z_{l-1})]}{n!}(z_{m+1} - z_{l-1})^n.
\end{aligned} \tag{4.2}$$

With $g^{(n)}(z_m^+) = g^{(n)}(z_m^-) + [g^{(n)}(z_m)]$, Taylor expansion for $g(z_{m+1}^-)$ at z_m^+ yields

$$\begin{aligned} g(z_{m+1}^-) &= \sum_{n=0}^{\infty} \frac{g^{(n)}(z_m^+)}{n!}(z_{m+1} - z_m)^n \\ &= \sum_{n=0}^{\infty} \frac{g^{(n)}(z_m^-)}{n!}(z_{m+1} - z_m)^n + \sum_{n=0}^{\infty} \frac{[g^{(n)}(z_m)]}{n!}(z_{m+1} - z_m)^n. \end{aligned}$$

Using recursion (4.2) repeatedly above gives the desired result. □

COROLLARY 4.2. *The Taylor expansion for $g(z_0^+)$ about z_{m+1}^- is*

$$g(z_0^+) = \sum_{n=0}^{\infty} \frac{g^{(n)}(z_{m+1}^-)}{n!}(z_0 - z_{m+1})^n + \sum_{l=1}^{m}\sum_{n=0}^{\infty} \frac{-[g^{(n)}(z_l)]}{n!}(z_0 - z_l)^n. \tag{4.3}$$

Jump conditions at the singular surface enter a finite difference scheme in the neighborhood of the singular surface. The form of the finite difference scheme can be found by applying Lemma 4.1 and Corollary 4.2. Here, we construct central finite difference schemes for first-order and second-order derivatives in the situation where there is a discontinuity point ξ between stencil points x_{i-1} and x_i and a discontinuity point η between stencil points x_i and x_{i+1}. If there are more discontinuity points in the stencil, they can easily be included in a similar manner.

LEMMA 4.3 (generalized central finite differences). *Let $x_{i+1} - x_i = x_i - x_{i-1} = h > 0$ and $x_{i-1} < \xi < x_i \le \eta < x_{i+1}$. Suppose that $u(x)$ is infinitely smooth except at discontinuity points of the first kind ξ and η. Further, $u(x)$ can be either continuous or discontinuous at x_{i+1} and x_{i-1}. Then*

$$\begin{aligned} \frac{\mathrm{d}u(x_i^-)}{\mathrm{d}x} &= \frac{u(x_{i+1}^-) - u(x_{i-1}^+)}{2h} \\ &\quad + \frac{1}{2h}\left(\sum_{n=0}^{2} \frac{-[u^{(n)}(\xi)]}{n!}(x_{i-1}-\xi)^n - \sum_{n=0}^{2} \frac{[u^{(n)}(\eta)]}{n!}(x_{i+1} - \eta)^n\right) + O(h^2), \end{aligned} \tag{4.4}$$

$$\begin{aligned} \frac{\mathrm{d}^2u(x_i^-)}{\mathrm{d}x^2} &= \frac{u(x_{i+1}^-) - 2u(x_i) + u(x_{i-1}^+)}{h^2} \\ &\quad - \frac{1}{h^2}\left(\sum_{n=0}^{3} \frac{-[u^{(n)}(\xi)]}{n!}(x_{i-1}-\xi)^n + \sum_{n=0}^{3} \frac{[u^{(n)}(\eta)]}{n!}(x_{i+1} - \eta)^n\right) + O(h^2). \end{aligned} \tag{4.5}$$

Other finite difference schemes with different orders can also be constructed based on Lemma 4.1 and Corollary 4.2, but the number of jump conditions sets an upper limit on the order of accuracy, as stated in the following proposition.

PROPOSITION 4.4. *The highest order of a finite difference scheme for $u^{(n)}(x)$ with a stencil containing a discontinuity point ζ is $m - n + 1$, where m is the maximum number for known jump conditions $[u^{(l)}(\zeta)]$ $(l = 0, 1, 2, \ldots, m)$.*

Navier–Stokes equations (2.1), (2.2) and pressure Poisson equation (2.3) have first- and second-order spatial derivatives. According to Proposition 4.4, to discretize the first-order spatial derivatives with second-order accuracy or the second-order spatial derivatives with first-order accuracy near the singular surface, the jump conditions of the velocity, the pressure, and their first- and second-order spatial derivatives are needed. To discretize the first-order spatial derivatives with third-order accuracy or the second-order spatial derivatives with second-order accuracy near the singular

surface, the jump conditions of their third-order derivatives are needed as well. All these spatial jump conditions are derived in section 5.

If the singular surface is moving, there will be jumps in the temporal derivatives of the velocity at a grid point whenever the surface crosses that grid point. Suppose that the singular surface passes the grid point at time $t_1, t_2, \ldots, t_m$ between time t_0 and t_{m+1}; then

$$u^i(t_{m+1}) = \sum_{n=0}^{\infty} \frac{\partial^n u^i(t_0)}{\partial t^n} \frac{(t_{m+1}-t_0)^n}{n!} + \sum_{l=1}^{m} \sum_{n=0}^{\infty} \left[\!\!\left[\frac{\partial^n u^i(t_l)}{\partial t} \right]\!\!\right] \frac{(t_{m+1}-t_l)^n}{n!}, \tag{4.6}$$

where $[\![\cdot]\!]$ denotes a jump at time t as $[\![\cdot]\!] := (\cdot)_{t^+} - (\cdot)_{t^-}$. Equation (4.6) follows directly from Lemma (4.1). Thus, to achieve first-order accuracy when discretizing $\frac{\partial u^i}{\partial t}$ in (2.1), we need $[\![u^i]\!]$ and $[\![\frac{\partial u^i}{\partial t}]\!]$; to achieve second-order accuracy, we also need $[\![\frac{\partial^2 u^i}{\partial t^2}]\!]$. All these temporal jump conditions are derived in section 6.1.

It should be noted that the spatial convergence rate of a simulation even in terms of the infinity norm can be of the same order as the numerical scheme away from a singular surface, even though the discretization of some derivatives in the Navier–Stokes equations and the pressure Poisson equation is of lower order accuracy near the singular surface. Examples can be found in Li and Lai [20] and Xu and Wang [36], where, with second-order central finite difference discretization of all spatial derivatives away from a singular surface, a simulation had second-order spatial convergence rates in terms of the infinity norm for both the velocity and the pressure, even though the discretization of the Laplace operator was only first-order accurate near the singular surface.

5. Jump conditions of spatial derivatives. The spatial jump conditions of velocity u^i and pressure p are given in (3.10) and (3.11). In this section we present how to derive the spatial jump conditions of all the first-, second-, and third-order velocity and pressure derivatives:

$$\left[\frac{\partial u^i}{\partial x^j}\right], \quad \left[\frac{\partial p}{\partial x^j}\right]; \quad \left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right], \quad \left[\frac{\partial^2 p}{\partial x^j \partial x^k}\right]; \quad \left[\frac{\partial^3 \mathbf{u}}{\partial x^i \partial x^j \partial x^k}\right], \quad \left[\frac{\partial^3 p}{\partial x^i \partial x^j \partial x^k}\right];$$

where $\mathbf{u}$ is used to represent the velocity vector and a superscript (i, j, or k) takes a value $1, 2$, or 3 in three-dimensional simulation.

5.1. Jump conditions $\left[\frac{\partial u^i}{\partial x^j}\right]$ and $\left[\frac{\partial p}{\partial x^j}\right]$. Differentiate (3.10) with respect to α^m $(m = 1, 2)$ to obtain

$$\frac{\partial [u^i]}{\partial \alpha^m} = \left[\frac{\partial u^i}{\partial x^k}\frac{\partial X^k}{\partial \alpha^m}\right] = \frac{\partial X^k}{\partial \alpha^m}\left[\frac{\partial u^i}{\partial x^k}\right] = 0. \tag{5.1}$$

Write (3.12) as

$$\left[\frac{\partial u^i}{\partial \mathbf{n}}\right] = n^j \left[\frac{\partial u^i}{\partial x^j}\right] = \frac{(f^k n_k) n^i - f^i}{\mu \mathcal{J}}. \tag{5.2}$$

Combining (5.1) and (5.2) gives

$$C_1 \begin{bmatrix} \dfrac{\partial u^i}{\partial x^1} \\ \dfrac{\partial u^i}{\partial x^2} \\ \dfrac{\partial u^i}{\partial x^3} \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \\ \dfrac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}} \end{pmatrix}, \tag{5.3}$$

where $\left[\vdots\right]$ denotes a jump for a column vector, and where the nonsingular coefficient matrix C_1 is defined in (3.7). From (5.3), jump conditions of first-order velocity derivatives can be solved.

Similarly, jump conditions of first-order pressure derivatives are found to satisfy

$$C_1 \begin{bmatrix} \dfrac{\partial p}{\partial x^1} \\ \dfrac{\partial p}{\partial x^2} \\ \dfrac{\partial p}{\partial x^3} \end{bmatrix} = \begin{pmatrix} \dfrac{\partial}{\partial \alpha^1}\left(\dfrac{f^i n_i}{\mathcal{J}}\right) \\ \dfrac{\partial}{\partial \alpha^2}\left(\dfrac{f^i n_i}{\mathcal{J}}\right) \\ \dfrac{1}{\mathcal{J}}\left(\dfrac{\partial \tilde{f}^1}{\partial \alpha^1} + \dfrac{\partial \tilde{f}^2}{\partial \alpha^2}\right) \end{pmatrix}, \tag{5.4}$$

from which, jump conditions of first-order pressure derivatives can be solved.

5.2. Jump conditions $\left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right]$ and $\left[\frac{\partial^2 p}{\partial x^j \partial x^k}\right]$. Differentiating equation (5.1) with respect to α^n $(n = 1, 2)$, we have

$$\frac{\partial^2 [u^i]}{\partial \alpha^m \partial \alpha^n} = \left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\frac{\partial X^j}{\partial \alpha^n}\frac{\partial X^k}{\partial \alpha^m} + \frac{\partial u^i}{\partial x^k}\frac{\partial^2 X^k}{\partial \alpha^m \partial \alpha^n}\right] = 0$$

$$\Rightarrow \frac{\partial X^j}{\partial \alpha^n}\frac{\partial X^k}{\partial \alpha^m}\left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right] = -\frac{\partial^2 X^k}{\partial \alpha^m \partial \alpha^n}\left[\frac{\partial u^i}{\partial x^k}\right]. \tag{5.5}$$

Differentiating (3.12) with respect to α^m $(m = 1, 2)$, we have

$$\frac{\partial}{\partial \alpha^m}\left[\frac{\partial u^i}{\partial \mathbf{n}}\right] = \left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\frac{\partial X^k}{\partial \alpha^m}n^j + \frac{\partial u^i}{\partial x^j}\frac{\partial n^j}{\partial \alpha^m}\right] = \frac{\partial}{\partial \alpha^m}\frac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}}$$

$$\Rightarrow n^j \frac{\partial X^k}{\partial \alpha^m}\left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right] = \frac{\partial}{\partial \alpha^m}\frac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}} - \frac{\partial n^j}{\partial \alpha^m}\left[\frac{\partial u^i}{\partial x^j}\right]. \tag{5.6}$$

With (3.2), equation (2.1) yields

$$\left[\rho\left(\frac{\partial u^i}{\partial t} + u^j \frac{\partial u^i}{\partial x^j}\right)\right] = -\left[\frac{\partial p}{\partial x^i}\right] + \mu\left[\frac{\partial^2 u^i}{\partial x^j \partial x^j}\right] = 0. \tag{5.7}$$

Combining (5.5), (5.6), and (5.7) gives

$$
\begin{aligned}
(5.8)\qquad C_2 \begin{bmatrix} \dfrac{\partial^2 u^i}{\partial x^1 \partial x^1} \\ \dfrac{\partial^2 u^i}{\partial x^1 \partial x^2} \\ \dfrac{\partial^2 u^i}{\partial x^1 \partial x^3} \\ \dfrac{\partial^2 u^i}{\partial x^2 \partial x^2} \\ \dfrac{\partial^2 u^i}{\partial x^2 \partial x^3} \\ \dfrac{\partial^2 u^i}{\partial x^3 \partial x^3} \end{bmatrix} &= \begin{pmatrix} 0 \\ 0 \\ 0 \\ \dfrac{\partial}{\partial \alpha^1} \dfrac{(f^k n_k) n^i - f^i}{\mu \mathcal{J}} \\ \dfrac{\partial}{\partial \alpha^2} \dfrac{(f^k n_k) n^i - f^i}{\mu \mathcal{J}} \\ \dfrac{1}{\mu} \left[\dfrac{\partial p}{\partial x^i} \right] \end{pmatrix} \\
&\quad - \begin{pmatrix} \dfrac{\partial^2 X^1}{\partial \alpha^1 \partial \alpha^1} & \dfrac{\partial^2 X^2}{\partial \alpha^1 \partial \alpha^1} & \dfrac{\partial^2 X^3}{\partial \alpha^1 \partial \alpha^1} \\ \dfrac{\partial^2 X^1}{\partial \alpha^2 \partial \alpha^2} & \dfrac{\partial^2 X^2}{\partial \alpha^2 \partial \alpha^2} & \dfrac{\partial^2 X^3}{\partial \alpha^2 \partial \alpha^2} \\ \dfrac{\partial^2 X^1}{\partial \alpha^1 \partial \alpha^2} & \dfrac{\partial^2 X^2}{\partial \alpha^1 \partial \alpha^2} & \dfrac{\partial^2 X^3}{\partial \alpha^1 \partial \alpha^2} \\ \dfrac{\partial n^1}{\partial \alpha^1} & \dfrac{\partial n^2}{\partial \alpha^1} & \dfrac{\partial n^3}{\partial \alpha^1} \\ \dfrac{\partial n^1}{\partial \alpha^2} & \dfrac{\partial n^2}{\partial \alpha^2} & \dfrac{\partial n^3}{\partial \alpha^2} \\ 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} \dfrac{\partial u^i}{\partial x^1} \\ \dfrac{\partial u^i}{\partial x^2} \\ \dfrac{\partial u^i}{\partial x^3} \end{bmatrix},
\end{aligned}
$$

where C_2 is

$$
\begin{aligned}
C_2 &= \begin{pmatrix} \tau_1^1\tau_1^1 & \tau_1^1\tau_1^2+\tau_1^2\tau_1^1 & \tau_1^1\tau_1^3+\tau_1^3\tau_1^1 & \tau_1^2\tau_1^2 & \tau_1^2\tau_1^3+\tau_1^3\tau_1^2 & \tau_1^3\tau_1^3 \\ \tau_2^1\tau_2^1 & \tau_2^1\tau_2^2+\tau_2^2\tau_2^1 & \tau_2^1\tau_2^3+\tau_2^3\tau_2^1 & \tau_2^2\tau_2^2 & \tau_2^2\tau_2^3+\tau_2^3\tau_2^2 & \tau_2^3\tau_2^3 \\ \tau_1^1\tau_2^1 & \tau_1^1\tau_2^2+\tau_1^2\tau_2^1 & \tau_1^1\tau_2^3+\tau_1^3\tau_2^1 & \tau_1^2\tau_2^2 & \tau_1^2\tau_2^3+\tau_1^3\tau_2^2 & \tau_1^3\tau_2^3 \\ \tau_1^1 n^1 & \tau_1^1 n^2+\tau_1^2 n^1 & \tau_1^1 n^3+\tau_1^3 n^1 & \tau_1^2 n^2 & \tau_1^2 n^3+\tau_1^3 n^2 & \tau_1^3 n^3 \\ \tau_2^1 n^1 & \tau_2^1 n^2+\tau_2^2 n^1 & \tau_2^1 n^3+\tau_2^3 n^1 & \tau_2^2 n^2 & \tau_2^2 n^3+\tau_2^3 n^2 & \tau_2^3 n^3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \\
(5.9)\qquad &:= \begin{pmatrix} C_2^{11} & C_2^{12} & C_2^{13} & C_2^{14} & C_2^{15} & C_2^{16} \\ C_2^{21} & C_2^{22} & C_2^{23} & C_2^{24} & C_2^{25} & C_2^{26} \\ C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} \\ C_2^{41} & C_2^{42} & C_2^{43} & C_2^{44} & C_2^{45} & C_2^{46} \\ C_2^{51} & C_2^{52} & C_2^{53} & C_2^{54} & C_2^{55} & C_2^{56} \\ C_2^{61} & C_2^{62} & C_2^{63} & C_2^{64} & C_2^{65} & C_2^{66} \end{pmatrix}.
\end{aligned}
$$

Nonsingularity of coefficient matrix C_2 is proved in Appendix B, and the right-hand side of (5.8) is known from (5.3). Thus, from (5.8), jump conditions of all second-order velocity derivatives can be found.

With the use of Poisson equation (2.3) for the pressure, jump conditions of all second-order pressure derivatives can be found in the similar way to satisfy

$$
\begin{aligned}
(5.10)\quad C_2 \begin{bmatrix} \dfrac{\partial^2 p}{\partial x^1 \partial x^1} \\ \dfrac{\partial^2 p}{\partial x^1 \partial x^2} \\ \dfrac{\partial^2 p}{\partial x^1 \partial x^3} \\ \dfrac{\partial^2 p}{\partial x^2 \partial x^2} \\ \dfrac{\partial^2 p}{\partial x^2 \partial x^3} \\ \dfrac{\partial^2 p}{\partial x^3 \partial x^3} \end{bmatrix} &= \begin{pmatrix} \dfrac{\partial^2}{\partial \alpha^1 \partial \alpha^1}\left(\dfrac{f^i n_i}{\mathcal{J}}\right) \\ \dfrac{\partial^2}{\partial \alpha^2 \partial \alpha^2}\left(\dfrac{f^i n_i}{\mathcal{J}}\right) \\ \dfrac{\partial^2}{\partial \alpha^1 \partial \alpha^2}\left(\dfrac{f^i n_i}{\mathcal{J}}\right) \\ \dfrac{\partial}{\partial \alpha^1}\left(\dfrac{1}{\mathcal{J}}\dfrac{\partial \tilde{f}^1}{\partial \alpha^1} + \dfrac{1}{\mathcal{J}}\dfrac{\partial \tilde{f}^2}{\partial \alpha^2}\right) \\ \dfrac{\partial}{\partial \alpha^2}\left(\dfrac{1}{\mathcal{J}}\dfrac{\partial \tilde{f}^1}{\partial \alpha^1} + \dfrac{1}{\mathcal{J}}\dfrac{\partial \tilde{f}^2}{\partial \alpha^2}\right) \\ -\rho\left[\dfrac{\partial u^i}{\partial x^j}\dfrac{\partial u^j}{\partial x^i}\right] \end{pmatrix} \\
&\quad - \begin{pmatrix} \dfrac{\partial^2 X^1}{\partial \alpha^1 \partial \alpha^1} & \dfrac{\partial^2 X^2}{\partial \alpha^1 \partial \alpha^1} & \dfrac{\partial^2 X^3}{\partial \alpha^1 \partial \alpha^1} \\ \dfrac{\partial^2 X^1}{\partial \alpha^2 \partial \alpha^2} & \dfrac{\partial^2 X^2}{\partial \alpha^2 \partial \alpha^2} & \dfrac{\partial^2 X^3}{\partial \alpha^2 \partial \alpha^2} \\ \dfrac{\partial^2 X^1}{\partial \alpha^1 \partial \alpha^2} & \dfrac{\partial^2 X^2}{\partial \alpha^1 \partial \alpha^2} & \dfrac{\partial^2 X^3}{\partial \alpha^1 \partial \alpha^2} \\ \dfrac{\partial n^1}{\partial \alpha^1} & \dfrac{\partial n^2}{\partial \alpha^1} & \dfrac{\partial n^3}{\partial \alpha^1} \\ \dfrac{\partial n^1}{\partial \alpha^2} & \dfrac{\partial n^2}{\partial \alpha^2} & \dfrac{\partial n^3}{\partial \alpha^2} \\ 0 & 0 & 0 \end{pmatrix} \begin{bmatrix} \dfrac{\partial p}{\partial x^1} \\ \dfrac{\partial p}{\partial x^2} \\ \dfrac{\partial p}{\partial x^3} \end{bmatrix}.
\end{aligned}
$$

Note that

$$
\begin{aligned}
\left[\frac{\partial u^i}{\partial x^j}\frac{\partial u^j}{\partial x^i}\right] &= 2\left(\frac{\partial u^i}{\partial x^j}\right)_{\mathcal{S}^+}\left[\frac{\partial u^j}{\partial x^i}\right] - \left[\frac{\partial u^i}{\partial x^j}\right]\left[\frac{\partial u^j}{\partial x^i}\right] \\
&= 2\left(\frac{\partial u^i}{\partial x^j}\right)_{\mathcal{S}^-}\left[\frac{\partial u^j}{\partial x^i}\right] + \left[\frac{\partial u^i}{\partial x^j}\right]\left[\frac{\partial u^j}{\partial x^i}\right],
\end{aligned}
$$

and $\left(\frac{\partial u^i}{\partial x^j}\right)_{\mathcal{S}^+}$ or $\left(\frac{\partial u^i}{\partial x^j}\right)_{\mathcal{S}^-}$ can be interpolated from the known velocity field which is used to solve pressure Poisson equation (2.3). The interpolation scheme is given in Xu and Wang [36]. From (5.10), jump conditions of all second-order pressure derivatives can be solved.

5.3. Jump conditions $\left[\frac{\partial^3 \mathbf{u}}{\partial x^i \partial x^j \partial x^k}\right]$ and $\left[\frac{\partial^3 p}{\partial x^i \partial x^j \partial x^k}\right]$. Differentiate (5.5) with respect to α^l $(l = 1, 2)$ and obtain

$$\frac{\partial^3[\mathbf{u}]}{\partial\alpha^l\partial\alpha^m\partial\alpha^n}=0\Rightarrow\frac{\partial X^j}{\partial\alpha^n}\frac{\partial X^i}{\partial\alpha^l}\frac{\partial X^k}{\partial\alpha^m}\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right]$$

$$=-\frac{\partial^3X^i}{\partial\alpha^l\partial\alpha^m\partial\alpha^n}\left[\frac{\partial\mathbf{u}}{\partial x^i}\right] \tag{5.11}$$

$$-\left(\frac{\partial^2X^k}{\partial\alpha^l\partial\alpha^m}\frac{\partial X^j}{\partial\alpha^n}+\frac{\partial^2X^k}{\partial\alpha^l\partial\alpha^n}\frac{\partial X^j}{\partial\alpha^m}+\frac{\partial^2X^j}{\partial\alpha^m\partial\alpha^n}\frac{\partial X^k}{\partial\alpha^l}\right)\left[\frac{\partial^2\mathbf{u}}{\partial x^j\partial x^k}\right].$$

As $(l,m,n)=(1,1,1),(2,2,2),(1,1,2)$, or $(2,2,1)$, we have four equations for $\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right]$. Differentiate (5.6) with respect to α^l $(l=1,2)$ and obtain

$$\frac{\partial^2}{\partial\alpha^l\partial\alpha^m}\left[\frac{\partial\mathbf{u}}{\partial\mathbf{n}}\right]=\frac{\partial^2}{\partial\alpha^l\partial\alpha^m}\frac{(f^kn_k)\mathbf{n}-\mathbf{f}}{\mu\mathcal{J}}\Rightarrow n^j\frac{\partial X^i}{\partial\alpha^l}\frac{\partial X^k}{\partial\alpha^m}\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right]$$

$$=\frac{\partial^2}{\partial\alpha^l\partial\alpha^m}\frac{(f^kn_k)\mathbf{n}-\mathbf{f}}{\mu\mathcal{J}}-\frac{\partial^2n^j}{\partial\alpha^l\partial\alpha^m}\left[\frac{\partial\mathbf{u}}{\partial x^j}\right] \tag{5.12}$$

$$-\left(\frac{\partial n^j}{\partial\alpha^m}\frac{\partial X^k}{\partial\alpha^l}+n^j\frac{\partial^2X^k}{\partial\alpha^l\partial\alpha^m}+\frac{\partial n^j}{\partial\alpha^l}\frac{\partial X^k}{\partial\alpha^m}\right)\left[\frac{\partial^2\mathbf{u}}{\partial x^j\partial x^k}\right].$$

As $(l,m)=(1,1),(2,2)$, or $(1,2)$, we have three other equations. Since a superscript (i, j, or k) can take values 1, 2, or 3, the number of unknowns $\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right]$ is ten. Three additional equations need to be found. Differentiating (2.1) with respect to x^k $(k=1,2,3)$ and applying (3.2), we find

$$\left[\frac{\partial^3\mathbf{u}}{\partial x^j\partial x^j\partial x^k}\right]=\frac{1}{\mu}\left[\frac{\partial}{\partial x^k}\nabla p\right]+\frac{\rho}{\mu}\left(\left[\frac{\partial^2\mathbf{u}}{\partial t\partial x^k}\right]+\left[\frac{\partial u^j}{\partial x^k}\frac{\partial\mathbf{u}}{\partial x^j}\right]+u^j\left[\frac{\partial^2\mathbf{u}}{\partial x^j\partial x^k}\right]\right), \tag{5.13}$$

which provides three additional equations if $\left[\frac{\partial^2\mathbf{u}}{\partial t\partial x^k}\right]$ is known. Jump conditions $\left[\frac{\partial^2\mathbf{u}}{\partial t\partial x^k}\right]$ are derived in section 6.1. Thus, combining (5.11), (5.12), and (5.13) gives a 10×10 system for ten unknowns $\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right]$.

To simplify the system, let

$$\mathbf{d}^j_{lm}=\frac{\partial X^i}{\partial\alpha^l}\frac{\partial X^k}{\partial\alpha^m}\left[\frac{\partial^3\mathbf{u}}{\partial x^i\partial x^j\partial x^k}\right], \tag{5.14}$$

and denote the right-hand sides of (5.11) and (5.12) as $\mathbf{r}^n_{lm}$ $(n=1,2)$ and $\mathbf{r}^3_{lm}$, respectively. Then, we can rewrite (5.11) and (5.12) as

$$C_1\begin{pmatrix}\mathbf{d}^1_{lm}\\ \mathbf{d}^2_{lm}\\ \mathbf{d}^3_{lm}\end{pmatrix}=\begin{pmatrix}\mathbf{r}^1_{lm}\\ \mathbf{r}^2_{lm}\\ \mathbf{r}^3_{lm}\end{pmatrix},$$

from which we can solve $\mathbf{d}^j_{lm}$ $(j=1,2,3)$ since C_1 is nonsingular. After $\mathbf{d}^j_{lm}$ is solved,

we can combine (5.14) and (5.13) to obtain a simplified system as

$$C_3 \begin{bmatrix} \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^1 \partial x^1} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^1 \partial x^2} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^1 \partial x^3} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^2 \partial x^2} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^2 \partial x^3} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^1 \partial x^3 \partial x^3} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^2 \partial x^2 \partial x^2} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^2 \partial x^2 \partial x^3} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^2 \partial x^3 \partial x^3} \\ \dfrac{\partial^3 \mathbf{u}}{\partial x^3 \partial x^3 \partial x^3} \end{bmatrix} = \begin{pmatrix} \mathbf{d}_{11}^1 \\ \mathbf{d}_{22}^1 \\ \mathbf{d}_{12}^1 \\ \mathbf{d}_{11}^2 \\ \mathbf{d}_{22}^2 \\ \mathbf{d}_{12}^2 \\ \mathbf{d}_{11}^3 \\ \mathbf{d}_{22}^3 \\ \mathbf{d}_{12}^3 \\ \mathbf{d}_{11}^4 \\ \mathbf{d}_{22}^4 \\ \mathbf{d}_{12}^4 \end{pmatrix}, \tag{5.15}$$

where

$$C_3 = \begin{pmatrix} C_2^{11} & C_2^{12} & C_2^{13} & C_2^{14} & C_2^{15} & C_2^{16} & 0 & 0 & 0 & 0 \\ C_2^{21} & C_2^{22} & C_2^{23} & C_2^{24} & C_2^{25} & C_2^{26} & 0 & 0 & 0 & 0 \\ C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} & 0 & 0 & 0 & 0 \\ 0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} & 0 \\ 0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} & 0 \\ 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} & 0 \\ 0 & 0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} \\ 0 & 0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} \\ 0 & 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \tag{5.16}$$

with matrix elements defined in (5.9) and

$$\mathbf{d}_{11}^4 = \frac{1}{\mu}\left[\frac{\partial}{\partial x^1}\nabla p\right] + \frac{\rho}{\mu}\left(\left[\frac{\partial^2 \mathbf{u}}{\partial t \partial x^1}\right] + \left[\frac{\partial u^j}{\partial x^1}\frac{\partial \mathbf{u}}{\partial x^j}\right] + u^j\left[\frac{\partial^2 \mathbf{u}}{\partial x^j \partial x^1}\right]\right),$$

$$\mathbf{d}_{22}^4 = \frac{1}{\mu}\left[\frac{\partial}{\partial x^2}\nabla p\right] + \frac{\rho}{\mu}\left(\left[\frac{\partial^2 \mathbf{u}}{\partial t \partial x^2}\right] + \left[\frac{\partial u^j}{\partial x^2}\frac{\partial \mathbf{u}}{\partial x^j}\right] + u^j\left[\frac{\partial^2 \mathbf{u}}{\partial x^j \partial x^2}\right]\right),$$

$$\mathbf{d}_{12}^4 = \frac{1}{\mu}\left[\frac{\partial}{\partial x^3}\nabla p\right] + \frac{\rho}{\mu}\left(\left[\frac{\partial^2 \mathbf{u}}{\partial t \partial x^3}\right] + \left[\frac{\partial u^j}{\partial x^3}\frac{\partial \mathbf{u}}{\partial x^j}\right] + u^j\left[\frac{\partial^2 \mathbf{u}}{\partial x^j \partial x^3}\right]\right).$$

To obtain a unique solution to (5.15) we need to show $\operatorname{rank}(C_3) = 10$, which is done in Appendix C.

Similarly, with the use of Poisson equation (2.3), we can obtain an equation system for $\left[\frac{\partial^3 p}{\partial x^i \partial x^j \partial x^k}\right]$ as

$$C_3 \begin{bmatrix} \frac{\partial^3 p}{\partial x^1 \partial x^1 \partial x^1} \\ \frac{\partial^3 p}{\partial x^1 \partial x^1 \partial x^2} \\ \frac{\partial^3 p}{\partial x^1 \partial x^1 \partial x^3} \\ \frac{\partial^3 p}{\partial x^1 \partial x^2 \partial x^2} \\ \frac{\partial^3 p}{\partial x^1 \partial x^2 \partial x^3} \\ \frac{\partial^3 p}{\partial x^1 \partial x^3 \partial x^3} \\ \frac{\partial^3 p}{\partial x^2 \partial x^2 \partial x^2} \\ \frac{\partial^3 p}{\partial x^2 \partial x^2 \partial x^3} \\ \frac{\partial^3 p}{\partial x^2 \partial x^3 \partial x^3} \\ \frac{\partial^3 p}{\partial x^3 \partial x^3 \partial x^3} \end{bmatrix} = \begin{pmatrix} D^1_{11} \\ D^1_{22} \\ D^1_{12} \\ D^2_{11} \\ D^2_{22} \\ D^2_{12} \\ D^3_{11} \\ D^3_{22} \\ D^3_{12} \\ D^4_{11} \\ D^4_{22} \\ D^4_{12} \end{pmatrix}, \tag{5.17}$$

in which

$$D^j_{lm} = \frac{\partial X^i}{\partial \alpha^l} \frac{\partial X^k}{\partial \alpha^m} \left[\frac{\partial^3 p}{\partial x^i \partial x^j \partial x^k}\right], \tag{5.18}$$

with $j = 1, 2, 3$ and $(l, m) = (1, 1), (2, 2)$, or $(1, 2)$, and

$$\begin{aligned} D^4_{11} =& -\frac{\partial \rho}{\partial x^i}\left(\left[\frac{\partial^2 u^i}{\partial t \partial x^1}\right] + u^j \left[\frac{\partial^2 u^i}{\partial x^j \partial x^1}\right] + \left[\frac{\partial u^j}{\partial x^1}\frac{\partial u^i}{\partial x^j}\right]\right) \\ & - \frac{\partial \rho}{\partial x^1}\left[\frac{\partial u^j}{\partial x^i}\frac{\partial u^i}{\partial x^j}\right] - 2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^1}\right], \\ D^4_{22} =& -\frac{\partial \rho}{\partial x^i}\left(\left[\frac{\partial^2 u^i}{\partial t \partial x^2}\right] + u^j \left[\frac{\partial^2 u^i}{\partial x^j \partial x^2}\right] + \left[\frac{\partial u^j}{\partial x^2}\frac{\partial u^i}{\partial x^j}\right]\right) \\ & - \frac{\partial \rho}{\partial x^2}\left[\frac{\partial u^j}{\partial x^i}\frac{\partial u^i}{\partial x^j}\right] - 2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^2}\right], \\ D^4_{12} =& -\frac{\partial \rho}{\partial x^i}\left(\left[\frac{\partial^2 u^i}{\partial t \partial x^3}\right] + u^j \left[\frac{\partial^2 u^i}{\partial x^j \partial x^3}\right] + \left[\frac{\partial u^j}{\partial x^3}\frac{\partial u^i}{\partial x^j}\right]\right) \\ & - \frac{\partial \rho}{\partial x^3}\left[\frac{\partial u^j}{\partial x^i}\frac{\partial u^i}{\partial x^j}\right] - 2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^3}\right]. \end{aligned}$$

If fluid density ρ is a constant, we have

$$D_{11}^4 = -2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^1}\right],$$

$$D_{22}^4 = -2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^2}\right],$$

$$D_{12}^4 = -2\rho\left[\frac{\partial u^j}{\partial x^i}\frac{\partial^2 u^i}{\partial x^j \partial x^3}\right].$$

We solve D_{lm}^j from

$$C_1\begin{pmatrix} D_{lm}^1 \\ D_{lm}^2 \\ D_{lm}^3 \end{pmatrix} = \begin{pmatrix} R_{lm}^1 \\ R_{lm}^2 \\ R_{lm}^3 \end{pmatrix},$$

where R_{lm}^n $(n = 1, 2)$ and R_{lm}^3 are

$$\begin{aligned} R_{lm}^n = {} & \frac{\partial^3}{\partial\alpha^l \partial\alpha^m \partial\alpha^n}\left(\frac{f^i n_i}{\mathcal{J}}\right) - \frac{\partial^3 X^i}{\partial\alpha^l \partial\alpha^m \partial\alpha^n}\left[\frac{\partial p}{\partial x^i}\right] \\ & - \left(\frac{\partial^2 X^k}{\partial\alpha^l \partial\alpha^m}\frac{\partial X^j}{\partial\alpha^n} + \frac{\partial^2 X^k}{\partial\alpha^l \partial\alpha^n}\frac{\partial X^j}{\partial\alpha^m} + \frac{\partial^2 X^j}{\partial\alpha^m \partial\alpha^n}\frac{\partial X^k}{\partial\alpha^l}\right)\left[\frac{\partial^2 p}{\partial x^j \partial x^k}\right], \\ R_{lm}^3 = {} & \frac{\partial^2}{\partial\alpha^l \partial\alpha^m}\left(\frac{1}{\mathcal{J}}\frac{\partial \tilde{f}^1}{\partial\alpha^1} + \frac{1}{\mathcal{J}}\frac{\partial \tilde{f}^2}{\partial\alpha^2}\right) - \frac{\partial^2 n^j}{\partial\alpha^l \partial\alpha^m}\left[\frac{\partial p}{\partial x^j}\right] \\ & - \left(\frac{\partial n^j}{\partial\alpha^m}\frac{\partial X^k}{\partial\alpha^l} + n^j\frac{\partial^2 X^k}{\partial\alpha^l \partial\alpha^m} + \frac{\partial n^j}{\partial\alpha^l}\frac{\partial X^k}{\partial\alpha^m}\right)\left[\frac{\partial^2 p}{\partial x^j \partial x^k}\right]. \end{aligned}$$

6. Jump conditions of temporal derivatives. When singular surface $\mathcal{S}$ is passing a fixed point $\mathbf{x}^*$ in space at time t^*, using $\mathbf{X}^*$ to denote the point on $\mathcal{S}$ which coincides with the point $\mathbf{x}^*$, for flow quantity ψ, we have the following relation between $[\![\psi(\mathbf{X}^*, t^*)]\!] = (\psi)_{t^{*+}} - (\psi)_{t^{*-}}$ and $[\psi(\mathbf{X}^*, t^*)] = (\psi)_{\mathcal{S}^+} - (\psi)_{\mathcal{S}^-}$:

$$[\![\psi]\!] = \begin{cases} [\psi], & \mathbf{u}(\mathbf{x}^*)\cdot\mathbf{n}(\mathbf{X}^*) < 0, \\ -[\psi], & \mathbf{u}(\mathbf{x}^*)\cdot\mathbf{n}(\mathbf{X}^*) > 0. \end{cases} \tag{6.1}$$

If $\mathbf{u}(\mathbf{x}^*)\cdot\mathbf{n}(\mathbf{X}^*) = 0$, we can approximate temporal derivatives at $\mathbf{x}^*$ by those at $\mathbf{X}^*|_{\mathcal{S}^+}$ or $\mathbf{X}^*|_{\mathcal{S}^-}$ with $[\![\cdot]\!] = 0$. Thus, instead of deriving $[\![u^i]\!]$, $\left[\!\left[\frac{\partial u^i}{\partial t}\right]\!\right]$, and $\left[\!\left[\frac{\partial^2 u^i}{\partial t^2}\right]\!\right]$, we turn to deriving $[u^i]$, $\left[\frac{\partial u^i}{\partial t}\right]$, and $\left[\frac{\partial^2 u^i}{\partial t^2}\right]$. We also need to derive $\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right]$, which appears in the right-hand side of (5.13). The spatial jump condition of velocity, $[u^i]$, is already given in (3.10).

Note that we need interpolation to obtain t^* and jump conditions evaluated at t^* in simulation practice. The interpolation procedures are given in Xu and Wang [36].

6.1. Jump conditions $\left[\frac{\partial \boldsymbol{u}^i}{\partial \boldsymbol{t}}\right]$, $\left[\frac{\partial^2 \boldsymbol{u}^i}{\partial \boldsymbol{t}^2}\right]$, and $\left[\frac{\partial^2 \boldsymbol{u}^i}{\partial \boldsymbol{t}\partial \boldsymbol{x}^k}\right]$. From (3.2), we directly have

$$\left[\frac{\partial u^i}{\partial t}\right] = -u^j\left[\frac{\partial u^i}{\partial x^j}\right]. \tag{6.2}$$

Differentiating (6.2) with respect to t, we have

$$\left[\frac{\partial^2 u^i}{\partial t^2}\right] = -2u^j\left[\frac{\partial^2 u^i}{\partial t \partial x^j}\right] - \left[\frac{\partial u^j}{\partial t}\frac{\partial u^i}{\partial x^j}\right] - u^j u^k\left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right] - u^k\left[\frac{\partial u^j}{\partial x^k}\frac{\partial u^i}{\partial x^j}\right], \tag{6.3}$$

where $\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right]$ also appears on the right-hand side of (5.13).

Differentiating (6.2) with respect to α^l $(l = 1, 2)$, we obtain

$$\frac{\partial X^k}{\partial \alpha^l}\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right] = \frac{\partial X^k}{\partial \alpha^l}\left(\left[\frac{\partial u^i}{\partial x^j}\frac{\partial u^j}{\partial x^k}\right] + u^j\left[\frac{\partial^2 u^i}{\partial x^j \partial x^k}\right]\right). \tag{6.4}$$

Differentiating (3.5) with respect to time t, we obtain

$$n^k\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right] = \frac{\mathrm{d}}{\mathrm{d}t}\frac{(f^k n_k)n^i - f^i}{\mu \mathcal{J}} - \frac{\mathrm{d}n^k}{\mathrm{d}t}\left[\frac{\partial u^i}{\partial x^k}\right] - n^k u^j\left[\frac{\partial^2 u^i}{\partial x^k \partial x^j}\right]. \tag{6.5}$$

Denote the right-hand side of (6.4) by r_t^{il} $(l = 1, 2)$ and the right-hand side of (6.5) by r_t^{i3}. Combining (6.4) and (6.5) then gives

$$C_1\begin{bmatrix} \dfrac{\partial^2 u^i}{\partial t \partial x^1} \\ \dfrac{\partial^2 u^i}{\partial t \partial x^2} \\ \dfrac{\partial^2 u^i}{\partial t \partial x^3} \end{bmatrix} = \begin{pmatrix} r_t^{i1} \\ r_t^{i2} \\ r_t^{i3} \end{pmatrix}, \tag{6.6}$$

from which we can solve $\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right]$.

6.2. A numerical example. Here, we implement the IIM in a one-dimensional linear wave equation with a moving singular source. This simple example is designed to illustrate the necessity of including temporal jump conditions in the discretization of temporal derivatives. We also examine the convergence property of the IIM in this example.

The system has the following form:

$$\begin{cases} \dfrac{\partial u}{\partial t} + a\dfrac{\partial u}{\partial x} = c(t)\delta(x - X(t)), \quad X(t) = bt, \quad L \le x \le R, \\ u(x, 0) = g(x), \\ \dfrac{\partial u(L, t)}{\partial x} = 0, \end{cases}$$

where a and b are constants satisfying $a > 0$ and $a \neq b$. The analytical solution to the problem is

$$u(x, t) = g(x - at) + \frac{h(x - bt) + h(at - x)}{2(a - b)} c\left(\frac{x - at}{b - a}\right),$$

where $h(x)$ is the step function.

Integrating the wave equation with respect to x from $X^-(t)$ to $X^+(t)$, the jump condition of u at $x = X(t)$ can be obtained as

$$[u] = \frac{c(t)}{a - b},$$

where $[\cdot] = (\cdot)_{X^+} - (\cdot)_{X^-}$. Following the procedure described in sections 5 and 6.1, we have derived the jump conditions at the singular source $x = X(t)$:

$$[u] = \frac{c(t)}{a-b}, \quad \left[\frac{\partial u}{\partial x}\right] = -\frac{1}{(a-b)^2}\frac{\mathrm{d}c(t)}{\mathrm{d}t}, \quad \left[\frac{\partial^2 u}{\partial t \partial x}\right] = 0, \quad \left[\frac{\partial^2 u}{\partial x^2}\right] = 0,$$
$$[\![u]\!] = -\frac{c(t)}{a-b}, \quad \left[\!\left[\frac{\partial u}{\partial t}\right]\!\right] = -\frac{a}{(a-b)^2}\frac{\mathrm{d}c(t)}{\mathrm{d}t}, \quad \left[\!\left[\frac{\partial^2 u}{\partial x \partial t}\right]\!\right] = 0, \quad \left[\!\left[\frac{\partial^2 u}{\partial t^2}\right]\!\right] = 0,$$

where $[\![\cdot]\!] = (\cdot)_{t^+} - (\cdot)_{t^-}$. At the wave front generated by the singular source, i.e., at $x = X(0) + at$, the solution is continuous but unsmooth, and the corresponding jump conditions are

$$[u] = 0, \quad \left[\frac{\partial u}{\partial x}\right] = \frac{1}{(a-b)^2}\frac{\mathrm{d}c(t)}{\mathrm{d}t}, \quad \left[\frac{\partial^2 u}{\partial t \partial x}\right] = 0, \quad \left[\frac{\partial^2 u}{\partial x^2}\right] = 0,$$
$$[\![u]\!] = 0, \quad \left[\!\left[\frac{\partial u}{\partial t}\right]\!\right] = \frac{a}{(a-b)^2}\frac{\mathrm{d}c(t)}{\mathrm{d}t}, \quad \left[\!\left[\frac{\partial^2 u}{\partial x \partial t}\right]\!\right] = 0, \quad \left[\!\left[\frac{\partial^2 u}{\partial t^2}\right]\!\right] = 0.$$

The Crank–Nicolson method is used to discretize the problem on a uniform mesh:

$$\left(\frac{u_i^{n+1} - u_i^n}{\Delta t} + CT_i^n\right) + \frac{a}{2}\left(\left(\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} + CX_i^n\right) + \left(\frac{u_{i+1}^{n+1} - u_{i-1}^{n+1}}{2\Delta x} + CX_i^{n+1} + CXT_i^n\right)\right) = 0,$$

where n denotes a time layer, i an interior grid point, Δt a time step, Δx a space step, CT the jump contribution in the finite difference of the temporal derivative, CX the jump contribution in the finite difference of the spatial derivative, and CXT the jump contribution to keep the second-order temporal accuracy of the Crank–Nicolson scheme. The jump contributions CT, CX, and CXT are caused by the unsmoothness of the solution. We calculate CT and CX according to generalized Taylor expansion to achieve $\mathcal{O}((\Delta t)^2)$ and $\mathcal{O}((\Delta x)^2)$ accuracy of the finite differences. The calculation of CXT originates from the following generalized Taylor expansion:

$$\left(\frac{\partial u}{\partial x}\right)^{n+1} = \left(\frac{\partial u}{\partial x}\right)^n + \left(\frac{\partial^2 u}{\partial x \partial t}\right)^n \Delta t + \left(\left[\!\left[\frac{\partial u}{\partial x}\right]\!\right] + \left[\!\left[\frac{\partial^2 u}{\partial x \partial t}\right]\!\right]\Delta t\right) + \mathcal{O}(\Delta t)^2.$$

If there is a grid point I satisfying $X^n < x_I \le X^{n+1}$ from time layer n to time layer $(n+1)$, then we let $x_I = X((n+\beta)\Delta t), 0 < \beta \le 1$, and compute CT_i^n and CXT_i^n as

$$\begin{cases} CT_I^n = -\dfrac{1}{\Delta t}\left([\![u]\!] + \left[\!\left[\dfrac{\partial u}{\partial t}\right]\!\right](1-\beta)\Delta t + \dfrac{1}{2}\left[\!\left[\dfrac{\partial^2 u}{\partial t^2}\right]\!\right](1-\beta)^2(\Delta t)^2\right), \\ CT_i^n = 0, \quad i \ne I. \end{cases}$$

and

$$\begin{cases} CXT_I^n = -\left(\left[\!\left[\dfrac{\partial u}{\partial x}\right]\!\right] + \left[\!\left[\dfrac{\partial^2 u}{\partial x \partial t}\right]\!\right]\Delta t\right), \\ CXT_i^n = 0, \quad i \ne I. \end{cases}$$

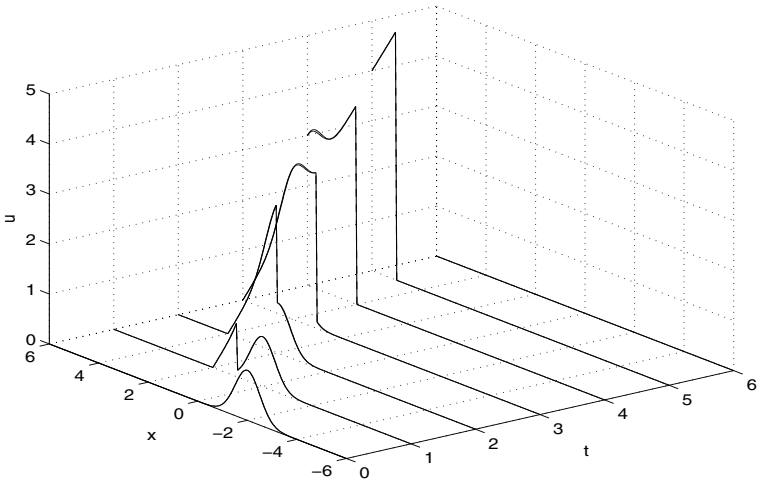


FIG. 6.1. *Comparison between the numerical solution (solid lines) and the analytical solution (dashed lines, hidden by solid lines). For the numerical solution,* $\Delta x = 0.02$ *and* $CFL = \frac{2\Delta t}{\Delta x} = 0.4$.

Otherwise, we have $CT_i^n = 0$ and $CXT_i^n = 0$ for any grid point i. If $x_{I-1} \le X^n < x_I$ at time n, then we can compute CX_i^n as

$$\begin{cases} CX_I^n = -\dfrac{1}{2\Delta x}\left([u] + \left[\dfrac{\partial u}{\partial x}\right](x_{I-1} - X^n) + \dfrac{1}{2}\left[\dfrac{\partial^2 u}{\partial x^2}\right](x_{I-1} - X^n)^2\right), \\ CX_{I-1}^n = -\dfrac{1}{2\Delta x}\left([u] + \left[\dfrac{\partial u}{\partial x}\right](x_I - X^n) + \dfrac{1}{2}\left[\dfrac{\partial^2 u}{\partial x^2}\right](x_I - X^n)^2\right), \\ CT_i^n = 0, \quad i \ne I, I-1. \end{cases}$$

Similarly, we can compute CX_i^{n+1}. At the outlet $x = R$ of the domain, an upwind finite difference scheme is used to approximate the spatial derivative, and the correction CX takes a different form. We also treat the effect of the discontinuity at $x = X(0) + at$ in the same way.

The results presented below are for $a = 2$, $b = 1$, $c(t) = t$, $L = -6$, $R = 6$, and $g(x) = \mathrm{e}^{-2(x+2)^2}$. Figure 6.1 shows the numerical solution and the analytical solution at times $t = 0, 1, 2, \ldots, 6$. After time $t = 6$, the initial wave, the singular source, and the wave generated by the singular source have all exited from the domain. The IIM produces the sharp jumps in the numerical solution and the correct wave-source interaction.

To check the accuracy of the numerical scheme near the singular source, we look at the solutions at time $t = 2$, when the position of the singular source is at $x = 2$. As spatial resolution changes, Figure 6.2(a) shows the change of the infinity norm of the error based on the analytical solution. For each spatial resolution, a very small time step corresponding to $CFL = \frac{2\Delta t}{\Delta x} = 0.002$ is used to ensure that the temporal discretization error is negligible compared with the spatial one. Second-order accuracy in space is indicated in Figure 6.2(a), as expected.

In order to check temporal accuracy, we obtain an accurate reference solution by using a very small time step corresponding to $CFL = 0.002$. We compute the numerical solution using different time steps with the same spatial resolution $\Delta x = 0.05$ as a reference. By subtracting the reference solution from a numerical solution

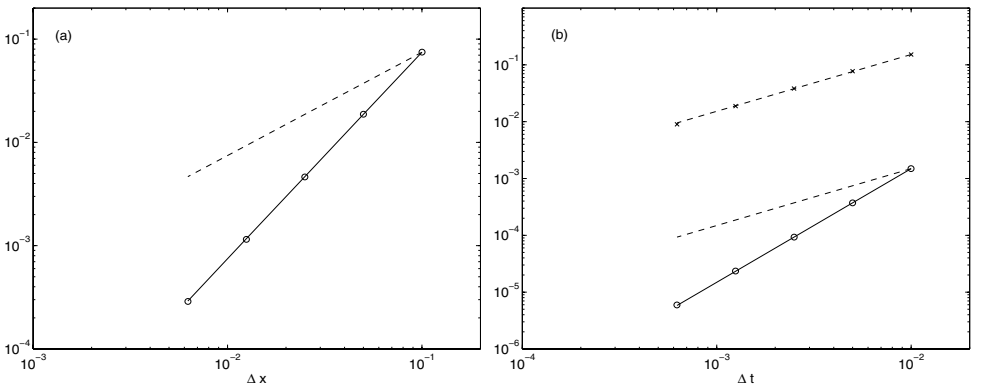


FIG. 6.2. *The infinity norm of the error as a function of* (a) *spatial resolution with* $CFL = 0.002$ *and* (b) *temporal resolution with* $\Delta x = 0.05$. *Open circles: numerical error with the jump contribution* CXT, *x-marks: numerical error without the jump contribution* CXT, *solid lines: second-order accuracy, dashed lines: first-order accuracy.*

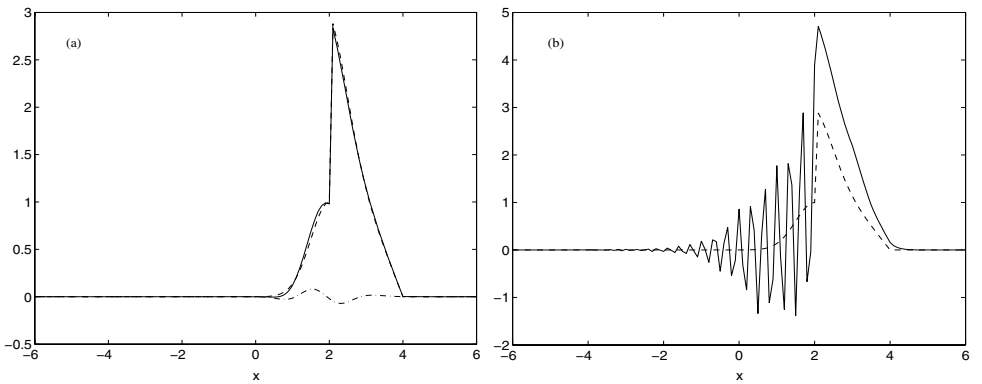


FIG. 6.3. *The effect of the jump contributions in the temporal discretization. Results* (a) *with temporal jump contributions,* (b) *without temporal jump contributions. The results are for* $t = 2$ *with* $\Delta x = 0.1$ *and* $CFL = 0.4$. *Solid lines: numerical results, dashed lines: analytical results, dash-dotted lines: difference between numerical and analytical results.*

calculated using the same space step but a different time step, we cancel out the spatial discretization error and obtain the temporal error. The results are plotted in Figure 6.2(b). Second-order accuracy in time is seen. If the jump contribution CXT is not included, only first-order accuracy in time can be achieved, as seen in Figure 6.2(b).

Figure 6.3(a) compares the numerical and analytical results at time $t = 2$ with $\Delta x = 0.1$ and $CFL = 0.4$. The amplitude of the jump contribution CT has the same order as the jump contribution CX when Δt is of the same order as Δx. If the jump contribution CT is not included, a numerical result can be totally wrong, as also shown in Figure 6.3(b).

As the velocity can be piecewise smooth across a singular surface in a viscous flow, jump contributions in the discretizations of the temporal terms in Navier–Stokes equations can be nonzero. If they are of the same order as the leading ones in spatial discretizations, the inclusion of the jump contributions in the temporal discretizations may be necessary. The moving interface problem simulated by Li and Lai [20] is

an exception. They considered the relaxation of a perturbed two-dimensional balloon immersed in an incompressible viscous fluid. The tangential force along the balloon surface is always zero in their case, so the velocity is smooth in both space and time, though the pressure is not. The jump contributions in temporal discretizations are always zero in this case. Recently, we designed an oscillating Taylor–Couette flow to look at the effect of jump contributions for temporal discretization on temporal convergence rate and temporal accuracy [36]. We found that the effect is very small. Whether this is true in general remains to be investigated.

7. Discussion. Naturally, we ask whether we can improve discretization accuracy in the IIM to arbitrarily high order for three-dimensional incompressible Navier–Stokes flows. We can derive equations for jump conditions by differentiating the principal jump conditions with respect to the Lagrangian parameters and differentiating the governing equations with respect to the Cartesian coordinates, as presented in sections 5 and 6.1. To examine this possibility, we need to know whether the number of derived equations is enough for unknown jump conditions and whether the equation system has a unique solution.

Regarding the number of equations and unknowns, we introduce a lemma.

LEMMA 7.1. *Paint m indistinctive balls with n different colors. Each ball has to be painted with one and only one color. The number of different nonordered color combinations for the m balls is*

$$\mathcal{H}_m^n = \sum_{k_1=0}^{m} \sum_{k_2=0}^{k_1} \sum_{k_3=0}^{k_2} \cdots \sum_{k_{n-2}=0}^{k_{n-3}} \sum_{k_{n-1}=0}^{k_{n-2}} \mathcal{H}_{k_{n-1}}^1, \tag{7.1}$$

with $\mathcal{H}_0^i = 1$ and $\mathcal{H}_i^1 = 1$, where i is a positive integer.

Proof. Paint $k_1\,(k_1 = 0, 1, 2, \ldots, m)$ balls using the same color. There are $n-1$ colors and $m - k_1$ balls left. Therefore, for a particular value of k_1, the number of different outcomes is $\mathcal{H}_{m-k_1}^{n-1}$. Summing $\mathcal{H}_{m-k_1}^{n-1}$ over all possible values of k_1 yields

$$\mathcal{H}_m^n = \sum_{k_1=0}^{m} \mathcal{H}_{m-k_1}^{n-1} = \sum_{k_1=0}^{m} \mathcal{H}_{k_1}^{n-1}. \tag{7.2}$$

Using recursion (7.2) repeatedly, the lemma follows. ∎

If we want the discretization accuracy of the second-order velocity and pressure derivatives to be order of $m-1\,(m \geq 2)$ in Navier–Stokes flow simulation, we need jump conditions of all velocity and pressure derivatives of order m. In a simulation of space dimension n, the number of unknown jump conditions of the velocity derivatives of order m is $\mathcal{H}_m^n$. Differentiating (3.10) with respect to the Lagrangian parameters can provide $\mathcal{H}_m^{n-1}$ equations for the unknowns, differentiating (3.12) can provide $\mathcal{H}_{m-1}^{n-1}$, and differentiating (2.1) with respect to the Cartesian coordinates and then subtracting the resulting equation at $\mathcal{S}^-$ from that at $\mathcal{S}^+$ can give $\mathcal{H}_{m-2}^n$ equations. For closure, we require

$$\mathcal{H}_m^{n-1} + \mathcal{H}_{m-1}^{n-1} + \mathcal{H}_{m-2}^n \geq \mathcal{H}_m^n.$$

The same requirement applies to the unknown jump conditions of the pressure derivatives of order m.

In three-dimensional simulations, $n = 3$, $\mathcal{H}_m^2 = m + 1$, and $\mathcal{H}_m^3 = \frac{(m+1)(m+2)}{2}$ according to (7.1). Hence, we have

$$\mathcal{H}_m^2 + \mathcal{H}_{m-1}^2 + \mathcal{H}_{m-2}^3 = \mathcal{H}_m^3;$$

that is, the number of equations is equal to the number of unknowns, supposing $\mathcal{H}^n_{m-2}$ jump conditions for temporal derivatives in the equations (for example, $\left[\frac{\partial^2 u^i}{\partial t \partial x^k}\right]$ in (5.13) in the case when $m = 3$ and $n = 3$) are known. The equation system may have a unique solution, which may be verified by induction with the use of the method given in Appendix C. However, the $\mathcal{H}^n_{m-2}$ jump conditions for the temporal derivatives are not directly available. We need to find equations for them too. As shown in section 6.1, a unique solution to the required temporal-derivative jump conditions can be obtained when $m = 3$. When $m > 3$, it is not clear whether a unique solution can be found. Thus with the method that we present in this paper for deriving jump conditions, ascending the discretization accuracy to arbitrarily high order is not achievable in the IIM for three-dimensional Navier–Stokes flows.

Appendix A. Proof of Theorem 3.3. First consider the situation in which G^i has finite jumps at $\mathcal{S}$. If necessary, smoothly extend $\mathcal{S}$ so that it cuts $\mathcal{V}$ into two separated regions. Form a banded region $\mathcal{V}_s$ which encloses $\mathcal{S}$ with surface $\mathcal{S}^+_{\epsilon/2}$ at one side of $\mathcal{S}$ and $\mathcal{S}^-_{\epsilon/2}$ at the other, where $\mathcal{S}^+_{\epsilon/2}$ and $\mathcal{S}^-_{\epsilon/2}$ are away from $\mathcal{S}$ by a small distance $\frac{\epsilon}{2}$. Thus,

$$\int_{\mathcal{V}_s} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} = \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \frac{\partial G^i}{\partial x^i} \mathrm{d}r\mathrm{d}\mathcal{S},$$

where $r = (\mathbf{x} - \mathbf{X}_{\mathrm{d}\mathcal{S}}) \cdot \mathbf{n}$ with $\mathbf{X}_{\mathrm{d}\mathcal{S}}$ representing a fixed point at the infinitesimal surface $\mathrm{d}\mathcal{S}$ and $\mathbf{n}$ normal at $\mathbf{X}_{\mathrm{d}\mathcal{S}}$ pointing to side $\mathcal{S}^+_{\epsilon/2}$. Denote the region between $\mathcal{S}^+_{\epsilon/2}$ and $\mathcal{S}$ as $\mathcal{V}_1$ and the region between $\mathcal{S}^-_{\epsilon/2}$ and $\mathcal{S}$ as $\mathcal{V}_2$. Analytically extend G^i in region $\mathcal{V}_1$ to region $\mathcal{V}_2$, and name the extended function G^i_1. Analytically extend G^i in region $\mathcal{V}_2$ to region $\mathcal{V}_1$, and name the extended function G^i_2. Introduce step function $h(r)$ as

$$h(r) = \begin{cases} 0, & r < 0, \\ 1, & r > 0. \end{cases}$$

Then,

$$\begin{aligned}
\int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \frac{\partial G^i}{\partial x^i} \mathrm{d}r\mathrm{d}\mathcal{S} &= \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \frac{\partial (h(r)G^i_1 + h(-r)G^i_2)}{\partial x^i} \mathrm{d}r\mathrm{d}\mathcal{S} \\
&= \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \frac{\partial (G^i_2 + h(r)(G^i_1 - G^i_2))}{\partial x^i} \mathrm{d}r\mathrm{d}\mathcal{S} \\
&= \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \left(h(r) \frac{\partial G^i_1}{\partial x^i} + h(-r) \frac{\partial G^i_2}{\partial x^i} + \frac{\partial h(r)}{\partial x^i} (G^i_1 - G^i_2) \right) \mathrm{d}r\mathrm{d}\mathcal{S} \\
&= \int_{\mathcal{V}_1} \frac{\partial G^i_1}{\partial x^i} \mathrm{d}\mathcal{V} + \int_{\mathcal{V}_2} \frac{\partial G^i_2}{\partial x^i} \mathrm{d}\mathcal{V} + \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \frac{\partial h(r)}{\partial x^i} (G^i_1 - G^i_2) \mathrm{d}r\mathrm{d}\mathcal{S}.
\end{aligned}$$

With $\frac{\partial h(r)}{\partial x^i} = \delta(r) n^i$, where $\delta(r)$ is the Dirac δ function, we thus have

$$\lim_{\epsilon \to 0} \int_{\mathcal{V}_s} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} = \int_{\mathcal{S}} \int_{-\frac{\epsilon}{2}}^{+\frac{\epsilon}{2}} \delta(r) n^i [G^i] \mathrm{d}r\mathrm{d}\mathcal{S} = \int_{\mathcal{S}} n^i [G^i] \mathrm{d}\mathcal{S},$$

where $[\cdot]$ denotes a jump calculated by $(\cdot)_{\mathcal{S}^+} - (\cdot)_{\mathcal{S}^-}$. Here $\mathcal{S}^+$ represents $\mathcal{S}^+_{\epsilon/2}(\epsilon \to 0)$, and $\mathcal{S}^-$ represents $\mathcal{S}^-_{\epsilon/2}(\epsilon \to 0)$.

$\mathcal{V}_s$ divides $\mathcal{V}$ into three regions: $\mathcal{V}_s$ itself, $\mathcal{V}^+$, and $\mathcal{V}^-$, where $\mathcal{V}^+$ and $\mathcal{S}^+$ are at the same side of $\mathcal{S}$ and $\mathcal{V}^-$ and $\mathcal{S}^-$ are at the same side. Hence, as $\epsilon \to 0$,

$$\begin{aligned}\int_{\mathcal{V}} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} &= \int_{\mathcal{V}^+} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} + \int_{\mathcal{V}^-} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} + \int_{\mathcal{V}_s} \frac{\partial G^i}{\partial x^i} \mathrm{d}\mathcal{V} \\ &= \oint_{\mathcal{A}} G^i N_i \mathrm{d}\mathcal{A} + \int_{\mathcal{S}^+} (-n_i G^i) \mathrm{d}\mathcal{S} + \int_{\mathcal{S}^-} n_i G^i \mathrm{d}\mathcal{S} + \int_{\mathcal{S}} n_i [G^i] \mathrm{d}\mathcal{S} = \oint_{\mathcal{A}} G^i N_i \mathrm{d}\mathcal{A},\end{aligned}$$

which completes the proof for the first situation that G^i has finite jumps at $\mathcal{S}$.

The second situation, in which G^i has a singularity of Dirac δ function type, can be proved by treating the Dirac δ function as a weak limit of a hat function and then using the result from the first situation.

Appendix B. Nonsingularity of C_2. By writing the third row of C_2 twice and then rearranging rows, we expand C_2 to C_2^e as follows:

$$C_2^e = \begin{pmatrix} \tau_1^1\tau_1^1 & \tau_1^1\tau_1^2 + \tau_1^2\tau_1^1 & \tau_1^1\tau_1^3 + \tau_1^3\tau_1^1 & \tau_1^2\tau_1^2 & \tau_1^2\tau_1^3 + \tau_1^3\tau_1^2 & \tau_1^3\tau_1^3 \\ \tau_1^1\tau_2^1 & \tau_1^1\tau_2^2 + \tau_1^2\tau_2^1 & \tau_1^1\tau_2^3 + \tau_1^3\tau_2^1 & \tau_1^2\tau_2^2 & \tau_1^2\tau_2^3 + \tau_1^3\tau_2^2 & \tau_1^3\tau_2^3 \\ \tau_1^1 n^1 & \tau_1^1 n^2 + \tau_1^2 n^1 & \tau_1^1 n^3 + \tau_1^3 n^1 & \tau_1^2 n^2 & \tau_1^2 n^3 + \tau_1^3 n^2 & \tau_1^3 n^3 \\ \tau_1^1\tau_2^1 & \tau_1^1\tau_2^2 + \tau_1^2\tau_2^1 & \tau_1^1\tau_2^3 + \tau_1^3\tau_2^1 & \tau_1^2\tau_2^2 & \tau_1^2\tau_2^3 + \tau_1^3\tau_2^2 & \tau_1^3\tau_2^3 \\ \tau_2^1\tau_2^1 & \tau_2^1\tau_2^2 + \tau_2^2\tau_2^1 & \tau_2^1\tau_2^3 + \tau_2^3\tau_2^1 & \tau_2^2\tau_2^2 & \tau_2^2\tau_2^3 + \tau_2^3\tau_2^2 & \tau_2^3\tau_2^3 \\ \tau_2^1 n^1 & \tau_2^1 n^2 + \tau_2^2 n^1 & \tau_2^1 n^3 + \tau_2^3 n^1 & \tau_2^2 n^2 & \tau_2^2 n^3 + \tau_2^3 n^2 & \tau_2^3 n^3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

We can rewrite C_2^e as

$$C_2^e = C_1^e C_2^* = \begin{pmatrix} C_1 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \\ 0 & 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

As C_1 is nonsingular, C_1^e is also nonsingular with inverse

$$(C_1^e)^{-1} = \begin{pmatrix} C_1^{-1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & C_1^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & 1 \end{pmatrix}.$$

Thus, if $\operatorname{rank}(C_2^*) = 6$, then C_2 is nonsingular. We present two methods to show $\operatorname{rank}(C_2^*) = 6$.

B.1. Method I. As $\tau_\mathbf{1} \neq 0$, one of τ_1^1, τ_1^2, or τ_1^3 must be nonzero. Through row and column permutations, C_2^* can be transformed to one of the following two matrices:

$$\begin{pmatrix} \tau_1^2 & \tau_1^3 & \tau_1^1 & 0 & 0 & 0 \\ 0 & \tau_1^2 & 0 & \tau_1^3 & \tau_1^1 & 0 \\ 0 & 0 & \tau_1^2 & 0 & \tau_1^3 & \tau_1^1 \\ \tau_2^2 & \tau_2^3 & \tau_2^1 & 0 & 0 & 0 \\ 0 & \tau_2^2 & 0 & \tau_2^3 & \tau_2^1 & 0 \\ 0 & 0 & \tau_2^2 & 0 & \tau_2^3 & \tau_2^1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}, \qquad \begin{pmatrix} \tau_1^3 & \tau_1^2 & \tau_1^1 & 0 & 0 & 0 \\ 0 & \tau_1^3 & 0 & \tau_1^2 & \tau_1^1 & 0 \\ 0 & 0 & \tau_1^3 & 0 & \tau_1^2 & \tau_1^1 \\ \tau_2^3 & \tau_2^2 & \tau_2^1 & 0 & 0 & 0 \\ 0 & \tau_2^3 & 0 & \tau_2^2 & \tau_2^1 & 0 \\ 0 & 0 & \tau_2^3 & 0 & \tau_2^2 & \tau_2^1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Both have the same structure and similar element ordering as C_2^*. Therefore, we suppose $\tau_1^1 \neq 0$ and need to work on only this case.

Using elementary operations, we can transform C_2^* to C_2^{**} as

$$C_2^* \to C_2^{**} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ 0 & 0 & 0 & -s_3 & -2\tau_1^2\tau_1^3 & -s_2 \\ 0 & 0 & 0 & 0 & e & f \\ 0 & 0 & 0 & 0 & g & h \\ 0 & 0 & 0 & 0 & N^3 & -N^2 \end{pmatrix},$$

with the following definitions:

$$\begin{aligned} s_2 &= \tau_1^1\tau_1^1 + \tau_1^3\tau_1^3, \\ s_3 &= \tau_1^1\tau_1^1 + \tau_1^2\tau_1^2, \\ N^2 &= \tau_1^3\tau_2^1 - \tau_2^3\tau_1^1, \\ N^3 &= \tau_1^1\tau_2^2 - \tau_2^1\tau_1^2, \\ e &= (\tau_2^2\tau_1^3 + \tau_1^2\tau_2^3)s_3 - 2\tau_1^2\tau_1^3 d_3, \\ f &= d_2 s_3 - d_3 s_2, \\ g &= N^2 s_3 + 2\tau_1^2\tau_1^3 N^3, \\ h &= s_2 N^3, \end{aligned}$$

where

$$\begin{aligned} d_2 &= \tau_1^1\tau_2^1 + \tau_1^3\tau_2^3, \\ d_3 &= \tau_1^1\tau_2^1 + \tau_1^2\tau_2^2. \end{aligned}$$

If $N^2 = N^3 = 0$, then as $\tau_1^1 \neq 0$,

$$\boldsymbol{\tau_1} \cdot (\boldsymbol{\tau_1} \times \boldsymbol{\tau_2}) = \tau_1^1 N^1 + \tau_1^2 N^2 + \tau_1^3 N^3 = 0 \Rightarrow N^1 = 0 \Rightarrow \boldsymbol{\tau_1} \times \boldsymbol{\tau_2} = 0,$$

where $N^1 = \tau_1^2\tau_2^3 - \tau_2^2\tau_1^3$. This is impossible. Thus, one of N^2 and N^3 must be nonzero. Suppose $N^3 \neq 0$. By elementary operations, we can transform C_2^{**} to

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ 0 & 0 & 0 & -s_3 & -2\tau_1^2\tau_1^3 & -s_2 \\ 0 & 0 & 0 & 0 & N^3 & -N^2 \\ 0 & 0 & 0 & 0 & 0 & N^3 f + N^2 e \\ 0 & 0 & 0 & 0 & 0 & N^3 h + N^2 g \end{pmatrix}.$$

Since

$$N^3 h + N^2 g = \tau_1^1\tau_1^1 \|\boldsymbol{\tau_1} \times \boldsymbol{\tau_2}\|^2 \neq 0,$$

we can conclude that $\operatorname{rank}(C_2^{**}) \geq 6$. Supposing $N^2 \neq 0$, we can similarly show $\operatorname{rank}(C_2^{**}) \geq 6$. C_2^e is expanded from C_2, and thus $\operatorname{rank}(C_2^e) \leq 6$. Therefore, $\operatorname{rank}(C_2^e) = \operatorname{rank}(C_2^*) = \operatorname{rank}(C_2^{**}) = 6$. The proof is completed. This proof gives a way to solve (5.8).

B.2. Method II. As $\tau_{\mathbf{1}} \times \tau_{\mathbf{2}} \neq 0$, there exist matrices P_r and P_c such that

$$P_r \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 \end{pmatrix} P_c = T,$$

where P_r is a 2×2 elementary matrix, P_c is a 3×3 elementary matrix without permutation operations, and $T = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 \\ e_b^1 & e_b^2 & e_b^3 \end{pmatrix}$ is one of

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Construct elementary matrices E_r and E_c as

$$E_r = \begin{pmatrix} P_r & \mathbf{0} \\ \mathbf{0} & I_{5\times 5} \end{pmatrix}, \quad E_c = \begin{pmatrix} P_c & \mathbf{0} \\ \mathbf{0} & I_{3\times 3} \end{pmatrix},$$

where $I_{n\times n}$ denotes an $n \times n$ unit matrix. With a series of actions by permutations and E_r and E_c, C_2^* can be transformed as follows:

$$\begin{aligned} C_2^* &\to E_r \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ 0 & 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} E_c = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ 0 & 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{pmatrix} \\ &\to E_r \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ e_t^2 & 0 & 0 & e_t^1 & e_t^3 & 0 \\ e_b^2 & 0 & 0 & e_b^1 & e_b^3 & 0 \\ 0 & 0 & \tau_1^2 & 0 & \tau_1^1 & \tau_1^3 \\ 0 & 0 & \tau_2^2 & 0 & \tau_2^1 & \tau_2^3 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} E_b = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ e_t^2 & 0 & 0 & e_t^1 & e_t^3 & 0 \\ e_b^2 & 0 & 0 & e_b^1 & e_b^3 & 0 \\ 0 & 0 & \tau_1^2 & 0 & \tau_1^1 & \tau_1^3 \\ 0 & 0 & \tau_2^2 & 0 & \tau_2^1 & \tau_2^3 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix} \\ &\to E_r \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & e_t^3 & 0 & e_t^1 & e_t^2 & 0 \\ 0 & e_b^3 & 0 & e_b^1 & e_b^2 & 0 \\ e_t^3 & 0 & 0 & e_t^2 & 0 & e_t^1 \\ e_b^3 & 0 & 0 & e_b^2 & 0 & e_b^1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} E_c = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ 0 & e_t^3 & 0 & e_t^1 & e_t^2 & 0 \\ 0 & e_b^3 & 0 & e_b^1 & e_b^2 & 0 \\ e_t^3 & 0 & 0 & e_t^2 & 0 & e_t^1 \\ e_b^3 & 0 & 0 & e_b^2 & 0 & e_b^1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}. \end{aligned}$$

As matrix T takes different choices, the last matrix above correspondingly takes

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

Thus, $\operatorname{rank}(C_2^*) = 6$.

Appendix C. rank(C_3) = 10. Expand C_3 to C_3^e as

$$C_3^e = \begin{pmatrix}
C_2^{11} & C_2^{12} & C_2^{13} & C_2^{14} & C_2^{15} & C_2^{16} & 0 & 0 & 0 & 0 \\
C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} & 0 & 0 & 0 & 0 \\
C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} & 0 & 0 & 0 & 0 \\
C_2^{21} & C_2^{22} & C_2^{23} & C_2^{24} & C_2^{25} & C_2^{26} & 0 & 0 & 0 & 0 \\
0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} & 0 \\
0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} & 0 \\
0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} & 0 \\
0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} & 0 \\
0 & 0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} \\
0 & 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} \\
0 & 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} \\
0 & 0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}.$$

If $\operatorname{rank}(C_3^e) = 10$, then $\operatorname{rank}(C_3) = 10$.

Define a corner block

$$C_3^c := \begin{pmatrix}
C_2^{11} & C_2^{12} & C_2^{13} & C_2^{14} & C_2^{15} & C_2^{16} \\
C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} \\
C_2^{31} & C_2^{32} & C_2^{33} & C_2^{34} & C_2^{35} & C_2^{36} \\
C_2^{21} & C_2^{22} & C_2^{23} & C_2^{24} & C_2^{25} & C_2^{26}
\end{pmatrix},$$

and decompose C_3^c as

$$\begin{aligned}
C_3^c &= \begin{pmatrix}
\tau_1^1\tau_1^1 & \tau_1^1\tau_1^2+\tau_1^2\tau_1^1 & \tau_1^1\tau_1^3+\tau_1^3\tau_1^1 & \tau_1^2\tau_1^2 & \tau_1^2\tau_1^3+\tau_1^3\tau_1^2 & \tau_1^3\tau_1^3 \\
\tau_1^1\tau_2^1 & \tau_1^1\tau_2^2+\tau_1^2\tau_2^1 & \tau_1^1\tau_2^3+\tau_1^3\tau_2^1 & \tau_1^2\tau_2^2 & \tau_1^2\tau_2^3+\tau_1^3\tau_2^2 & \tau_1^3\tau_2^3 \\
\tau_1^1\tau_2^1 & \tau_1^1\tau_2^2+\tau_1^2\tau_2^1 & \tau_1^1\tau_2^3+\tau_1^3\tau_2^1 & \tau_1^2\tau_2^2 & \tau_1^2\tau_2^3+\tau_1^3\tau_2^2 & \tau_1^3\tau_2^3 \\
\tau_2^1\tau_2^1 & \tau_2^1\tau_2^2+\tau_2^2\tau_2^1 & \tau_2^1\tau_2^3+\tau_2^3\tau_2^1 & \tau_2^2\tau_2^2 & \tau_2^2\tau_2^3+\tau_2^3\tau_2^2 & \tau_2^3\tau_2^3
\end{pmatrix} \\
&:= L_1R_1 = \begin{pmatrix}
\tau_1^1 & 0 & \tau_1^2 & 0 & \tau_1^3 & 0 \\
\tau_2^1 & 0 & \tau_2^2 & 0 & \tau_2^3 & 0 \\
0 & \tau_1^1 & 0 & \tau_1^2 & 0 & \tau_1^3 \\
0 & \tau_2^1 & 0 & \tau_2^2 & 0 & \tau_2^3
\end{pmatrix}
\begin{pmatrix}
\tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\
\tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\
0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\
0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \\
0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\
0 & 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3
\end{pmatrix}.
\end{aligned}$$

Through permutations, we can transform C_3^c as

$$C_3^c \to L_1^*R_1^* := \begin{pmatrix}
\tau_1^2 & 0 & \tau_1^3 & 0 & \tau_1^1 & 0 \\
\tau_2^2 & 0 & \tau_2^3 & 0 & \tau_2^1 & 0 \\
0 & \tau_1^2 & 0 & \tau_1^3 & 0 & \tau_1^1 \\
0 & \tau_2^2 & 0 & \tau_2^3 & 0 & \tau_2^1
\end{pmatrix}
\begin{pmatrix}
\tau_1^2 & \tau_1^3 & \tau_1^1 & 0 & 0 & 0 \\
\tau_2^2 & \tau_2^3 & \tau_2^1 & 0 & 0 & 0 \\
0 & \tau_1^2 & 0 & \tau_1^3 & \tau_1^1 & 0 \\
0 & \tau_2^2 & 0 & \tau_2^3 & \tau_2^1 & 0 \\
0 & 0 & \tau_1^2 & 0 & \tau_1^3 & \tau_1^1 \\
0 & 0 & \tau_2^2 & 0 & \tau_2^3 & \tau_2^1
\end{pmatrix}$$

or

$$C_3^c \to L_1^{**} R_1^{**} := \begin{pmatrix} \tau_1^3 & 0 & \tau_1^2 & 0 & \tau_1^1 & 0 \\ \tau_2^3 & 0 & \tau_2^2 & 0 & \tau_2^1 & 0 \\ 0 & \tau_1^3 & 0 & \tau_1^2 & 0 & \tau_1^1 \\ 0 & \tau_2^3 & 0 & \tau_2^2 & 0 & \tau_2^1 \end{pmatrix} \begin{pmatrix} \tau_1^3 & \tau_1^2 & \tau_1^1 & 0 & 0 & 0 \\ \tau_2^3 & \tau_2^2 & \tau_2^1 & 0 & 0 & 0 \\ 0 & \tau_1^3 & 0 & \tau_1^2 & \tau_1^1 & 0 \\ 0 & \tau_2^3 & 0 & \tau_2^2 & \tau_2^1 & 0 \\ 0 & 0 & \tau_1^3 & 0 & \tau_1^2 & \tau_1^1 \\ 0 & 0 & \tau_2^3 & 0 & \tau_2^2 & \tau_2^1 \end{pmatrix}.$$

Since L_1, L_1^*, and L_1^{**} have the same structure and similar element ordering, and so do R_1, R_1^*, and R_1^{**}, we suppose $\tau_1^1 \neq 0$ and need to work on only this case.

As $\tau_1^1 \neq 0$ and $\boldsymbol{\tau_1 \times \tau_2} \neq 0$, there exists matrix P such that

$$\begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 \end{pmatrix} P = T,$$

where P is a 3×3 elementary matrix without permutation operations and $T = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 \\ e_b^1 & e_b^2 & e_b^3 \end{pmatrix}$ is one of

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Construct elementary matrix E as

$$E = \begin{pmatrix} P & \mathbf{0} \\ \mathbf{0} & I_{3\times 3} \end{pmatrix}.$$

With a series of actions by permutations and E, C_3^c can be transformed as follows:

$$\begin{aligned} C_3^c = L_1 R_1 &\to L_1 R_1 E = L_1 \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \\ 0 & 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 \\ 0 & 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 \end{pmatrix} \\ &\to \begin{pmatrix} \tau_1^2 & 0 & \tau_1^1 & 0 & \tau_1^3 & 0 \\ \tau_2^2 & 0 & \tau_2^1 & 0 & \tau_2^3 & 0 \\ 0 & \tau_1^2 & 0 & \tau_1^1 & 0 & \tau_1^3 \\ 0 & \tau_2^2 & 0 & \tau_2^1 & 0 & \tau_2^3 \end{pmatrix} \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ e_t^2 & 0 & 0 & e_t^1 & e_t^3 & 0 \\ e_b^2 & 0 & 0 & e_b^1 & e_b^3 & 0 \\ 0 & 0 & \tau_1^2 & 0 & \tau_1^1 & \tau_1^3 \\ 0 & 0 & \tau_2^2 & 0 & \tau_2^1 & \tau_2^3 \end{pmatrix} \\ &:= L_2 R_2 \to L_2 R_2 E = L_2 \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ e_t^2 & 0 & 0 & e_t^1 & e_t^3 & 0 \\ e_b^2 & 0 & 0 & e_b^1 & e_b^3 & 0 \\ 0 & 0 & \tau_1^2 & 0 & \tau_1^1 & \tau_1^3 \\ 0 & 0 & \tau_2^2 & 0 & \tau_2^1 & \tau_2^3 \end{pmatrix} \\ &\to \begin{pmatrix} \tau_1^3 & 0 & \tau_1^1 & 0 & \tau_1^2 & 0 \\ \tau_2^3 & 0 & \tau_2^1 & 0 & \tau_2^2 & 0 \\ 0 & \tau_1^3 & 0 & \tau_1^1 & 0 & \tau_1^2 \\ 0 & \tau_2^3 & 0 & \tau_2^1 & 0 & \tau_2^2 \end{pmatrix} \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ e_t^3 & 0 & 0 & e_t^2 & 0 & e_t^1 \\ e_b^3 & 0 & 0 & e_b^2 & 0 & e_b^1 \\ 0 & e_t^3 & 0 & e_t^1 & e_t^2 & 0 \\ 0 & e_b^3 & 0 & e_b^1 & e_b^2 & 0 \end{pmatrix} \end{aligned}$$

$$:= L_3R_3 \to L_3R_3E = L_3 \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 & 0 & 0 & 0 \\ e_b^1 & e_b^2 & e_b^3 & 0 & 0 & 0 \\ e_t^3 & 0 & 0 & e_t^2 & 0 & e_t^1 \\ e_b^3 & 0 & 0 & e_b^2 & 0 & e_b^1 \\ 0 & e_t^3 & 0 & e_t^1 & e_t^2 & 0 \\ 0 & e_b^3 & 0 & e_b^1 & e_b^2 & 0 \end{pmatrix} := L_4R_4.$$

When

$$T = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 \\ e_b^1 & e_b^2 & e_b^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix},$$

we continue to transform C_3^c as

$$\begin{aligned} L_4R_4 &= \begin{pmatrix} \tau_1^3 & 0 & 0 & \tau_1^2 & 0 & \tau_1^1 \\ \tau_2^3 & 0 & 0 & \tau_2^2 & 0 & \tau_2^1 \\ 0 & \tau_1^3 & 0 & \tau_1^1 & \tau_1^2 & 0 \\ 0 & \tau_2^3 & 0 & \tau_2^1 & \tau_2^2 & 0 \end{pmatrix} \to \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \end{pmatrix} E \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & \tau_1^1 & 0 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & \tau_2^1 & 0 & \tau_2^2 & \tau_2^3 & 0 \end{pmatrix} \to \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} E \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

When

$$T = \begin{pmatrix} e_t^1 & e_t^2 & e_t^3 \\ e_b^1 & e_b^2 & e_b^3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

we continue to transform C_3^c as

$$\begin{aligned} L_4R_4 &= \begin{pmatrix} \tau_1^3 & 0 & 0 & \tau_1^2 & 0 & \tau_1^1 \\ \tau_2^3 & 0 & 0 & \tau_2^2 & 0 & \tau_2^1 \\ \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & 0 & \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & 0 & \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 \end{pmatrix} E \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 \\ 0 & 0 & \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 \end{pmatrix} \to \begin{pmatrix} \tau_1^1 & \tau_1^2 & \tau_1^3 & 0 & 0 & 0 \\ \tau_2^1 & \tau_2^2 & \tau_2^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} E \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \end{aligned}$$

Combining all the row-operating elementary matrices as a 4×4 elementary matrix E_r^c and all the column-operating elementary matrices as a 6×6 elementary matrix E_c^c, we can write transformations to C_3^c as

$$C_3^c \to E_r^c C_3^c E_c^c = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \gamma^1 & \gamma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where (γ^1, γ^2) equals $(1, 0)$ or $(0, 1)$.

Construct elementary matrices Q_r and Q_c as

$$Q_r = \begin{pmatrix} E_r^c & \mathbf{0} \\ \mathbf{0} & I_{11\times 11} \end{pmatrix}, \quad Q_c = \begin{pmatrix} E_c^c & \mathbf{0} \\ \mathbf{0} & I_{4\times 4} \end{pmatrix}.$$

With a series of actions by permutations, Q_r and Q_c, C_3^e can be transformed as follows:

$$C_3^e \to Q_r C_3^e Q_c = \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \gamma^1 & \gamma^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} & 0 \\
0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} & 0 \\
0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} & 0 \\
0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} & 0 \\
0 & 0 & C_2^{11} & 0 & C_2^{12} & C_2^{13} & 0 & C_2^{14} & C_2^{15} & C_2^{16} \\
0 & 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} \\
0 & 0 & C_2^{31} & 0 & C_2^{32} & C_2^{33} & 0 & C_2^{34} & C_2^{35} & C_2^{36} \\
0 & 0 & C_2^{21} & 0 & C_2^{22} & C_2^{23} & 0 & C_2^{24} & C_2^{25} & C_2^{26} \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1
\end{pmatrix}$$

$$\to \cdots \to \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \gamma^1 & \gamma^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & \gamma^2 & 0 & 0 & 0 & 0 & 0 & \gamma^1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\gamma^2 & 0 & 0 & 0 & 0 & 0 & 0 & \gamma^1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}.$$

When (γ^1, γ^2) equals $(1, 0)$ or $(0, 1)$, it can be seen that the rank of the last matrix above is 10. Thus $\operatorname{rank}(C_3) = \operatorname{rank}(C_3^e) = 10$.

REFERENCES

[1] K. M. Arthurs, L. C. Moore, C. S. Peskin, E. B. Pitman, and H. E. Layton, *Modeling arteriolar flow and mass transport using the immersed boundary method*, J. Comput. Phys., 147 (1998), pp. 402–440.

[2] R. P. Beyer and R. J. Leveque, *Analysis of a one-dimensional model for the immersed boundary method*, SIAM J. Numer. Anal., 29 (1992), pp. 332–364.

[3] J. M. Birch and M. H. Dickinson, *Spanwise flow and the attachment of the leading-edge vortex on insect wings*, Nature, 412 (2001), pp. 729–733.

[4] D. M. Bushnell, J. N. Hefner, and R. L. Ash, *Effects of compliant wall motion on turbulent boundary layers*, Phys. Fluids A, 20 (1977), pp. s31–48.

[5] D. Calhoun, *A Cartesian grid method for solving the two-dimensional streamfunction-vorticity equations in irregular regions*, J. Comput. Phys., 176 (2002), pp. 231–275.
[6] P. W. Carpenter, C. Davies, and A. D. Lucey, *Hydrodynamics and compliant walls: Does the dolphin have a secret?*, Current Science, 79 (2000), pp. 758–765.
[7] R. Cortez and M. Minion, *The blob projection method for immersed boundary problems*, J. Comput. Phys., 161 (2000), pp. 428–453.
[8] M. H. Dickinson, F. O. Lehmann, and S. P. Sane, *Wing rotation and the aerodynamic basis of insect flight*, Science, 284 (1999), pp. 1954–1960.
[9] C. P. Ellington, C. van den Berg, A. P. Willmott, and A. L. R. Thomas, *Leading-edge vortices in insect flight*, Nature, 384 (1996), pp. 626–630.
[10] L. J. Fauci and C. S. Peskin, *A computational model of aquatic animal locomotion*, J. Comput. Phys., 77 (1988), pp. 85–108.
[11] A. L. Fogelson and C. S. Peskin, *A fast numerical method for solving the three-dimensional Stokes' equations in the presence of suspended particles*, J. Comput. Phys., 79 (1988), pp. 50–69.
[12] A. L. Fogelson and J. P. Keener, *Immersed interface methods for Neumann and related problems in two and three dimensions*, SIAM J. Sci. Comput., 22 (2000), pp. 1630–1654.
[13] S. N. Fry, R. Sayaman, and M. H. Dickinson, *The aerodynamics of free-flight maneuvers in Drosophila*, Science, 300 (2003), pp. 495–498.
[14] M. Gad-El-Hak, *Compliant coatings: The simpler alternative*, Experimental Thermal and Fluid Science, 16 (1998), pp. 141–156.
[15] M. O. Kramer, *The dolphin's secret*, Journal of American Society for Naval Engineering, 73 (1961), pp. 103–107.
[16] M.-C. Lai and C. S. Peskin, *An immersed boundary method with formal second-order accuracy and reduced numerical viscosity*, J. Comput. Phys., 160 (2000), pp. 705–719.
[17] M.-C. Lai and Z. Li, *A remark on jump conditions for the three-dimensional Navier–Stokes equations involving an immersed moving membrane*, Appl. Math. Lett., 14 (2001), pp. 149–154.
[18] R. J. LeVeque and Z. Li, *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 31 (1994), pp. 1019–1044.
[19] R. J. LeVeque and Z. Li, *Immersed interface methods for Stokes flow with elastic boundaries or surface tension*, SIAM J. Sci. Comput., 18 (1997), pp. 709–735.
[20] Z. Li and M.-C. Lai, *The immersed interface method for the Navier–Stokes equations with singular forces*, J. Comput. Phys., 171 (2001), pp. 822–842.
[21] A. Mayo, *The fast solution of Poisson's and the biharmonic equations on irregular regions*, SIAM J. Numer. Anal., 21 (1984), pp. 285–299.
[22] A. McKenney, L. Greengard, and A. Mayo, *A fast Poisson solver for complex geometries*, J. Comput. Phys., 118 (1995), pp. 348–355.
[23] C. S. Peskin, *Flow patterns around heart valves: A numerical method*, J. Comput. Phys., 10 (1972), pp. 252–271.
[24] C. S. Peskin, *Numerical analysis of blood flow in the heart*, J. Comput. Phys., 25 (1977), pp. 220–252.
[25] C. S. Peskin and D. M. McQueen, *A three-dimensional computational method for blood flow in the heart* I. *Immersed elastic fibers in a viscous incompressible fluid*, J. Comput. Phys., 81 (1989), pp. 372–405.
[26] C. S. Peskin and B. F. Printz, *Improved volume conservation in the computation of flows with immersed elastic boundaries*, J. Comput. Phys., 105 (1993), pp. 33–46.
[27] A. M. Roma, C. S. Peskin, and M. J. Berger, *An adaptive version of the immersed boundary method*, J. Comput. Phys., 153 (1999), pp. 509–534.
[28] D. Russell and Z. J. Wang, *A Cartesian grid method for modeling multiple moving objects in 2D incompressible viscous flow*, J. Comput. Phys., 191 (2003), pp. 177–205.
[29] C. Tu and C. S. Peskin, *Stability and instability in the computation of flows with moving immersed boundaries: A comparison of three methods*, SIAM J. Statist. Comput., 13 (1992), pp. 1361–1376.
[30] H. S. Udaykumar, R. Mittal, P. Rampunggoon, and A. Khanna, *A sharp interface Cartesian grid method for simulating flows with complex moving boundaries*, J. Comput. Phys., 174 (2001), pp. 345–380.
[31] Z. J. Wang, *Vortex shedding and frequency selection in flapping flight*, J. Fluid Mech., 410 (2000), pp. 323–341.
[32] Z. J. Wang, *Two dimensional mechanism for insect hovering*, Phys. Rev. Lett., 85 (2000), pp. 2216–2219.

[33] A. Wiegmann and K. P. Bube, *The immersed interface method for nonlinear differential equations with discontinuous coefficients and singular sources*, SIAM J. Numer. Anal., 35 (1998), pp. 177–200.

[34] A. Wiegmann and K. P. Bube, *The explicit-jump immersed interface method: Finite difference methods for PDEs with piecewise smooth solutions*, SIAM J. Numer. Anal., 37 (2000), pp. 827–862.

[35] S. Xu, D. Rempfer, and J. Lumley, *Turbulence over a compliant surface: Numerical simulation and analysis*, J. Fluid Mech., 478 (2003), pp. 11–34.

[36] S. Xu and Z. J. Wang, *An immersed interface method for simulating the interaction of a fluid with moving boundaries*, J. Comput. Phys., to appear.

[37] L. Zhu and C. S. Peskin, *Simulation of a flapping flexible filament in a flowing soap film by the immersed boundary method*, J. Comput. Phys., 179 (2002), pp. 452–468.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 1981–2009
© 2006 Society for Industrial and Applied Mathematics

DETERMINISTIC SIMULATION OF THE BOLTZMANN–POISSON SYSTEM IN GAAS-BASED SEMICONDUCTORS*

MARÍA J. CÁCERES†, JOSÉ A. CARRILLO‡, AND ARMANDO MAJORANA§

Abstract. In this work we present a deterministic numerical scheme solving the one-dimensional Boltzmann–Poisson system for semiconductor materials with a complicated band structure such as GaAs. The model is based on a system of two kinetic equations for the electron population in each of the relevant valleys coupled through the Poisson equation to consider the self-consistent potential. Interaction of electrons with the crystal is taken into account by treating the most important inter- and intravalley scattering mechanisms in GaAs. The numerical scheme follows the main ideas in [J. A. Carrillo, I. M. Gamba, A. Majorana, and C.-W. Shu, *J. Comput. Phys.*, 184 (2003), pp. 498–525] already developed for Si-based devices. The newest numerical features are the intervalley scattering, the singularity of the kernels of impurities, and optical polar phonon scattering. The charge conservation of the scheme is achieved by a suitable discretization of the collision operator within the accuracy of the middle-point integration rule. A detailed study of this numerical scheme is made by showing results on the bulk material, numerically validating the results by comparison to direct simulation Monte Carlo (DSMC) results, and testing our results for diodes and Gunn oscillations.

Key words. GaAs device, WENO methods, semiconductor simulation, deterministic scheme, Gunn oscillations

AMS subject classifications. 82D37, 65M06, 65Z05, 82C70

DOI. 10.1137/040607526

1. Introduction.

The most simplified, but relevant, band structure for GaAs semiconductor devices needs to take into account transport on the lowest energy conduction band. In its description, the most important information comes from the absolute and local minima since they typically concentrate most of the electrons in the conduction band. These minima are called valleys, and we will consider them as different populations in our model.

In GaAs the most interesting part of the valley structure is given by an absolute minimum at the center of the Brillouin zone, called the Γ-valley, and four local minima through the crystal orientation, called the L-valleys, which are considered equivalent [18]. The dispersion relation in each valley is given by the Kane model:

$$\mathcal{E}_A(\mathsf{k}_A) = \frac{\hbar^2|\mathsf{k}_A|^2}{m_A^* + \sqrt{(m_A^*)^2 + 2\,\alpha_A\, m_A^*\,\hbar^2|\mathsf{k}_A|^2}}, \qquad A = \Gamma, L, \tag{1.1}$$

where α_A is called the nonparabolicity factor.

*Received by the editors April 29, 2004; accepted for publication (in revised form) May 19, 2005; published electronically February 3, 2006. This work was partially supported by the European IHP network "Hyperbolic and Kinetic Equations: Asymptotics, Numerics, Applications," RNT2 2001 349. The first and second authors were also supported by DGI-MCYT/FEDER project MTM2005-08024.

http://www.siam.org/journals/sisc/27-6/60752.html

†Departamento de Matemática Aplicada, Universidad de Granada, 18071 Granada, Spain (caceresg@ugr.es).

‡ICREA (Institució Catalana de Recerca i Estudis Avançats) and Departament de Matemàtiques, Universitat Autònoma de Barcelona, E-08193 Bellaterra, Spain (carrillo@mat.uab.es).

§Dipartimento di Matematica e Informatica, Università di Catania, Catania, Italy (majorana@dmi.unict.it).

We study the transport of electrons in each valley by a linear transport Boltzmann equation:

$$\frac{\partial f_\Gamma}{\partial t} + \frac{1}{\hbar}\nabla_{\mathsf{k}_\Gamma}\mathcal{E}_\Gamma \cdot \nabla_{\mathsf{x}} f_\Gamma - \frac{e}{\hbar}E \cdot \nabla_{\mathsf{k}_\Gamma} f_\Gamma = Q_\Gamma(f_\Gamma, f_L), \tag{1.2}$$

$$\frac{\partial f_L}{\partial t} + \frac{1}{\hbar}\nabla_{\mathsf{k}_L}\mathcal{E}_L \cdot \nabla_{\mathsf{x}} f_L - \frac{e}{\hbar}E \cdot \nabla_{\mathsf{k}_L} f_L = Q_L(f_L, f_\Gamma), \tag{1.3}$$

and the only relation between electrons is given by electrostatic interaction through the electric potential V and its field E given by

$$\Delta V = \frac{e}{\epsilon_0}\left[\rho(t,\mathsf{x}) - C(\mathsf{x})\right], \qquad E = -\nabla_{\mathsf{x}} V, \tag{1.4}$$

where $f_A(t,\mathsf{x},\mathsf{k}_A)$ is the density function in phase space of electrons in the valley A at time $t \geq 0$ and position x with k_A its corresponding wave-number. Physical parameters and functions are the electron charge e, reduced Planck constant $\hbar$, dielectric constant ϵ_0, effective mass m_A^*, and donor-acceptor concentration C.

The macroscopic density of electrons is computed by the weighted average of the densities in each of the valleys,

$$\rho(t,\mathsf{x}) = Z_{L\Gamma}\rho_\Gamma(t,\mathsf{x}) + Z_{\Gamma L}\rho_L(t,\mathsf{x}), \tag{1.5}$$

where $\rho_\Gamma,\ \rho_L$ denote

$$\rho_\Gamma(t,\mathsf{x}) = \int_{\mathbb{R}^3} f_\Gamma(\mathsf{k}_\Gamma)\ d\mathsf{k}_\Gamma \quad \text{and} \quad \rho_L(t,\mathsf{x}) = \int_{\mathbb{R}^3} f_L(\mathsf{k}_L)\ d\mathsf{k}_L,$$

and $Z_{\Gamma L}$ and $Z_{L\Gamma}$ are the degeneration numbers from Γ-valley to L-valley and vice versa, namely, the number of possibilities for transferring electrons after scattering from Γ-valley to L-valley and vice versa. $Z_{\Gamma L}$ is 4, since there are four equivalent L-valleys, and $Z_{L\Gamma}$ is 1 (see Table 1.1 and Figure 1). The scattering operators on the

TABLE 1.1
Parameters.

m	Electron Mass	$9.1095 \cdot 10^{-31}$Kg
m_Γ^*	Effective electron mass in the Γ-valley	0.067 m
m_L^*	Effective electron mass in the L-valley	0.35 m
ρ_0	Density lattice	5360 Kg/m^3
v_s	Longitudinal sound speed	5240 m/s
α_Γ	Nonparabolicity factor in the Γ-valley	0.611 eV^{-1}
α_L	Nonparabolicity factor in the L-valley	0.242 eV^{-1}
ϵ_r	Relative dielectric constant	12.90
ϵ_∞	Relative dielectric constant at optical frequency	10.92
ϵ_v	Vacuum dielectric constant	$8.85419 \cdot 10^{-12}$ F/m
Ξ_d	Acoustic-phonon deformation potential	7 eV
$D_t K$	Nonpolar optical phonon deformation potential	10^{11} eV/m
$\hbar\omega_{np}$	Nonpolar optical phonon energy	0.032 eV
$\hbar\omega_p$	Polar optical phonon energy	0.032 eV
$\mathcal{E}_{0\Gamma}$	Γ-valley bottom energy	0 eV
$\mathcal{E}_{0L}$	L-valley bottom energy	0.32 eV
$Z_{\Gamma L}$	Degeneracy from Γ to L valleys	4
$Z_{L\Gamma}$	Degeneracy form L to Γ valley	1
Z_{LL}	Degeneracy from L to L valleys	3
N_i	Intrinsic concentration	$1.79 \cdot 10^6$ cm^{-3}
N_I	Impurity concentration	10^{14} cm^{-3}
$Ze = e$	Elementary charge	$1.6021 \cdot 10^{-19}$ C

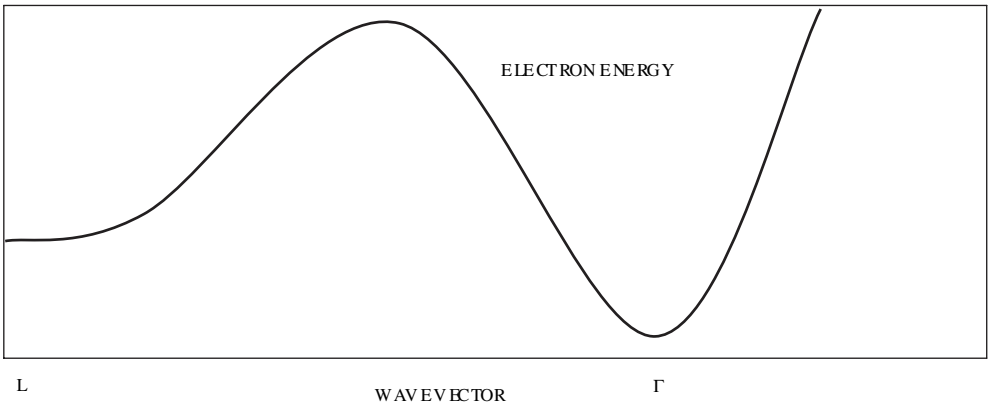


FIG. 1. *Scheme of the conduction band for GaAs.*

right-hand side, of (1.2) and (1.3) take into account the interaction of the electrons with the crystal vibrations, and therefore $Q_D(f_A, f_B)$, with $A, B, D = \Gamma, L$, are *linear* operators of the distribution functions. The main scattering mechanisms in GaAs [12, 18] are intravalley scatterings—impurities, acoustic phonon and polar optical phonon—while we take into account nonpolar optical intervalley scattering. These *linear* operators can be written in the form

$$Q_A^{(\gamma)}(f_A, f_A) = \int_{\mathbb{R}^3} \left[S^{(\gamma)}(\mathsf{k}'_A, \mathsf{k}_A) f_A(\mathsf{k}'_A) - S^{(\gamma)}(\mathsf{k}_A, \mathsf{k}'_A) f_A(\mathsf{k}_A) \right] d\mathsf{k}'_A, \tag{1.6}$$

$$Q_A^{(\gamma)}(f_A, f_B) = Z_{AB} \int_{\mathbb{R}^3} \left[S^{(\gamma)}(\mathsf{k}_B, \mathsf{k}_A) f_B(\mathsf{k}_B) - S^{(\gamma)}(\mathsf{k}_A, \mathsf{k}_B) f_A(\mathsf{k}_A) \right] d\mathsf{k}_B, \tag{1.7}$$

where $S^{(\gamma)}(\mathsf{k}, \mathsf{k}')$ is the transition probability from state k to k' per unit of time and the superscript (γ) identifies each scattering mechanism.

These scattering operators satisfy the detailed balance

$$S^{(\gamma)}(\mathsf{k}_B, \mathsf{k}_A) = S^{(\gamma)}(\mathsf{k}_A, \mathsf{k}_B) \exp\left(-\frac{\mathcal{E}_A - \mathcal{E}_B + \Delta_{AB}}{k_B T} \right),$$

where k_B is the Boltzmann constant, T is the lattice temperature, and $\Delta_{AB} = \mathcal{E}_{0A} - \mathcal{E}_{0B}$ is the energy gap between the minima of the two valleys. Therefore, the collision operators which appear in (1.2)–(1.3) (denoted in general as $Q_D(f_A, f_B)$, with $A, B, D = \Gamma, L$ above) are summations of the corresponding scattering processes $Q^{(\gamma)}$. The detailed expression of this transition probability depends on each scattering mechanism. In the case of scattering with randomly placed impurities, it reads

$$S^{(imp)}(\mathsf{k}_A, \mathsf{k}'_A) = \frac{K_{imp}}{(|\mathsf{k}_A - \mathsf{k}'_A|^2 + \beta^2)^2} \, \delta(\mathcal{E}'_A - \mathcal{E}_A),$$

with

$$K_{imp} = \frac{N_I Z^2 e^4}{4\pi^2 \hbar \epsilon_0^2}, \qquad \beta = \sqrt{\frac{e^2 N_I}{\epsilon_0 k_B T}},$$

where β is the inverse of the Debye length and N_I and $Z\,e$ are the impurities concentration and its charge, respectively. All physical parameters involved in the following are explained and detailed in Table 1.1.

The scattering with crystal vibrations in the acoustic mode is taken into account in the elastic approximation and is given by

$$S^{(ac)}(\mathsf{k}_A, \mathsf{k}'_A) = K_{ac}\, \delta(\mathcal{E}'_A - \mathcal{E}_A),$$

with

$$K_{ac} = \frac{k_B T \Xi_d^2}{4\pi^2 \hbar \rho_0 v_s^2}.$$

Nonpolar optical phonon interaction is considered in the inelastic approximation, and thus we consider emission and absorption of phonons of energy $\hbar\omega_{np}$:

$$S^{(np)}(\mathsf{k}_A, \mathsf{k}_B) = K_{np}\, \left[n_{np} \delta(\mathcal{E}_B - \mathcal{E}_A - \hbar\omega^+_{np}) + (1 + n_{np}) \delta(\mathcal{E}_B - \mathcal{E}_A + \hbar\omega^-_{np}) \right],$$

with

$$K_{np} = \frac{(D_t K)^2}{8\pi^2 \rho_0 \omega_{np}}, \qquad n_{np} = \frac{1}{\exp(\hbar\omega_{np}/(k_B T)) - 1},$$
$$\hbar\omega^+_{np} = \hbar\omega_{np} + \Delta_{AB}, \qquad \hbar\omega^-_{np} = \hbar\omega_{np} - \Delta_{AB}.$$

The occupation number n_{np} is given by the equilibrium Bose–Einstein distribution, and Z_{AB} is the degeneration number from A- to B-valley. GaAs solid-state properties imply that nonpolar optical phonon scattering contributes to the intervalley scattering for both valleys (Γ and L) and to the intravalley scattering only for the L-valley, since there exists only one Γ-valley, i.e., $Z_{\Gamma\Gamma} = 0$ (see [18] and (6.31) in the appendix).

Finally, polar optical phonon scattering has a transition probability

$$S^{(p)}(\mathsf{k}_A, \mathsf{k}'_A) = K_p \frac{G(\mathsf{k}_A, \mathsf{k}'_A)}{|\mathsf{k}_A - \mathsf{k}'_A|^2} \left[n_p \delta(\mathcal{E}'_A - \mathcal{E}_A - \hbar\omega_p) + (1 + n_p) \delta(\mathcal{E}'_A - \mathcal{E}_A + \hbar\omega_p) \right],$$

with

$$K_p = \frac{\hbar\omega_p e^2}{8\pi^2 \hbar} \left(\frac{1}{\epsilon_\infty} - \frac{1}{\epsilon_0} \right), \qquad n_p = \frac{1}{\exp(\hbar\omega_p/(k_B T)) - 1};$$

here $\hbar\omega_p$ is the polar optical phonon energy, n_p its occupation number, and the overlapping factor

$$G(\mathsf{k}_A, \mathsf{k}'_A) = \left[a_A a'_A + c_A c'_A \cos(\mathsf{k}_A, \mathsf{k}'_A) \right]^2,$$

with

$$a_A = \sqrt{\frac{1 + \alpha_A \mathcal{E}_A}{1 + 2\,\alpha_A \mathcal{E}_A}}, \qquad a'_A = \sqrt{\frac{1 + \alpha_A \mathcal{E}'_A}{1 + 2\,\alpha_A \mathcal{E}'_A}},$$
$$c_A = \sqrt{\frac{\alpha_A \mathcal{E}_A}{1 + 2\,\alpha_A \mathcal{E}_A}}, \qquad c'_A = \sqrt{\frac{\alpha_A \mathcal{E}'_A}{1 + 2\,\alpha_A \mathcal{E}'_A}}.$$

The equations are simplified by changing variables to suitable energy-type variables in which the scattering operators become either simple evaluations or linear integral operators with singular kernels. This change of variables was introduced first

in [9] in the parabolic band approximation and in [13] for general dispersion relations for the energy formula (1.1). This change of variables and a suitable nondimensionalization is performed for GaAs devices and summarized in section 2. Each of the scattering mechanisms gives a contribution to the nondimensional collision operator, as is fully detailed in the appendix.

The numerical scheme is based on ideas analogous to the one introduced in [3] for Si-based devices and quickly revisited and adapted to our case in section 2. Here, the new numerical difficulties are the preservation of total charge due to the intervalley scattering mechanisms and the singular character of the kernels of the impurities and polar optical phonon scattering. The rest of section 2 focuses on these new difficulties: we propose a charge conservative numerical approximation of the collision operator in which all integrals involving the singular kernels can be computed explicitly.

In this paper, we show the flexibility of the numerical scheme proposed in [3] in dealing with several scattering mechanisms. In order to make this work more readable the somewhat tedious previous work of deriving the expressions of these scattering mechanisms in the energy-based variables is relegated to the appendix. Nevertheless, once this preprocessing work is done, the deterministic numerical results deal with high gradient regions of the macroscopic quantities of interest in a remarkable way without the appearance of numerical noise typical of direct simulation Monte Carlo (DSMC) or particle-based methods [6, 7, 8, 11, 18], due to the high order accuracy of the method. Other advantages of our scheme are the accurate computation of the distribution function, and thus of all its moments, and the possibility of the transient study of the solution. The main drawback is the high dimensionality of the problem resulting in a much higher computational cost than the DSMC or particle methods but achievable with today's computer technology in a timely fashion in the one-dimensional case and by parallelized-coding techniques in the realistic two-dimensional case. Standard coding techniques have been used in a two-dimensional metal semiconductor field effect transistor Si-(MESFET) recently in [4] but require a huge computational time. In the case of GaAs-based devices, this drawback is more pronounced due to the higher number of unknowns.

Finally, we validate our scheme in the one-dimensional case by computing the electron valley occupancy, velocity, and energy of the homogeneous case (bulk material) and by comparing the results with the DSMC results for matching experimental data in [18]. Later we use this scheme to analyze the transport of electrons in $N^+ - N_i - N^+$ diodes, where N_i stands for the intrinsic concentration of the material, and we take advantage of this to apply our scheme to the Gunn-oscillator [1, 5, 16]. All these numerical tests are reported in section 3.

2. Equivalent system and numerical scheme. In this section, we will focus on explaining the numerical scheme applied to a system equivalent to (1.2), (1.3), and (1.4) and on highlighting the main differences with respect to the Si case [3].

The system equivalent to (1.2), (1.3), and (1.4) is obtained upon changing variables in (6.1) from the wave-momentum vector k_A, $A = \Gamma, L$, to energy variables and by choosing suitable units for the system. The new variables are the A-valley dimensionless electron energy w_A, the polar ϕ_A, and the azimuthal angles of the wave-momentum vector k_A. However, it is easier to work with the cosine of the azimuthal angle μ_A for computational reasons. Moreover, since we are considering the one-dimensional spatial case, we assume that the doping profile and the potential depend only on the third spatial component z of $\mathsf{x} \in \mathbb{R}^3$ and that the distribution function is cylindrically symmetric with respect to the axis of the electric field, and thus, that f_A does not depend on the polar angle ϕ_A.

Following the details of this change of variables (6.1) and the nondimensionalization, which we postponed to the appendix (see subsection 6.1.1) for reader's sake, our system can be written equivalently as a system of two transport equations in conservative form coupled to the Poisson equation, i.e.,

$$\frac{\partial \Phi_A}{\partial t} + \frac{\partial}{\partial z}\left(a_1^A\,\Phi_A\right) + \frac{\partial}{\partial w_A}\left(a_2^A \Phi_A\right) + \frac{\partial}{\partial \mu_A}\left(a_3^A \Phi_A\right) = \bar{Q}(\Phi_A), \qquad A = \Gamma, L, \tag{2.1}$$

where Φ_A is the new unknown function depending on dimensionless time $t \geq 0$, dimensionless spatial variable z, dimensionless energy w_A, and cosine of the angle with respect to field direction μ_A for each valley $A = \Gamma, L$. Here, the flux coefficients are

$$\begin{aligned} a_1^A &= a_1^A(w_A, \mu_A) = c_{1,A}\,\frac{\mu_A\,\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}{1+2\,\bar{\alpha}_A w_A}, \\ a_2^A &= a_2^A(w_A, \mu_A) = -c_{2,A}\,\bar{E}\,\mu_A\,\frac{\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}{1+2\bar{\alpha}_A w_A}, \\ a_3^A &= a_3^A(w_A, \mu_A) = -c_{2,A}\,\bar{E}\,\frac{1-\mu_A^2}{\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}, \end{aligned} \tag{2.2}$$

with $c_{1,A}$, $c_{2,A}$ constants (6.4) depending on the effective masses of each valley and the lattice temperature (assumed constant along the evolution). These equations are coupled to the dimensionless Poisson equation

$$\begin{aligned} \Delta_z \bar{V}(t,z) = c_p \Bigg(& c_\Gamma \int_0^\infty \int_{-1}^1 \Phi_\Gamma(t,z,w_\Gamma,\mu_\Gamma)\,d\mu_\Gamma\,dw_\Gamma \\ & + c_L \int_0^\infty \int_{-1}^1 \Phi_L(t,z,w_L,\mu_L)\,d\mu_L\,dw_L - \bar{C}(z)\Bigg), \end{aligned} \tag{2.3}$$

with c_p, c_Γ, and c_L defined in (6.33). We refer to the appendix for the exact values of all the constants involved. Here, $\bar{Q}(\Phi_A)$ is the nondimensional collision operator after the change of variables (6.31).

Let us point out that coefficients a_1^A and a_2^A are regular as functions of the variables w_A and μ_A. However, the coefficient a_3^A presents a singularity when the energy vanishes ($w_A = 0$), but we can expect that the factor $s_A(w_A)$, the Jacobian of the coordinate transformation up to constants (6.2), in the definition of Φ_A (6.3) compensates this singularity. This fact also makes it appealing to approximate numerically the derivative in μ_A by a conservative flux-finite difference method in which we approximate the whole flux.

For every valley, $A = \Gamma$, L, we will consider as initial value a local Maxwellian distribution with temperature T,

$$\Phi_A(0,z,w_A,\mu_A) = \mathcal{M}\,\bar{C}(z)\,M_A(w_A), \qquad A = \Gamma,\, L,$$

where

$$M_A(w_A) = s_A(w_A)\,e^{w_A - \bar{\mathcal{E}}_{0A}}, \qquad A = \Gamma,\, L,$$

and $\mathcal{M}$ is a constant chosen in such way that the initial density is equal to the doping density $\bar{C}(z)$.

2.1. Numerical scheme. We consider the same mesh for both valleys (Γ and L):

$$
\begin{aligned}
z_i &= \left(i - \frac{1}{2}\right)\Delta z, && i = 1, 2, \dots, N_z,\\
w_j &= \left(j - \frac{1}{2}\right)\Delta w, && j = 1, 2, \dots, N_w,\\
\mu_k &= -1 + \left(k - \frac{1}{2}\right)\Delta \mu, && k = 1, 2, \dots, N_\mu,
\end{aligned}
\tag{2.4}
$$

where $\Delta z = L/N_z$, $\Delta w = w_{max}/N_w$, and $\Delta\mu = 2/N_\mu$ are grid steps in each direction; w_{max} is taken as a multiple of $\alpha_n = \alpha_{np}$ (we will denote this quantity as α); and $\Delta w = \alpha/n_{multiple}$. Here, $n_{multiple}$ is an integer chosen in order that the evaluations of the distribution functions Φ_A needed in the collision term are on the grid, and thus we avoid interpolations that might introduce additional errors. We remark that $n_{multiple}$ has to be chosen in a suitable way to take into account all the scattering processes. The meshes are cell-centered in order to impose the boundary conditions and to avoid the singularity of a_3^A at $w_A = 0$. Let us point out that the energy gap between valleys is taken also as a multiple of the phonon energy to avoid again an additional source of errors due to interpolation. When these ratios are not integers, an interpolation procedure has to be done [14].

For every valley, system (2.1) is numerically solved using a conservative finite difference weighted essentially nonoscillatory (WENO) scheme of fifth order in the transport variables. The evolution on time is done by a third order Runge–Kutta TVD scheme. Details are referred to [3] since the numerical scheme has been widely explained therein. Boundary conditions are treated in the same way as in [3], and basically they amount to imposing zero flux conditions on the boundaries in energy and angle, and inflow Maxwellian boundary conditions in space. Poisson's equation is discretized and numerically solved by a standard central finite differences scheme. Stability in time for the resulting scheme is ensured by computing time steps verifying the following CFL condition: given

$$
t_A = \left(\frac{\max(|a_1^A|)}{\Delta z} + \frac{\max(|a_2^A|)}{\Delta w} + \frac{\max(|a_3^A|)}{\Delta \mu}\right), \qquad A = \Gamma, L,
$$

then $\Delta t = CFL/\max(t_\Gamma, t_L)$. Let us remark that the maximum is taken over all the points in the mesh, and since a_3^A is singular for $w_A = 0$, we may consider this CFL condition quite restrictive. Except for the circuit equation, in which we are interested in the transients for a long period of time compared to the typical relaxation time of the collision operator, the CFL condition is not restrictive in practice. In our simulations we considered CFL = 0.6 for the stationary case and 0.9 for the Gunn diode. Improvements of the method can come by using implicit-explicit Runge–Kutta methods [17] or solving the equation by splitting methods and/or implicit time-solvers. A semi-Lagrangian approach for the convective part of the equation may avoid the CFL condition [10].

2.2. Charge conservation. The Boltzmann equation for this system gives as a result a conservation law for the total charge. Therefore also, the proposed numerical scheme should preserve it. For a silicon device, this property holds numerically, since the collision operator is approximated by simple evaluations in mesh points. In our

case the complexity of the collision operator produces a new numerical difficulty, since it involves the functions I_4 and I_5, which are not smooth enough (see the appendix).

To be more precise, we deal with the following general problem: *Consider* Φ *the unknown in the Boltzmann equation and assume* $Q(\Phi) = Q_G(\Phi) - Q_L(\Phi)$*, where the gain term is*

$$Q_G(\Phi)(w,\mu) = s(w)\int_{-1}^{+1} I(w,\mu,\mu')\Phi(w,\mu')d\mu',$$

$I(w,\mu,.) \in C^\infty((-1,1)) \cap L^1((-1,1))$ *for all* $\mu \in (-1,1)$ *and* $I(w,\mu,\mu') = I(w,\mu',\mu)$*, and the loss term is*

$$Q_L(\Phi) = N(w)s(w)\Phi(w,\mu).$$

The functions I *and* N *are related by*

$$\int_{-1}^{+1} I(w,\mu,\mu')\,d\mu = N(w)$$

(independently of μ'*), and thus*

$$\int_{-1}^{1}\int_{0}^{+\infty} Q(\Phi)\,dw\,d\mu = 0. \tag{2.5}$$

We will apply this general framework to the singular functions in the angle variable μ, $I_{4,A}$ and $I_{5,A}$, which appear in the impurity and polar scattering process (see the appendix for a detailed expression of these functions, (6.22)–(6.23), (6.25)–(6.28)).

Expression (2.5), which means that we are dealing with a conservation law, is a consequence of the relation between I and N, since it can be rewritten as

$$\int_{-1}^{1}\int_{0}^{+\infty}\left[s(w)\int_{-1}^{+1} I(w,\mu,\mu')\Phi(w,\mu')d\mu'\right]dw\,d\mu = \int_{-1}^{1}\int_{0}^{+\infty} N(w)s(w)\Phi(w,\mu)\,dw\,d\mu.$$

In order to obtain a numerical method which preserves the charge, we consider the following approximation for the gain term:

$$Q_G(\Phi)(w,\mu_k) \approx s(w)\sum_{j=1}^{N_\mu}\Phi(w,\mu_j')\int_{I_j}\left[\frac{1}{\Delta\mu}\int_{I_k} I(w,\mu,\mu')d\mu\right]d\mu', \tag{2.6}$$

where

$$\mu_j' := -1 + (j-0.5)\Delta\mu\,, \quad I_j = [-1+(j-1)\Delta\mu, -1+j\Delta\mu]\,, \quad j = 1,2,\dots,N_\mu.$$

The numerical conservation of charge is obtained, taking into account (2.6), within the accuracy of the composite middle-point integration rule. In fact, we deduce that

$$\int_{-1}^{1} Q_G(\Phi)(w,\mu)\,d\mu \approx s(w)\Delta\mu\sum_{k=1}^{N_\mu}\sum_{j=1}^{N_\mu}\Phi(w,\mu_j')\int_{I_j}\left[\frac{1}{\Delta\mu}\int_{I_k} I(w,\mu,\mu')d\mu\right]d\mu'$$

$$= s(w)N(w)\Delta\mu\sum_{j=1}^{N_\mu}\Phi(w,\mu_j') \approx \int_{-1}^{1} Q_L(\Phi)(w,\mu)\,d\mu.$$

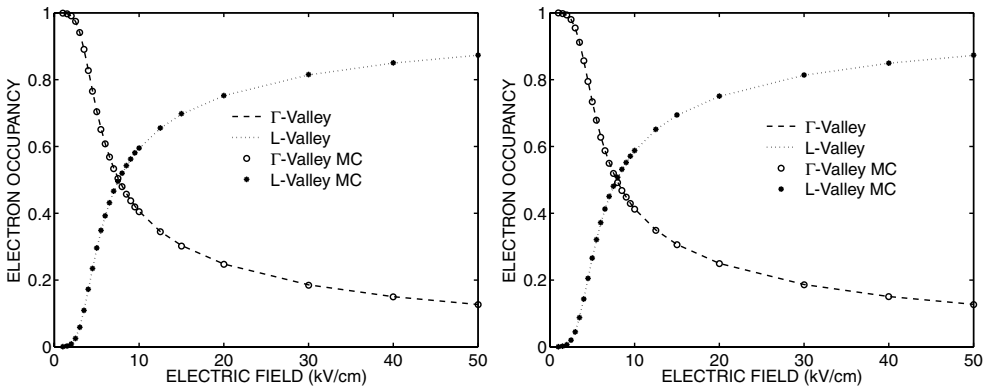


FIG. 2. *Electron occupation at room temperature for two different impurity concentrations: left* $N_I = 1 \times 10^{14}\, cm^{-3}$*, right* $N_I = 1 \times 10^{17}\, cm^{-3}$.

The double integrals

$$\int_{I_j} \left[\frac{1}{\Delta\mu} \int_{I_k} I(w, \mu, \mu') d\mu \right] d\mu'$$

may be evaluated numerically, using high order accurate integration formulas as adaptive Gaussian formulas. In our case, both for I_4 and I_5, it is possible to compute the double integration analytically. Here, we do not give the cumbersome formulas.

3. Numerical validation of the scheme: Bulk case. In this section, we check the main properties of the solution in a homogeneous-in-space setting for which there is an extensive literature from experiments [12] and from simulation with the Monte Carlo (MC) method [12, 18]. We will show that our results in the bulk material agree remarkably well with the results of MC codes in [18] that were already cross-checked with experiments.

In the results of this section we use two different values for the impurity concentration, $N_I = 10^{14}\,\mathrm{cm}^{-3}$ and $N_I = 10^{17}\,\mathrm{cm}^{-3}$, to which we will refer as low and high impurities cases. We will compute for each case the occupancy of each valley (rate of density of each valley with respect to total density), mean electron velocity and energy in terms of the constant force field, and the time evolution of mean electron velocity. We have run our program till stabilization of the mean velocity and energy up to six digits.

In this way, we recover in our model the following physical phenomena well-known in the literature:

- Figure 2: Γ-occupancy (resp., L-occupancy) decreases (resp., increases) monotonically for higher fields due to the jump of high energy electrons from the Γ- to the L-valley. Let us point out that the occupancies with different impurity concentrations are quite similar, since the values of the electric field at which the maximum of the velocity and the energy are reached are almost equal (see Figures 3 and 4).
- Figure 3: The mean electron energy behaves almost linearly for low fields and high fields, with a quick transition in the region in which the transfer of electrons from Γ-valley to the L-valley takes place.
- Figure 4: We observe that the mean electron velocity is not monotonically increasing as in the case of silicon, and it preserves a global maximum, after

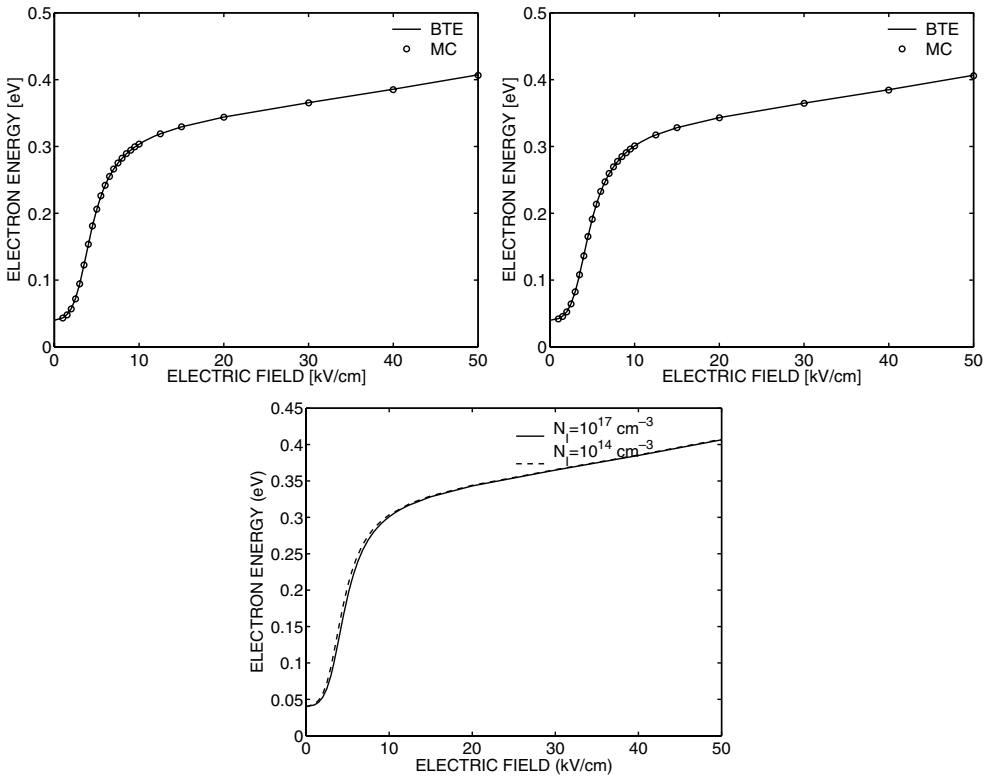


FIG. 3. *Electron energy at room temperature for two different impurity concentrations: top left* $N_I = 1 \times 10^{14}\, cm^{-3}$*, top right* $N_I = 1 \times 10^{17}\, cm^{-3}$*. We compare both results in the bottom figure.*

which it decreases to finally increase towards a *saturation speed value*. This phenomenon is known as the *negative differential conductivity*. It can be checked that the global maximum (maximum speed) and the initial slope (low-field mobility) decrease by increasing the impurity concentration. This is due to the lower value of the effective mass for the Γ-valley than for the L-valley.

- Figure 5: We can observe the *velocity overshooting* phenomenon in which the evolution of the mean velocity in time is not increasing and achieves a maximum before decaying monotonically to the stationary speed value. This is due to the lower momentum relaxation time compared to the energy relaxation time for high energies, i.e., high fields.

As a conclusion from previous figures and simulations, we have validated our numerical scheme, since we recover both qualitative and quantitative improvements by comparison to MC simulations (cf. the results in [12, 18]). We finally show in Figure 6 the steady state distribution function for a particular case.

4. Inhomogenous case. Once our numerical scheme has been validated, we show its flexibility by simulating a one-dimensional diode computing its transients, evolution of moments, and stationary states. On the other hand, we analyze a periodic-in-time circuit in which the main issue is to resolve in an effective way the oscillation of the potential. For all the inhomogeneous cases we choose the lower value for impurity concentration (see Table 1.1).

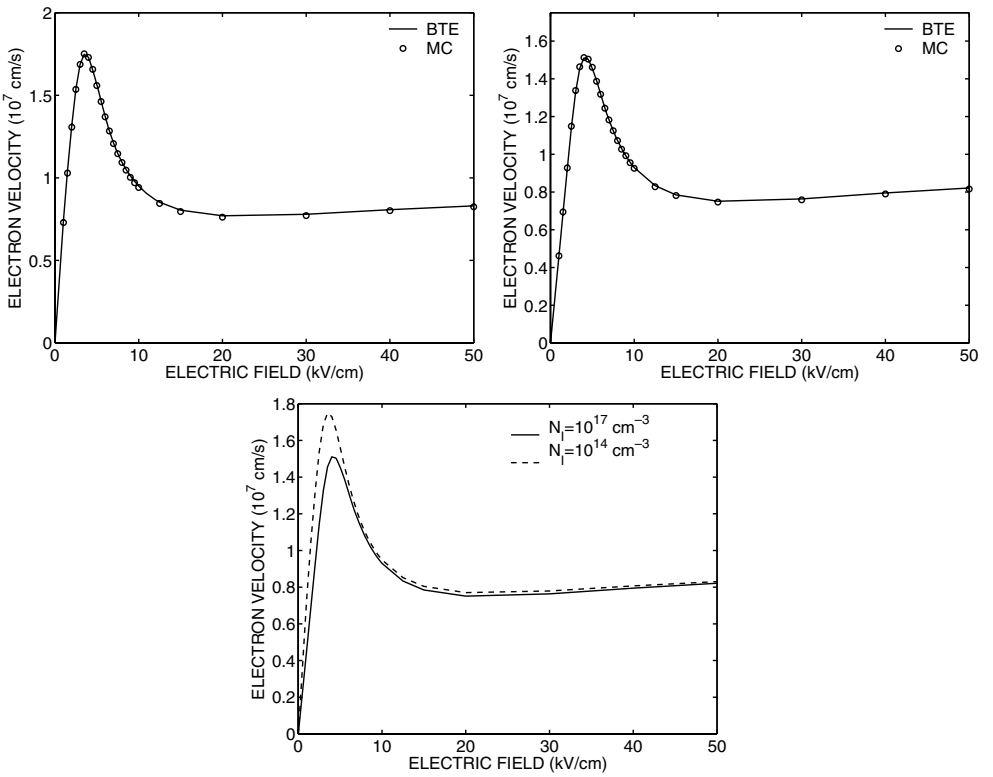


FIG. 4. *Electron velocity at room temperature for two different impurity concentrations: top left* $N_I = 1 \times 10^{14}\, cm^{-3}$, *top right* $N_I = 1 \times 10^{17}\, cm^{-3}$. *We compare both results in the bottom figure.*

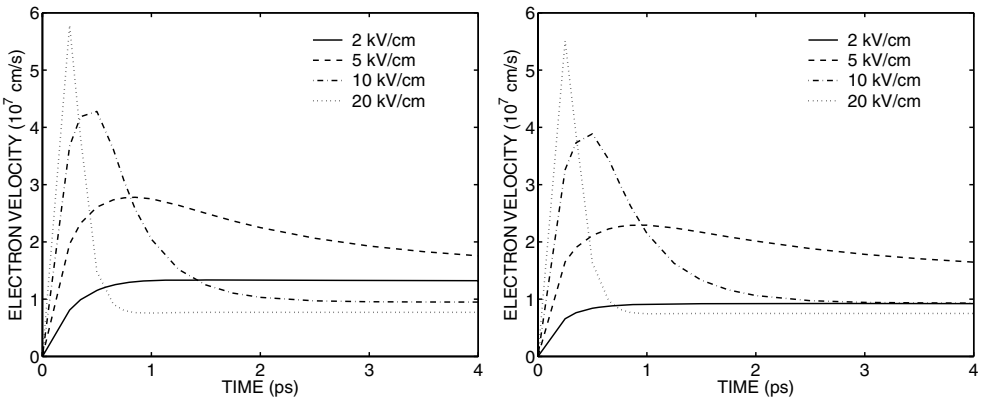


FIG. 5. *Time evolution of the electron velocity at room temperature for two different impurity concentrations: left* $N_I = 1 \times 10^{14}\, cm^{-3}$, *right* $N_I = 1 \times 10^{17}\, cm^{-3}$.

4.1. Stationary case. In this subsection we simulate the stationary state of a $N^+ - N_i - N^+$ diode based in GaAs, which means an active zone N_i-layer of 0.25 μm (see Table 1.1) sandwiched between two highly doped N^+-layers of 0.15 μm (cathode and anode), with $N^+ = 2{\times}10^{17}\mathrm{cm}^{-3}$. The intrinsic concentration is the concentration of the material itself, when the impurities are low and the concentration in the valence

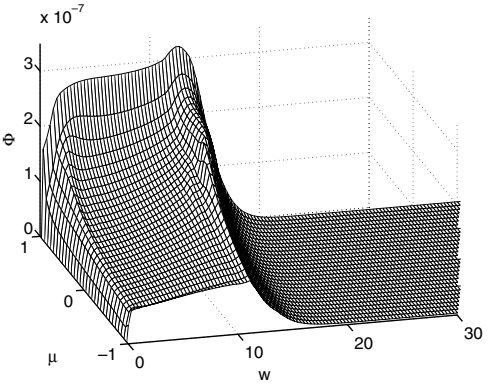


FIG. 6. *Snapshot of the distribution function for bulk material in valley* Γ *with* $E = 10\,kV/cm$ *applied field at equilibrium.*

band is the same as the concentration in the conduction band. This common value is the intrinsic concentration:

$$N_i = \left(\frac{2\pi k_B T\sqrt{m_n m_p}}{\hbar^2}\right)^{3/2} \exp\left(\frac{-E_g}{2k_B T}\right),$$

where m_n and m_p are the electron and hole effective masses, respectively, and E_g is the band gap energy. Therefore, the device has a length of 0.55 μm with two contacts: at 0.15 μm and 0.4 μm from the cathode.

We will focus our simulations on this device considered at room temperature with 0.25 V and 0.75 V applied bias. Our results are validated by MC simulations in Figure 7 by comparing density, mean velocity, energy, potential, and force field after reaching the stationary state. We consider the steady state to have been achieved once the energy, mean velocity, and energy are stabilized up to six digits. The results show a good agreement, the difference at the end of the channel being due in part to the need of having different meshes for both valleys in energy to improve the resolution in energy of the distribution function in the L-valley.

In Figures 8 and 9, we observe the evolution in time of macroscopic quantities, with an applied potential of 0.25 V and 0.75 V, respectively. Computation time was 9.26 ps when the system reached the stationary state in the first case, while we ran for only 3 ps in the second one. In both cases, we clearly see that the evolution towards stabilization is quite fast and that results for time 2.5 ps are already relevant (2 digits stabilized) for comparison with the MC steady state values in the 0.25 V case.

In the evolution of the velocity for the 0.75 V case, we appreciate the presence of a wave in the channel, which travels from left to right, still slightly noticeable after 3 ps. Similar phenomena happen for the density. This phenomenon is due to the electron accumulation in the active zone, close to the anode. The transit of electrons around the valleys Γ and L determines the shape of the macroscopic quantities. We observe that the mean velocity attains its maximum value around $z = 0.3\,\mu$m, and after that it quickly decreases to the anode. On the other hand, if we analyze the density picture, we find that it is again around $z = 0.3\,\mu$m, where the charge is accumulated. When the applied potential is 0.25 V, the charge accumulation disappears.

Finally, we conclude our stationary study by analyzing the transients for the distribution functions. As happens for an Si device with a 50 nm channel and a large applied potential [3], the distribution functions in some points of the device are far

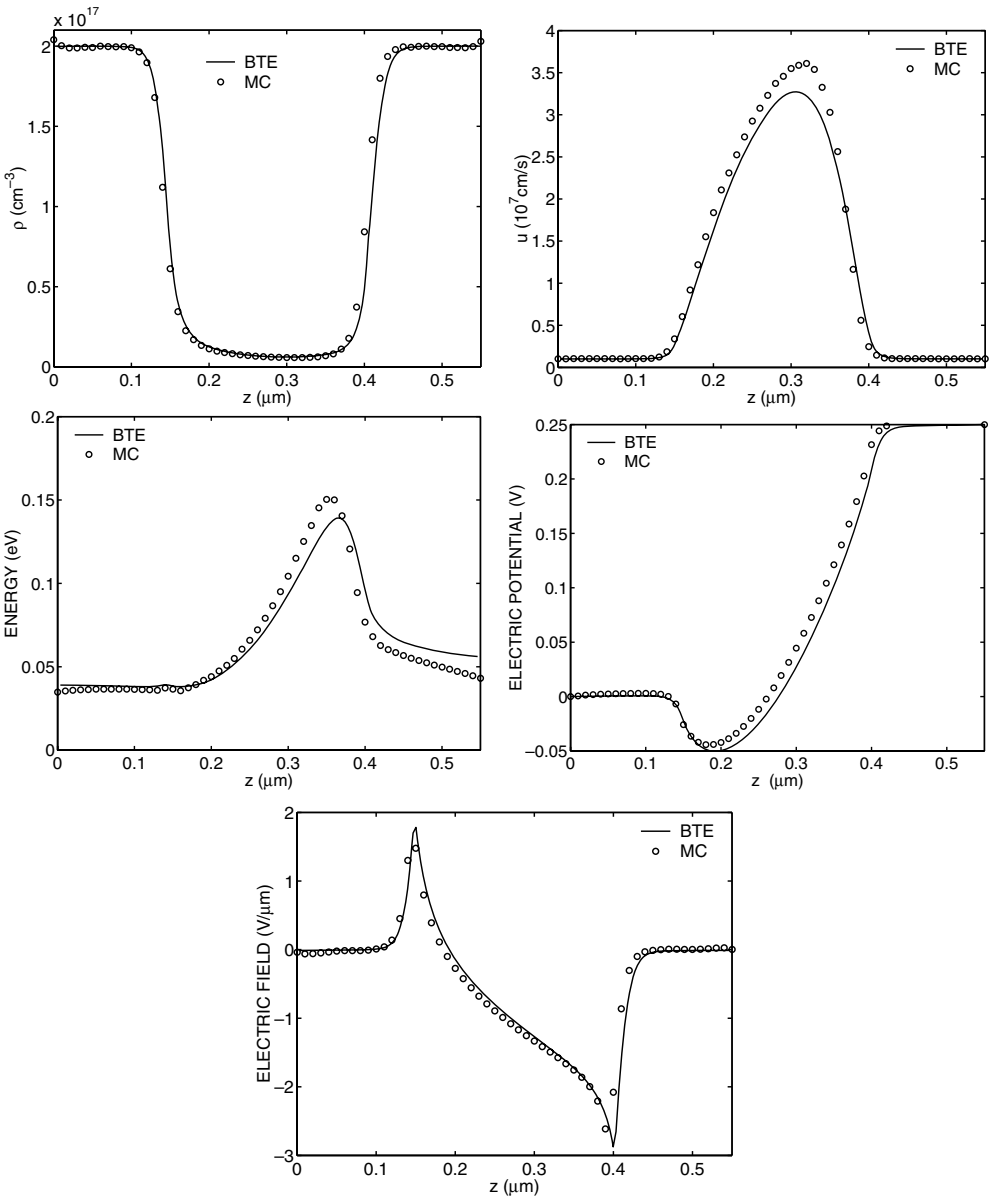


FIG. 7. *Macroscopic quantities after* 9 *ps for the* $N^+ - N_i - N^+$ *diode with* 0.25 *V applied bias at room temperature: density (top left), velocity (top right), energy (middle left), potential (middle right), and force field (bottom left).*

from being shifted Maxwellians typical of dynamics well described by hydrodynamical systems (remember $\mu = 1$ is the direction of the field and $\mu = -1$ is the opposite). The distribution function close to the second junction is not spherically symmetric.

The behavior of the distribution functions in both valleys is similar. Before the first contact and after the second one, they are practically Maxwellians. However, in the middle of the channel and just before the second contact, the distribution functions present purely kinetic features. We observe this fact in Figure 10, in which we show the time evolution of the distribution function in the Γ-valley just at the

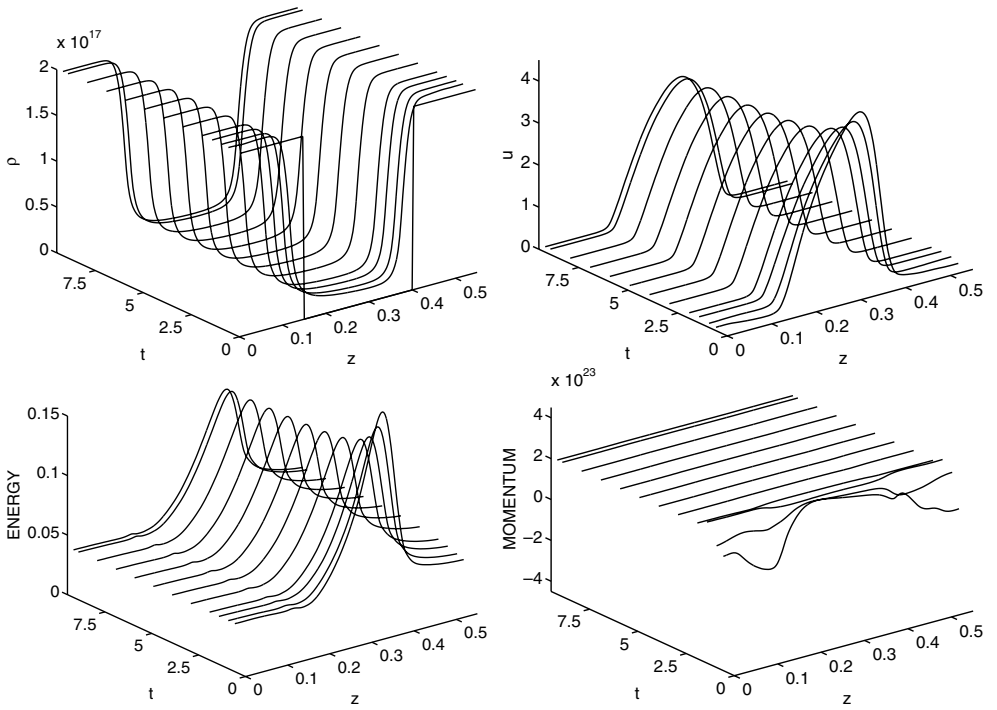


FIG. 8. *Macroscopic quantities evolution for the diode* $N^+ - N_i - N^+$ *under* 0.25 *V applied bias at room temperature* 300 *K: density* (cm^{-3}, *top left), velocity* ($10^7\, cm/s$, *top right), energy* (*eV*, *bottom left), and momentum* ($cm^{-2}\, s^{-1}$, *bottom right).*

edge of the channel. We show the distribution Γ-function at times 1, 1.5, 2, 2.5, 3 ps and the distribution L-function at time 3 ps.

4.2. Gunn diode. We conclude our numerical study by considering a GaAs notched diode. Simulations for this kind of device were shown in [1, 5, 16] through hydrodynamical models and the MC method. Here we use the same physical parameters considered in [1, 5]. We consider the same diode as it was dealt in previous references; a diode with $\mathcal{L} = 2\mu$m of length and the doping profile

$$C(\mathsf{x}) = \begin{cases} 10^{23}\,\mathrm{m}^{-3} & \text{if} \quad \mathsf{x} < 0.125\mu\mathrm{m}, \\ 10^{22}\,\mathrm{m}^{-3} & \text{if} \quad 0.125\mu\mathrm{m} < \mathsf{x} < 0.15\mu\mathrm{m}, \\ 0.5 \times 10^{22}\,\mathrm{m}^{-3} & \text{if} \quad 0.15\mu\mathrm{m} < \mathsf{x} < 0.1875\mu\mathrm{m}, \\ 10^{22}\,\mathrm{m}^{-3} & \text{if} \quad 0.1875\mu\mathrm{m} < \mathsf{x} < 1.875\mu\mathrm{m}, \\ 10^{23}\,\mathrm{m}^{-3} & \text{if} \quad 1.875\mu\mathrm{m} < \mathsf{x}. \end{cases} \tag{4.1}$$

In this case we substitute the V_{bias} by a circuit in order to simulate Gunn oscillations, as was done by Chen et al. [5]; the voltage V across the device and the current I through the battery satisfy differential equations

$$\frac{dV}{dt} = \frac{1}{C_{ap}}\left(I - I_d - \frac{V}{R}\right), \qquad \frac{dI}{dt} = \frac{1}{\Lambda}\left(V_B - V\right), \tag{4.2}$$

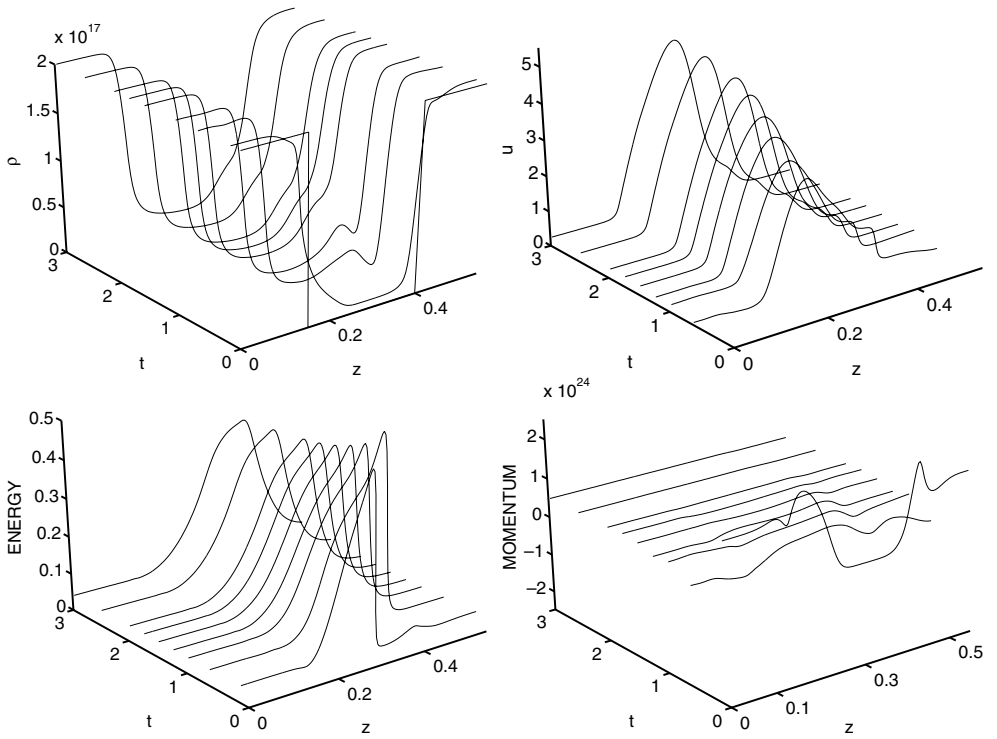


FIG. 9. *Macroscopic quantities evolution for the diode* $N^+ - N_i - N^+$ *under* 0.75 *V applied bias at room temperature* 300 K*: density (*cm^{-3}*, top left), velocity (*$10^7\, cm/s$*, top right), energy (eV, bottom left), and momentum (*$cm^{-2}\, s^{-1}$*, bottom right).*

where

$$I_d = -\frac{eA}{\mathcal{L}} \int_0^{\mathcal{L}} \left(Z_{L\Gamma} \rho_\Gamma u_\Gamma + Z_{\Gamma L} \rho_L u_L \right)\, dx \tag{4.3}$$

is the particle current in the device and where u_Γ and u_L are the mean velocities for every valley (see subsection 6.2 for details). Physical parameters for the capacitance, C_{ap}, resistance, R, inductance, Λ, and cross-section area, A, of the circuit are

$$C_{ap} = \frac{\epsilon A}{\mathcal{L}} + 0.82 \times 10^{-12}, \quad R = 25, \quad \Lambda = 3.5 \times 10^{-12}, \quad A = 10^{-9}, \quad V_B = 2.$$

Units are detailed in Table 4.1. As in the previous references about this device, we engage the circuit after 75 ps, when the diode is assumed to have reached the steady state. Therefore, to solve the oscillator (4.2) we consider the initial value at $t_0 = 75$ ps when the circuit is connected:

$$V(t_0) = V_B, \quad I(t_0) = 0.$$

The evolution on time of the voltage is shown in Figure 11. Our results reflect that at the beginning, when the circuit is engaged, the voltage evolves with one large oscillation during 10 ps, and afterwards the voltage oscillations are damped until 150 ps. Finally, the voltage becomes almost constant at 2 V. These results are in qualitative agreement with the results obtained by hydrodynamic models in [15, 16] and by MC methods in [1], but in our case voltage oscillations are damped faster. Finally,

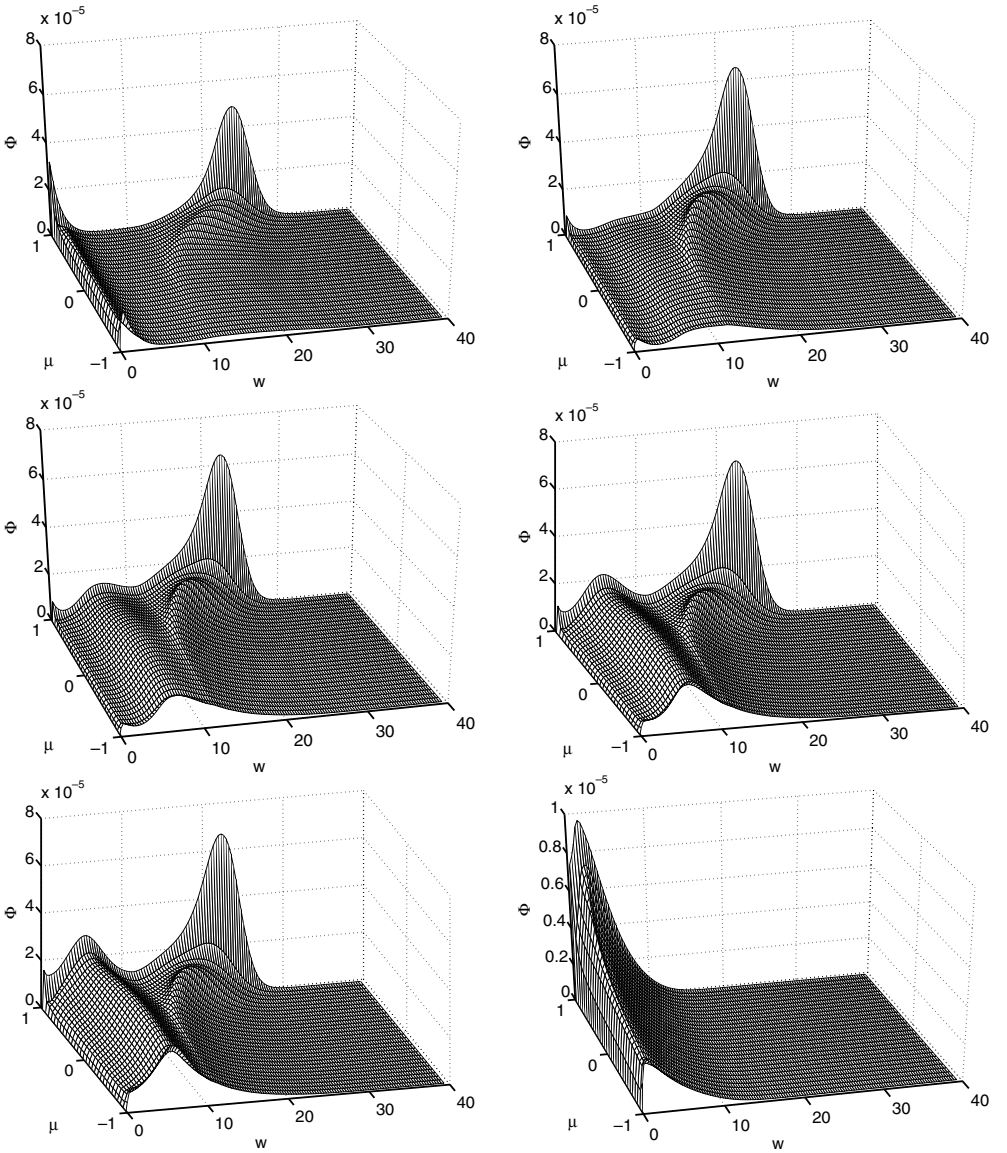


FIG. 10. *Snapshots of the distribution function in valley* Γ *with* 0.75 V *applied bias at room temperature* 300 K *at point* $z = 0.399667\,\mu m$, *for* $t = 1\,ps$ *(top left),* $t = 1.5\,ps$ *(top right),* $t = 2\,ps$ *(middle left),* $t = 2.5\,ps$ *(middle right),* $t = 3\,ps$ *(bottom left); and of the distribution function in valley* L *for* $t = 3\,ps$ *(bottom right).*

TABLE 4.1
Units.

x: μm	Potential: V
Temperature: K	Velocity: 10^7cm/s
Density: cm^{-3}	Energy: eV
Momentum: cm$^{-2}\,s^{-1}$	Current: A
Capacitance: F	Resistance: Ohm
Inductance: Henry	Cross-sectional area: m^2

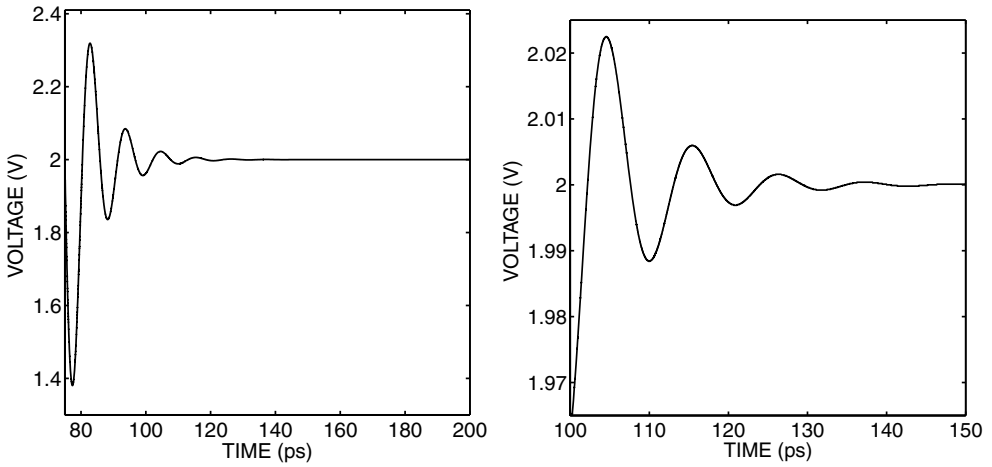


FIG. 11. *Time evolution of the voltage for the Gunn-oscillator circuit.*

let us point out that parabolic-band results for the evolution on time of the voltage are similar to the nonparabolic ones. We refer to [2] and the references therein for connected problems and information about the Gunn effect in superlattices.

5. Conclusions. A deterministic numerical scheme for solving the one-dimensional Boltzmann–Poisson system for semiconductor devices based on GaAs has been presented in this work. We have followed the same philosophy as in silicon case [3]: spherical change of coordinates, Boltzmann equation rewritten in conservative form, and finally numerical resolution using a high-order WENO-scheme in finite differences.

The band structure of a semiconductor based on GaAs is more complicated than for that based on silicon. The main numerical issue posed by this fact has been the charge conservation. We have proposed a numerical treatment of the gain and loss term giving us charge conservation within the accuracy of the middle-point integration rule.

The numerical results have been split in bulk and inhomogenous cases. The results obtained for the bulk case are excellent and validate our scheme (compared to Monte Carlo results). As a conclusion, we have checked the known phenomena typical of this material: distribution of electrons around the different valleys (occupation), negative differential conductivity, and velocity overshoot.

For the inhomogenous case we have considered two different problems: simulations of the steady state of $N^+ - N_i - N^+$ diodes and of the oscillation potential when a circuit is engaged to the device. The stationary state of $N^+ - N_i - N^+$ diode is analyzed by means of its macroscopic quantities and transients of the distribution function for each valley. Simulations for the evolution in time of the distribution functions in each valley show that the steady states in some points of the device are far from being shifted Maxwellian distributions.

In the inhomogeneous case, the agreement between MC simulations and deterministic solutions is still good, but not excellent. We have carefully checked that the differences arise from the boundary conditions. In fact, we use a condition on the fluxes (as in [3]) when the Boltzmann–Poisson system is solved and the well-known charge neutrality condition for the DSMC. We remark that in the stochastic model, at each time step, some particles (not few in our simulations) are injected, and they increase their energy when they arrive near junctions. We believe that, due to the high velocity in the channel, the particles do not reach a stationary state here. In

fact, we observe velocity and energy DSMC values greater then the corresponding values obtained by solving the Boltzmann–Poisson system. Further understanding of boundary conditions at the level of DSMC methods compared to deterministic kinetic boundary conditions needs to be done. As a summary, let us remark that the DSMC method and the deterministic method are based on the same equations and the same physical parameters, and thus the discrepancies between them are due to their different treatment of the boundary conditions.

For the Gunn-oscillator case the results are in qualitative concordance with the results obtained by hydrodynamic models and MC schemes; however, the voltage oscillations in our case are damped faster.

We have shown in this paper the flexibility of the approach in [3] for solving Boltzmann transport equation (BTE) for semiconductors in including many relevant scattering effects and different valley populations and thus the possibility of solving them for different semiconductor materials with complicated band structure.

6. Appendix: Change of variables and nondimensionalization. This appendix is devoted to summarizing the computations which transform system (1.2)–(1.4) into system (2.1), (2.3). The appendix is split in two subsections: the first one is devoted to the change of variables and nondimensionalization, while the last subsection reports the precise expression of the hydrodynamic quantities shown in the previous section. Tedious but simple computations are skipped.

6.1. Change of variables and nondimensionalization. We consider the change of variables

(6.1)
$$\mathsf{k}_A = \frac{\sqrt{m_A^* k_B T}}{\hbar}\sqrt{2\,w_A\,(1+\bar\alpha_A\,w_A)}\left(\sqrt{1-\mu_A^2}\cos\phi_A, \sqrt{1-\mu_A^2}\,\mathrm{sen}\,\phi_A, \mu_A\right),$$

where $\bar\alpha_A = k_B T\,\alpha_A$ is the nonparabolicity factor without dimension, w_A is the dimensionless energy variable, μ_A is the cosine of the angle between the wave vector k_A and the z-axis, and ϕ_A is the polar angle ($A = \Gamma, L$). Since we assume that the doping profile and the potential (and consequently the electric field) are only z dependent, our device will be considered in the z direction with $\mathsf{x} = (x, y, z)$.

In order to introduce the dimensionless parameters and variables, let us consider t_* and l_* the characteristic time and length, respectively. We denote by $\bar{}$ the dimensionless variables and parameters:

$$t = t_*\,\bar t, \qquad \mathsf{x} = l_*\bar{\mathsf{x}},$$
$$\mathsf{k}_A = \frac{1}{\hbar}\sqrt{m_A^* k_B T}\,\bar{\mathsf{k}}_A, \qquad \frac{\mathcal{E}(\mathsf{k}_A)}{k_B T} = \bar{\mathcal{E}}_A(\bar{\mathsf{k}}_A) = \frac{\|\bar{\mathsf{k}}_A\|^2}{1+\sqrt{1+2\,\bar\alpha_A\|\bar{\mathsf{k}}_A\|^2}},$$
$$E(t,\mathsf{x}) = \frac{\sqrt{m_X^* k_B T}}{e\,t_*}\,\bar E(\bar t,\bar{\mathsf{x}}), \qquad V(t,\mathsf{x}) = \frac{l_*\sqrt{m_X^* k_B T}}{e\,t_*}\,\bar V(\bar t,\bar{\mathsf{x}}),$$
$$C(\mathsf{x}) = \left(\frac{\sqrt{m_X^* k_B T}}{\hbar}\right)^3 \bar C(\bar{\mathsf{x}}), \qquad f_A(t,\mathsf{x},\mathsf{k}_A) = \bar f_A(\bar t,\bar{\mathsf{x}},\bar{\mathsf{k}}_A),$$
$$\alpha_p = \frac{\hbar\omega_p}{k_B T}, \qquad a_p = \frac{n_p+1}{n_p} = \exp(\alpha_p),$$
$$\alpha_{np} = \frac{\hbar\omega_{np}}{k_B T}, \qquad a_{np} = \frac{n_{np}+1}{n_{np}} = \exp(\alpha_{np}),$$
$$\Delta_{AB} = k_B T\,\bar\Delta_{AB}, \qquad \bar\alpha_A = k_B T\,\alpha_A.$$

In the following, according to the same rule, we denote by $\bar{g}$ the function g after introducing dimensionless variables.

As we can observe later (in the Poisson equation), we need to introduce a characteristic mass m_X^*. This mass might be chosen as m_Γ^* or m_L^*. In our simulations we have considered m_X^* as the electron mass (see Table 1.1).

The unknown f_A in terms of the new variables is denoted by F_A. We denote by s_A the Jacobian of the dimensionless change of variables $\bar{\mathsf{k}}_A(w_A, \mu_A, \phi_A)$. It can be easily verified to be

$$s_A(w_A) = \sqrt{2\, w_A\, (1 + \bar{\alpha}_A w_A)}\, (1 + 2\, \bar{\alpha}_A w_A), \tag{6.2}$$

and we consider the new unknown

$$\Phi_A(t, z, w_A, \mu_A) = F_A(t, z, w_A, \mu_A)\, s_A(w_A), \tag{6.3}$$

with $z \in [0, L]$, $w_A \in [0, w_{max}]$, and $\mu_A \in [-1, 1]$, where w_{max} is the maximum value of the energy, chosen in such a way that for our simulations

$$F_A(t, z, w_A, \mu_A) \approx 0 \qquad \text{for } w_A \geq w_{max}, \quad \forall\, t, z, \mu_A,$$

for both valleys.

Remark 6.1. Our nondimensionalization fixes $\bar{\mathcal{E}}_A(\bar{\mathsf{k}}_A) = w_A$, and thus w_A is the dimensionless energy. Moreover, since the function $s_A(w_A)$ is proportional to the Jacobian of the change of variables, we can evaluate moments of the distribution function directly using Φ_A.

The rest of the section is devoted to writing the Boltzmann–Poisson system in terms of the new unknown Φ_A. We split the computations into three parts: transport term, collision operator, and Poisson equation.

6.1.1. Transport term. *Nondimensionalization.* Defining

$$c_{1,A} = \frac{t_*}{l_*} \sqrt{\frac{k_B T}{m_A^*}} \quad \text{and} \quad c_{2,A} = \sqrt{\frac{m_X^*}{m_A^*}}, \tag{6.4}$$

we obtain

$$\frac{1}{\hbar} \nabla_{\mathsf{k}_A} \mathcal{E}_A \cdot \nabla_{\mathsf{x}} f_A = \frac{c_{1,A}}{t_*} \frac{\bar{\mathsf{k}}_A}{1 + 2\, \bar{\alpha}_A \bar{\mathcal{E}}_A} \cdot \nabla_{\bar{\mathsf{x}}} \bar{f}_A$$

and

$$-\frac{e}{\hbar} E \cdot \nabla_{\mathsf{k}_A} f_A = -\frac{c_{2,A}}{t_*} \bar{E} \cdot \nabla_{\bar{\mathsf{k}}_A} \bar{f}_A$$

Therefore, the transport operator in terms of $\bar{f}_A$ is written as

$$\frac{1}{t_*} \left(\frac{\partial \bar{f}_A}{\partial \bar{t}} + c_{1,A} \frac{\bar{\mathsf{k}}_A}{1 + 2\, \bar{\alpha}_A \bar{\mathcal{E}}_A} \cdot \nabla_{\bar{\mathsf{x}}} \bar{f}_A - c_{2,A} \bar{E} \cdot \nabla_{\bar{\mathsf{k}}_A} \bar{f}_A \right). \tag{6.5}$$

Change of coordinates. The next step is to find the expression of the transport operator in terms of F_A:

$$c_{1,A} \frac{\bar{\mathsf{k}}_A}{1 + 2\, \bar{\alpha}_A \bar{\mathcal{E}}_A} \cdot \nabla_{\bar{\mathsf{x}}} \bar{f}_A = c_{1,A} \frac{\mu_A \sqrt{2\, w_A (1 + \bar{\alpha}_A w_A)}}{1 + 2\, \bar{\alpha}_A w_A} \cdot \nabla_z F_A.$$

Since we are considering the one-dimensional case taking the direction of the electric field, in order to avoid a tedious notation we use $\bar{E}$ indistinctly to denote $\bar{E} = (0, 0, \bar{E}_z)$ and $\bar{E}_z$. Thus,

$$\bar{E} \cdot \nabla_{\bar{\mathsf{k}}_A} \bar{f}_A = \bar{E} \left(\frac{\partial F_A}{\partial w_A} \frac{\partial w_A}{\partial \bar{\mathsf{k}}_{A3}} + \frac{\partial F_A}{\partial \mu_A} \frac{\partial \mu_A}{\partial \bar{\mathsf{k}}_{A3}} \right), \tag{6.6}$$

where $\bar{\mathsf{k}}_A = (\bar{\mathsf{k}}_{A1}, \bar{\mathsf{k}}_{A2}, \bar{\mathsf{k}}_{A3})$. Since $|\bar{\mathsf{k}}_A|^2 = 2w_A(1 + \bar{\alpha}_A w_A)$,

$$\nabla_{\bar{\mathsf{k}}_A} |\bar{\mathsf{k}}_A|^2 = \nabla_{\bar{\mathsf{k}}_A} \left[2w_A(1 + \bar{\alpha}_A w_A) \right],$$

and we deduce

$$2\,\bar{\mathsf{k}}_A = 2\,(1 + 2\bar{\alpha}_A w_A) \nabla_{\bar{\mathsf{k}}_A} w_A.$$

In this way,

$$\nabla_{\bar{\mathsf{k}}_A} w_A = \frac{\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}{1 + 2\bar{\alpha}_A w_A} \left(\sqrt{1 - \mu_A^2} \cos\phi_A, \sqrt{1 - \mu_A^2}\, \mathrm{sen}\, \phi_A, \mu_A \right),$$

and consequently

$$\frac{\partial w_A}{\partial \bar{\mathsf{k}}_{A3}} = \frac{\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}{1 + 2\bar{\alpha}_A w_A}\, \mu_A. \tag{6.7}$$

On the other hand, recalling $\bar{\mathsf{k}}_{A3} = \mu_A \sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}$ and

$$\frac{\partial}{\partial \bar{\mathsf{k}}_{A3}} \left(\mu_A \sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)} \right) = \frac{\partial \bar{\mathsf{k}}_{A3}}{\partial \bar{\mathsf{k}}_{A3}} = 1,$$

we get

$$\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}\, \frac{\partial \mu_A}{\partial \bar{\mathsf{k}}_{A3}} + \mu_A \frac{1 + 2\,\bar{\alpha}_A w_A}{\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}\, \frac{\partial w_A}{\partial \bar{\mathsf{k}}_{A3}} = 1,$$

and then using (6.7),

$$\frac{\partial \mu_A}{\partial \bar{\mathsf{k}}_{A3}} = \frac{1 - \mu_A^2}{\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}. \tag{6.8}$$

By substituting (6.7) and (6.8) into (6.6), the final expression for the field transport term is deduced.

Final expression. Therefore, after the change of coordinates and nondimensionalization, the transport operator is rewritten as

$$\begin{aligned} &\frac{1}{t_*} \left(\frac{\partial F_A}{\partial \bar{t}} + c_{1,A} \frac{\mu_A \sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}{1 + 2\,\bar{\alpha}_A w_A} \frac{\partial}{\partial z} F_A \right) \\ &\qquad - \frac{c_{2,A} \bar{E}}{t_*} \left(\frac{\partial F_A}{\partial w_A} \frac{\sqrt{2\,w_A(1 + \bar{\alpha}_A w_A)}}{1 + 2\bar{\alpha}_A w_A} \mu_A + \frac{\partial F_A}{\partial \mu_A} \frac{1 - \mu_A^2}{\sqrt{2 w_A(1 + \bar{\alpha}_A w_A)}} \right). \end{aligned} \tag{6.9}$$

Finally, multiplying the previous expression by $s_A(w_A)$, we obtain

$$\frac{1}{t_*}\left[\frac{\partial \Phi_A}{\partial \bar{t}} + \frac{\partial}{\partial z}\left(a_1^A\,\Phi_A\right) + \frac{\partial}{\partial w_A}\left(a_2^A \Phi_A\right) + \frac{\partial}{\partial \mu_A}\left(a_3^A \Phi_A\right)\right], \tag{6.10}$$

where

$$a_1^A = a_1^A(w_A,\mu_A) = c_{1,A}\,\frac{\mu_A\,\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}{1+2\,\bar{\alpha}_A w_A},$$

$$a_2^A = a_2^A(w_A,\mu_A) = -c_{2,A}\,\bar{E}\,\mu_A\frac{\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}{1+2\bar{\alpha}_A w_A},$$

and

$$a_3^A = a_3^A(w_A,\mu_A) = -c_{2,A}\,\bar{E}\,\frac{1-\mu_A^2}{\sqrt{2\,w_A(1+\bar{\alpha}_A w_A)}}.$$

Therefore, considering (6.10) multiplied by t_*, we obtain the dimensionless left-hand side

$$\frac{\partial \Phi_A}{\partial \bar{t}} + \frac{\partial}{\partial z}\left(a_1^A\,\Phi_A\right) + \frac{\partial}{\partial w_A}\left(a_2^A \Phi_A\right) + \frac{\partial}{\partial \mu_A}\left(a_3^A \Phi_A\right). \tag{6.11}$$

6.1.2. Collision operator. *Intravalley scattering.* The expression of the kernel is given by

(6.12)

$$\begin{aligned} S_A(\mathsf{k}_A,\mathsf{k}'_A) = &\left[K_{ac} + \frac{K_{imp}}{(|\mathsf{k}_A-\mathsf{k}'_A|^2+\beta^2)^2}\right]\delta(\mathcal{E}'_A-\mathcal{E}_A)\\ &+ K_P\,\frac{G(\mathsf{k}_A,\mathsf{k}'_A)}{|\mathsf{k}_A-\mathsf{k}'_A|^2}\left[n_p\delta(\mathcal{E}'_A-\mathcal{E}_A-\hbar\omega_p)+(1+n_p)\delta(\mathcal{E}'_A-\mathcal{E}_A+\hbar\omega_p)\right]\\ &+ Z_{AA}K_{np}\left[n_{np}\delta(\mathcal{E}'_A-\mathcal{E}_A-\hbar\omega_{np})+(1+n_{np})\delta(\mathcal{E}'_A-\mathcal{E}_A+\hbar\omega_{np})\right]. \end{aligned}$$

Using the new dimensionless variables, (6.12) is rewritten as

$$\begin{aligned} S_A(\mathsf{k}_A,\mathsf{k}'_A) = &\left[K_{ac} + \frac{K_{imp}\,\hbar^4\,(m_A^* k_B T)^{-2}}{\left[\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)+{\beta_A}^2\right]^2}\right]\delta(k_BT(w'_A-w_A))\\ &+ K_p\,\frac{\bar{G}(w_A,w'_A,\mu_A,\mu'_A)}{\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)}\,n_p\\ &\quad\cdot\left[\delta(k_BT(w'_A-w_A-\alpha_p))+a_p\delta(k_BT(w'_A-w_A+\alpha_p))\right]\\ &+ Z_{AA}K_{np}n_{np}\\ &\quad\cdot\left[\delta(k_BT(w'_A-w_A-\alpha_{np}))+a_{np}\delta(k_BT(w'_A-w_A+\alpha_{np}))\right], \end{aligned} \tag{6.13}$$

$\bar{G}(w_A,w'_A,\mu_A,\mu'_A)$ is the corresponding dimensionless function $G(\mathsf{k}_A,\mathsf{k}'_A)$ written in the new variables, and

$$\begin{aligned} \varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A) = {} & q_A(w_A)+q_A(w'_A)-2\sqrt{q_A(w_A)\,q_A(w'_A)}\\ &\cdot\left(\sqrt{1-{\mu_A}^2}\,\sqrt{1-{\mu'_A}^2}\cos(\phi_A-\phi'_A)+\mu_A\mu'_A\right), \end{aligned}$$

with

$$q_A(w_A) = 2w_A(1+\bar{\alpha}_A w_A), \qquad \beta_A = \frac{\hbar\,\beta}{\sqrt{m_A^* k_B T}},$$

(6.14)

$$\begin{aligned} S_A(\mathsf{k}_A,\mathsf{k}'_A)\,d\mathsf{k}'_A = &\left[\left[K_{ac} + \frac{K_{imp}\,\hbar^4\,(m_A^* k_B T)^{-2}}{\left[\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)+{\beta_A}^2\right]^2}\right]\delta(k_B T(w'_A - w_A))\right. \\ &+ K_p\,\frac{\bar{G}(w_A,w'_A,\mu_A,\mu'_A)}{\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)}\,n_p \\ &\cdot[\delta(k_B T(w'_A - w_A - \alpha_p)) + a_p\delta(k_B T(w'_A - w_A + \alpha_p))] \\ &+ Z_{AA}K_{np}n_{np} \\ &\left.\cdot\,[\delta(k_B T(w'_A - w_A - \alpha_{np})) + a_{np}\delta(k_B T(w'_A - w_A + \alpha_{np}))]\right] \\ &\cdot\,\frac{(m_A^* k_B T)^{3/2}}{\hbar^3}\,dw'_A\,d\mu'_A\,d\phi'_A. \end{aligned}$$

Using the properties of the Dirac delta function,

(6.15)

$$\begin{aligned} S_A(\mathsf{k}_A,\mathsf{k}'_A)\,d\mathsf{k}'_A = &\left[\left[K_{ac} + \frac{K_{imp}\,\hbar^4\,(m_A^* k_B T)^{-2}}{\left[\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)+{\beta_A}^2\right]^2}\right]\delta(w'_A - w_A)\right. \\ &+ K_p\,\frac{\bar{G}(w_A,w'_A,\mu_A,\mu'_A)}{\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)}\,n_p \\ &\cdot[\delta(w'_A - w_A - \alpha_p) + a_p\delta(w'_A - w_A + \alpha_p)] \\ &\left.+\,Z_{AA}K_{np}n_{np}\,[\delta(w'_A - w_A - \alpha_{np}) + a_{np}\delta(w'_A - w_A + \alpha_{np})]\right] \\ &\cdot\,\frac{(m_A^* k_B T)^{3/2}}{k_B T\hbar^3}\,dw'_A\,d\mu'_A\,d\phi'_A. \end{aligned}$$

In order to obtain an easier notation, we introduce the constants

$$c_{3,A} = K_{ac}\,\frac{t_*(m_A^*)^{3/2}(k_B T)^{1/2}}{\hbar^3}, \qquad c_{4,A} = K_{imp}\,\frac{t_*\hbar}{k_B T\sqrt{m_A^* k_B T}},$$

$$c_{5,A} = K_p\,\frac{t_* n_p\sqrt{m_A^*}}{\hbar\sqrt{k_B T}}, \qquad c_{6,A} = Z_{AA}K_{np}\,\frac{t_*(m_A^*)^{3/2}(k_B T)^{1/2}}{\hbar^3};$$

thus

$$
\begin{aligned}
S_A(\mathsf{k}_A, \mathsf{k}'_A)\, d\mathsf{k}'_A = \Bigg[\Bigg[c_{3,A} + \frac{c_{4,A}}{\left[\varphi(w_A, w'_A, \mu_A, \mu'_A, \phi_A, \phi'_A) + {\beta_A}^2\right]^2}\Bigg]\delta(w'_A - w_A) \\
+ \frac{c_{5,A}\,\bar{G}(w_A, w'_A, \mu_A, \mu'_A)}{\varphi(w_A, w'_A, \mu_A, \mu'_A, \phi_A, \phi'_A)} \\
\cdot \left[\delta(w'_A - w_A - \alpha_p) + a_p \delta(w'_A - w_A + \alpha_p)\right] \\
+ c_{6,A}\left[\delta(w'_A - w_A - \alpha_{np}) + a_{np}\delta(w'_A - w_A + \alpha_{np})\right]\Bigg] \\
\cdot \frac{1}{t_*}\, dw'_A\, d\mu'_A\, d\phi'_A.
\end{aligned} \tag{6.16}
$$

Intervalley scatterings.

$$
\begin{aligned}
S(\mathsf{k}_A, \mathsf{k}_B) = K_{np}\left[n_{np}\delta(\mathcal{E}_B - \mathcal{E}_A - \hbar\omega_{np} - \Delta_{AB})\right. \\
\left. + (1 + n_{np})\delta(\mathcal{E}_B - \mathcal{E}_A + \hbar\omega_{np} - \Delta_{AB})\right].
\end{aligned} \tag{6.17}
$$

After the nondimensionalization and change of coordinates,

$$
\begin{aligned}
S(\mathsf{k}_A, \mathsf{k}_B) = K_{np} n_{np}\left[\delta(k_B T(w_B - w_A - \alpha_{np} - \bar{\Delta}_{AB}))\right. \\
\left. + a_{np}\delta(k_B T(w_B - w_A + \alpha_{np} - \bar{\Delta}_{AB}))\right]
\end{aligned} \tag{6.18}
$$

and

$$
\begin{aligned}
S(\mathsf{k}_A, \mathsf{k}_B)\, d\mathsf{k}_D = K_{np} n_{np}\left[\delta(k_B T(w_B - w_A - \alpha_{np} - \bar{\Delta}_{AB}))\right. \\
\left. + a_{np}\delta(k_B T(w_B - w_A + \alpha_{np} - \bar{\Delta}_{AB}))\right] \\
\cdot \frac{(m^*_D k_B T)^{3/2}}{\hbar^3}\, dw_D\, d\mu_D\, d\phi_D.
\end{aligned} \tag{6.19}
$$

And finally, denoting

$$
c_{7,D} = K_{np} n_{np} \frac{t_* {m^*_D}^{3/2}\sqrt{k_B T}}{\hbar^3},
$$

we obtain

$$
\begin{aligned}
S(\mathsf{k}_A, \mathsf{k}_B)\, d\mathsf{k}_D = c_{7,D}\left[\delta(w_B - w_A - \alpha_{np} - \bar{\Delta}_{AB})\right. \\
\left. + a_{np}\delta(w_B - w_A + \alpha_{np} - \bar{\Delta}_{AB})\right]\frac{1}{t_*}\, dw_D\, d\mu_D\, d\phi_D.
\end{aligned} \tag{6.20}
$$

To conclude this subsection it just remains to compute

$$\int_{\mathbb{R}^3} S(\mathsf{k}_A,\mathsf{k}'_A)d\mathsf{k}'_A, \qquad \int_{\mathbb{R}^3} S(\mathsf{k}'_A,\mathsf{k}_A)f_A(\mathsf{k}'_A)d\mathsf{k}'_A,$$

$$\int_{\mathbb{R}^3} S(\mathsf{k}_B,\mathsf{k}_A)d\mathsf{k}'_A, \qquad \int_{\mathbb{R}^3} S(\mathsf{k}_A,\mathsf{k}_B)f_A(\mathsf{k}_A)d\mathsf{k}'_A.$$

These quantities are easily obtained using (6.16) and (6.20). Let us denote the positive part of $g(s)$ by $g^+(s)$,

(6.21)

$$\begin{aligned}
&\int_{\mathbb{R}^3} S_A(\mathsf{k}_A,\mathsf{k}'_A)\,d\mathsf{k}'_A \\
&\quad= \int_0^\infty\!\!\int_{-1}^1\!\!\int_0^{2\pi} \left[\left[c_{3,A}+\frac{c_{4,A}}{\left[\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)+{\beta_A}^2\right]^2}\right]\delta(w'_A-w_A)\right. \\
&\qquad + \frac{c_{5,A}\bar{G}(w_A,w'_A,\mu_A,\mu'_A)}{\varphi(w_A,w'_A,\mu_A,\mu'_A,\phi_A,\phi'_A)}\left[\delta(w'_A-w_A-\alpha_p)+a_p\delta(w'_A-w_A+\alpha_p)\right] \\
&\qquad \left. + c_{6,A}\left[\delta(w'_A-w_A-\alpha_{np})+a_{np}\delta(w'_A-w_A+\alpha_{np})\right]\right]\frac{s_A(w'_A)}{t_*}\;d\phi'_A\;d\mu'_A\;dw'_A \\
&\quad= \left[4\pi\,c_{3,A}s_A(w_A)+4\pi\,c_{4,A}\frac{s_A(w_A)}{{\beta_A}^2\left(4q_A(w_A)+{\beta_A}^2\right)}\right. \\
&\qquad + c_{5,A}\left[N_{5,A}(w_A,w_A+\alpha_p)\,s_A(w_A+\alpha_p)\right. \\
&\qquad\qquad \left. + a_pN^+_{5,A}(w_A,w_A-\alpha_p)\,s^+_A(w_A-\alpha_p)\right] \\
&\qquad \left. + c_{6,A}\,4\pi\left[s_A(w_A+\alpha_{np})+a_{np}s^+_A(w_A-\alpha_{np})\right]\right]\frac{1}{t_*}.
\end{aligned}$$

Here,

$$N_{5,A}(w_A,w'_A)=\int_{-1}^1\!\!\int_0^{2\pi}\frac{\bar{G}(\bar{\mathsf{k}}_A,\bar{\mathsf{k}}'_A)}{|\bar{\mathsf{k}}_A-\bar{\mathsf{k}}'_A|^2}\;d\phi'_A\;d\mu'_A, \tag{6.22}$$

which is written as

$$N_{5,A}(w_A,w'_A)=2\pi\left[2\hat{B}\left(2\hat{A}\hat{D}-\hat{B}\hat{C}\right)\hat{D}^{-2}+\hat{D}^{-3}\left(\hat{A}\hat{D}-\hat{B}\hat{C}\right)^2\log\left|\frac{\hat{C}+\hat{D}}{\hat{C}-\hat{D}}\right|\right], \tag{6.23}$$

with

$$\hat{A} = \bar{a}_A\,\bar{a}'_A, \quad \hat{B} = \bar{c}_A\,\bar{c}'_A, \quad \hat{C} = q_A(w_A) + q_A(w'_A), \quad \hat{D} = -2\sqrt{q_A(w_A)\,q_A(w'_A)}$$

after some computations

(6.24)

$$\begin{aligned}
&\int_{\mathbb{R}^3} S_A(\mathsf{k}'_A, \mathsf{k}_A) f_A(\mathsf{k}'_A)\, d\mathsf{k}'_A \\
&\quad = \int_0^{\infty}\int_{-1}^{1}\int_0^{2\pi} \left[\left[c_{3,A} + \frac{c_{4,A}}{\left[\varphi(w_A, w'_A, \mu_A, \mu'_A, \phi_A, \phi'_A) + {\beta_A}^2\right]^2}\right]\delta(w'_A - w_A)\right. \\
&\qquad\qquad + c_{5,A}\frac{\bar{G}(w_A, w'_A, \mu_A, \mu'_A)}{\varphi(w_A, w'_A, \mu_A, \mu'_A, \phi_A, \phi'_A)} \\
&\qquad\qquad\qquad \cdot \left[\delta(w'_A - w_A - \alpha_p) + a_p\delta(w'_A - w_A + \alpha_p)\right] \\
&\qquad\qquad \left. + c_{6,A}\left[\delta(w'_A - w_A - \alpha_{np}) + a_{np}\delta(w'_A - w_A + \alpha_{np})\right]\right] \\
&\qquad \cdot \frac{s_A(w'_A)\, F_A(w'_A, \mu'_A)}{t_*}\; d\phi'_A\; d\mu'_A\; dw'_A \\
&\quad = \int_{-1}^{1}\Big[2\pi c_{3,A} s_A(w_A) F_A(w_A, \mu'_A) \\
&\qquad + c_{4,A} I_{4,A}(w_A, \mu_A, \mu'_A) F_A(w_A, \mu'_A) s_A(w_A) \\
&\qquad + c_{5,A}\Big[I^+_{5,A}(w_A, w_A - \alpha_p, \mu_A, \mu'_A) F^+_A(w_A - \alpha_p, \mu'_A)\, s^+_A(w_A - \alpha_p) \\
&\qquad + a_p I_{5,A}(w_A, w_A + \alpha_p, \mu_A, \mu'_A) F_A(w_A + \alpha_p, \mu'_A)\, s_A(w_A + \alpha_p)\Big] \\
&\qquad + c_{6,A}\, 2\pi\left[F^+_A(w_A - \alpha_{np}, \mu'_A) s^+_A(w_A - \alpha_{np})\right. \\
&\qquad \left. + \; a_{np} F_A(w_A + \alpha_{np}, \mu'_A) s_A(w_A + \alpha_{np})\right]\Big]\, \frac{1}{t_*}\; d\mu'_A.
\end{aligned}$$

Where

$$I_{4,A}(w_A, \mu_A, \mu'_A) = \int_0^{+\infty}\int_0^{2\pi} \frac{1}{\left(|\bar{\mathsf{k}}_A - \bar{\mathsf{k}}'_A|^2 + \beta_A^2\right)^2}\delta(w'_A - w_A)\; d\phi'_A\; dw'_A \tag{6.25}$$

and we show that

(6.26)

$$I_{4,A}(w_A, \mu_A, \mu'_A) = \frac{2\,\pi\left[2\,q_A(w_A)\,(1 - \mu_A\,\mu'_A) + \beta_A^2\right]}{\left[4\,q_A^2(w_A)\,(\mu_A - \mu'_A)^2 + \beta_A^4 + 4\,\beta_A^2\,q_A(w_A)\,(1 - \mu_A\mu'_A)\right]^{3/2}}.$$

While

$$I_{5,A}(w_A, w'_A, \mu_A, \mu'_A) = \int_0^{2\pi} \frac{\bar{G}(\bar{\mathsf{k}}_A, \bar{\mathsf{k}}'_A)}{|\bar{\mathsf{k}}_A - \bar{\mathsf{k}}'_A|^2}\, d\phi'_A, \tag{6.27}$$

again after some computations we obtain

$$\begin{aligned} I_{5,A}(w_A, w'_A, \mu_A, \mu'_A) =& \frac{\pi}{2} A_* \frac{4 + \bar{\alpha}_A \left(4 + \bar{\alpha}_A C_*\right) C_*}{\sqrt{C_*^2 + 4B_*^2 \left(\mu_A^2 + {\mu'_A}^2 - 1\right) - 4B_* C_* \mu_A \mu'_A}} \\ & - \frac{\pi}{2} A_* \bar{\alpha}_A \left(2\bar{\alpha}_A B_* \mu_A \mu'_A + \bar{\alpha}_A C_* + 4\right) \end{aligned} \tag{6.28}$$

with

$$\begin{aligned} A_* &= \frac{1}{2(1 + 2\bar{\alpha}_A w_A)(1 + 2\bar{\alpha}_A w'_A)}, \\ B_* &= \frac{\sqrt{w_A w'_A}}{\sqrt{(1 + \bar{\alpha}_A w_A)(1 + \bar{\alpha}_A w'_A)}}, \\ C_* &= \frac{w_A}{1 + \bar{\alpha}_A w_A} + \frac{w'_A}{1 + \bar{\alpha}_A w'_A}. \end{aligned}$$

The last two terms are easily computed, and we obtain

$$\int_{\mathbb{R}^3} S(\mathsf{k}_B, \mathsf{k}_A)\, d\mathsf{k}_A = 4\pi\, c_{7,A} \left[s_A^+(w_B + \alpha_{np} + \bar{\Delta}_{BA}) + a_{np} s_A^+(w_B - \alpha_{np} + \bar{\Delta}_{BA}) \right] \frac{1}{t_*}, \tag{6.29}$$

$$\begin{aligned} &\int_{\mathbb{R}^3} S(\mathsf{k}_A, \mathsf{k}_B) f_A(\mathsf{k}_A)\, d\mathsf{k}_A \\ &\quad = 2\pi\, c_{7,A} \int_{-1}^{1} \left[F_A^+(w_B - \alpha_{np} - \bar{\Delta}_{AB}, \mu_A) s_A^+(w_B - \alpha_{np} - \bar{\Delta}_{AB}) \right. \\ &\qquad \left. + a_{np} F_A^+(w_B + \alpha_{np} - \bar{\Delta}_{AB}, \mu_A) s_A^+(w_B + \alpha_{np} - \bar{\Delta}_{AB}) \right] \frac{1}{t_*}\, d\mu_A. \end{aligned} \tag{6.30}$$

Finally, collecting (6.21), (6.24), (6.29), and (6.30) and multiplying them by $t_* s_A(w_A)$,

the collision operator can be rewritten as

(6.31)

$$\begin{aligned}\bar{Q}(\Phi_A) = s_A(w_A)\int_{-1}^{1}\Big\{&2\,\pi\,c_{3,A}\Phi_A(w_A,\mu_A')+c_{4,A}I_{4,A}(w_A,\mu_A,\mu_A')\Phi_A(w_A,\mu_A')\\ &+c_{5,A}\Big[I_{5,A}^{+}(w_A,w_A-\alpha_p,\mu_A,\mu_A')\Phi_A^{+}(w_A-\alpha_p,\mu_A')\\ &\quad+a_pI_{5,A}(w_A,w_A+\alpha_p,\mu_A,\mu_A')\Phi_A(w_A+\alpha_p,\mu_A')\Big]\\ &+c_{6,A}\,2\,\pi\left[\Phi_A^{+}(w_A-\alpha_{np},\mu_A')+a_{np}\,\Phi_A(w_A+\alpha_{np},\mu_A')\right]\Big\}\;d\mu_A'\\ -\Big\{&4\,\pi\,c_{3,A}s_A(w_A)+4\,\pi\,c_{4,A}\frac{s_A(w_A)}{{\beta_A}^2(4\,q_A(w_A)+{\beta_A}^2)}\\ &+c_{5,A}\Big[N_{5,A}(w_A,w_A+\alpha_p)s_A(w_A+\alpha_p)\\ &\quad+a_p\,N_{5,A}^{+}(w_A,w_A-\alpha_p)s_A^{+}(w_A-\alpha_p)\Big]\\ &+c_{6,A}\,4\pi\,\left[s_A(w_A+\alpha_{np})+a_{np}\,s_A^{+}(w_A-\alpha_{np})\right]\Big\}\,\Phi_A(w_A,\mu_A)\\ +Z_{AB}s_A(w_A)\,c_{7,B}\,2\pi\int_{-1}^{1}&\left[\Phi_B^{+}(w_A-\alpha_{np}-\bar{\Delta}_{BA},\mu_B)\right.\\ &\left.+a_{np}\Phi_B^{+}(w_A+\alpha_{np}-\bar{\Delta}_{BA},\mu_B)\right]\,d\mu_B\\ -Z_{AB}\,c_{7,B}\,4\pi&\left[s_B^{+}(w_A+\alpha_{np}+\bar{\Delta}_{AB})\right.\\ &\left.+a_{np}\,s_B^{+}(w_A-\alpha_{np}+\bar{\Delta}_{AB})\right]\Phi_A(w_A,\mu_A).\end{aligned}$$

6.1.3. Poisson equation. The right-hand side of the Poisson equation (1.4) satisfies

$$\Delta_{\bar{\mathsf{x}}}\bar{V}(\bar{t},\bar{\mathsf{x}}) = \frac{e\,t_*\,l_*}{\sqrt{m_X^*\,k_BT}}\,\Delta_{\mathsf{x}}V(t,\mathsf{x}).$$

The dimensionless ρ_Γ and ρ_L are given by

$$\rho_\Gamma(t,\mathsf{x}) = 2\,\pi\,\left(\frac{\sqrt{m_\Gamma^* k_BT}}{\hbar}\right)^3\int_0^\infty\int_{-1}^{1}\,\Phi_\Gamma(t,z,w_\Gamma,\mu_\Gamma)d\mu_\Gamma\;dw_\Gamma$$

and

$$\rho_L(t,\mathsf{x}) = 2\,\pi\,\left(\frac{\sqrt{m_L^* k_BT}}{\hbar}\right)^3\int_0^\infty\int_{-1}^{1}\Phi_L(t,z,w_L,\mu_L)\;d\mu_L\;dw_L.$$

Finally, using the dimensionless doping density, we obtain the dimensionless Poisson equation

$$\Delta_z \bar{V}(\bar{t},z) = c_p \left(c_\Gamma \int_0^\infty \int_{-1}^1 \Phi_\Gamma(\bar{t},z,w_\Gamma,\mu_\Gamma)\; d\mu_\Gamma\; dw_\Gamma + c_L \int_0^\infty \int_{-1}^1 \Phi_L(\bar{t},z,w_L,\mu_L)\; d\mu_L\; dw_L - \bar{C}(z) \right), \tag{6.32}$$

where

$$c_p = \frac{e^2\, t_*\, l_*\, m_X^*\, k_B T}{\epsilon_0\, \hbar^3}, \quad c_\Gamma = 2\,\pi\, Z_{L\Gamma} \left(\frac{m_\Gamma^*}{m_X^*}\right)^{3/2}, \quad c_L = 2\,\pi\, Z_{\Gamma L} \left(\frac{m_L^*}{m_X^*}\right)^{3/2}. \tag{6.33}$$

6.2. Macroscopic quantities. To conclude, we give the expression of the macroscopic quantities: mean density,

$$\rho(t,\mathsf{x}) = Z_{L\Gamma}\rho_\Gamma(t,\mathsf{x}) + Z_{\Gamma L}\rho_L(t,\mathsf{x}),$$

mean velocity,

$$u(t,\mathsf{x}) = \frac{1}{\hbar\,\rho(t,\mathsf{x})} \left(Z_{L\Gamma} \int_{\mathbb{R}^3} \nabla_{\mathsf{k}_\Gamma} \mathcal{E}_\Gamma(\mathsf{k}_\Gamma)\, f_\Gamma(t,\mathsf{x})\; d\mathsf{k}_\Gamma + Z_{\Gamma L} \int_{\mathbb{R}^3} \nabla_{\mathsf{k}_L} \mathcal{E}_L(\mathsf{k}_L)\, f_L(t,\mathsf{x})\; d\mathsf{k}_L \right),$$

energy,

$$\frac{1}{\rho(t,\mathsf{x})} \left(Z_{L\Gamma} \int_{\mathbb{R}^3} \mathcal{E}_\Gamma(\mathsf{k}_\Gamma)\, f_\Gamma(t,\mathsf{x})\; d\mathsf{k}_\Gamma + Z_{\Gamma L} \int_{\mathbb{R}^3} \mathcal{E}_L(\mathsf{k}_L)\, f_L(t,\mathsf{x})\; d\mathsf{k}_L \right),$$

and momentum,

$$\frac{1}{\hbar} \left(Z_{L\Gamma} \int_{\mathbb{R}^3} \nabla_{\mathsf{k}_\Gamma} \mathcal{E}_\Gamma(\mathsf{k}_\Gamma)\, f_\Gamma(t,\mathsf{x})\; d\mathsf{k}_\Gamma + Z_{\Gamma L} \int_{\mathbb{R}^3} \nabla_{\mathsf{k}_L} \mathcal{E}_L(\mathsf{k}_L)\, f_L(t,\mathsf{x})\; d\mathsf{k}_L \right).$$

Acknowledgments. We thank the anonymous referees, who certainly helped in improving the results and the presentation of this work.

REFERENCES

[1] A. M. Anile and S. D. Hern, *Two-valley hydrodynamical models for electron transport in gallium arsenide: Simulation of Gunn oscillations*, VLSI Design, 15 (2002), pp. 681–693.

[2] L. L. Bonilla and H. T. Grahn, *Two-population hydrodynamical models for carrier transport in gallium arsenide: Simulation of Gunn oscillations*, Rep. Prog. Phys., 68 (2005), pp. 577–683.

[3] J. A. Carrillo, I. M. Gamba, A. Majorana, and C.-W. Shu, *A WENO-solver for the transients of Boltzmann–Poisson system for semiconductor devices: Performance and comparisons with Monte Carlo methods*, J. Comput. Phys., 184 (2003), pp. 498–525.

[4] J. A. Carrillo, I. M. Gamba, A. Majorana, and C.-W. Shu, *A direct solver for 2D nonstationary Boltzmann–Poisson systems for semiconductor devices: A MESFET simulation by WENO-Boltzmann schemes*, J. Comput. Electronics, 2 (2003), pp. 375–380.

[5] G.-Q. Chen, J. W. Jerome, C. W. Shu, and D. Wang, *Two carrier semiconductor device models with geometric structure and symmetry properties*, in Modelling and Computation for Applications in Mathematics, Science, and Engineering, J. Jerome, ed., Oxford University Press, London, 1998, pp. 103–140.

[6] P. Degond, F. Delaurens, and F. J. Mustieles, *Semiconductor modelling via the Boltzmann equation*, Comput. Methods Appl. Sci. Engrg., (1990), pp. 311–324.

[7] F. Delaurens and F. J. Mustieles, *A deterministic particle method for solving kinetic transport equations: The semiconductor Boltzmann equation case*, SIAM J. Appl. Math., 52 (1992), pp. 973–998.

[8] P. Degond, B. Niclot, and F. Poupaud, *Deterministic particle simulations the Boltzmann transport equation of semiconductors*, J. Comput. Phys., 78 (1988), pp. 313–349.

[9] E. Fatemi and F. Odeh, *Upwind finite difference solution of Boltzmann equation applied to electron transport in semiconductor devices*, J. Comput. Phys., 108 (1993), pp. 209–217.

[10] F. Filbet, E. Sonnendrücker, and P. Bertrand, *Conservative numerical schemes for the Vlasov equation*, J. Comput. Phys., 172 (2001), pp. 166–187.

[11] C. Jacoboni and P. Lugli, *The Monte Carlo Method for Semiconductor Device Simulation*, Springer-Verlag, New York, 1989.

[12] C. Jacoboni and L. Reggiani, *The Monte Carlo method for the solution of charge transport in semiconductor with application to covalent materials*, Rev. Modern Phys., 55 (1983), pp. 645–705.

[13] A. Majorana and R. Pidatella, *A finite difference scheme solving the Boltzmann–Poisson system for semiconductor devices*, J. Comput. Phys., 174 (2001), pp. 649–668.

[14] A. Majorana, O.Muscato, and C.Milazzo, *Comparison of Monte Carlo and Boltzmann equation simulations for* 1*-D silicon devices*, COMPEL, 23 (2004), pp. 410–435.

[15] G. Mascali and V. Romano, *Hydrodynamical model of charge transport in GaAs based on the maximum entropy principle*, Contin. Mech. Thermodynam., 14 (2002), pp. 405–423.

[16] G. Mascali and V. Romano, *Simulation of Gunn oscillations with a non-parabolic hydrodynamical model based on the maximum entropy principle*, COMPEL, 24 (2005), pp. 35–54.

[17] L. Pareschi and G. Russo, *Implicit-explicit Runge–Kutta schemes for stiff systems of differential equations*, in Recent Trends in Numerical Analysis, L. Brugnano and D. Trigiante, eds., Nova Science Publ., New York, 3 (2000), pp. 269–289.

[18] K. Tomizawa, *Numerical Simulation of Submicron Semiconductor Devices*, Artech House, Boston, 1993.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2010–2031
© 2006 Society for Industrial and Applied Mathematics

GENERAL LINEAR METHODS FOR VOLTERRA INTEGRO-DIFFERENTIAL EQUATIONS WITH MEMORY*

CHENGJIAN ZHANG† AND STEFAN VANDEWALLE‡

Abstract. A new class of numerical methods for Volterra integro-differential equations with memory is developed. The methods are based on the combination of general linear methods with compound quadrature rules. Sufficient conditions that guarantee global and asymptotic stability of the solution of the differential equation and its numerical approximation are established. Numerical examples illustrate the convergence and effectiveness of the numerical methods.

Key words. stability, general linear methods, Volterra delay-integro-differential equation

AMS subject classifications. 65R20, 45L05, 65L20

DOI. 10.1137/040607058

1. Introduction. Volterra integro-differential equations (VIDEs) arise widely in the mathematical modeling of physical and biological phenomena. Significant advances in the theoretical analysis of such equations and in the numerical analysis and implementation of time-integration techniques for these problems have been made in the last few decades. For a survey of early results we refer the reader to the books [9, 26]. More recently, one has found that VIDEs with memory (MVIDEs), also called Volterra delay-integro-differential equations (VDIDEs), can be even more effective than standard VIDEs for the modeling of real-life problems; see, e.g., [8, 22]. This fact has led researchers to develop a theory and numerical analysis for MVIDEs. For example, numerical time-integration techniques of one-step collocation and Runge–Kutta (RK) type were investigated in [10, 17, 11, 23, 34, 21]. Linear multistep based methods were studied in [1, 2, 3, 27, 33].

So far, very few papers have discussed the *nonlinear stability* of MVIDEs. This is quite different from the case of delay differential equations (DDEs) without distributed delay, for which many results have been reported. For example, linear stability results for DDEs from [28] were extended in [29] to the nonlinear DDE case. In that paper it was proved that certain RK methods satisfy a so-called B_ρ-stability property. For the implicit Euler rule, [30] shows a contractivity result, which is a quite strong nonlinear stability property. Based on the concept of BN_f-stability, the authors of [6, 7, 31, 32] obtained a series of stability and contractivity results for continuous RK methods. In [32] it was shown that the requirement of BN_f-stability for a numerical method is a very strong requirement that leads to an order barrier. In order to avoid that order barrier, in [19, 20] numerical stability results have been derived for nonlinear DDEs based on the concept of algebraic stability of the underlying methods.

Stability results for MVIDEs were obtained in [5, 33, 34], where the authors investigated the nonlinear stability of continuous RK methods, discrete RK methods,

*Received by the editors April 20, 2004; accepted for publication (in revised form) May 19, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-6/60705.html
†Department of Mathematics, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China (Cjzhang@mail.hust.edu.cn). This author's research was supported by Fellowship F/02/019 of the Research Council of the K. U. Leuven and by NSFC (No. 10571066).
‡Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium (Stefan.Vandewalle@cs.kuleuven.be).

and backward differentiation (BDF) methods, respectively. For a study of MVIDEs of neutral type we refer the reader to [12], where the concept of BN_f-stability is used as the basic assumption. For adaptations of RK methods towards general Volterra functional differential equations, we would like to point out the recent paper [25]. There, stability results have been derived based on B-stability of the underlying schemes. These results are applicable for time-integration on a finite time interval. Here, we will study the nonlinear stability of numerical schemes for MVIDEs based on the use of general linear (GL) methods. Our stability results will be applicable on infinite time intervals.

This paper is organized as follows. In section 2, a fairly general class of MVIDEs is defined. This class contains the problems considered in [33, 34]. We present a stability criterion for such problems, which generalizes the criteria in the above references. In section 3, a class of extended GL methods is derived for solving MVIDEs. They are obtained by combining classical GL methods with compound quadrature rules. In section 4, some technical lemmas are derived. These lemmas will play a key role in the derivation of the numerical stability results. In sections 5 and 6, we study the global and asymptotic stability of the extended GL methods. In section 7 we present some numerical examples in order to illustrate the convergence of the methods. These results show that the new methods are quite effective. Finally, in section 8, we end with some concluding remarks.

2. A class of MVIDEs and its stability.

2.1. A general stability result. We consider a complex N-dimensional system of MVIDEs with constant delay $\tau > 0$ of the form

$$\begin{cases} y'(t) = f(t, y(t), y(t-\tau), \int_{t-\tau}^{t} g(t, v, y(v))dv), & t \in [t_0, +\infty), \\ y(t) = \varphi(t), & t \in [t_0 - \tau, t_0], \end{cases} \tag{2.1}$$

where the functions f, g, and φ are smooth enough such that system (2.1) has a unique solution $y(t)$. We assume in particular that the following conditions are satisfied:

$$\Re\langle f(t, x, y, z) - f(t, \tilde{x}, \tilde{y}, \tilde{z}), x - \tilde{x}\rangle \le \alpha\|x - \tilde{x}\|^2 + \beta\|y - \tilde{y}\|^2 + \sigma\|z - \tilde{z}\|^2, \tag{2.2}$$

$$\|g(t, v, x) - g(t, v, \tilde{x})\| \le \gamma\|x - \tilde{x}\|, \quad (t, v) \in \mathbb{D}, \tag{2.3}$$

for $t \in [t_0, +\infty)$, $\mathbb{D} = \{(t, v) : t \in [t_0, +\infty),\ v \in [t-\tau, t]\}$, and $x, y, z, \tilde{x}, \tilde{y}, \tilde{z} \in \mathbb{C}^N$. The notations $\langle\cdot,\cdot\rangle$ and $\|\cdot\|$ denote a given inner product in $\mathbb{C}^N$ and its induced norm. The constants $(-\alpha)$, β, σ, and γ are nonnegative. Problems of type (2.1) with (2.2) and (2.3) will be called *problems of class* $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$. Some examples will be given in section 7. For problems of class $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$, we can derive the following stability result.

THEOREM 2.1. *Assume that system* (2.1) *belongs to the class* $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ *with* $\beta + \sigma\gamma^2\tau^2 < -\alpha$. *Then the following global and asymptotic stability properties hold:*

$$\|y(t) - \tilde{y}(t)\| \le \max_{\theta \in [t_0 - \tau,\, t_0]} \|\varphi(\theta) - \psi(\theta)\| \quad \forall t \ge t_0, \tag{2.4}$$

$$\lim_{t \to +\infty} \|y(t) - \tilde{y}(t)\| = 0, \tag{2.5}$$

where $\tilde{y}(t)$ *is the solution of* (2.1) *with* $\varphi(t)$ *replaced by* $\psi(t)$.

Proof. Denote by $\Delta y(t)$ the difference $y(t) - \tilde{y}(t)$. Using (2.2) and (2.3), we find

$$\begin{aligned}\frac{d}{dt}(\|\Delta y(t)\|^2) &= 2\Re\langle \Delta y(t), (\Delta y(t))'\rangle \\ &\le \alpha\|\Delta y(t)\|^2 + \beta\|\Delta y(t-\tau)\|^2 + \sigma\left\|\int_{t-\tau}^{t}[g(t,v,y(v)) - g(t,v,\tilde{y}(v))]dt\right\|^2 \\ &\le \alpha\|\Delta y(t)\|^2 + \beta\|\Delta y(t-\tau)\|^2 + \sigma\gamma^2\tau^2 \max_{t-\tau\le v\le t}\|\Delta y(v)\|^2 \\ &\le \alpha\|\Delta y(t)\|^2 + (\beta + \sigma\gamma^2\tau^2)\max_{t-\tau\le v\le t}\|\Delta y(v)\|^2.\end{aligned}$$

Application of the generalized Halanay inequality from [4] leads to

$$\|\Delta y(t)\|^2 \le \max_{\theta\in[t_0-\tau,\,t_0]}\|\varphi(\theta)-\psi(\theta)\|^2 \ \ \forall t \ge t_0 \quad \text{and} \quad \lim_{t\to+\infty}\|\Delta y(t)\|^2 = 0.$$

Thus, (2.4) and (2.5) are satisfied. ∎

REMARK 2.2. *The* $\mathbb{GRI}$ *class includes the* $\mathbb{DI}$ *and* $\mathbb{RI}$ *problem classes that were investigated in* [33, 34]. *The relation among these classes is as follows:*

$$\mathbb{DI}(\alpha,\beta,(\sigma_1,\sigma_2),\gamma) \subset \mathbb{RI}(\alpha,\beta\sigma_1,\beta\sigma_2,\gamma) \subset \mathbb{GRI}(\alpha+\beta(\sigma_1+\sigma_2)/2,\beta\sigma_1/2,\beta\sigma_2/2,\gamma).$$

The research in the framework of class $\mathbb{GRI}$ *extends the results from* [33, 34].

REMARK 2.3. *When the distributed delay argument is absent from* (2.1), *the problem becomes a DDE. In that case, Theorem* 2.1 *can be applied with* $\gamma = \sigma = 0$, *which leads to the stability condition* $\beta < -\alpha$.

2.2. Relation to earlier work. The result from Remark 2.3 can be related to earlier results on analytical stability of DDEs. In [28, 29], Reverdy dealt respectively with linear and nonlinear DDEs of class $\mathbb{GRI}(-\rho,\rho,0,0)$, with $\rho > 0$. This class contains, for example, the linear DDE

$$\begin{cases} y'(t) = py(t) + qy(t-\tau), & t \in [t_0,+\infty), \\ y(t) = \varphi(t), & t \in [t_0-\tau, t_0], \end{cases} \tag{2.6}$$

with $\Re(p) \le -|q|$, by setting $\rho = -\Re(p)/2$. It was proved for $\mathbb{GRI}(-\rho,\rho,0,0)$ that

$$\mathcal{N}[y(t)-\tilde{y}(t)] \le \mathcal{N}[y(t_0)-\tilde{y}(t_0)] \quad \forall t \ge t_0, \tag{2.7}$$

where $\mathcal{N}[y(t)] = \frac{1}{2}\|y(t)\|^2 + \rho\int_{t-\tau}^{t}\|y(\theta)\|^2 d\theta$. Note that stability property (2.7) follows from our somewhat stronger result (2.4).

In [30] Torelli considered DDEs with a variable delay $\tau(t)$:

$$\begin{cases} y'(t) = f(t,y(t),y(t-\tau(t))), & t \in [t_0,+\infty), \\ y(t) = \varphi(t), & t \in \left[\inf\limits_{t\ge t_0}\{t-\tau(t)\}, t_0\right], \end{cases} \tag{2.8}$$

with $\beta(t) \le -\alpha(t)$ and $\alpha(t) < 0$ for $t \ge t_0$, where $\alpha(t)$ and $\beta(t)$ are defined as

$$\alpha(t) = \sup_{x,\tilde{x},y,\in\mathbb{C}^n,\, x\ne\tilde{x}} \Re\langle f(t,x,y) - f(t,\tilde{x},y), x-\tilde{x}\rangle/\|x-\tilde{x}\|^2, \tag{2.9}$$

$$\beta(t) = \sup_{x,y,\tilde{y}\in\mathbb{C}^n,\, y\ne\tilde{y}} \|f(t,x,y) - f(t,x,\tilde{y})\|/\|y-\tilde{y}\|. \tag{2.10}$$

In that paper it was proven that (2.8) satisfies the global stability property (2.4).

System (2.8) was analyzed by Zennaro in [32] under a different set of conditions:

(I) there exists a constant τ_0 such that $\tau(t) \geq \tau_0$ for all $t \geq t_0$;
(II) the delayed argument $t - \tau(t)$ is a strictly increasing function for all $t \geq t_0$;
(III) $\lim_{t\to+\infty} t - \tau(t) = +\infty$;
(IV) there are two continuous functions, $\hat{\alpha}(t)$ and $\hat{\beta}(t)$, and two real numbers, $0 \leq r < 1$ and $\alpha_0 < 0$, such that $\alpha(t) \leq \hat{\alpha}(t) \leq \alpha_0$ and $\beta(t) \leq \hat{\beta}(t) \leq -r\hat{\alpha}(t)$ for $t \geq t_0$.

It was proven in [32] that, under those conditions, system (2.8) satisfies the asymptotic stability (2.5).

The above results by Torelli and by Zennaro can be applied to the constant delay case, i.e., with $\tau(t) \equiv \tau$. Assume there exists an α and β such that $\alpha := \sup_{t\geq t_0} \alpha(t) < 0$ and $\beta := \sup_{t\geq t_0} \beta(t) < -\alpha$. Then, system (2.8) satisfies the stability properties (2.4) and (2.5) by the results from [30] and [32], respectively. This conclusion, however, is just the special case from our results, as expressed in Remark 2.3.

3. The extended GL methods. The numerical methods we suggest for (2.1) will be based on GL methods for ODEs of the form $y'(t) = f(t, y(t))$ with $y(t_0) = y_0$; see [13, 15, 16]. A GL method can be formulated as follows:

$$\text{(3.1)} \qquad \begin{cases} Y_i^{(n)} = h\displaystyle\sum_{j=1}^{s} c_{ij}^{(11)} f(t_n + c_j h, Y_j^{(n)}) + \sum_{j=1}^{r} c_{ij}^{(12)} y_j^{(n-1)}, \quad i = 1, 2, \ldots, s, \\ y_i^{(n)} = h\displaystyle\sum_{j=1}^{s} c_{ij}^{(21)} f(t_n + c_j h, Y_j^{(n)}) + \sum_{j=1}^{r} c_{ij}^{(22)} y_j^{(n-1)}, \quad i = 1, 2, \ldots, r, \end{cases}$$

with real coefficients $c_{ij}^{(IJ)}$ for $I, J = 1, 2$, with a step size $h > 0$, and with $t_n = t_0 + nh$ for $n \geq 0$. The vector $Y_i^{(n)}$ approximates $y(t_n + c_i h)$. The precise nature of vector $y_i^{(n)}$ differs from method to method; generally, it contains the necessary information for doing the next time-integration step. For use in the subsequent analysis, we introduce the coefficient matrices $C_{IJ} = (c_{ij}^{(IJ)})$ and the abscissa vector $c = (c_1, c_2, \ldots, c_s)^T$.

An important subclass of the GL methods, which will be considered further on, is the multistep Runge–Kutta (MRK) methods (see, e.g., [14, 24]):

$$\text{(3.2)} \qquad \begin{cases} Y_i^{(n)} = h\displaystyle\sum_{j=1}^{s} a_{ij} f(t_n + c_j h, Y_j^{(n)}) + \sum_{j=1}^{r} \hat{a}_{ij} y_{n+j-1}, \quad i = 1, 2, \ldots, s, \\ y_{n+r} = h\displaystyle\sum_{j=1}^{s} b_j f(t_n + c_j h, Y_j^{(n)}) + \sum_{j=1}^{r} \hat{b}_j y_{n+j-1}. \end{cases}$$

These methods can be written in the form (3.1) when setting $y_{n+i} = y_i^{(n)}$. With $b = (b_1, b_2, \ldots, b_s)^T$ and $\hat{b} = (\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_r)^T$, we can identify the coefficient matrices:

$$C_{11} = (a_{ij}), \quad C_{12} = (\hat{a}_{ij}), \quad C_{21} = \begin{bmatrix} 0 \\ b^T \end{bmatrix}, \quad C_{22} = \begin{bmatrix} 0 & I_{r-1} \\ & \hat{b}^T \end{bmatrix}.$$

For the formulation of other numerical methods as a GL method, we refer the reader to [16].

First, we recall a number of elementary concepts of GL methods that will be important for our stability analysis.

DEFINITION 3.1. *Let* $R(z) = C_{22} + zC_{21}(I_s - zC_{11})^{-1}C_{12}$, *where* $z \in \mathbb{C}$ *and where* I_s *denotes the* $s \times s$ *identity matrix. A GL method is called strictly stable at infinity if* $R(\infty) := \lim_{z\to\infty} R(z)$ *exists and if its spectral radius satisfies* $\rho(R(\infty)) < 1$.

DEFINITION 3.2 (cf. [13]). *A GL method is called* (k, l)*-algebraically stable if there exist real constants* k *and* l, *a symmetric positive-definite matrix* $G = (g_{ij}) \in \mathbb{R}^{r\times r}$, *and a nonnegative diagonal matrix* $D = \mathrm{diag}(d_1, d_2, \ldots, d_s) \in \mathbb{R}^{s\times s}$ *such that matrix* $M = (m_{ij}) \in \mathbb{R}^{(r+s)\times(r+s)}$ *is nonnegative definite, with*

$$M = \begin{bmatrix} kG - C_{22}^T GC_{22} - 2lC_{12}^T GC_{12} & C_{12}^T D - C_{22}^T GC_{21} - 2lC_{12}^T DC_{11} \\ DC_{12} - C_{21}^T GC_{22} - 2lC_{11}^T DC_{12} & DC_{11} + C_{11}^T D - C_{21}^T GC_{21} - 2lC_{11}^T DC_{11} \end{bmatrix}.$$

In particular, a $(1, 0)$*-algebraically stable method is called algebraically stable.*

Adapting method (3.1) to MVIDE (2.1) and considering the case where the step size $h = \tau/m$ for m a given positive integer yield the following numerical scheme:

(3.3)
$$\begin{cases} Y_i^{(n)} = h\displaystyle\sum_{j=1}^{s} c_{ij}^{(11)} f(t_n + c_j h, Y_j^{(n)}, Y_j^{(n-m)}, Z_j^{(n)}) + \sum_{j=1}^{r} c_{ij}^{(12)} y_j^{(n-1)}, \quad i = 1, 2, \ldots, s, \\ y_i^{(n)} = h\displaystyle\sum_{j=1}^{s} c_{ij}^{(21)} f(t_n + c_j h, Y_j^{(n)}, Y_j^{(n-m)}, Z_j^{(n)}) + \sum_{j=1}^{r} c_{ij}^{(22)} y_j^{(n-1)}, \quad i = 1, 2, \ldots, r. \end{cases}$$

$Z_j^{(n)}$ is an approximation to

$$Z(t_n + c_j h) := \int_{t_{n-m}+c_j h}^{t_n+c_j h} g(t_n + c_j h, v, y(v))dv$$

and is computed by a convergent, compound quadrature formula

$$Z_j^{(n)} = h\sum_{q=0}^{m} \nu_q g(t_n + c_j h, t_{n-q} + c_j h, Y_j^{(n-q)}), \quad j = 1, 2, \ldots, s. \tag{3.4}$$

Such a quadrature formula can be derived from a uniform repeated rule (cf. [3, 9, 33]). For our stability analysis we need the rule to satisfy the following condition:

$$h\sqrt{(m+1)\sum_{q=0}^{m} |\nu_q|^2} < \nu \quad \text{with } mh = \tau \text{ and a positive constant } \nu. \tag{3.5}$$

This condition holds for many of the common quadrature rules. For example, the compound trapezoidal (CT) rule of second order,

$$\int_0^\tau \Phi(s)ds \cong h\left[\frac{1}{2}\Phi(0) + \sum_{q=1}^{m-1} \Phi((m-q)h) + \frac{1}{2}\Phi(mh)\right], \quad \text{where } mh = \tau,$$

satisfies

$$h\sqrt{(m+1)\sum_{q=0}^{m} |\nu_q|^2} = h\sqrt{(m+1)[(1/2)^2 + m - 1 + (1/2)^2]} \le h(m+1) \le 2mh = 2\tau.$$

The compound Gregory (CG) rule of order 3 (cf. [9]),

$$\int_0^\tau \Phi(s)ds \cong \frac{h}{12}\left[5\Phi(0) + 13\Phi(h) + 12\sum_{q=2}^{m-2} \Phi((m-q)h) + 13\Phi((m-1)h) + 5\Phi(mh)\right],$$

TABLE 1
Value for the bound ν in (3.5) for different compound quadrature rules.

	CT rule	CG rule	CS rule	CN rule
ν	$2\,\tau$	$2\,\tau$	$\sqrt{20}\,\tau/3$	$2\sqrt{1194}\,\tau/45$

with $mh = \tau$, satisfies

$$h\sqrt{(m+1)\sum_{q=0}^{m}|\nu_q|^2} = h\sqrt{(m+1)\left[\frac{5^2+13^2+(m-3)12^2+13^2+5^2}{12^2}\right]}$$
$$= h\sqrt{(m+1)\left(m-\frac{11}{36}\right)} \le h(m+1) \le 2mh = 2\tau.$$

The values for ν for the fourth order compound Simpson (CS) rule and for the sixth order compound Newton–Cotes (CN) rule have been derived in [33]; see Table 1.

We end this section with some more notational conventions. When the extended GL method is applied to system (2.1) with the initial function $\varphi(t)$ replaced by $\psi(t)$, the resulting numerical approximations will be denoted by $\tilde{Y}_j^{(n)}$, $\tilde{Z}_j^{(n)}$, and $\tilde{y}_i^{(n)}$. Also, when a time point t_n or $t_n + c_j h$ falls in the initial interval $[t_0 - \tau, t_0]$, we set the approximation at that point equal to the corresponding (known) true solution.

4. Some elementary lemmas. In this section, we will derive some technical lemmas, which are important for the derivation of the main results in the subsequent sections. First, we introduce some more notation:

$$\Delta y_j^{(n)} = y_j^{(n)} - \tilde{y}_j^{(n)}, \quad \Delta Y_j^{(n)} = Y_j^{(n)} - \tilde{Y}_j^{(n)}, \quad \Delta Z_j^{(n)} = Z_j^{(n)} - \tilde{Z}_j^{(n)},$$

$$\Delta f_j^{(n)} = f(t_j^{(n)}, Y_j^{(n)}, Y_j^{(n-m)}, Z_j^{(n)}) - f(t_j^{(n)}, \tilde{Y}_j^{(n)}, \tilde{Y}_j^{(n-m)}, \tilde{Z}_j^{(n)}),$$

$$\Delta y^{(n)} = \begin{pmatrix} \Delta y_1^{(n)} \\ \Delta y_2^{(n)} \\ \vdots \\ \Delta y_r^{(n)} \end{pmatrix}, \quad \Delta Y^{(n)} = \begin{pmatrix} \Delta Y_1^{(n)} \\ \Delta Y_2^{(n)} \\ \vdots \\ \Delta Y_s^{(n)} \end{pmatrix}, \quad \Delta f^{(n)} = \begin{pmatrix} \Delta f_1^{(n)} \\ \Delta f_2^{(n)} \\ \vdots \\ \Delta f_s^{(n)} \end{pmatrix}.$$

With (3.3) we can write the relation between the above quantities compactly as

$$\begin{cases} \Delta Y^{(n)} = h(C_{11} \otimes I_N)\Delta f^{(n)} + (C_{12} \otimes I_N)\Delta y^{(n-1)}, \\ \Delta y^{(n)} = h(C_{21} \otimes I_N)\Delta f^{(n)} + (C_{22} \otimes I_N)\Delta y^{(n-1)}, \end{cases} \tag{4.1}$$

where the symbol $\otimes$ denotes the Kronecker product. We will use the following inner product and norms on $\mathbb{C}^{Ns}$:

$$\langle U, V\rangle = \sum_{i=1}^{s}\langle u_i, v_i\rangle, \quad \|U\| = \sqrt{\langle U, U\rangle}, \quad \|U\|_G = \sqrt{\langle U, GU\rangle},$$

where vectors $U = (u_1^T, u_2^T, \ldots, u_s^T)^T$, $V = (v_1^T, v_2^T, \ldots, v_s^T)^T \in \mathbb{C}^{Ns}$, and $u_i, v_i \in \mathbb{C}^N$.

LEMMA 4.1. *Suppose that the GL method* (3.1) *is* (k,l)*-algebraically stable with* $0 < k \le \theta \le 1$. *Suppose that the quadrature formula* (3.4) *satisfies* (3.5) *and that the conditions* (2.2) *and* (2.3) *hold. Set* $d = \sum_{j=1}^{s} d_j$. *Then, method* (3.3) *satisfies*

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 \le \theta^n \|\Delta y^{(0)}\|_G^2 &+ 2\beta\tau\theta^{n-m} d \max_{-m\le i\le -1} \max_{1\le j\le s} \{\|\Delta Y_j^{(i)}\|^2\} \\ &+ 2[(h\alpha - l)\theta^m + h\beta] \sum_{i=0}^{n-1} \theta^{n-m-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i)}\|^2 \\ &+ \frac{2h\sigma(\nu\gamma)^2}{m+1} \sum_{i=0}^{n-1} \theta^{n-i-1} \sum_{j=1}^{s} \sum_{q=0}^{m} d_j \|\Delta Y_j^{(i-q)}\|^2, \quad n \ge 1. \end{aligned} \tag{4.2}$$

Proof. It follows from a fairly straightforward (but tedious) computation and (k,l)-algebraic stability that (compare also [13, 20])

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 - k\|\Delta y^{(n-1)}\|_G^2 - 2\sum_{j=1}^{s} d_j \Re\langle \Delta Y_j^{(n)}, h\Delta f_j^{(n)} - l\Delta Y_j^{(n)}\rangle &= -\sum_{i=1}^{r+s}\sum_{i=1}^{r+s} m_{ij}\langle \omega_i, \omega_j\rangle \\ &\le 0,\end{aligned}$$

where $M = (m_{ij})$, as defined in Definition 3.2, and

$$\omega_i = \begin{cases} y_i^{(n-1)}, & 1 \le i \le r, \\ h\Delta f_i^{(n)}, & r+1 \le i \le r+s. \end{cases}$$

This, together with the condition $0 < k \le \theta$, implies

$$\|\Delta y^{(n)}\|_G^2 \le \theta\|\Delta y^{(n-1)}\|_G^2 + 2\sum_{j=1}^{s} d_j \Re\langle \Delta Y_j^{(n)}, h\Delta f_j^{(n)} - l\Delta Y_j^{(n)}\rangle. \tag{4.3}$$

By condition (2.2) one has

$$\sum_{j=1}^{s} d_j \Re\langle \Delta Y_j^{(n)}, h\Delta f_j^{(n)}\rangle \le h\sum_{j=1}^{s} d_j[\alpha\|\Delta Y_j^{(n)}\|^2 + \beta\|\Delta Y_j^{(n-m)}\|^2 + \sigma\|\Delta Z_j^{(n)}\|^2].$$

When this bound is inserted into (4.3), one finds

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 \le \theta\|\Delta y^{(n-1)}\|_G^2 &+ 2(h\alpha - l)\sum_{j=1}^{s} d_j \|\Delta Y_j^{(n)}\|^2 \\ &+ 2h\beta\sum_{j=1}^{s} d_j \|\Delta Y_j^{(n-m)}\|^2 + 2h\sigma\sum_{j=1}^{s} d_j \|\Delta Z_j^{(n)}\|^2. \end{aligned} \tag{4.4}$$

An induction argument applied to (4.4) yields

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 \le \theta^n\|\Delta y^{(0)}\|_G^2 &+ 2(h\alpha - l)\sum_{i=0}^{n-1} \theta^{n-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i)}\|^2 \\ &+ 2h\beta\sum_{i=0}^{n-1} \theta^{n-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i-m)}\|^2 + 2h\sigma\sum_{i=0}^{n-1} \theta^{n-i-1} \sum_{j=1}^{s} d_j \|\Delta Z_j^{(i)}\|^2. \end{aligned} \tag{4.5}$$

With (2.3), (3.5), and the Cauchy inequality, we have

$$\|\Delta Z_j^{(i)}\|^2 \le (h\gamma)^2 \left(\sum_{q=0}^{m} |\nu_q| \|\Delta Y_j^{(i-q)}\|\right)^2 \le (h\gamma)^2 \left(\sum_{q=0}^{m} |\nu_q|^2\right) \left(\sum_{q=0}^{m} \|\Delta Y_j^{(i-q)}\|^2\right)$$

$$\le \frac{(\gamma\nu)^2}{m+1} \sum_{q=0}^{m} \|\Delta Y_j^{(i-q)}\|^2. \tag{4.6}$$

Moreover, we have

$$h \sum_{i=0}^{n-1} \theta^{n-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i-m)}\|^2$$

$$= h \sum_{i=0}^{n-m-1} \theta^{n-m-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i)}\|^2 + h \sum_{i=-m}^{-1} \theta^{n-m-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i)}\|^2$$

$$\le h \sum_{i=0}^{n-1} \theta^{n-m-i-1} \sum_{j=1}^{s} d_j \|\Delta Y_j^{(i)}\|^2 + \tau \theta^{n-m} d \max_{-m\le i\le -1} \max_{1\le j\le s} \{\|\Delta Y_j^{(i)}\|^2\}. \tag{4.7}$$

Substituting (4.6) and (4.7) into (4.5) results in (4.2). ∎

The second lemma gives an existence condition for the stability function $R(z)$ that is referred to in Definition 3.1.

LEMMA 4.2. *Suppose that a GL method is (k,l)-algebraically stable for a symmetric positive-definite matrix $G \in \mathbb{R}^{r\times r}$ and a positive diagonal matrix $D \in \mathbb{R}^{s\times s}$. Then the limit $R(\infty) := \lim_{z\to\infty} R(z)$ exists.*

Proof. It suffices to prove the existence of $C_0 := \lim_{\varepsilon\to 0} C_{21}(C_{11} + \varepsilon I_s)^{-1}$. This can be shown in a similar way as Lemma 3.8 of [20] is proven (compare also [18]), i.e., by considering the nonnegative definiteness of the bottom right-hand principal minor $DC_{11} + C_{11}^T D - C_{21}^T G C_{21} - 2l C_{11}^T D C_{11}$ of matrix M. ∎

REMARK 4.3. *In* [18, 20] *the positivity of the diagonal matrix D is guaranteed by requiring the underlying methods to be irreducible and algebraically stable. Here, we loosen these conditions by requesting positivity for D explicitly. This allows us to consider more general (k,l)-algebraically stable GL methods.*

REMARK 4.4. *When the conditions in Lemma* 4.2 *are satisfied, it holds that*

$$R(\infty) = \begin{cases} C_{22} - C_{21}C_{11}^{-1}C_{12} & \text{if } C_{11} \text{ is invertible,} \\ C_{22} - C_0 C_{12} & \text{if } C_{11} \text{ is singular.} \end{cases}$$

Two inequalities from [34] will also play important roles in the subsequent sections.

LEMMA 4.5. *Suppose that $\{A_i\}_{i=0}^n$ and $\{B_i\}_{i=-m}^n$ are two arbitrary nonnegative real sequences. Then the following two inequalities hold for all $n, m \ge 0$:*

$$\sum_{i=0}^{n} \left(A_i \sum_{j=0}^{m} B_{i-j}\right) \le \sum_{j=0}^{m}\sum_{i=0}^{n} A_{i+j}B_i + \left(\sum_{j=1}^{m}\sum_{i=1}^{j} A_{j-i}\right) \max_{-m\le q\le -1}\{B_q\}, \tag{4.8}$$

$$\sum_{i=0}^{n}\sum_{j=0}^{m} B_{i-j} \le (m+1)\sum_{i=0}^{n} B_i + \frac{m(m+1)}{2} \max_{-m\le q\le -1}\{B_q\}. \tag{4.9}$$

Finally, we mention an asymptotic property of vector difference equations that can be derived easily from a combination of Theorems 105B and 123D of [16].

LEMMA 4.6. *Given a matrix $\mathcal{A} \in \mathbb{C}^{Q\times Q}$ and a sequence $V_n \in \mathbb{C}^Q$, the solution sequence $\{X_n\} \subseteq \mathbb{C}^Q$ of the linear difference equation $X_n = \mathcal{A}X_{n-1} + V_n$ satisfies $\lim_{n\to\infty} \|X_n\| = 0$ if and only if $\rho(\mathcal{A}) < 1$ and $\lim_{n\to\infty} \|V_n\| = 0$.*

5. Global stability of the extended GL methods. Numerical stability is an important feature of an effective numerical method. An unstable numerical method may be consistent of high order, yet arbitrarily small perturbations will eventually cause large deviations from the true solution. In this section, we will focus on the global stability of the extended GL methods.

DEFINITION 5.1. *The extended GL method* (3.3) *and* (3.4) *is called globally stable for problems of class $\mathbb{GRI}(\alpha,\beta,\sigma,\gamma)$ if there exists a constant $\mathcal{H} > 0$, which depends only on $\alpha,\beta,\sigma,\gamma,\tau,\nu$, and the method, such that*

$$\|\Delta y^{(n)}\| \le \mathcal{H} \max\left\{\|\Delta y^{(0)}\|, \max_{-m\le i\le -1}\max_{1\le j\le s}\{\|\Delta Y_j^{(i)}\|\}\right\} \quad \forall n \ge 1. \tag{5.1}$$

REMARK 5.2. *For the extended one-step RK methods, we have $\Delta y^{(n)} = \Delta y_n := y_{n+1} - \tilde{y}_{n+1}$. Also, under the usual assumption that $0 \le c_i \le 1$, it holds that*

$$\max\left\{\|\Delta y_0\|, \max_{-m\le i\le -1}\max_{1\le j\le s}\{\|\Delta Y_j^{(i)}\|\}\right\} \le \max_{t_0-\tau\le t\le t_0}\|\varphi(t) - \psi(t)\|.$$

Hence, we have for (5.1) *the following equivalent formulation:*

$$\|\Delta y_n\| \le \mathcal{H} \max_{t_0-\tau\le t\le t_0}\|\varphi(t) - \psi(t)\|.$$

Thus, the concept of global stability in Definition 5.1 *is slightly stronger than that in* [34].

THEOREM 5.3. *Suppose that the GL method* (3.1) *is (k,l)-algebraically stable for a nonnegative diagonal matrix $D = \mathrm{diag}(d_1,d_2,\ldots,d_s) \in \mathbb{R}^{s\times s}$ and a symmetric positive-definite matrix $G = (g_{ij}) \in \mathbb{R}^{r\times r}$, where $0 < k \le 1$, and suppose the quadrature formula* (3.4) *satisfies condition* (3.5). *Then the extended GL method* (3.3) *and* (3.4) *is globally stable for the class $\mathbb{GRI}(\alpha,\beta,\sigma,\gamma)$ with stability constant*

$$\mathcal{H} = \sqrt{\frac{\lambda_{max}^G + d\tau(2\beta + \sigma\gamma^2\nu^2)}{\lambda_{min}^G}}, \tag{5.2}$$

where λ_{min}^G and λ_{max}^G denote the minimum and maximum eigenvalues of matrix G and $d = \sum_{j=1}^s d_j$, when the following condition holds:

$$h(\alpha + \beta + \sigma\gamma^2\nu^2) \le l. \tag{5.3}$$

Proof. Setting $\theta = 1$ in (4.2) gives

(5.4)

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 \le{}& \|\Delta y^{(0)}\|_G^2 + 2\beta\tau d \max_{-m\le i\le -1}\max_{1\le j\le s}\{\|\Delta Y_j^{(i)}\|^2\}\\ &+ 2[h(\alpha+\beta) - l]\sum_{i=0}^{n-1}\sum_{j=1}^{s} d_j\|\Delta Y_j^{(i)}\|^2 + \frac{2h\sigma(\nu\gamma)^2}{m+1}\sum_{i=0}^{n-1}\sum_{j=1}^{s}\sum_{q=0}^{m} d_j\|\Delta Y_j^{(i-q)}\|^2.\end{aligned}$$

Also, by virtue of inequality (4.9), we have for $n \geq 1$

(5.5)
$$\sum_{i=0}^{n-1}\sum_{q=0}^{m}\|\Delta Y_j^{(i-q)}\|^2 \leq (m+1)\sum_{i=0}^{n-1}\|\Delta Y_j^{(i)}\|^2 + \frac{m(m+1)}{2}\max_{-m\leq i\leq -1}\{\|\Delta Y_j^{(i)}\|^2\}.$$

Inserting (5.5) into (5.4) and using the condition that $mh = \tau$ yield

$$\begin{aligned}\|\Delta y^{(n)}\|_G^2 \leq \|\Delta y^{(0)}\|_G^2 &+ d\tau(2\beta + \sigma\gamma^2\nu^2)\max_{-m\leq i\leq -1}\max_{1\leq j\leq s}\{\|\Delta Y_j^{(i)}\|^2\}\\ &+ 2[h(\alpha+\beta+\sigma\gamma^2\nu^2) - l]\sum_{i=0}^{n-1}\sum_{j=1}^{s} d_j\|\Delta Y_j^{(i)}\|^2. \end{aligned} \tag{5.6}$$

With condition (5.3) this leads to

$$\|\Delta y^{(n)}\|_G^2 \leq \|\Delta y^{(0)}\|_G^2 + d\tau(2\beta + \sigma\gamma^2\nu^2)\max_{-m\leq i\leq -1}\max_{1\leq j\leq s}\{\|\Delta Y_j^{(i)}\|^2\}.$$

Considering that $\lambda_{min}^G\|\Delta y^{(n)}\|^2 \leq \|\Delta y^{(n)}\|_G^2$ and that $\|\Delta y^{(0)}\|_G^2 \leq \lambda_{max}^G\|\Delta y^{(0)}\|^2$, we arrive at stability inequality (5.1) with stability constant (5.2). □

Example 5.1. As an application of Theorem 5.3, we consider an extended GL method with $s = 1$ and $r = 2$, induced by the combination of the two-step BDF method and the CT rule:

$$\begin{cases} Y^{(n)} = \frac{2}{3}hF^{(n)} + \left[\left(\frac{4}{3}, -\frac{1}{3}\right)\otimes I_N\right] y^{(n-1)},\\ y^{(n)} = h\left[\begin{pmatrix}\frac{2}{3}\\ 0\end{pmatrix}\otimes I_N\right]F^{(n)} + \left[\begin{pmatrix}\frac{4}{3} & -\frac{1}{3}\\ 1 & 0\end{pmatrix}\otimes I_N\right] y^{(n-1)}, \end{cases} \tag{5.7}$$

where $Y^{(n)} = y_{n+2}$, $y_n \cong y(t_n)$, $y^{(n)} = (y_{n+2}^T, y_{n+1}^T)^T$, and

$$\begin{aligned} F^{(n)} &= f_{n+2} := f\left(t_{n+2}, Y^{(n)}, Y^{(n-m)}, Z^{(n)}\right),\\ Z^{(n)} &= h{\sum_{q=0}^{m}}'' g\left(t_{n+2}, t_{n+2-q}, Y^{(n-q)}\right), \end{aligned} \tag{5.8}$$

where the special summation symbol stands for a sum with the first and last term halved. By Table 1 we know that (5.8) satisfies (3.5) with $\nu = 2\tau$. In [15], it was shown that the two-step BDF method is $(1/(1-2l), l)$-algebraically stable with $l \leq 0$ for $D = 1$ and the 2×2 real symmetric positive-definite matrix

$$G := \begin{pmatrix}(5-2l)/2 & -1\\ -1 & 1/2\end{pmatrix}.$$

Note that $0 < k \leq 1$ whenever $l \leq 0$ and that there always exists an l such that (5.3) holds whenever $\beta + \sigma\gamma^2\nu^2 \leq -\alpha$. Hence, we may conclude that the method is globally stable for the class $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ whenever $\beta + \sigma\gamma^2\nu^2 \leq -\alpha$. For the parameters in the formula of the stability constant, the following values are used:

$$d = 1, \quad \nu = 2\tau, \quad \lambda_{min}^G = \frac{3 - l - \sqrt{(l-2)^2+4}}{2}, \quad \lambda_{max}^G = \frac{3 - l + \sqrt{(l-2)^2+4}}{2}.$$

TABLE 2
l-value minimizing (5.9) *and corresponding stability constant* $\mathcal{H}$ *of the extended two-step BDF method for the MVIDEs in* (7.1) *and* (7.2).

	MVIDE (7.1)		MVIDE (7.2)	
m	l	$\mathcal{H}$	l	$\mathcal{H}$
8	−0.2224	6.4650	−0.0554	7.1616
16	−0.1112	6.8497	−0.0277	7.2997
32	−0.0556	7.0912	−0.0138	7.3733
64	−0.0278	7.2279	−0.0069	7.4116

The range of l is given by $h(\alpha+\beta+\sigma\gamma^2\nu^2) \le l \le 0$. In particular, when the extended two-step BDF method with a fixed step size h is applied to a concrete problem of class $\mathbb{GRI}(\alpha,\beta,\sigma,\gamma)$, one should look for the value of l that minimizes the stability constant $\mathcal{H}$. This leads one to consider the following constrained minimization problem:

$$\begin{cases} \text{minimize} \quad \mathcal{H} := \sqrt{\frac{3-l+\sqrt{(l-2)^2+4}+4\tau(\beta+2\sigma\gamma^2\tau^2)}{3-l-\sqrt{(l-2)^2+4}}} \\ \text{subject to} \quad h(\alpha+\beta+\sigma\gamma^2\nu^2) \le l \le 0. \end{cases} \tag{5.9}$$

In section 7 results of numerical experiments will be reported for two concrete MVIDEs. The first of those, specified in (7.1), belongs to class $\mathbb{GRI}(-4, \frac{1}{2}, \frac{1}{2}, 1)$; the second one, specified in (7.2), belongs to class $\mathbb{GRI}(-4+\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}, 2)$. We have numerically solved problem (5.9) for these two examples. The l-value at which the minimum is obtained and the corresponding $\mathcal{H}$-value are given in Table 2 as a function of the parameter m, with $h = \tau/m$.

The method in the above example is also algebraically stable. For such methods, a general result can be derived immediately from Theorem 5.3.

COROLLARY 5.4. *Suppose that GL method* (3.1) *is algebraically stable for a nonnegative diagonal matrix* $D = \mathrm{diag}(d_1, d_2, \ldots, d_s) \in \mathbb{R}^{s\times s}$ *and a symmetric positive-definite matrix* $G = (g_{ij}) \in \mathbb{R}^{r\times r}$*, and suppose the quadrature formula* (3.4) *satisfies condition* (3.5)*. Then the extended GL method* (3.3) *and* (3.4) *is globally stable for the class* $\mathbb{GRI}(\alpha,\beta,\sigma,\gamma)$ *with stability constant* (5.2) *when the following condition holds:*

$$\beta + \sigma\gamma^2\nu^2 \le -\alpha. \tag{5.10}$$

In [14], Burrage has derived a class of s-stage MRK methods of order $2s$ of the form (3.2) that satisfy

$$\begin{cases} B(2s),\ C(s),\ E(s);\ c_i \ne c_j \ \text{ whenever } \ i \ne j; \\ \displaystyle\sum_{j=1}^{r} \hat{b}_j = 1,\ i = 1,2,\ldots,s;\ \hat{b}_1 > 0,\ \hat{b}_j \ge 0,\ j = 2,3,\ldots,r, \end{cases} \tag{5.11}$$

where $B(2s)$, $C(s)$, and $E(s)$ denote the usual order conditions. It was shown that these methods are algebraically stable for the matrices

$$G := \mathrm{diag}\left(\hat{b}_1, \hat{b}_1+\hat{b}_2, \ldots, \sum_{j=1}^{r}\hat{b}_j\right), \quad D := \mathrm{diag}(b_1, b_2, \ldots, b_s). \tag{5.12}$$

TABLE 3
Coefficients of the fourth order MRK method from Example 5.2.

a_{11}	a_{12}	a_{21}	a_{22}
0.47790690818421	0.87165188291653	−0.08663699023763	0.50361252124048
$\hat{a}_{11}$	$\hat{a}_{12}$	$\hat{a}_{21}$	$\hat{a}_{22}$
0.75576439912123	0.24423560087877	0.97380878183171	0.02619121816829
b_1	b_2	$\hat{b}_1$	$\hat{b}_2$
0.95532987568936	0.79063681672548	$2\sqrt{15}-7$	$8-2\sqrt{15}$
	c_1	c_2	
	1.59379439197950	0.44316674917114	

Obviously, those are positive-definite. Hence, (5.10) is sufficient as a condition for global stability of the corresponding extended GL method.

Example 5.2. By combining the fourth order MRK method from [14] with the CS quadrature rule of order 4, we can obtain an extended GL method with $r = 2$ and $s = 2$. It is written schematically below; for the values of the coefficients we refer the reader to Table 3.

$$\begin{cases} Y^{(n)} = h\left[\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \otimes I_N\right] F^{(n)} + \left[\begin{pmatrix} \hat{a}_{11} & \hat{a}_{12} \\ \hat{a}_{21} & \hat{a}_{22} \end{pmatrix} \otimes I_N\right] y^{(n-1)}, \\ y^{(n)} = h\left[\begin{pmatrix} 0 & 0 \\ b_1 & b_2 \end{pmatrix} \otimes I_N\right] F^{(n)} + \left[\begin{pmatrix} 0 & 1 \\ \hat{b}_1 & \hat{b}_2 \end{pmatrix} \otimes I_N\right] y^{(n-1)}, \end{cases} \tag{5.13}$$

where $Y^{(n)} = ({Y_1^{(n)}}^T, {Y_2^{(n)}}^T)^T$ and $y^{(n)} = ({y_{n+1}}^T, {y_{n+2}}^T)^T$, with $Y_i^{(n)} \cong y(t_n + c_i h)$ and $y_n \cong y(t_n)$, and

$$F^{(n)} = \left(f\left(t_n + c_1 h, Y_1^{(n)}, Y_1^{(n-m)}, Z_1^{(n)}\right)^T, f\left(t_n + c_2 h, Y_2^{(n)}, Y_2^{(n-m)}, Z_2^{(n)}\right)^T \right)^T.$$

$Z_j^{(n)}$ is computed by the CS rule with an even integer $m \geq 4$:

$$\begin{aligned} Z_j^{(n)} = \frac{h}{3}\Bigg[& g\left(t_{n,j}, t_{n,j}, Y_j^{(n)}\right) + 4\sum_{q=1}^{m/2} g\left(t_{nj}, t_{n-2q+1,j}, Y_j^{(n-2q+1)}\right) \\ & + 2\sum_{q=1}^{(m-2)/2} g\left(t_{n,j}, t_{n-2q,j}, Y_j^{(n-2q)}\right) + g\left(t_{n,j}, t_{n-m,j}, Y_j^{(n-m)}\right)\Bigg]. \end{aligned} \tag{5.14}$$

Here the notation $t_{n,j}$ is used as shorthand for $t_n + c_j h$. This rule satisfies (3.5) for $\nu = \frac{\sqrt{20}}{3}\tau$. Under condition (5.10) this extended MRK method is globally stable. The stability constant (5.2) can be computed using $d = 1.7460$, $\nu = \frac{\sqrt{20}}{3}\tau$, $\lambda_{min}^G = 2\sqrt{15} - 7$, and $\lambda_{max}^G = 1$.

In [24], the MRK methods that satisfy (5.11) are called *MRK methods of the first class*. Five more classes of MRK methods are identified by Li in the above reference. All of those are proven to be algebraically stable for the matrices (5.12). Hence, Corollary 5.4 is immediately applicable to the corresponding extended GL methods.

TABLE 4
Coefficients of the third order MRK method from Example 5.3.

a_{11}	a_{12}	a_{21}	a_{22}
0.41623635782678	0.43372361014760	−0.22963576074350	0.62694215804990
$\hat{a}_{11}$	$\hat{a}_{12}$	$\hat{a}_{21}$	$\hat{a}_{22}$
0.04995996797438	0.95004003202562	0.48484848484848	0.51515151515152
b_1	b_2	$\hat{b}_1$	$\hat{b}_2$
0.59227996965099	0.55772003034901	0.15	0.85
	c_1	c_2	
	1.80000000000000	0.91245791245791	

Example 5.3. We present an example of the second MRK class. Using formulas (3.11)–(3.16) of [24], we compute the coefficients for a two-stage third order MRK method. Its combination with the CG rule of order 3 (cf. [9]) yields another extended GL method of the form (5.13). The coefficients are given in Table 4. The $Z_j^{(n)}$-values are computed as follows:

$$\begin{aligned} Z_j^{(n)} = \frac{h}{12}\Bigg[& 5\, g(t_{n,j}, t_{n,j}, Y_j^{(n)}) + 13\, g(t_{n,j}, t_{n-1,j}, Y_j^{(n-1)}) \\ & + 12 \sum_{q=2}^{m-2} g(t_{n,j}, t_{n-q,j}, Y_j^{(n-q)}) + 13\, g(t_{n,j}, t_{n-m+1,j}, Y_j^{(n-m+1)}) \\ & + 5\, g(t_{n,j}, t_{n-m,j}, Y_j^{(n-m)}) \Bigg]. \end{aligned} \tag{5.15}$$

Since the corresponding underlying method is algebraically stable, and since the CG rule satisfies (3.5) with $\nu = 2\tau$, we immediately have global stability of the extended GL method under condition (5.11). The stability constant (5.2) can be computed using $d = 1.1500$, $\nu = 2\tau$, $\lambda_{min}^G = 0.15$, and $\lambda_{max}^G = 1$.

6. Asymptotic stability of the extended GL methods. In this section, we focus on the concept of asymptotic stability.

DEFINITION 6.1. *The extended GL method* (3.3) *and* (3.4) *is called asymptotically stable for problems of class* $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ *if* $\lim_{n\to\infty} \|\Delta y^{(n)}\| = 0$.

THEOREM 6.2. *Suppose that GL method* (3.1) *is* (k, l)*-algebraically stable for a nonnegative diagonal matrix* $D = \mathrm{diag}(d_1, d_2, \ldots, d_s) \in \mathbb{R}^{s\times s}$ *and a real symmetric positive-definite matrix* $G = (g_{ij}) \in \mathbb{R}^{r\times r}$*, where* $0 < k < 1$*, and suppose the quadrature formula* (3.4) *satisfies* (3.5)*. Then the extended GL method* (3.3) *and* (3.4) *is asymptotically stable for the class* $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ *when the following condition holds:*

$$h(\alpha + \beta + \sigma\gamma^2\nu^2) < l. \tag{6.1}$$

Proof. Define the quantity $\theta = \max\{k, [\frac{h(\beta+\sigma\gamma^2\nu^2)}{l-h\alpha}]^{\frac{1}{m}}\}$. From $0 < k < 1$ and (6.1) it follows that $0 < k \le \theta < 1$. Hence, bound (4.2) from Lemma 4.1 holds. We will simplify its right-hand side. By using (4.8), we can derive the following bound:

$$h\sum_{i=0}^{n-1}\theta^{-i}\sum_{q=0}^{m}\|\Delta Y_j^{(i-q)}\|^2$$
$$\le h\left[\sum_{q=0}^{m}\sum_{i=0}^{n-1}\theta^{-(i+q)}\|\Delta Y_j^{(i)}\|^2+\left(\sum_{q=1}^{m}\sum_{i=1}^{q}\theta^{-(q-i)}\right)\max_{-m\le\hat{q}\le-1}\{\|\Delta Y_j^{(\hat{q})}\|^2\}\right]$$
$$\le (m+1)h\sum_{i=0}^{n-1}\theta^{-(i+m)}\|\Delta Y_j^{(i)}\|^2+m\tau\theta^{1-m}\max_{-m\le i\le-1}\{\|\Delta Y_j^{(i)}\|^2\}. \tag{6.2}$$

Substituting (6.2) into the last term of the right-hand side of (4.2) yields

$$\|\Delta y^{(n)}\|_G^2\le\theta^n\|\Delta y^{(0)}\|_G^2+2\tau\theta^{n-m}d\left[\beta+\frac{m\sigma\gamma^2\nu^2}{m+1}\right]\max_{-m\le i\le-1}\max_{1\le j\le s}\{\|\Delta Y_j^{(i)}\|^2\}$$
$$+2[(h\alpha-l)\theta^m+h(\beta+\sigma\gamma^2\nu^2)]\sum_{i=0}^{n-1}\theta^{n-m-i-1}\sum_{j=1}^{s}d_j\|\Delta Y_j^{(i)}\|^2. \tag{6.3}$$

By the definition of θ and with the inequalities $h\alpha\le h(\alpha+\beta+\sigma\gamma^2\nu^2)<l$, one finds

$$(h\alpha-l)\theta^m+h(\beta+\sigma\gamma^2\nu^2)\le 0.$$

This inequality, together with the knowledge that $0<\theta<1$ and (6.3), leads to $\lim_{n\to\infty}\|\Delta y^{(n)}\|_G=0$ or $\lim_{n\to\infty}\|\Delta y^{(n)}\|=0$. ∎

REMARK 6.3. *The major difference between Theorems* 6.2 *and* 5.3 *lies in the strict inequalities present in both* $k<1$ *and* (6.1).

Example 6.1. As an illustration we consider the method (5.7) and (5.8) again. Recall that (5.8) satisfies (3.5) with $\nu=2\tau$ and that the underlying GL method (5.7) is $(1/(1-2l),l)$-algebraically stable for any $l\le 0$. Thus, we have $0<k<1$ whenever $l<0$. Moreover, there always exists an $l\,(<0)$ such that $h(\alpha+\beta+\sigma\gamma^2\nu^2)<l$ whenever $\beta+\sigma\gamma^2\nu^2<-\alpha$. Hence, the latter is sufficient as a condition for asymptotic stability.

From Theorem 6.2, it is not immediately possible to derive a corollary that applies specifically for algebraically stable methods because of the condition $k<1$. To remedy this situation, we present an alternative approach for proving asymptotic stability, where the parameter k will be allowed to take the value 1.

THEOREM 6.4. *Suppose that the GL method* (3.1) *is strictly stable at infinity and* (k,l)*-algebraically stable for a positive diagonal matrix* $D=\operatorname{diag}(d_1,d_2,\dots,d_s)\in\mathbb{R}^{s\times s}$ *and a real symmetric positive-definite matrix* $G=(g_{ij})\in\mathbb{R}^{r\times r}$*, where* $0<k\le 1$*. Suppose the quadrature formula* (3.4) *satisfies* (3.5)*. Then the extended GL method* (3.3) *and* (3.4) *is asymptotically stable for the class* $\mathbb{GRI}(\alpha,\beta,\sigma,\gamma)$ *when* (6.1) *holds.*

Proof. Inequality (5.6) can be written as

$$\|\Delta y^{(n)}\|_G^2+2[l-h(\alpha+\beta+\sigma\gamma^2\nu^2)]\sum_{i=0}^{n-1}\sum_{j=1}^{s}d_j\|\Delta Y_j^{(i)}\|^2$$
$$\le\|\Delta y^{(0)}\|_G^2+d\tau(2\beta+\sigma\gamma^2\nu^2)\max_{-m\le i\le-1}\max_{1\le j\le s}\{\|\Delta Y_j^{(i)}\|^2\}. \tag{6.4}$$

Since D is a positive diagonal matrix and since (6.1) holds, we may conclude from (6.4) that $\lim_{n\to\infty}\|\Delta Y_j^{(n)}\|^2=0$ for $j=1,2,\dots,s$, and thus

$$\lim_{n\to\infty}\|\Delta Y^{(n)}\|=\lim_{n\to\infty}\sqrt{\sum_{j=1}^{s}\|\Delta Y_j^{(n)}\|^2}=0. \tag{6.5}$$

The remainder of the proof is divided into two parts. First, we consider the case where matrix C_{11} is invertible. Then, by (4.1), we have

$$\Delta y^{(n)} = [R(\infty) \otimes I_N]\Delta y^{(n-1)} + [C_{21}C_{11}^{-1} \otimes I_N]\Delta Y^{(n)}, \tag{6.6}$$

where $R(\infty) = C_{22} - C_{21}C_{11}^{-1}C_{12}$. Because of the strict stability at infinity, one has

$$\rho[R(\infty) \otimes I_N] = \rho[R(\infty)] < 1. \tag{6.7}$$

Therefore, applying Lemma 4.6 to (6.6) yields $\lim_{n\to\infty} \|\Delta y^{(n)}\| = 0$.

Next, we consider the case that matrix C_{11} is singular. We replace it in (6.6) by the invertible matrix $(C_{11} + \varepsilon I_s)$ and let $\varepsilon \to 0$. This gives

$$\Delta y^{(n)} = [R(\infty) \otimes I_N]\Delta y^{(n-1)} + (C_0 \otimes I_N)\Delta Y^{(n)}, \tag{6.8}$$

where $R(\infty) = C_{22} - C_0C_{12}$ and $C_0 = \lim_{\varepsilon\to 0} C_{21}(C_{11} + \varepsilon I_s)^{-1}$, whose existence is ensured by Lemma 4.2. Another application of Lemma 4.6 completes the proof. ∎

When the GL method satisfies algebraic stability, Theorem 6.4 can be simplified.

COROLLARY 6.5. *Suppose that the GL method* (3.1) *is strictly stable at infinity and algebraically stable for a positive diagonal matrix* $D = \mathrm{diag}(d_1, d_2, \ldots, d_s) \in \mathbb{R}^{s\times s}$ *and a real symmetric positive-definite matrix* $G = (g_{ij}) \in \mathbb{R}^{r\times r}$*, and suppose the quadrature formula* (3.4) *satisfies* (3.5)*. Then the extended GL method* (3.3) *and* (3.4) *is asymptotically stable for the class* $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ *when the following condition holds:*

$$\beta + \sigma\gamma^2\nu^2 < -\alpha. \tag{6.9}$$

Example 6.2. Consider the extended GL method from Example 5.2. A computation shows that $\rho[R(\infty)] \cong 0.9816 < 1$, which implies strict stability. So, the method is asymptotically stable for problem class $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ under condition (6.9).

Example 6.3. Consider the extended GL method from Example 5.3. Now, we find that $\rho[R(\infty)] \cong 0.6158 < 1$. Hence, also for this method (6.9) is a sufficient condition to guarantee asymptotic stability.

7. Numerical experiments and convergence. In the previous sections we have proven that the extended GL methods possess an excellent stability behavior under a set of suitable conditions. Naturally, one also wishes to find out the actual computational performance and their accuracy in particular. Therefore, we will apply the methods from Examples 5.1, 5.2, and 5.3 to two MVIDE systems. In order to measure the quality of the numerical methods, we introduce the following:

$$err := \|y(t_n) - y_n\|_\infty, \quad \varepsilon := \max_{0\le n\le N_f} \|y(t_n) - y_n\|_\infty, \quad p := \ln(\varepsilon)\,/\,\ln(h),$$

where N_f is a positive integer such that $N_f h = T$ with T being the length of the solution interval. The symbols *err*, ε, and p denote the error at time point t_n, the maximum error over the interval, and the convergence order of the method, respectively.

We will pay particular attention to the estimate of the convergence order. Our numerical results will seem to indicate that the convergence order of the extended GL method equals the minimum of the orders of the underlying GL method and the quadrature rule. These results are not entirely unexpected (see, e.g., the discussion of the combined order problem in [9, section 3.4]), and will be verified through a number

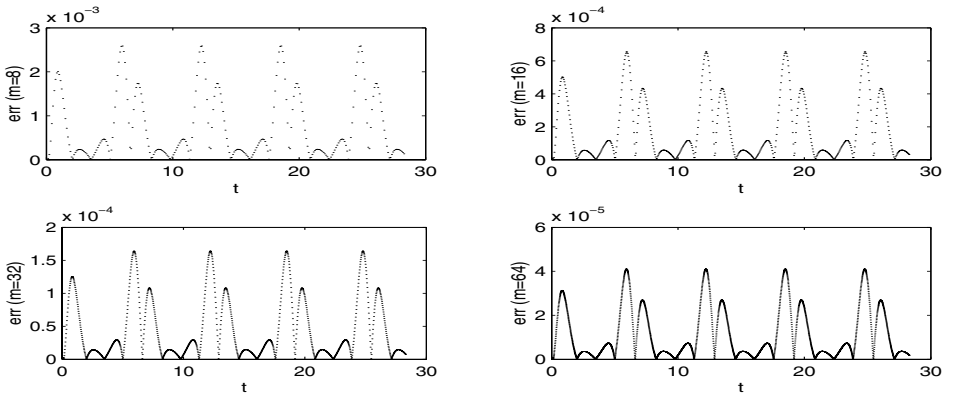


FIG. 7.1. *Error of the method from Example* 5.1 *with* $m = 8, 16, 32, 64$ *for* (7.1) *on* $[0, 9\pi]$.

of examples. In the first two examples, the methods of Examples 5.1, 5.2, and 5.3 will be used. Those are characterized by a correct match between the orders of the GL method and the quadrature rule (order 2 for Example 5.1, order 3 for Example 5.3, and order 4 for Example 5.2). Further on, we will also consider methods where the order of the GL method is different from that of the quadrature rule.

Example 7.1. Consider the linear system with partially variable coefficients:

$$y'(t) = -(6 + \sin t)y(t) + y\left(t - \frac{\pi}{4}\right) - \int_{t-\frac{\pi}{4}}^{t} \sin(v)y(v)dv + 5\exp(\cos t), \ t \geq 0, \tag{7.1}$$

with an initial condition on $[-\pi/4, 0]$ such that the exact solution is given by $y(t) = \exp(\cos t)$. One may check that the problem belongs to class $\mathbb{GRI}(-4, \frac{1}{2}, \frac{1}{2}, 1)$. With Theorem 2.1 we conclude that system (7.1) satisfies stability properties (2.4) and (2.5). Moreover, based on our earlier discussions, we deduce that the numerical methods from Examples 5.1, 5.2, and 5.3 retain the stability properties of the analytical solution; for this problem they are also globally and asymptotically stable.

In order to check the convergence behavior and accuracy of the extended GL methods, they will be applied using a sequence of step lengths characterized by the parameter m on the time interval $[0, 9\pi]$. The accuracy of the obtained numerical solutions is displayed in Figures 7.1–7.3. The numerically estimated convergence orders are given in Table 5. The numerical results show that the methods are quite effective and preserve the inherent order of accuracy of the underlying components.

Example 7.2. Consider the following two-dimensional nonlinear system:

$$\begin{aligned} \frac{d}{dt}\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} &= -4\begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} + \begin{pmatrix} 0 & \sin t \\ \cos t & 0 \end{pmatrix}\begin{pmatrix} y_1(t-\frac{\pi}{5}) \\ y_2(t-\frac{\pi}{5}) \end{pmatrix} \\ &\quad + \frac{1}{\sqrt{2}}\int_{t-\frac{\pi}{5}}^{t} \begin{pmatrix} \frac{(1+\sin^2 v)y_1^2(v)}{1+y_1^2(v)} \\ \frac{(1+\cos^2 v)y_2^2(v)}{1+y_2^2(v)} \end{pmatrix} dv + \begin{pmatrix} f(t) \\ g(t) \end{pmatrix}, \quad 0 \leq t. \end{aligned} \tag{7.2}$$

Functions $f(t)$ and $g(t)$ and the initial condition on $[-\pi/5, 0]$ are chosen in such a way that the solution equals $y(t) = (\sin t, \cos t)^T$. This problem belongs to class $\mathbb{GRI}(-4 + \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{4}, 2)$. It follows that (7.2) satisfies stability properties (2.4) and

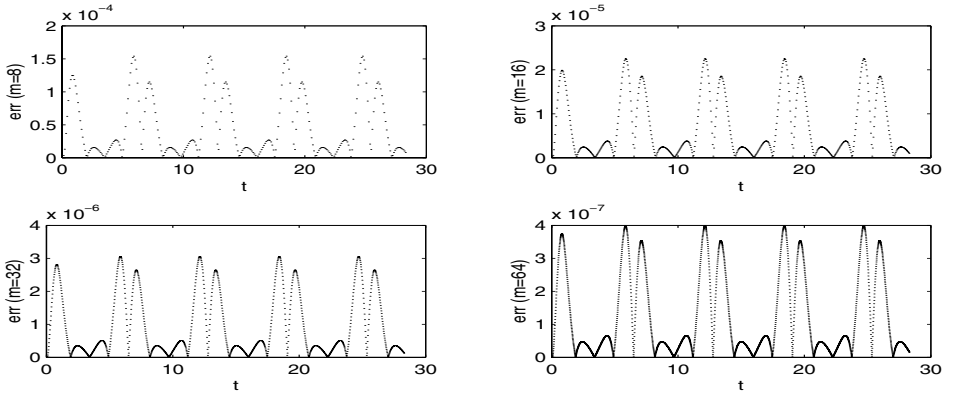


FIG. 7.2. *Error of the method from Example* 5.3 *with* $m = 8, 16, 32, 64$ *for* (7.1) *on* $[0, 9\pi]$.

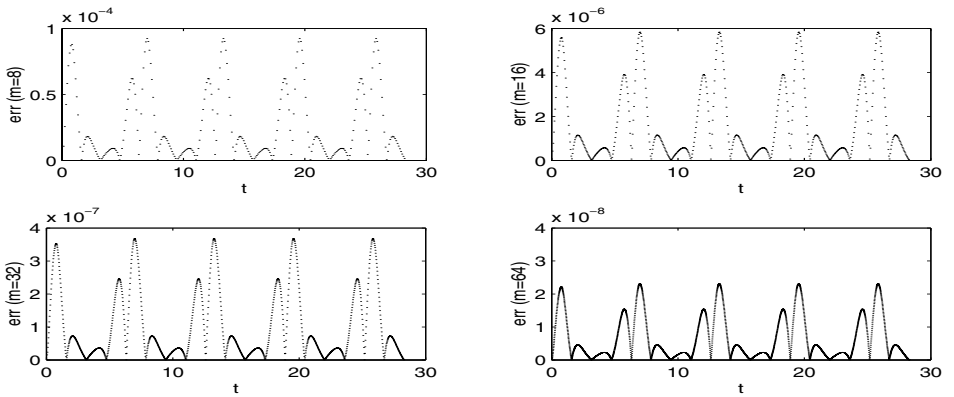


FIG. 7.3. *Error of the method from Example* 5.2 *with* $m = 8, 16, 32, 64$ *for* (7.1) *on* $[0, 9\pi]$.

TABLE 5

Convergence order of extended GL methods for system (7.1). *Example* 5.1*: 2nd order BDF + 2nd order quadrature; Example* 5.3*: 3rd order MRK + 3rd order quadrature; Example* 5.2*: 4th order MRK + 4th order quadrature.*

m	Example 5.1	Example 5.3	Example 5.2
8	2.5666	3.7821	4.0022
16	2.4327	3.5499	3.9982
32	2.3510	3.4254	3.9964
64	2.2956	3.3491	3.9960

(2.5) and that our three GL methods are globally and asymptotically stable. We consider the numerical results obtained when these methods are applied to (7.2). Note that a nonlinear equation is to be solved in every time-step. To that end, we adopted a Newton iteration, with initial values obtained by a backward interpolation formula

$$\hat{Y}_i^{(n)} = y_{n+1} + \sum_{j=1}^{Q} \frac{1}{j!} (c_i - 1)c_i(c_i + 1)(c_i + 2) \cdots (c_i + j - 2)\nabla^j y_{n+1}, \quad i = 1, 2, \ldots,$$

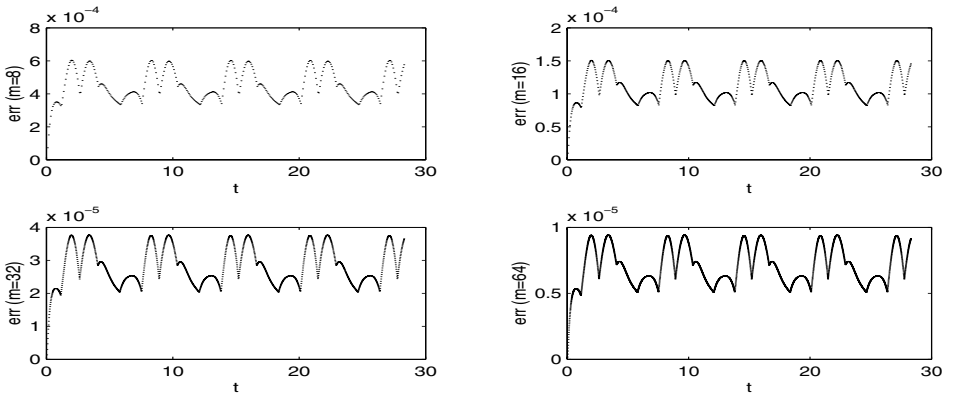


FIG. 7.4. *Error of the method from Example* 5.1 *with* $m = 8, 16, 32, 64$ *for* (7.2) *on* $[0, 9\pi]$.

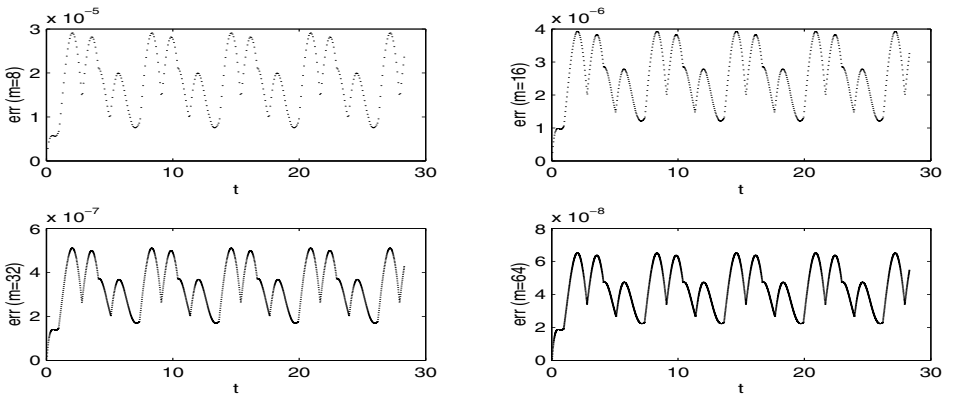


FIG. 7.5. *Error of the method from Example* 5.3 *with* $m = 8, 16, 32, 64$ *for* (7.2) *on* $[0, 9\pi]$.

where $\hat{Y}_i^{(n)} \cong y(t_n + c_i h)$ $(= y(t_{n+1} + (c_i - 1)h))$, $0 < c_i \leq 2$. More precisely, in order to get a starting value with accuracy order matching the estimated order of the extended GL methods, we take $Q = 2$, 3, 4 for the methods of Example 5.1, 5.3, and 5.2, respectively. This proved to provide a great savings in computational cost. The accuracy of the numerical solutions are shown in Figures 7.4–7.6, and the estimated convergence orders are given in Table 6. As in the previous example, the methods again seem to preserve the inherent accuracy of the underlying components.

Example 7.3. Naturally, one also wishes to know what happens to an extended GL method when the order of the underlying method differs from that of the quadrature formula. In order to gain some insight, we considered the third order MRK method from Example 5.3 combined with the first order quadrature formula

$$Z_j^{(n)} = h \sum_{q=0}^{m} g(t_{n,j}, t_{n-q,j}, Y_j^{(n-q)}), \quad j = 1, 2, \tag{7.3}$$

combined with the second order quadrature formula

$$Z_j^{(n)} = h \sum_{q=0}^{m}{}'' \, g\left(t_{n,j}, t_{n-q,j}, Y_j^{(n-q)}\right), \quad j = 1, 2, \tag{7.4}$$

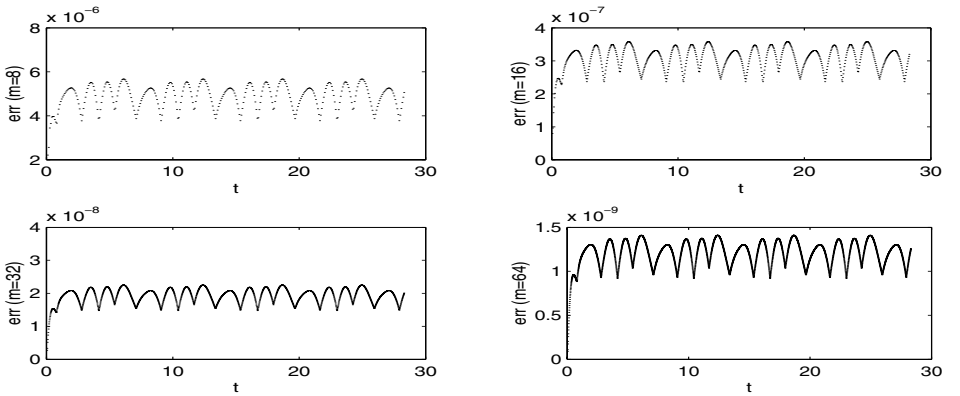


FIG. 7.6. *Error of the method from Example* 5.2 *with* $m = 8, 16, 32, 64$ *for* (7.2) *on* $[0, 9\pi]$.

TABLE 6

Convergence order of extended GL methods for system (7.2). *Example* 5.1*: 2nd order BDF + 2nd order quadrature; Example* 5.3*: 3rd order MRK + 3rd order quadrature; Example* 5.2*: 4th order MRK + 4th order quadrature.*

m	Example 5.1	Example 5.3	Example 5.2
8	2.9132	4.1053	4.7479
16	2.7183	3.8449	4.5847
32	2.5918	3.6858	4.4804
64	2.5029	3.5787	4.4078

TABLE 7

Convergence order of extended GL methods for (7.2)*: 3rd order MRK method combined with 1st order method* (7.3)*, combined with 2nd order method* (7.4)*, combined with 3rd order method* (5.15)*, and combined with 4th order method* (5.14)*.*

m	(7.3)	(7.4)	(5.15)	(5.14)
8	1.6318	3.5703	4.1053	4.2630
16	1.4976	3.2562	3.8449	3.9556
32	1.4102	3.0393	3.6858	3.7715
64	1.3489	2.8821	3.5787	3.6492

and combined with the fourth order quadrature formula (5.14). The estimated convergence orders are given in Table 7. For completeness, we also added the second column from Table 6, where the method was combined with a third order quadrature method. Finally, in Table 8, we present similar results for the fourth order MRK method from Example 5.2 combined with four quadrature rules of different orders.

All of these results seem to indicate that the order of the extended GL method is equal to the minimum of the orders of its two components: the underlying GL method and the quadrature rule. A proof for this observation and a derivation of the conditions under which it would hold for class $\mathbb{GRI}(\alpha, \beta, \sigma, \gamma)$ remain to be found.

8. Concluding remarks. We have investigated the stability of extended GL methods for a class of nonlinear MVIDEs. Numerical examples illustrate the convergence and the effectiveness of the methods. Our findings extend and improve

TABLE 8
Convergence order of extended GL methods for (7.2)*: 4th order MRK method combined with* 1*st order method* (7.3)*, combined with* 2*nd order method* (7.4)*, combined with* 3*rd order method* (5.15)*, and combined with* 4*th order method* (5.14)*.*

m	(7.3)	(7.4)	(5.15)	(5.14)
8	1.6320	3.5895	4.3429	4.0022
16	1.4976	3.2555	4.1125	3.9982
32	1.4102	3.0354	3.9351	3.9964
64	1.3489	2.8804	3.8030	3.9960

our earlier results reported in [33]. Specializing our results to the DDE $y'(t) = f(t, y(t), y(t-\tau))$, we find our results to be consistent with those in [20], slightly weaker than those in [30], and substantially different from those in [28, 29, 6, 7, 31, 32, 25]. As a conclusion, we would like to point out the precise differences.

In [30], the author studied the implicit Euler method for DDEs and proved that this method satisfies the following stability property:

$$\|y_n - \tilde{y}_n\| \leq \max_{t_0-\tau\leq t\leq t_0} \|\varphi(t) - \psi(t)\| \quad \forall n \geq 0. \tag{8.1}$$

This strong property could be derived thanks to the relatively simple structure of the Euler scheme. The approach cannot be generalized to more complex methods.

In [28, 29], linear and nonlinear DDEs of class $\mathbb{GRI}(-\rho, \rho, 0, 0)$ with $\rho > 0$ are considered. When an A-stable RK method is used in the linear case or an algebraically stable RK method is used in the nonlinear case, a specialized stability result can be derived, called B_ρ-stability. In particular, it is shown that

$$\lceil y_{n+1} - \tilde{y}_{n+1} \rceil \leq \lceil y_n - \tilde{y}_n \rceil, \tag{8.2}$$

where the notation $\lceil \cdot \rceil$ is defined as follows:

$$\lceil y_n \rceil^2 = \frac{1}{2}\|y_n\|^2 + h\rho \sum_{n-m\leq j\leq n-1} \sum_{i=1}^{s} b_i \|y_i^{(j)}\|.$$

This clearly shows the perturbation behavior of a combination of the numerical solution and the stage value approximations. In practice, however, one usually wishes to check the perturbation behavior of the numerical solution only.

In [6, 31], based on the concept of (semi-) BN_f-stability of the underlying continuous RK methods, the authors derived (RN-) GRN-stability results for RK methods for DDEs, with a stability inequality characterized by (8.1). The special cases of continuous RK methods applied to linear autonomous and nonautonomous DDE problems are also considered. The concept of BN_f-stability is then specialized into that of A_f-stability and that of AN_f-stability, respectively. Under those conditions, (8.1) is obtained again. For more details and some extensions we refer the reader to [7]. The assumption of (semi-) BN_f-stability is rather strong, however, as it leads to an order barrier for the underlying RK methods. Indeed, in [32] it is shown that four is a sure order barrier for any BN_f-stable RK method. Moreover, at this moment, only two BN_f-stable RK methods have been found: the implicit Euler method and the Lobatto IIIC method with linear interpolation. Our research on the contrary is based on algebraic stability, which does not usually suffer from such a strong order barrier.

In [25], Li studies RK methods applied to general Volterra functional differential equations. Under the condition of B-stability, it is shown that the numerical solutions y_n and $\tilde{y}_n$ obtained from two different sets of initial data $\{\varphi(t), y_1, y_2, \ldots, y_k\}$ and $\{\psi(t), \tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_k\}$ satisfy the following stability inequality:

$$\|y_n - \tilde{y}_n\| \leq \exp(c(t_n - t_k)) \max \left\{ \max_{1\leq i\leq k} \|y_i - \tilde{y}_i\|, \max_{t_0-\tau\leq t\leq t_0} \|\varphi(t) - \psi(t)\| \right\}, \tag{8.3}$$

where c is a positive constant and h_n a variable step size. Note that the stability function $\mathcal{H}(t) := \exp(c(t-t_k))$ in (8.3) is increasing for $t \geq t_0$ and will tend to infinity when $t \to +\infty$. Hence, (8.3) is applicable only on a finite time interval. Our stability results on the contrary can be applied on the infinite time interval since the stability function is a constant $\mathcal{H}$ and is independent of the time variable t.

REFERENCES

[1] C. T. H. Baker and N. J. Ford, *Convergence of linear multistep methods for a class of delay integro-differential equations*, in Numerical Mathematics, Singapore 1988, Internat. Schriftenreihe Numer. Math. 86, R. P. Agarwal, Y. M. Chow, and S. J. Wilson, eds., Birkhäuser, Basel, 1988, pp. 47–59.

[2] C. T. H. Baker and N. J. Ford, *Asymptotic error expansions for linear multistep methods for a class of delay integro-differential equations*, Bull. Soc. Math. Grèce (N.S.), 31 (1990), pp. 5–18.

[3] C. T. H. Baker and N. J. Ford, *Stability properties of a scheme for the approximate solution of a delay integro-differential equation*, Appl. Numer. Math., 9 (1992), pp. 357–370.

[4] C. T. H. Baker and A. Tang, *Generalized Halanay inequalities for Volterra functional differential equations and discrete versions*, in Volterra Equations and Applications (Arlington, TX, 1996), Stability Control Theory Methods Appl. 10, C. Corduneanu and I. W. Sandberg, eds., Gordon and Breach, Amsterdam, 2000, pp. 39–55.

[5] C. T. H. Baker and N. J. Ford, *Stability analysis of continuous Runge-Kutta methods for Volterra integro-differential systems with unbounded delays*, Appl. Numer. Math., 24 (1997), pp. 153–173.

[6] A. Bellen and M. Zennaro, *Strong contractivity properties of numerical methods for ordinary and delay differential equations*, Appl. Numer. Math., 9 (1992), pp. 321–346.

[7] A. Bellen and M. Zennaro, *Numerical Methods for Delay Differential Equations*, Oxford University Press, Oxford, UK, 2003.

[8] F. Bloom, *Ill-Posed Problems for Integrodifferential Equations in Mechanics and Electromagnetic Theory*, SIAM Stud. Appl. Math. 3, SIAM, Philadelphia, 1981.

[9] H. Brunner and P. J. van der Houwen, *The Numerical Solution of Volterra Equations*, CWI Monogr. 3, North–Holland, Amsterdam, 1986.

[10] H. Brunner, *The numerical solution of neutral Volterra integro-differential equations with delay arguments*, Ann. Numer. Math., 1 (1994), pp. 309–322.

[11] H. Brunner, *The discretization of neutral functional integro-differential equations by collocation methods*, Z. Anal. Anwendungen, 18 (1999), pp. 393–406.

[12] H. Brunner and R. Vermiglio, *Stability of solutions of neutral functional integro-differential equations and their discretizations*, Computing, 71 (2003), pp. 229–245.

[13] K. Burrage, *Non-linear stability of a general class of differential equations methods*, BIT, 20 (1980), pp. 185–203.

[14] K. Burrage, *High order algebraically stable multistep Runge–Kutta methods*, SIAM J. Numer. Anal., 24 (1987), pp. 106–115.

[15] J. C. Butcher, *Stability properties for a general class of methods for ordinary differential equations*, SIAM J. Numer. Anal., 18 (1981), pp. 37–44.

[16] J. C. Butcher, *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*, John Wiley & Sons, Chichester, UK, 1987.

[17] W. H. Enright and M. Hu, *Continuous Runge-Kutta methods for neutral Volterra integro-differential equations with delay*, Appl. Numer. Math., 24 (1997), pp. 175–190.

[18] E. Hairer and M. Zennaro, *On error growth of Runge-Kutta methods*, Appl. Numer. Math., 22 (1996), pp. 205–216.

[19] C. Huang, H. Fu, S. Li, and G. Chen, *Stability analysis of Runge-Kutta methods for nonlinear delay differential equations*, BIT, 39 (1999), pp. 270–280.

[20] C. Huang, S. Li, H. Fu, and G. Chen, *Nonlinear stability of general linear methods for delay differential equations*, BIT, 42 (2002), pp. 380–392.
[21] C. Huang and S. Vandewalle, *Stability of Runge-Kutta-Pouzet Methods for Volterra Integral and Integro-differential Equations with Delays*, Technical report TW 361, Department of Computer Science, Katholieke Universiteit Leuven, Leuven, Belgium, 2003.
[22] A. J. Jerri, *Introduction to Integral Equations with Applications*, Wiley-Interscience, New York, 1999.
[23] T. Koto, *Stability of Runge-Kutta methods for delay integro-differential equations*, J. Comput. Appl. Math., 145 (2002), pp. 483–492.
[24] S. Li, *Stability and B-convergence properties of multistep Runge-Kutta methods*, Math. Comp., 69 (1999), pp. 1481–1504.
[25] S. Li, *B-theory of Runge-Kutta methods for stiff Volterra functional differential equations*, Sci. China Ser. A, 46 (2003), pp. 662–674.
[26] P. Linz, *Analytical and Numerical Methods for Volterra Equations*, SIAM Stud. Appl. Math. 7, SIAM, Philadelphia, 1985.
[27] T. Luzyanina, K. Engelborghs, and D. Roose, *Computing stability of differential equations with bounded and distributed delays*, Numer. Algorithms, 34 (2003), pp. 41–66.
[28] J. Reverdy, *Sur l'approximation d'équations d'évolution linéaires du premier ordre à retard par des méthodes de type Runge-Kutta*, Thèse de doctorat, Université Paul Sabatier, Toulouse, France, 1981.
[29] J. Reverdy, *Sur la B-stabilité pour une équation différentielle à retard*, C. R. Acad. Sci. Paris Sér. I Math., 310 (1990), pp. 461–463.
[30] L. Torelli, *Stability of numerical methods for delay differential equations*, J. Comput. Appl. Math., 25 (1989), pp. 15–26.
[31] M. Zennaro, *Contractivity of Runge-Kutta methods with respect to forcing terms*, Appl. Numer. Math., 10 (1993), pp. 321–345.
[32] M. Zennaro, *Asymptotic stability analysis of Runge-Kutta methods for nonlinear systems of delay differential equations*, Numer. Math., 77 (1997), pp. 549–563.
[33] C. Zhang and S. Vandewalle, *Stability analysis of delay-integro-differential equations and their backward differentiation time-discretization*, J. Comput. Appl. Math., 164/165 (2004), pp. 797–814.
[34] C. Zhang and S. Vandewalle, *Stability analysis of Runge-Kutta methods for nonlinear Volterra delay-integro-differential equations*, IMA J. Numer. Anal., 24 (2004), pp. 193–214.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2032–2052
© 2006 Society for Industrial and Applied Mathematics

CONTACT ANALYSIS OF CABLE NETWORKS BY USING SECOND-ORDER CONE PROGRAMMING*

YOSHIHIRO KANNO† AND MAKOTO OHSAKI‡

Abstract. A method based on mathematical programming is proposed for large deformation and contact analysis of cable networks. By explicitly considering these nonsmooth behaviors, we formulate the linear complementarity problems over symmetric cones under some practically acceptable assumptions. We also present the equivalent second-order cone programming (SOCP) problems, which can be regarded as the minimization problem of total potential energy and complementary energy with the subsidiary constraints on the displacements and contact forces, respectively. By solving the presented SOCP problems by using the primal-dual interior-point method, the equilibrium configurations and internal forces of several cable networks are obtained without any assumptions on stress states and contact conditions.

Key words. second-order cone programming, cable network, contact problem, complementarity problem, primal-dual interior-point method

AMS subject classifications. 74M15, 90C25, 90C51

DOI. 10.1137/S1064827503431946

1. Introduction.

Nonsmooth behavior of mechanical systems, such as contact mechanics, slackening of cables, and wrinkling of membranes, has received increasing attention [14, 31]. In particular, contact problems have been extensively studied in the literature. The variational inequality formulations of the elastostatic frictional unilateral contact problem were reviewed in [28, 29]. The frictionless unilateral contact problem, which is well known as the Signorini problem, is classical but still important [1, 11, 20, 23]. For multi–rigid-body systems, both formulations and algorithms for the static, quasi-static, and dynamic contact problems were investigated [3, 34, 38, 41]. The general frameworks for the frictional contact problem were presented based on the complementarity problems [33]. For two-body contact problems, various numerical approaches, such as the finite element method, have been developed [15, 16, 27].

Stress-unilateral behaviors have also received much attention due to their wide field of application, which includes equilibrium analysis of masonry structures [8, 10] and membrane structures [6, 12, 25, 35]. For cable networks, however, few studies have dealt explicitly with the slackening of cables [5, 9, 32].

This paper deals with two types of nonsmooth behaviors; i.e., we consider the cable networks where each member can transmit only tension force, and suppose that some members may possibly make contact with each other or with rigid obstacles at the equilibrium state. Geometrical nonlinearity due to large deformation is also considered.

In the conventional approaches to contact problems, we typically choose a set of contact candidates. Each candidate is defined as the pair of node and contact surface for unilateral contact problems [28], or that of two nodes for two-body contact

*Received by the editors July 23, 2003; accepted for publication (in revised form) May 12, 2005; published electronically February 3, 2006.
http://www.siam.org/journals/sisc/27-6/43194.html
†Department of Urban and Environmental Engineering, Kyoto University, Kyoto 606-8501, Japan (kanno@archi.kyoto-u.ac.jp).
‡Department of Architecture and Architectural Engineering, Kyoto University, Kyoto 615-8540, Japan (ohsaki@archi.kyoto-u.ac.jp).

problems [20]. We should choose an appropriate set of candidates satisfying some assumptions, e.g., the assumption that the contact occurs only at these candidates. It is difficult, however, to choose such sets generally, especially if we allow finite deformation. We shall show that, for contact problems of cable networks, we do not have to identify the locations of candidate nodes but only have to choose some members as candidates. Therefore, it is rather easy to select an appropriate set of candidates, which is illustrated by using an example in section 4. The contact conditions between the two cables, and between the cable and obstacle, respectively, are formulated as the complementarity conditions in sections 3 and 4.

Concerning stress-unilateral behavior, in the existing incremental methods based on the tangent stiffness [4], the tangent stiffness matrix should be constructed by using the information on the stress states. Since such information and the obtained displacement increments may conflict, these methods sometimes cannot find a convergent solution [19]. Another difficulty exists in the singularity of the tangent stiffness matrix in the case where many cable members are slackening. In contrast, we formulate the constitutive law in terms of the internal force vector and the member deformation vector as the complementarity condition over *second-order cones* [7].

By explicitly considering these nonsmooth behaviors, we formulate the linear complementarity problems over symmetric cones to obtain the equilibrium configurations of cable networks under the contact conditions. We also present the equivalent *second-order cone programming* (SOCP) [7] problems. SOCP is a class of nonlinear convex programming including linear programming and convex quadratic programming. The primal-dual interior-point algorithms have been developed for SOCP, which can obtain the global optimal solution in time polynomial in the problem size [7, 30, 39, 40]. It should be emphasized that the equilibrium configuration can be obtained by solving an SOCP problem which is convex and independent of both the stress states of cable members and contact states.

This paper is organized as follows. In section 2, we introduce SOCP and, in order to make this paper self-contained, we revisit the SOCP formulation presented in [19], which has the same optimizer as the minimization of total potential energy for cable networks without contact conditions. In section 2.2, we formulate the constitutive law considering the slackening of cable and geometrical nonlinearity into the complementarity conditions over the second-order cones which will play a key role in sections 3 and 4. Sections 3 and 4, respectively, are devoted to formulating SOCP problems for the two-body contact problem and unilateral contact problem of cable networks. By introducing the concept of auxiliary frictionless node, we define the contact candidates as the pairs of two members of the cable network, or those of members and edges of the obstacle. Under some practically acceptable assumptions, we formulate the complementarity problems over second-order cones and present the equivalent SOCP problems. In section 5, equilibrium configurations of cable networks with various contact conditions are obtained by solving the proposed SOCP problem by using the primal-dual interior-point method [39]. Some technical results are proved in section 6. We conclude the paper in section 7.

2. Preliminaries.

2.1. Second-order cone program. For any $\boldsymbol{p} \in \mathbf{R}^n$, let $\|\boldsymbol{p}\|$ denote the standard Euclidean norm defined by $\|\boldsymbol{p}\| = (\boldsymbol{p}^\top \boldsymbol{p})^{1/2}$. All the vectors are assumed to be column vectors. However, for vectors $\boldsymbol{p} \in \mathbf{R}^n$ and $\boldsymbol{q} \in \mathbf{R}^m$, we often write $(\boldsymbol{p}, \boldsymbol{q}) = (\boldsymbol{p}^\top, \boldsymbol{q}^\top)^\top \in \mathbf{R}^{n+m}$ in order to simplify the notation.

Let $\mathbf{R}^n_+ \subset \mathbf{R}^n$ and $\mathbf{L}^n_+ \subset \mathbf{R}^n$ denote the nonnegative orthant and the *second-order cone* (or the *Lorentz cone*) [7], respectively, which are defined as

$$\mathbf{R}^n_+ = \{\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{R}^n \,|\, x_i \geq 0 \ (i = 1, \ldots, n)\},$$
$$\mathbf{L}^n_+ = \{\boldsymbol{x} = (x_0, \boldsymbol{x}_1) \in \mathbf{R}^n \,|\, x_0 \geq \|\boldsymbol{x}_1\|\}.$$

Let $\mathcal{K} \subset \mathbf{R}^n$ denote a convex cone. $\mathcal{K}^* \subset \mathbf{R}^n$ denotes the dual cone of $\mathcal{K}$ defined by $\mathcal{K}^* = \{\boldsymbol{s} \in \mathbf{R}^n \,|\, \boldsymbol{x}^\top \boldsymbol{s} \geq 0 \ (\forall \boldsymbol{x} \in \mathcal{K})\}$. The following pair of problems is known to be a primal-dual pair of *conic linear programming* (cone LP) problems [7]:

$$(\mathrm{P}_{\mathrm{cone}}): \quad \min \quad \boldsymbol{c}^\top \boldsymbol{x} \qquad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} \in \mathcal{K}, \tag{2.1}$$

$$(\mathrm{D}_{\mathrm{cone}}): \quad \max \quad \boldsymbol{b}^\top \boldsymbol{y} \qquad \text{s.t.} \quad \boldsymbol{c} - \boldsymbol{A}^\top \boldsymbol{y} \in \mathcal{K}^*, \tag{2.2}$$

where $\boldsymbol{A} \in \mathbf{R}^{m \times n}$, $\boldsymbol{b} \in \mathbf{R}^m$, and $\boldsymbol{c} \in \mathbf{R}^n$ are constants, and $\boldsymbol{x} \in \mathbf{R}^n$ and $\boldsymbol{y} \in \mathbf{R}^m$ are variables. In this paper, we deal with the special class of cone LP problems; i.e., suppose that $\mathcal{K} \subset \mathbf{R}^n$ is written as $\mathcal{K} = \mathcal{K}_1 \times \cdots \times \mathcal{K}_k$, where $\mathcal{K}_i = \mathbf{R}^{n_i}_+$ or $\mathbf{L}^{n_i}_+$, and $n = \sum_{i=1}^k n_i$. Since $\mathbf{R}^n_+$ and $\mathbf{L}^n_+$ have self-dual properties [7], i.e., $(\mathbf{R}^n_+)^* = \mathbf{R}^n_+$ and $(\mathbf{L}^n_+)^* = \mathbf{L}^n_+$, we have $\mathcal{K}^* = \mathcal{K}$. By introducing $\boldsymbol{x}_i \in \mathbf{R}^{n_i}$, $\boldsymbol{A}_i \in \mathbf{R}^{m \times n_i}$, and $\boldsymbol{c}_i \in \mathbf{R}^{n_i}$ $(i = 1, \ldots, k)$, (2.1) and (2.2) are explicitly rewritten as

$$\begin{cases} (\mathrm{P}): \quad \min \quad \displaystyle\sum_{i=1}^k \boldsymbol{c}_i^\top \boldsymbol{x}_i \\ \qquad\quad \text{s.t.} \quad \displaystyle\sum_{i=1}^k \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}, \quad \boldsymbol{x}_i \in \mathcal{K}_i \quad (i = 1, \ldots, k), \end{cases} \tag{2.3}$$

$$\begin{cases} (\mathrm{D}): \quad \max \quad \boldsymbol{b}^\top \boldsymbol{y} \\ \qquad\quad \text{s.t.} \quad \boldsymbol{c}_i - \boldsymbol{A}_i^\top \boldsymbol{y} \in \mathcal{K}_i \quad (i = 1, \ldots, k), \end{cases} \tag{2.4}$$

which are classified as the SOCP problems [2]. Since $\mathbf{R}^n_+ = (\mathbf{L}_+)^n$, SOCP includes the *linear programming* (LP) problem as a special case. SOCP has received increasing attention for its wide fields of application [7, 17, 24, 37]. The primal-dual interior-point methods, which were developed for LP [21] at first, have been naturally extended to SOCP [2, 30, 40]. It is theoretically guaranteed that the primal-dual interior-point method converges to the global optimal solution of (P) and (D) within the number of arithmetic operations bounded by a polynomial of m and n.

We make the following assumptions:

1. (2.3) and (2.4) have interior feasible solutions;
2. rows of the matrix $(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_n) \in \mathbf{R}^{m \times n}$ are linearly independent.

Under the first assumption, it is known that (2.3) and (2.4) have optimal solutions which have the same optimal value, and that solving (2.3) and (2.4) is equivalent to finding the following $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$, $\boldsymbol{y}$ and $\boldsymbol{s} = (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k)$ [7, 40]:

$$\begin{cases} (\mathrm{PD}): \quad \boldsymbol{s}_i^\top \boldsymbol{x}_i = 0 \quad (i = 1, \ldots, k), \\ \qquad\quad \displaystyle\sum_{i=1}^k \boldsymbol{A}_i \boldsymbol{x}_i = \boldsymbol{b}, \quad \boldsymbol{x}_i \in \mathcal{K}_i \quad (i = 1, \ldots, k), \\ \qquad\quad \boldsymbol{s}_i = \boldsymbol{c}_i - \boldsymbol{A}_i^\top \boldsymbol{y} \in \mathcal{K}_i \quad (i = 1, \ldots, k). \end{cases} \tag{2.5}$$

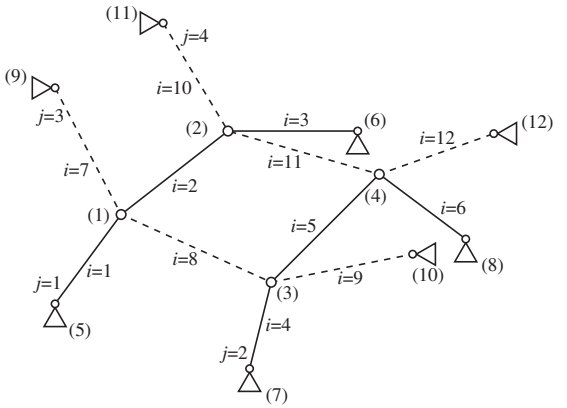


FIG. 1. *Cable network with frictionless joints.*

2.2. Complementarity conditions for frictionless cable networks. Consider a cable network in three-dimensional space. A physically continuous cable connecting a support and a pin-joint or two pin-joints is simply referred to as a *cable*. A cable is divided into several *members* by *frictionless joints*, which can move freely along the cable.

Let n^{m} and n^{c} denote the numbers of members and cables, respectively. $\boldsymbol{x} \in \mathbf{R}^{n^{\mathrm{d}}}$ denotes the vector of coordinates of internal nodes at the deformed state, where n^{d} is the number of degrees of freedom of displacements. Let $\boldsymbol{h}_i \in \mathbf{R}^3$ denote the vector that has the same length and direction as those of the ith member at the deformed state. Then $\boldsymbol{h}_i$ is written as

$$\boldsymbol{h}_i = \boldsymbol{B}_i \boldsymbol{x} - \boldsymbol{d}_i \quad (i = 1, \ldots, n^{\mathrm{m}}), \tag{2.6}$$

where $\|\boldsymbol{h}_i\|$ coincides with the member length at the deformed state. For each $i = 1, \ldots, n^{\mathrm{m}}$, $\boldsymbol{B}_i \in \mathbf{R}^{3 \times n^{\mathrm{d}}}$ is a constant matrix determined only by the connectivity of nodes and the ith member, each element of which is either $\{-1, 0, 1\}$. $\boldsymbol{d}_i \in \mathbf{R}^3$ is a constant vector that consists of the specified coordinates of support if the ith member is connected to the support; otherwise $\boldsymbol{d}_i = \boldsymbol{0}$. Consider a simple example of a cable network illustrated in Figure 1, which consists of four cables indicated as $j = 1, \ldots, 4$. These cables are connected by frictionless joints (1)–(4), whereas nodes (5)–(12) are fixed supports. Hence, we see $n^{\mathrm{c}} = 4$, $n^{\mathrm{m}} = 12$, and $n^{\mathrm{d}} = 12$.

The subset $\mathcal{I}_j$ of $\{1, 2, \ldots, n^{\mathrm{m}}\}$ is defined for each $j = 1, \ldots, n^{\mathrm{c}}$ such that the ith member satisfying $i \in \mathcal{I}_j$ belongs to the jth cable, where $\mathcal{I}_{j_1} \cap \mathcal{I}_{j_2} = \emptyset$ $(j_1 \neq j_2)$ and $\mathcal{I}_1 \cup \cdots \cup \mathcal{I}_{n^{\mathrm{c}}} = \{1, 2, \ldots, n^{\mathrm{m}}\}$. For example, in the case of the cable network illustrated in Figure 1, the cable $j = 1$ is divided into members $i = 1, 2, 3$; i.e., we have $\mathcal{I}_1 = \{1, 2, 3\}$, $\mathcal{I}_2 = \{4, 5, 6\}$, $\mathcal{I}_3 = \{7, 8, 9\}$, and $\mathcal{I}_4 = \{10, 11, 12\}$. Letting ℓ_j^0 and c_j denote the specified initial unstressed length and the elongation at the deformed state of the jth cable, respectively, we see that

$$c_j = \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\| - \ell_j^0 \quad (j = 1, \ldots, n^{\mathrm{c}}), \tag{2.7}$$

which is regarded as the geometrically exact compatibility condition. Assume that

the strain energy w_j and the axial force q_j of the jth cable are written as

$$w_j(c_j) = \begin{cases} k_j c_j^2/2 & (c_j \geq 0), \\ 0 & (-\ell_j^0 \leq c_j < 0), \end{cases} \qquad q_j = \frac{\partial w_j(c_j)}{\partial c_j}, \tag{2.8}$$

where $k_j > 0$ denotes the specified extensional stiffness. Notice here that the constitutive law (2.8) represents the nonsmooth behavior due to slackening of cables. The relation between the member deformation $\boldsymbol{h}_i \in \mathbf{R}^3$ and the internal force $\boldsymbol{v}_i \in \mathbf{R}^3$ of the ith member is written as

$$\boldsymbol{v}_i \|\boldsymbol{h}_i\| = q_j \boldsymbol{h}_i. \tag{2.9}$$

If $\boldsymbol{h}_i \neq \boldsymbol{0}$, then (2.9) is reduced to

$$\boldsymbol{v}_i = \begin{cases} q_j \boldsymbol{h}_i / \|\boldsymbol{h}_i\| & (q_j \geq 0), \\ \boldsymbol{0} & (q_j = 0), \end{cases} \tag{2.10}$$

which implies that $\boldsymbol{h}_i$ and $\boldsymbol{v}_i$ must share a common direction.

For each $j = 1, \ldots, n^{\mathrm{c}}$, consider the conditions

$$\begin{aligned} &q_j = k_j y_j, \quad q_j(y_j + \ell_j^0) - \sum_{i \in \mathcal{I}_j} \boldsymbol{v}_i^\top \boldsymbol{h}_i = 0, \\ &y_j + \ell_j^0 \geq \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|, \quad q_j \geq \|\boldsymbol{v}_i\| \quad (i \in \mathcal{I}_j), \end{aligned} \tag{2.11}$$

which can be regarded as the linear complementarity conditions over second-order cones $\mathbf{L}_+^4$ (see, e.g., [13]). The following lemma implies that (2.8) and (2.9) can be reformulated into (2.11).

LEMMA 2.1. *For each $j = 1, \ldots, n^{\mathrm{c}}$, a set of $\boldsymbol{h}_i$, c_j, y_j, q_j, and $\boldsymbol{v}_i$ $(i \in \mathcal{I}_j)$ satisfies* (2.7)–(2.9) *and*

$$y_j = \begin{cases} c_j & (c_j \geq 0), \\ 0 & (c_j < 0) \end{cases} \tag{2.12}$$

if and only if (2.7) *and* (2.11) *are satisfied.*

Proof. See section 6. ∎

Notice here that the nonsmooth property due to slackening of cable as well as the compatibility conditions can be expressed via the complementarity conditions (2.11). In (2.11), we see that the compatibility conditions $y_j + \ell_j^0 = \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|$ and the relations $q_j = \|\boldsymbol{v}_i\|$ $(i \in \mathcal{I}_j)$ are relaxed into inequalities. Moreover, (2.11) lacks the relation (2.9) between $\boldsymbol{h}_i$ and $\boldsymbol{v}_i$. Lemma 2.1 implies that all these conditions are satisfied at a solution of (2.7) and (2.11). See [18] for detailed discussions of this issue for pin-jointed cable networks without frictionless joints.

Let $\mathcal{L}_{\mathrm{f}}$ denote the set of all indices for both frictionless joints and pin-joints. In accordance with the notation used in the following sections, we put here $\mathcal{L}_{\mathrm{f}} = \{1, 2, \ldots, n^{\mathrm{n}}\}$, where n^{n} denotes the number of unconstrained nodes. For example, in Figure 1, we have $n^{\mathrm{n}} = 4$ and $\mathcal{L}_{\mathrm{f}} = \{1, 2, 3, 4\}$. The coordinates of the lth node $\boldsymbol{x}_l \in \mathbf{R}^3$ can be rewritten as

$$\boldsymbol{x}_l = \boldsymbol{C}_l \boldsymbol{x} \quad (l \in \mathcal{L}_{\mathrm{f}}), \tag{2.13}$$

where $\boldsymbol{C}_l \in \mathbf{R}^{3\times n^{\mathrm{d}}}$ is a constant matrix determined only by the indices of freedom of displacements of the lth node, and each element of $\boldsymbol{C}_l$ is either $\{0, 1\}$. We specify the vector of external loads $\boldsymbol{f}_l \in \mathbf{R}^3$ applied at the lth node. The equilibrium equations allowing the finite deformations are written as

$$\sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{B}_i^{\top} \boldsymbol{v}_i = \sum_{l\in\mathcal{L}_{\mathrm{f}}} \boldsymbol{C}_l^{\top} \boldsymbol{f}_l. \tag{2.14}$$

The equilibrium configuration and internal forces can be obtained by solving the system of (2.6)–(2.9) and (2.14). Suppose that a set of $\boldsymbol{x}$, $\boldsymbol{h}_i$, y_j, q_j, and $\boldsymbol{v}_i$ satisfies the system of (2.6), (2.11), and (2.14). Then c_j is obtained from (2.7). Lemma 2.1 implies that the obtained set of $\boldsymbol{x}$, $\boldsymbol{h}_i$, c_j, q_j, and $\boldsymbol{v}_i$ solves the system (2.6)–(2.9) and (2.14). The relation between y_j and c_j is given by (2.12). Thus, in order to obtain the equilibrium configurations and internal forces of cable networks, we can solve the second-order cone complementarity problem (2.6), (2.11), and (2.14) instead of the system of (2.6)–(2.9) and (2.14).

3. Two-body contact problem of cable networks. In this section, suppose that some members of the cable network may possibly make contact with each other. Let $\mathcal{N}^{\mathrm{m}} = \{1, 2, \ldots, n^{\mathrm{m}}\}$. We define contact candidates as

$$\boldsymbol{\zeta}_l = (i_l^{\mathrm{U}}, i_l^{\mathrm{D}}), \quad i_l^{\mathrm{U}} \in \mathcal{N}^{\mathrm{m}}, \quad i_l^{\mathrm{D}} \in \{\mathcal{N}^{\mathrm{m}} \setminus i_l^{\mathrm{U}}\} \quad (l \in \mathcal{L}_{\mathrm{c}}),$$

each of which is a pair of the indices i_l^{U} and i_l^{D} of members. $\mathcal{L}_{\mathrm{c}}$ denotes the set of all indices of the candidates. At the equilibrium state, the i_l^{U}th member should exist *above* the i_l^{D}th member (rigorously defined below).

For each candidate $\boldsymbol{\zeta}_l$ ($l \in \mathcal{L}_{\mathrm{c}}$), we assign a pair of auxiliary nodes $\boldsymbol{x}_l^{\mathrm{U}} \in \mathbf{R}^3$ and $\boldsymbol{x}_l^{\mathrm{D}} \in \mathbf{R}^3$ on the i_l^{U}th and the i_l^{D}th members, respectively, which are regarded as frictionless joints. These nodes are added to the list of the unconstrained nodes. Recalling that n^{n} denotes the number of unconstrained nodes, we see $\mathcal{L}_{\mathrm{c}} \subset \mathcal{N}^{\mathrm{n}} := \{1, 2, \ldots, n^{\mathrm{n}}\}$. The set $\mathcal{L}_{\mathrm{f}}$ of all indices of original nodes satisfies $\mathcal{L}_{\mathrm{f}} = \{\mathcal{N}^{\mathrm{n}} \setminus \mathcal{L}_{\mathrm{c}}\}$. Notice here that $\mathcal{L}_{\mathrm{f}}$, $\mathcal{L}_{\mathrm{c}}$, and $\mathcal{N}^{\mathrm{n}}$ are not necessarily taken as sets of successive sequence of natural numbers; e.g., we may take $\mathcal{L}_{\mathrm{f}} = \{1, 3\}$, $\mathcal{L}_{\mathrm{c}} = \{2, 5\}$, and $\mathcal{N}^{\mathrm{n}} = \{1, 2, 3, 5\}$ without any mismatching.

In a manner similar to (2.13), the coordinates of auxiliary nodes can be written as

$$\boldsymbol{x}_l^{\mathrm{D}} = \boldsymbol{C}_l^{\mathrm{D}} \boldsymbol{x} \quad (l \in \mathcal{L}_{\mathrm{c}}),$$

where $\boldsymbol{C}_l^{\mathrm{U}}$ and $\boldsymbol{C}_l^{\mathrm{D}} \in \mathbf{R}^{3\times n^{\mathrm{d}}}$ are constant matrices and $\boldsymbol{x}_l^{\mathrm{U}}$ and $\boldsymbol{x}_l^{\mathrm{D}}$ are the nodal coordinates at the deformed state. For each $l \in \mathcal{L}_{\mathrm{c}}$, we also assign a constant orthogonal matrix $\boldsymbol{T}_l \in \mathbf{R}^{3\times 3}$. The local orthogonal coordinate system $\boldsymbol{\xi}_l = (\xi_{l1}, \xi_{l2}, \xi_{l3})$ is defined by using $\boldsymbol{T}_l$ such that the locations of auxiliary nodes $\boldsymbol{x}_l^{\mathrm{U}}$ and $\boldsymbol{x}_l^{\mathrm{D}}$ are expressed as

$$\boldsymbol{\xi}_l^{\mathrm{U}} = \boldsymbol{T}_l \boldsymbol{x}_l^{\mathrm{U}}, \quad \boldsymbol{\xi}_l^{\mathrm{D}} = \boldsymbol{T}_l \boldsymbol{x}_l^{\mathrm{D}}. \tag{3.1}$$

Define the vector $\boldsymbol{g}_l \in \mathbf{R}^3$ ($l \in \mathcal{L}_{\mathrm{c}}$) by

$$\boldsymbol{g}_l = \boldsymbol{\xi}_l^{\mathrm{U}} - \boldsymbol{\xi}_l^{\mathrm{D}}, \tag{3.2}$$

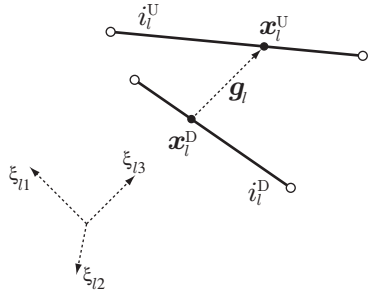


FIG. 2. *Local coordinate* $\boldsymbol{\xi}_l$.

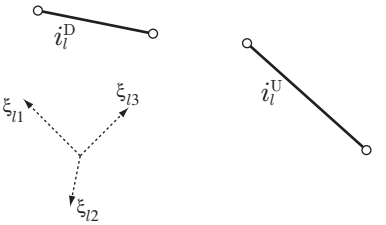


FIG. 3. *Inadmissible case of Assumption* 3.1.

which can be rewritten as $\boldsymbol{g}_l = \boldsymbol{A}_l \boldsymbol{x}$, where a constant matrix $\boldsymbol{A}_l \in \mathbf{R}^{3\times n^{\mathrm{d}}}$ is defined by

$$\boldsymbol{A}_l := \boldsymbol{T}_l(\boldsymbol{C}_l^{\mathrm{U}} - \boldsymbol{C}_l^{\mathrm{D}}).$$

The relations among $\boldsymbol{x}_l^{\mathrm{U}}$, $\boldsymbol{x}_l^{\mathrm{D}}$, $\boldsymbol{\xi}_l$, and $\boldsymbol{g}_l$ are illustrated in Figure 2.

In the conventional approach to two-body contact problems [22], we define the gap vector by using the closest points of two bodies. Notice here that $\boldsymbol{g}_l$ coincides with the usual gap vector only if we take $\boldsymbol{T}_l$ such that $\boldsymbol{g}_l$ is normal to both the i_l^{U}th and i_l^{D}th members. Since $\boldsymbol{T}_l$ satisfying these conditions depends on the deformation of the cable network, it is difficult to estimate such $\boldsymbol{T}_l$ a priori for the case of large deformation. Therefore, we assume the following quite weak condition on $\boldsymbol{T}_l$.

Assumption 3.1. Suppose that the cable network attains the equilibrium state, which is referred to as the actual equilibrium state. For each $l \in \mathcal{L}_{\mathrm{c}}$, a system of coordinates $\boldsymbol{\xi}_l$ is defined. We assume that there exist the points $\boldsymbol{\xi}_l^{\mathrm{U}}$ and $\boldsymbol{\xi}_l^{\mathrm{D}}$ on the i_l^{U}th and i_l^{D}th members, respectively, satisfying

$$\xi_{l1}^{\mathrm{U}} = \xi_{l1}^{\mathrm{D}}, \quad \xi_{l2}^{\mathrm{U}} = \xi_{l2}^{\mathrm{D}}, \quad \xi_{l3}^{\mathrm{U}} \geq \xi_{l3}^{\mathrm{D}} \tag{3.3}$$

at the actual equilibrium state.

In other words, we assume that the orthogonal projections of the two members onto the $\xi_{l1}\xi_{l2}$-plane at the actual equilibrium state have a nonempty intersection. We assign the inverse images of the intersection which exist on the i_l^{U}th and the i_l^{D}th members as $\boldsymbol{x}_l^{\mathrm{U}}$ and $\boldsymbol{x}_l^{\mathrm{D}}$, respectively. Then the i_l^{U}th member exists *above* the i_l^{D}th member in the sense that the ξ_{l3} coordinate of $\boldsymbol{x}_l^{\mathrm{U}}$ is greater than or equal to that of $\boldsymbol{x}_l^{\mathrm{D}}$. An example of an inadmissible case of Assumption 3.1 is as illustrated in Figure 3, where there exists no intersection of the orthogonal projections of the i_l^{U}th and i_l^{D}th members onto the $\xi_{l1}\xi_{l2}$-plane. Since $\boldsymbol{\xi}_l$ is required to satisfy only Assumption 3.1, $\boldsymbol{g}_l$ defined by (3.2) does not necessarily correspond to the gap vector defined by using the closest points of the i_l^{U}th and the i_l^{D}th members. Hence, we may assign an appropriate $\boldsymbol{T}_l$ satisfying Assumption 3.1 more easily than in the conventional methods. On the other hand, it is not still clear what kind of constraints on kinematic and/or static variables should be considered in order to formulate the analysis problems.

Defining the cone $\mathcal{G} \subset \mathbf{R}^3$ by

$$\mathcal{G} = \{\boldsymbol{z} = (z_i) \in \mathbf{R}^3 |\, z_1 = z_2 = 0,\ z_3 \geq 0\}, \tag{3.4}$$

the conditions (3.3) are reduced to $\boldsymbol{g}_l \in \mathcal{G}$ $(l \in \mathcal{L}_{\rm c})$. The dual cone $\mathcal{G}^* \subset \mathbf{R}^3$ of $\mathcal{G}$ is obtained as

$$\mathcal{G}^* = \{\boldsymbol{\tau} = (\tau_i) \in \mathbf{R}^3 | \, \boldsymbol{\tau}^\top \boldsymbol{z} \geq 0 \ (\forall \boldsymbol{z} \in \mathcal{G})\} = \{\boldsymbol{\tau} \in \mathbf{R}^3 | \, \tau_3 \geq 0\}.$$

Let $\boldsymbol{\tau}_l \in \mathbf{R}^3$ $(l \in \mathcal{L}_{\rm c})$ denote the vector of contact forces, which may act between the nodes $\boldsymbol{x}_l^{\rm U}$ and $\boldsymbol{x}_l^{\rm D}$. Notice again that $\boldsymbol{g}_l$ does not correspond to the conventional gap vector, from which it follows that τ_{l3} is not necessarily equal to the normal reaction in general. Recall that, for the contact candidate $\boldsymbol{\zeta}_l$ $(l \in \mathcal{L}_{\rm c})$, the nodes $\boldsymbol{x}_l^{\rm U}$ and $\boldsymbol{x}_l^{\rm D}$ are modeled as frictionless joints, which move along with the $i_l^{\rm D}$th and the $i_l^{\rm U}$th members freely without friction. Hence, if $g_{l3} > 0$ and $\boldsymbol{\tau}_l \neq \boldsymbol{0}$, then these frictionless nodes move on the members, which implies that this situation does not correspond to an equilibrium state. Therefore, the equilibrium equations guarantee that $\boldsymbol{\tau}_l = \boldsymbol{0}$ is satisfied at an equilibrium state automatically if $g_{l3} > 0$. Consequently, the contact conditions can be written as the following complementarity conditions:

$$\boldsymbol{\tau}_l^\top \boldsymbol{g}_l = 0, \quad \boldsymbol{g}_l \in \mathcal{G}, \quad \boldsymbol{\tau}_l \in \mathcal{G}^* \quad (l \in \mathcal{L}_{\rm c}). \tag{3.5}$$

Let $\boldsymbol{v}_i \in \mathbf{R}^3$ $(i = 1, \ldots, n^{\rm m})$ denote the internal force vector of the ith member. The equilibrium equations allowing the finite deformations are written as

$$\sum_{i=1}^{n^{\rm m}} \boldsymbol{B}_i^\top \boldsymbol{v}_i = \sum_{l \in \mathcal{L}_{\rm c}} \boldsymbol{A}_l^\top \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\rm f}} \boldsymbol{C}_l^\top \boldsymbol{f}_l. \tag{3.6}$$

It follows from Lemma 2.1, (3.5), and (3.6) that the equilibrium configuration of a cable network can be obtained as a solution to the following second-order cone complementarity problem (3.7)–(3.10):

$$\text{(CP)}: \quad q_j = k_j y_j \quad (j = 1, \ldots, n^{\rm c}), \tag{3.7}$$

$$q_j(y_j + \ell_j^0) - \sum_{i \in \mathcal{I}_j} \boldsymbol{v}_i^\top \boldsymbol{h}_i = 0, \tag{3.8}$$

$$y_j + \ell_j^0 \geq \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|, \quad q_j \geq \|\boldsymbol{v}_i\| \quad (i \in \mathcal{I}_j; \ j = 1, \ldots, n^{\rm c}),$$

$$\boldsymbol{\tau}_l^\top \boldsymbol{A}_l \boldsymbol{x} = 0, \quad \boldsymbol{A}_l \boldsymbol{x} \in \mathcal{G}, \quad \boldsymbol{\tau}_l \in \mathcal{G}^* \quad (l \in \mathcal{L}_{\rm c}), \tag{3.9}$$

$$\sum_{i=1}^{n^{\rm m}} \boldsymbol{B}_i^\top \boldsymbol{v}_i = \sum_{l \in \mathcal{L}_{\rm c}} \boldsymbol{A}_l^\top \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\rm f}} \boldsymbol{C}_l^\top \boldsymbol{f}_l, \tag{3.10}$$

where the relation between y_j and the actual elongation c_j of the jth cable is given by (2.12). The relation between $\boldsymbol{h}_i$ and $\boldsymbol{x}$ is given by (2.6). Notice here that both $\mathcal{G}$ and $\mathcal{G}^*$ can be represented as second-order cones, i.e., $\mathbf{L}_+^1$, which verifies that (3.7)–(3.10) can be embedded into (2.5). We can show that the system (3.7)–(3.10) can be embedded into (2.5) and coincides with the Karush–Kuhn–Tucker (KKT) conditions

[36, Theorem 4.2] of the following primal-dual pair of SOCP problems:

$$
\left\{
\begin{array}{ll}
\min & \phi = \displaystyle\sum_{j=1}^{n^{\mathrm{c}}} \frac{1}{2} k_j y_j^2 - \sum_{l \in \mathcal{L}_{\mathrm{f}}} \boldsymbol{f}_l^{\top} \boldsymbol{C}_l \boldsymbol{x} \\
\text{s.t.} & y_j + \ell_j^0 \geq \displaystyle\sum_{i \in \mathcal{I}_j} \| \boldsymbol{B}_i \boldsymbol{x} - \boldsymbol{d}_i \| \quad (j = 1, \ldots, n^{\mathrm{c}}), \\
& \boldsymbol{A}_l \boldsymbol{x} \in \mathcal{G} \quad (l \in \mathcal{L}_{\mathrm{c}}),
\end{array}
\right. \tag{3.11}
$$

$$
\left\{
\begin{array}{ll}
\min & \phi^* = \displaystyle\sum_{j=1}^{n^{\mathrm{c}}} \frac{q_j^2}{2k_j} + \sum_{j=1}^{n^{\mathrm{c}}} \ell_j^0 q_j + \sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{d}_i^{\top} \boldsymbol{v}_i \\
\text{s.t.} & \displaystyle\sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{B}_i^{\top} \boldsymbol{v}_i = \sum_{l \in \mathcal{L}_{\mathrm{c}}} \boldsymbol{A}_l^{\top} \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\mathrm{f}}} \boldsymbol{C}_l^{\top} \boldsymbol{f}_l, \\
& q_j \geq \| \boldsymbol{v}_i \| \quad (i \in \mathcal{I}_j;\ j = 1, \ldots, n^{\mathrm{c}}), \\
& \boldsymbol{\tau}_l \in \mathcal{G}^* \quad (l \in \mathcal{L}_{\mathrm{c}}).
\end{array}
\right. \tag{3.12}
$$

Hence, we can obtain the equilibrium configuration and the internal forces, respectively, of the cable network by solving the SOCP problems (3.11) and (3.12). It follows from Lemma 2.1 that $\boldsymbol{h}_i$ and $\boldsymbol{v}_i$ share a common direction as investigated in (2.10).

The authors have shown that ϕ^* in (3.12) is the complementary energy function for cable network in geometrical nonlinear theory [18]. In (3.12), we minimize ϕ^* over the equilibrium equations, and an optimal solution coincides with a set of internal forces and contact forces at the equilibrium state; i.e., (3.12) is the minimization problem of complementary energy with contact conditions only in terms of stress components, although it allows finite deformation.

Notice again that we can choose any matrix $\boldsymbol{T}_l$ satisfying Assumption 3.1, because the solution to (3.7)–(3.10) does not depend on $\boldsymbol{T}_l$. Let y_j^*, q_j^* $(j = 1, \ldots, n^{\mathrm{c}})$, $\boldsymbol{h}_i^*$, $\boldsymbol{v}_i^*$ $(i = 1, \ldots, n^{\mathrm{m}})$, $\boldsymbol{\tau}_l^*$ $(l \in \mathcal{L}_{\mathrm{c}})$, and $\boldsymbol{x}^*$ solve the complementarity problem (3.7)–(3.10) for an appropriately defined $\boldsymbol{T}_l$. Suppose that we replace $\boldsymbol{T}_l$ with $\boldsymbol{T}_l'$ satisfying Assumption 3.1. Then $\boldsymbol{A}_l$ in (3.9) and (3.10) is replaced with $\boldsymbol{A}_l' := \boldsymbol{T}_l'(\boldsymbol{C}_l^{\mathrm{U}} - \boldsymbol{C}_l^{\mathrm{D}})$. Let $\boldsymbol{\tau}_l'$ denote the vector satisfying $\boldsymbol{A}_l^{\top} \boldsymbol{\tau}_l^* = (\boldsymbol{A}_l')^{\top} \boldsymbol{\tau}_l'$. Then we see that $\boldsymbol{v}_i^*$ $(i = 1, \ldots, n^{\mathrm{m}})$, $\boldsymbol{\tau}_l'$ $(l \in \mathcal{L}_{\mathrm{c}})$, and $\boldsymbol{x}^*$ satisfy (3.9) and (3.10). Thus, $\boldsymbol{x}^*$ remains to be a solution of (3.7)–(3.10). The contact forces $\boldsymbol{T}_l^{\top} \boldsymbol{\tau}_l$ $(l \in \mathcal{L}_{\mathrm{c}})$ in the global coordinate system are also independent of the choice $\boldsymbol{T}_l$, because we have $\boldsymbol{T}_l^{\top} \boldsymbol{\tau}_l^* = (\boldsymbol{T}_l')^{\top} \boldsymbol{\tau}_l'$.

4. Unilateral contact problem with rigid convex obstacles. In this section, we consider a class of frictionless unilateral contact problem, or Signorini's problem [20], where the rigid obstacle is modeled as the union of some convex polyhedra. Suppose that there exist some rigid obstacles $\mathcal{H}_p$ $(p \in \mathcal{P})$, each of which is a convex polyhedron fixed in three-dimensional space, and that $\mathcal{P}$ denotes the set of indices of $\mathcal{H}_p$. π_q denotes a plane which contains a surface of a polyhedron and is defined as

$$
\pi_q = \{ \boldsymbol{z} \in \mathbf{R}^3 \,|\, \boldsymbol{a}_q^{\top} (\boldsymbol{z} - \boldsymbol{b}_q) = 0 \} \quad (q \in \mathcal{Q}). \tag{4.1}
$$

Here, $\boldsymbol{a}_q \in \mathbf{R}^3$ is a constant unit outer normal vector along a surface of $\mathcal{H}_p$, $\boldsymbol{b}_q \in \mathbf{R}^3$ is a constant location vector on π_q, and $\mathcal{Q}$ is a set of indices of π_q. By using the appropriate subset $\mathcal{Q}_p$ of $\mathcal{Q}$, we can give the expression of $\mathcal{H}_p$ as

$$
\mathcal{H}_p = \bigcap_{q \in \mathcal{Q}_p} \{ \boldsymbol{z} \in \mathbf{R}^3 \,|\, \boldsymbol{a}_q^{\top} (\boldsymbol{z} - \boldsymbol{b}_q) \leq 0 \} \tag{4.2}
$$

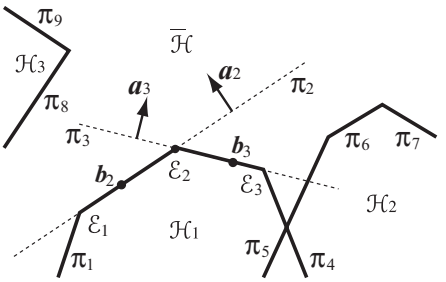


FIG. 4. *Definitions of $\mathcal{H}_p$ and π_q.*

without redundancy. An example is illustrated in Figure 4, where $\mathcal{P} = \{1, 2, 3\}$, $\mathcal{Q}_1 = \{1, 2, 3, 4\}$, $\mathcal{Q}_2 = \{5, 6, 7\}$, $\mathcal{Q}_3 = \{8, 9\}$, and $\mathcal{Q} = \{1, \ldots, 9\}$. Although we consider the polyhedra $\mathcal{H}_p$ ($p \in \mathcal{P}$) in three-dimensional space, Figure 4 illustrates only the cross sections of $\mathcal{H}_p$ so that a plane π_q is reduced to a line. Let $\mathcal{H} = \bigcup_{p\in\mathcal{P}} \mathcal{H}_p$. The admissible region $\mathcal{X}^{\mathrm{ad}} \subset \mathbf{R}^3$ of the location vector of any point on the cable network is given by

$$\mathcal{X}^{\mathrm{ad}} = \bigcap_{p\in\mathcal{P}} \left\{ \bigcup_{q\in\mathcal{Q}_p} \{\boldsymbol{z} \in \mathbf{R}^3 |\, \boldsymbol{a}_q^\top (\boldsymbol{z} - \boldsymbol{b}_q) \geq 0\} \right\}.$$

Although $\mathcal{H}_p$ is convex, $\mathcal{X}^{\mathrm{ad}}$ is not convex in general. The equilibrium configuration can be obtained by directly incorporating the constraints into the minimization problem of total potential energy such that all the points on the cable network should exist in $\mathcal{X}^{\mathrm{ad}}$. However, this problem is not tractable, since it is a nonconvex problem, not to mention the fact that it requires infinite number of constraints. As a conventional approach to contact problems [20], we may assign some contact candidates, each of which is a pair of the node of the cable network and the particle of the surface of $\mathcal{H}$. To verify that the cable network contacts $\mathcal{H}$ only at these candidates, we must solve complementarity problems successively with respect to infinitesimal displacements with a large number of candidates. This is the major difficulty of contact problems, and it motivates us to formulate an SOCP problem such that the optimal solution coincides with the equilibrium configuration under some weak assumptions.

Let $\mathcal{F}_q$ and $\mathcal{E}_j$ denote the faces and edges, respectively, of the polyhedra $\mathcal{H}_p$ ($p \in \mathcal{P}$). $\mathcal{J}$ denotes the set of indices of $\mathcal{E}_j$. By using the appropriate $\mathcal{Q}_j^{\mathcal{E}} \subseteq \mathcal{Q}_p$ satisfying $|\mathcal{Q}_j^{\mathcal{E}}| \geq 2$, we can write

$$\mathcal{F}_q = \mathcal{H}_p \cap \pi_q \qquad (p \in \mathcal{P},\ q \in \mathcal{Q}), \tag{4.3}$$

$$\mathcal{E}_j = \mathcal{H}_p \cap \left\{ \bigcap_{q\in\mathcal{Q}_j^{\mathcal{E}}} \pi_q \right\} \quad (p \in \mathcal{P},\ j \in \mathcal{J}). \tag{4.4}$$

Recall the example in Figure 4, where the cross sections of edges $\mathcal{E}_j$ are illustrated as the points. For the edges $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ of $\mathcal{H}_1$, we have $\mathcal{Q}_1^{\varepsilon} = \{1, 2\}$, $\mathcal{Q}_2^{\varepsilon} = \{2, 3\}$, and $\mathcal{Q}_3^{\varepsilon} = \{3, 4\}$. For the convex reformulation, we define contact candidates as

$$\boldsymbol{\zeta}_l^{\mathrm{aux}} = (i_l, j_l) \in \mathcal{N}^{\mathrm{m}} \times \mathcal{J} \quad (l \in \mathcal{L}^{\mathrm{aux}}),$$
$$\boldsymbol{\zeta}_l^{\mathrm{org}} = (l, q_l) \in \{l\} \times \mathcal{Q} \quad (l \in \mathcal{L}^{\mathrm{org}} \subseteq \mathcal{L}_{\mathrm{f}}).$$

For each $l \in \mathcal{L}^{\rm aux}$, the first and second elements of $\boldsymbol{\zeta}_l^{\rm aux}$ denote the indices of a member and an edge, respectively, where the i_lth member may contact $\mathcal{H}$ at the edge $\mathcal{E}_{j_l}$ at the equilibrium state. Then we set an auxiliary node $\boldsymbol{x}_l \in \mathbf{R}^3$ on the i_lth member, which is regarded as a frictionless joint. Note that these nodes are added to the list of unconstrained nodes. For each $l \in \mathcal{L}^{\rm org}$, q_l is defined as an index of a face, where the original node $\boldsymbol{x}_l$ may contact $\mathcal{H}$ on the face $\mathcal{F}_{q_l}$. $\mathcal{L}^{\rm aux}$ and $\mathcal{L}^{\rm org}$, respectively, denote the sets of indices of $\boldsymbol{\zeta}_l^{\rm aux}$ and $\boldsymbol{\zeta}_l^{\rm org}$, where $\mathcal{L}^{\rm aux} \cap \mathcal{L}^{\rm org} = \emptyset$.

Recall that $\mathcal{L}_{\rm f}$ denotes the set of all indices of original frictionless joints and pin-joints, which implies $\mathcal{L}^{\rm org} \subseteq \mathcal{L}_{\rm f}$. The auxiliary nodes are added to the list of the unconstrained nodes, i.e., $\mathcal{L}^{\rm aux} \cup \mathcal{L}_{\rm f} = \mathcal{N}^{\rm n}$ and $\mathcal{L}^{\rm aux} \cap \mathcal{L}_{\rm f} = \emptyset$. Define $\mathcal{G}_l \subset \mathbf{R}^3$ and $\mathcal{X}_l \subset \mathbf{R}^3$ as

$$\mathcal{G}_l = \begin{cases} \{\boldsymbol{z} \in \mathbf{R}^3 | \, \boldsymbol{a}_q^\top \boldsymbol{z} \geq 0 \ (q \in \mathcal{Q}_{j_l}^{\mathcal{E}})\} & (l \in \mathcal{L}^{\rm aux}), \\ \{\boldsymbol{z} \in \mathbf{R}^3 | \, \boldsymbol{a}_{q_l}^\top \boldsymbol{z} \geq 0\} & (l \in \mathcal{L}^{\rm org}), \end{cases}$$

$$\mathcal{X}_l = \begin{cases} \{\boldsymbol{z} \in \mathbf{R}^3 | \, \boldsymbol{a}_q^\top (\boldsymbol{z} - \boldsymbol{b}_q) \geq 0 \ (q \in \mathcal{Q}_{j_l}^{\mathcal{E}})\} & (l \in \mathcal{L}^{\rm aux}), \\ \{\boldsymbol{z} \in \mathbf{R}^3 | \, \boldsymbol{a}_{q_l}^\top (\boldsymbol{z} - \boldsymbol{b}_q) \geq 0\} & (l \in \mathcal{L}^{\rm org}), \end{cases}$$

where $(i_l, j_l) = \boldsymbol{\zeta}_l^{\rm aux}$ for $l \in \mathcal{L}^{\rm aux}$ and $(l, q_l) = \boldsymbol{\zeta}_l^{\rm org}$ for $l \in \mathcal{L}^{\rm org}$. We assume that the cable network may contact $\mathcal{H}$ only at the contact candidates $l \in \mathcal{L}^{\rm aux} \cup \mathcal{L}^{\rm org}$. Moreover, these contact candidates should be chosen to satisfy the following assumption.

Assumption 4.1. Define the actual equilibrium configuration as the equilibrium configuration of the cable network, with the subsidiary conditions such that any point on the cables should exist in $\mathcal{X}^{\rm ad}$. We assume that the actual equilibrium configuration satisfies the following assumptions:

(i) for each $l \in \mathcal{L}^{\rm org}$, $\boldsymbol{x}_l \in \mathcal{X}_l$ is satisfied;

(ii) for each $l \in \mathcal{L}^{\rm aux}$, $\boldsymbol{\zeta}_l = (i_l, j_l)$ satisfies that the i_lth member and $\mathcal{X}_l$ have a nonempty intersection.

Let $\boldsymbol{e}_j \in \mathbf{R}^3$ denote a position vector on the edge $\mathcal{E}_j$ $(j \in \mathcal{J})$ or on the face $\mathcal{F}_q$ $(q \in \mathcal{Q})$. Define the vector $\boldsymbol{g}_l \in \mathbf{R}^3$ as

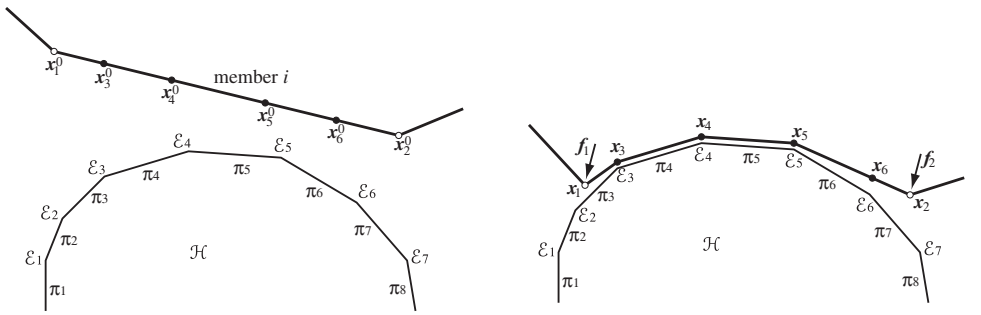
$$\boldsymbol{g}_l = \boldsymbol{x}_l - \boldsymbol{e}_l \quad (l \in \mathcal{L}^{\rm aux} \cup \mathcal{L}^{\rm org}), \tag{4.5}$$

where $\boldsymbol{x}_l$ is related to $\boldsymbol{x}$ by (2.13). Then conditions (i) and (ii) in Assumption 4.1 are equivalent to $\boldsymbol{g}_l \in \mathcal{G}_l$ $(l \in \mathcal{L}^{\rm org} \cup \mathcal{L}^{\rm aux})$.

Note that, for $l \in \mathcal{L}^{\rm aux}$, $\boldsymbol{g}_l$ defined by (4.5) is different from the conventional gap vector in contact mechanics. Let $\boldsymbol{p}_l^0 \in \mathbf{R}^3$ denote the coordinate of the lth node, which is not a frictionless joint but a pin-joint, on the lth member at the undeformed state. For $l \in \mathcal{L}^{\rm aux}$, the conventional gap vector is defined as the unique vector $\boldsymbol{g}_l' := \boldsymbol{p}_l^0 - \boldsymbol{e}_l'$, where $\boldsymbol{p}_l$ may contact the obstacle at the point $\boldsymbol{e}_l'$. Thus, we must specify the position vector $\boldsymbol{e}_l'$ on the obstacle. However, it is difficult to estimate such $\boldsymbol{e}_l'$ and $\boldsymbol{p}_l^0$ a priori if we allow the large deformation in the formulations. On the other hand, $\boldsymbol{g}_l$ $(l \in \mathcal{L}^{\rm aux})$ in (4.5) is defined with respect to the nodal coordinates at the deformed state and an arbitrary position vector $\boldsymbol{e}_l$ on the edge. Hence, we have to estimate neither $\boldsymbol{e}_l'$ nor $\boldsymbol{p}_l^0$. Moreover, by introducing $\boldsymbol{g}_l$ along with the concept of frictionless auxiliary joints, we can formulate the contact conditions allowing large deformations in the linear complementarity conditions over convex cones.

On the contrary, for $l \in \mathcal{L}^{\rm org}$, we have to estimate the face $\mathcal{F}_l$ corresponding to the lth node, which is the same task as that required in conventional formulation.

Since we use $\boldsymbol{g}_l$ defined by (4.5) instead of the conventional gap vector, it is not clear what kind of constraints should be included in the minimization problem of



(a) undeformed state (b) equilibrium state

FIG. 5. *Example of Remark* 1.

total potential energy in order to find the equilibrium configurations. Notice here that $\boldsymbol{g}_l = \boldsymbol{0}$ if and only if $\boldsymbol{x}_l$ is in contact with $\mathcal{H}$. Clearly, Assumption 4.1 requires no information as to whether $\boldsymbol{x}_l$ will contact $\mathcal{H}$ or not. It should be emphasized that we need not identify the coordinate of a contact node $\boldsymbol{p}_l^0$ on members, but need only identify a pair of a member and an edge.

Remark 1. The advantage of our approach relies on the fact that, in practical situations, the candidates $\boldsymbol{\zeta}_l^{\mathrm{aux}}$ satisfying Assumption 4.1 often can be easily found. Indeed, if we can find $\boldsymbol{\zeta}_l^{\mathrm{org}}$ satisfying $\boldsymbol{x}_l \in \mathcal{X}_l$ ($l \in \mathcal{L}^{\mathrm{org}}$), it is rather easy to look for appropriate $\boldsymbol{\zeta}_l^{\mathrm{aux}}$; i.e., if we know the face $\mathcal{F}_{q_l}$ satisfying $\boldsymbol{x}_l \in \mathcal{X}_l$ for each $\boldsymbol{x}_l$ ($l \in \mathcal{L}^{\mathrm{org}}$), we can easily conjecture whether the i'th member has a nonempty intersection with $\mathcal{X}_{j'}$ or not. If the conjecture has a positive answer, we should add a candidate $\boldsymbol{\zeta}_l^{\mathrm{aux}} = (i', j')$. Consequently, Assumption 4.1 is practically acceptable, even for very complicated obstacles and/or cable networks, if we can estimate the correct $\boldsymbol{\zeta}_l^{\mathrm{org}} = (l, q_l)$. For example, consider the ith member and the obstacle $\mathcal{H}$, as illustrated in Figure 5, where undeformed cable and only the cross section of the obstacle are illustrated in Figure 5(a). Figure 5(b) illustrates the equilibrium configuration under the existence of external forces. $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the original nodes, i.e., $\mathcal{L}_{\mathrm{f}} = \{1, 2\}$. The surface of $\mathcal{H}$ consists of the planes $\pi_1, \ldots, \pi_8$. The edges $\mathcal{E}_1, \ldots, \mathcal{E}_7$ are defined by $\mathcal{Q}_j^{\mathcal{E}} = \{j, j+1\}$ ($j = 1, \ldots, 7$); i.e., $\mathcal{E}_j$ is the intersection of π_j and π_{j+1}. We may regard all of the original nodes as the contact candidates, i.e., $\mathcal{L}^{\mathrm{org}} = \mathcal{L}_{\mathrm{f}}$. Suppose that we guess that

$$\boldsymbol{a}_3^\top (\boldsymbol{x}_1 - \boldsymbol{b}_3) \geq 0, \quad \boldsymbol{a}_7^\top (\boldsymbol{x}_2 - \boldsymbol{b}_7) \geq 0 \tag{4.6}$$

are satisfied at the actual equilibrium state; i.e., $q_1 = 3$ and $q_2 = 7$ in accordance with Assumption 4.1. From this, we can see that the ith member may possibly contact $\mathcal{H}$ only at $\mathcal{E}_3, \ldots, \mathcal{E}_6$. Hence, we should consider more candidates $\boldsymbol{x}_3, \ldots, \boldsymbol{x}_6$, i.e., $\mathcal{L}^{\mathrm{aux}} = \{3, 4, 5, 6\}$ and $\boldsymbol{\zeta}_l^{\mathrm{aux}} = (i, j)$ ($j = 3, \ldots, 6$). Here, the definitions of $\mathcal{X}_1$ and $\mathcal{X}_6$ are illustrated in Figure 6. If (4.6) holds at the actual equilibrium state, it is easy to see that the actual equilibrium configuration satisfies condition (ii) in Assumption 4.1; i.e., the ith member has a nonempty intersection with each of $\mathcal{X}_3, \ldots, \mathcal{X}_6$. Hence, in this example, we only have to find appropriate q_1 and q_2. □

Let $\boldsymbol{\tau}_l \in \mathbf{R}^3$ ($l \in \mathcal{L}_{\mathrm{c}}$) denote the vector of contact forces at $\boldsymbol{x}_l$. The equilibrium equations are written as

$$\sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{B}_i^\top \boldsymbol{v}_i = \sum_{l \in \mathcal{L}^{\mathrm{aux}} \cup \mathcal{L}^{\mathrm{org}}} \boldsymbol{C}_l^\top \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\mathrm{f}}} \boldsymbol{C}_l^\top \boldsymbol{f}_l. \tag{4.7}$$

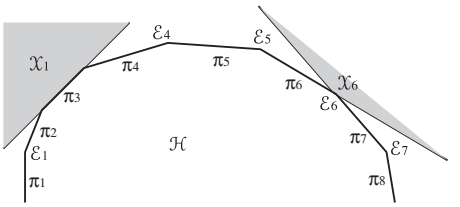


FIG. 6. *Definitions of* $\mathcal{X}_1$ *and* $\mathcal{X}_6$.

Notice here that we do not consider the original nonpenetration constraints such that any point on the cable network should exist in $\mathcal{X}^{\rm ad}$ in what follows. Moreover, for $l \in \mathcal{L}^{\rm aux}$, the definition of $\mathcal{X}_l$ does not coincide with the conventional admissible region of the position vector of contact candidate node. Therefore, it is not clear at present whether the original nonpenetration conditions can be expressed by using some conditions in terms of the kinematic and static variables of auxiliary nodes.

Since the i_lth member may contact the edge $\mathcal{E}_{j_l}$ without friction, $\boldsymbol{\tau}_l \in \mathcal{G}_l^*$ is satisfied, where the dual cone $\mathcal{G}_l^*$ of $\mathcal{G}_l$ is obtained as

$$\mathcal{G}_l^* = \begin{cases} \left\{ \sum_{q \in \mathcal{Q}_{j_l}^{\mathcal{E}}} \lambda_q \boldsymbol{a}_q \middle| \lambda_q \geq 0 \ (q \in \mathcal{Q}_{j_l}^{\mathcal{E}}) \right\} & (l \in \mathcal{L}^{\rm aux}), \\ \{ \lambda_{q_l} \boldsymbol{a}_{q_l} | \lambda_{q_l} \geq 0 \} & (l \in \mathcal{L}^{\rm org}). \end{cases}$$

For $l \in \mathcal{L}^{\rm aux}$, the i_lth member may contact $\mathcal{H}$ at the edge $\mathcal{E}_{j_l}$, and the contact force $\boldsymbol{\tau}_l$ should be expressed as the nonnegative linear combination of $\boldsymbol{a}_q$'s, each of which is normal to the plane π_q that includes $\mathcal{E}_{j_l}$. On the contrary, for $l \in \mathcal{L}^{\rm org}$, the original node $\boldsymbol{x}_l$ may contact the face $\mathcal{F}_{q_l}$, and the contact force $\boldsymbol{\tau}_l$ should be in the direction of the outer normal vector of $\mathcal{F}_{q_l}$.

Recall that $\boldsymbol{x}_l$ ($l \in \mathcal{L}^{\rm aux}$) is regarded as the frictionless joint. If the i_lth member does not contact the edge $\mathcal{E}_{j_l}$ at the actual equilibrium state, then $\boldsymbol{x}_l$ satisfies the equilibrium conditions at any point on the i_lth member. Assumption 4.1 guarantees the existence of a point in the intersection of $\mathcal{X}_l$ and the i_lth member at the actual equilibrium state. On the other hand, if the i_lth member contacts $\mathcal{E}_{j_l}$ at the actual equilibrium state, then the intersection of the i_lth member and $\mathcal{X}_l$ is reduced to a point in $\mathcal{E}_{j_l} \cap \mathcal{X}_l$. Hence, at each contact candidate $l \in \mathcal{L}^{\rm aux} \cup \mathcal{L}^{\rm org}$, the contact condition can be written as

$$\boldsymbol{\tau}_l^\top \boldsymbol{g}_l = 0, \quad \boldsymbol{g}_l \in \mathcal{G}_l, \quad \boldsymbol{\tau}_l \in \mathcal{G}_l^*, \tag{4.8}$$

which is a linear complementarity condition.

It follows from Lemma 2.1, (4.7), and (4.8) that the equilibrium configuration of the cable network can be obtained as a solution to the following system:

$$\text{(CP)}: \quad q_j = k_j y_j \quad (j = 1, \ldots, n^{\rm c}), \tag{4.9}$$

$$q_j (y_j + \ell_j^0) - \sum_{i \in \mathcal{I}_j} \boldsymbol{v}_i^\top \boldsymbol{h}_i = 0,$$

$$y_j + \ell_j^0 \geq \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|, \quad q_j \geq \|\boldsymbol{v}_i\| \quad (i \in \mathcal{I}_j;\ j = 1, \ldots, n^{\rm c}), \tag{4.10}$$

$$\boldsymbol{\tau}_l^\top \boldsymbol{g}_l = 0, \quad \boldsymbol{g}_l \in \mathcal{G}_l, \quad \boldsymbol{\tau}_l \in \mathcal{G}_l^* \quad (l \in \mathcal{L}^{\rm aux} \cup \mathcal{L}^{\rm org}), \tag{4.11}$$

$$\sum_{i=1}^{n^{\rm m}} \boldsymbol{B}_i^\top \boldsymbol{v}_i = \sum_{l \in \mathcal{L}^{\rm aux} \cup \mathcal{L}^{\rm org}} \boldsymbol{C}_l^\top \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\rm f}} \boldsymbol{C}_l^\top \boldsymbol{f}_l. \tag{4.12}$$

Since both $\mathcal{G}$ and $\mathcal{G}^*$ can be represented as the second-order cones, the system (4.9)–(4.12) can also be embedded into (2.5). In a manner similar to that in section 3, we can show that (4.9)–(4.12) coincide with the KKT conditions of the following primal-dual pair of SOCP problems:

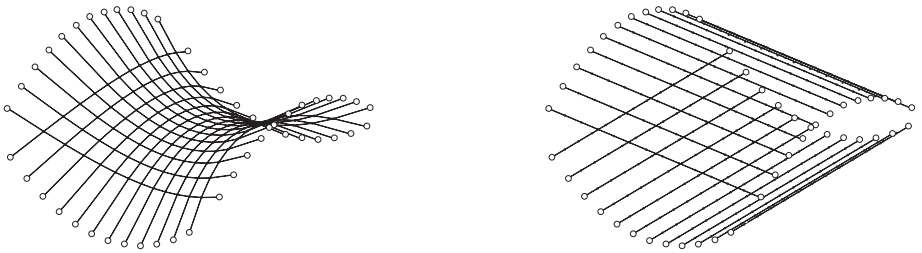
$$
(4.13)\qquad \begin{cases} \min & \displaystyle\sum_{j=1}^{n^{\mathrm{c}}} \frac{1}{2} k_j y_j^2 - \sum_{l \in \mathcal{L}_{\mathrm{f}}} \boldsymbol{f}_l^{\top} \boldsymbol{C}_l \boldsymbol{x} \\ \text{s.t.} & y_j + \ell_j^0 \geq \displaystyle\sum_{i \in \mathcal{I}_j} \| \boldsymbol{B}_i \boldsymbol{x} - \boldsymbol{d}_i \| \quad (j = 1, \ldots, n^{\mathrm{c}}), \\ & \boldsymbol{C}_l \boldsymbol{x} \in \mathcal{X}_l \quad (l \in \mathcal{L}^{\mathrm{aux}} \cup \mathcal{L}^{\mathrm{org}}), \end{cases}
$$

$$
(4.14)\qquad \begin{cases} \min & \displaystyle\sum_{j=1}^{n^{\mathrm{c}}} \frac{q_j^2}{2k_j} + \sum_{j=1}^{n^{\mathrm{c}}} \ell_j^0 q_j + \sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{d}_i^{\top} \boldsymbol{v}_i + \sum_{l \in \mathcal{L}^{\mathrm{aux}} \cup \mathcal{L}^{\mathrm{org}}} \sum_{q \in \mathcal{Q}_{j_l}^{\mathcal{E}}} \boldsymbol{b}_q^{\top} \lambda_q^l \boldsymbol{a}_q \\ \text{s.t.} & \displaystyle\sum_{i=1}^{n^{\mathrm{m}}} \boldsymbol{B}_i^{\top} \boldsymbol{v}_i = \sum_{l \in \mathcal{L}^{\mathrm{aux}} \cup \mathcal{L}^{\mathrm{org}}} \boldsymbol{C}_l^{\top} \boldsymbol{\tau}_l + \sum_{l \in \mathcal{L}_{\mathrm{f}}} \boldsymbol{C}_l^{\top} \boldsymbol{f}_l, \\ & q_j \geq \| \boldsymbol{v}_i \| \quad (i \in \mathcal{I}_j;\ j = 1, \ldots, n^{\mathrm{c}}), \\ & \boldsymbol{\tau}_l = \displaystyle\sum_{q \in \mathcal{Q}_{j_l}^{\mathcal{E}}} \lambda_q^l \boldsymbol{a}_q \in \mathcal{G}_l^* \quad (l \in \mathcal{L}^{\mathrm{aux}}), \\ & \boldsymbol{\tau}_l = \lambda_{q_l} \boldsymbol{a}_{q_l} \in \mathcal{G}_l^* \quad (l \in \mathcal{L}^{\mathrm{org}}). \end{cases}
$$

Note that (4.13) is independent of both the stress states and contact states of cable members. Moreover, (4.13) and (4.14) are embedded into the dual and primal standard forms of SOCP problems (2.4) and (2.3), respectively, which can be solved easily by using the primal-dual interior-point method [7].

5. Numerical examples. In order to obtain the equilibrium configurations of cable networks, the presented SOCP problems are transformed into the pair of standard forms (2.3) and (2.4) and are solved by using SeDuMi [39], which implements the self-dual embedding technique for optimization over self-dual homogeneous cones and operates under the MATLAB environment. Computation has been carried out on a Pentium III (844MHz with 256MB memory) with MATLAB Version 6.5 [26] and SeDuMi Version 1.05. In the following examples, the elastic modulus and the cross-sectional area, respectively, are specified as $E = 205.8$ GPa and $A_j = 10^{-2}$ cm^2 for each cable, and the elongation stiffness is computed as $k_j = EA_j/\ell_j^0$.

5.1. Two-body contact problems of cable networks. Equilibrium configurations are found for two cable networks with 10×10 and 12×12 grids, which are referred to as Models (I) and (II), respectively. Each cable network projected to the horizontal plane makes a grid with 1 m $\times$ 1 m squares. The (x, y)-axes are defined along the grid, and the origin is set at the center of each cable network. Then the z-coordinates of the nodes are given as $z = (x^2 - y^2)/(4\alpha)$, where $\alpha = 11$ and 13 (m), respectively, for Models (I) and (II). The nodes indicated by circles in Figure 7(a) are supported. n^{d}, n^{m}, and n^{c} of each model are listed in Table 1. For each contact candidate $\boldsymbol{\zeta}_l$ $(l \in \mathcal{L}_{\mathrm{c}})$, there exist two auxiliary nodes. However, from the definition (3.4) of $\mathcal{G}$, the constraint $\boldsymbol{A}_l \boldsymbol{x} \in \mathcal{G}$ in Problem (3.11) implies that, for each pair of auxiliary nodes related to $\boldsymbol{\zeta}_l$, the number of degrees of freedom becomes four.



(a) Model (II)-(a) (b) Model (II)-(b)

FIG. 7. *Model* (II).

TABLE 1
Results of Models (I) *and* (II).

		m	n	CPU	n^{d}	n^{m}	n^{c}	Iter.	$\mathcal{K}$
(I)	Pin	740	1540	1.44	300	220	—	15	$(\mathbf{L}^3_+)^{220} \times (\mathbf{L}^4_+)^{220}$
	Frictionless	540	940	2.85	300	220	20	24	$(\mathbf{L}^3_+)^{20} \times (\mathbf{L}^4_+)^{220}$
	Contact-(a)	640	1040	2.65	400	220	20	20	$\mathbf{R}^{100}_+ \times (\mathbf{L}^3_+)^{20} \times (\mathbf{L}^4_+)^{220}$
	Contact-(b)	640	1040	2.46	400	220	20	20	$\mathbf{R}^{100}_+ \times (\mathbf{L}^3_+)^{20} \times (\mathbf{L}^4_+)^{220}$
(II)	Contact-(a)	912	1464	4.23	576	312	24	22	$\mathbf{R}^{144}_+ \times (\mathbf{L}^3_+)^{24} \times (\mathbf{L}^4_+)^{312}$
	Contact-(b)	912	1464	3.98	576	312	24	21	$\mathbf{R}^{144}_+ \times (\mathbf{L}^3_+)^{24} \times (\mathbf{L}^4_+)^{312}$

Set $\ell^0_1 = \cdots = \ell^0_{n^{\mathrm{c}}} = 8$ (m) and $k_1 = \cdots = k_{n^{\mathrm{c}}}$. The self-equilibrium configurations with $\boldsymbol{f} = \boldsymbol{0}$ are computed. Note that the self-equilibrium configurations are independent of k_j. We assume $\boldsymbol{T}_l = \boldsymbol{I}$ (for all $l \in \mathcal{L}_{\mathrm{c}}$) in (3.1), where $\boldsymbol{I} \in \mathbf{R}^{3\times 3}$ denotes a unit matrix. We consider two problems for each model, which are referred to as *contact*-(a) and *contact*-(b), respectively. In the problem contact-(a), we suppose that all the cables parallel to the y-axis are above the cables parallel to the x-axis. In contrast, in the problem contact-(b), all the cables parallel to the x-axis are supposed to be above the cables parallel to the y-axis.

The primal-dual pair of problems (3.11) and (3.12) is solved in order to obtain the configurations, internal forces, and contact forces at the equilibrium state. For each model, m, n, $\mathcal{K}$, CPU time in seconds, and number of iterations are listed in Table 1. The equilibrium configurations of Model (II) are shown in Figure 7. Note that all the pairs of contact candidate members are in contact state at a solution of contact-(a), whereas they are free at a solution of contact-(b). We see that the obtained configuration of contact-(a) is the same as the case where all the internal joints are frictionless in the direction of the cables, while all the cables of contact-(b) make straight lines. It can be observed from Table 1 that the CPU time does not increase drastically even if we incorporate the contact conditions as additional constraints.

5.2. Signorini's problem of a cable. Consider a cable with one (original) frictionless joint and a fixed parabolic rigid obstacle $\mathcal{H}$. We specify the initial unstressed length $\ell^0_j = 50.0$ cm of the cable, and the two end points are fixed. $\mathcal{H}$ is approximated as a convex polyhedron, the surface of which consists of 18 planes as

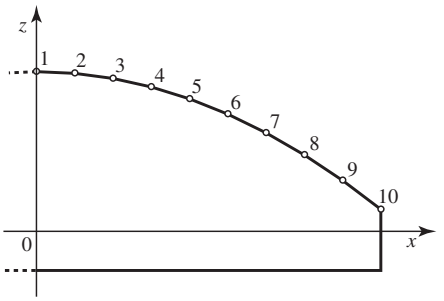
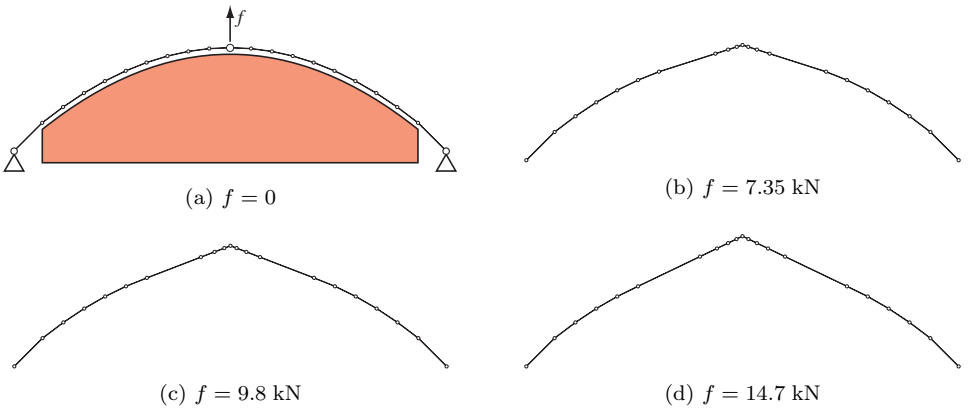


FIG. 8. *Symmetric half-part of* $\mathcal{H}$.

TABLE 2
Nodal coordinates (cm) *of* $\mathcal{H}$.

Edge	x	z
(1)	0.0	10.0
(2)	2.2222222	9.9012346
(3)	4.4444444	9.6049383
(4)	6.6666667	9.1111112
(5)	8.8888889	8.4197531
(6)	11.111111	7.5308643
(7)	13.333333	6.4444445
(8)	15.555556	5.1604939
(9)	17.777778	3.6790124
(10)	20.000000	1.1879670



(a) $f = 0$ (b) $f = 7.35$ kN

(c) $f = 9.8$ kN (d) $f = 14.7$ kN

FIG. 9. *Equilibrium configurations of cable.*

shown in Figure 8, where the coordinates of the edges are listed in Table 2 for one of two symmetric parts. Thus, we have 18 auxiliary frictionless joints and one original frictionless joint, i.e., $n^{\mathrm{d}} = 57$, $n^{\mathrm{m}} = 20$, and $n^{\mathrm{c}} = 1$.

The external force f is applied in the positive direction of the z-axis at the original frictionless joint. The equilibrium configuration without external force is as shown in Figure 9(a), where the cable is in tension and contacts $\mathcal{H}$ at all the auxiliary joints. The pair of standard forms (2.3) and (2.4) with $\mathcal{K} = \mathbf{R}_+^{38} \times \mathbf{L}_+^3 \times (\mathbf{L}_+^4)^{20}$ is solved for various values of f, and the corresponding equilibrium configurations are as shown in Figure 9(b)–(d).

5.3. 2 × 9 cable network with rigid obstacle. Consider a cable network, the image of which projected to the horizontal plane makes a grid with 10 cm × 10.2 cm squares, illustrated in Figure 10. The nodes indicated by ○ are fixed, where the z-coordinates (cm) of the nodes are given as $z = -\gamma(y/1.02 - 5)(y/1.02 + 5) - 7$ with a positive parameter γ. The nodes indicated by ⊕ are also fixed in the xy-plane. We consider the obstacle $\mathcal{H}$ defined by (4.2) with $\mathcal{P} = \{1\}$, $\mathcal{Q}_1 = \{1, 2\}$, $\boldsymbol{a}_1 = (-3/\sqrt{109}, 0, 10/\sqrt{109})$, $\boldsymbol{a}_2 = (3/\sqrt{109}, 0, 10/\sqrt{109})$, and $\boldsymbol{b}_1 = \boldsymbol{b}_2 = (0, 0, -3)$. The nodes indicated by ⊗ and ● are pin-joints and auxiliary frictionless joints, respectively, where $n^{\mathrm{m}} = 56$, $n^{\mathrm{c}} = 47$, and $n^{\mathrm{d}} = 81$. The initial unstressed length of each cable is specified as $\ell_j^0 = 10.3$ cm. The external force 49 kN is applied at each node indicated by ⊗ in the negative direction of the z-axis. The equilibrium configuration at $\gamma = 0.4$ without external forces is illustrated in Figure 11.

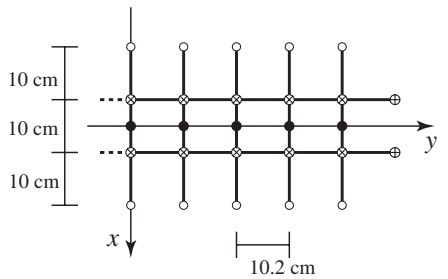


FIG. 10. *Symmetric half-part of Model* (III).

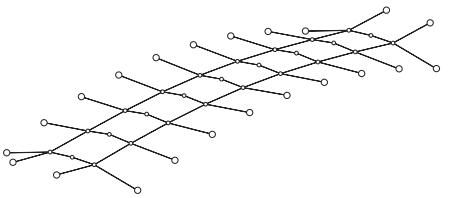


FIG. 11. *Equilibrium configuration of Model* (III) ($f = 0$, $\gamma = 0.4$).

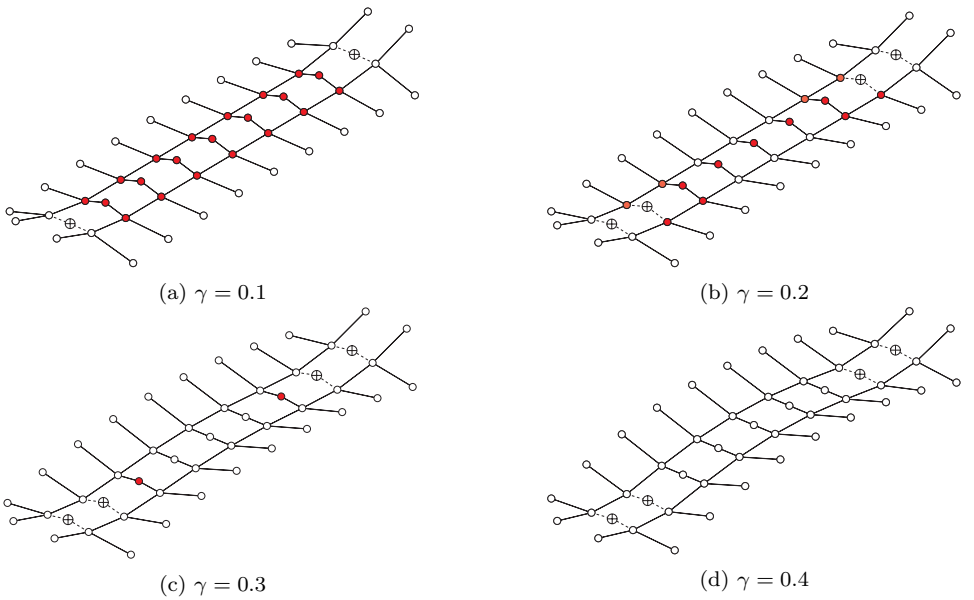


FIG. 12. *Equilibrium configurations of Model* (III).

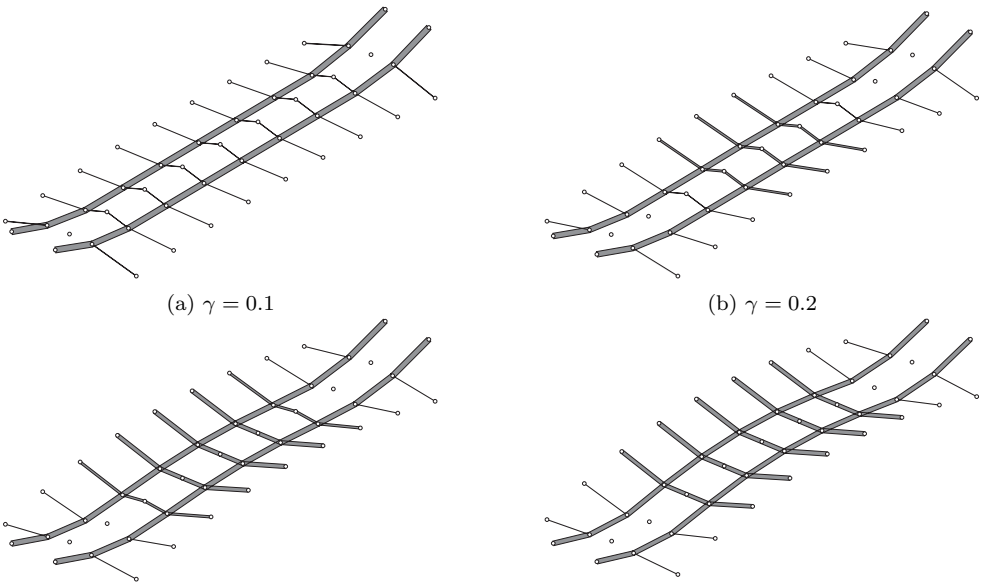
For various magnitudes of γ, the primal-dual pair of problems (4.13) and (4.14) is solved, where $\mathcal{K} = \mathbf{R}_+^{36} \times (\mathbf{L}_+^3)^{47} \times (\mathbf{L}_+^4)^{56}$, $m = 184$, and $n = 401$ in (2.4). The obtained configurations are shown in Figure 12. The cable network contacts $\mathcal{H}$ at the nodes indicated by $\bullet$ and $\oplus$, respectively, with and without contact forces. The dotted lines indicate slackening members. The stress state of each case is shown in Figure 13, where the width of each member is proportional to its axial force.

6. Proof of Lemma 2.1. Observe that the "only if" part of the lemma is immediately shown by substituting (2.7)–(2.9) and (2.12) into (2.11).

Next, we show the "if" part; i.e., suppose that (2.7) and (2.11) hold. The last inequality in (2.11) implies $q_j \geq 0$. On the other hand, from the first equality in (2.11), we have $y_j = q_j/k_j$. Consequently, we obtain

$$y_j \geq 0. \tag{6.1}$$

The two situations such that $y_j + \ell_j^0 = \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|$ and $y_j + \ell_j^0 > \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|$ are



(a) $\gamma = 0.1$ (b) $\gamma = 0.2$

(c) $\gamma = 0.3$ (d) $\gamma = 0.4$

FIG. 13. *Stress states of Model* (III).

investigated. Assume

$$y_j + \ell_j^0 = \sum_{i \in \mathcal{I}_j} \|\boldsymbol{h}_i\|. \tag{6.2}$$

From (2.7) and (6.2), we see that

$$y_j = c_j. \tag{6.3}$$

It is easy to see that (6.2) implies that

$$q_j(y_j + \ell_j^0) - \sum_{i \in \mathcal{I}_j} \boldsymbol{v}_i^\top \boldsymbol{h}_i = \sum_{i \in \mathcal{I}_j} (q_j \|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i). \tag{6.4}$$

By substituting Schwarz's inequality $-\boldsymbol{v}_i^\top \boldsymbol{h}_i \geq -\|\boldsymbol{v}_i\|\|\boldsymbol{h}_i\|$ into (6.4), we obtain

$$q_j \|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i \geq (q_j - \|\boldsymbol{v}_i\|)\|\boldsymbol{h}_i\| \geq 0, \tag{6.5}$$

where the second inequality follows the fourth condition in (2.11). Observe that, as a consequence of (6.4), the second condition in (2.11) is satisfied if and only if

$$q_j \|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i = 0 \quad (i \in \mathcal{I}_j). \tag{6.6}$$

If $\boldsymbol{h}_i = \boldsymbol{0}$, then it is easy to see that (2.9) is satisfied. On the other hand, if $\boldsymbol{h}_i \neq \boldsymbol{0}$ $(i \in \mathcal{I}_j)$, then it follows from the second inequality in (6.5) that (6.6) holds if and only if

$$q_j = \|\boldsymbol{v}_i\| \tag{6.7}$$

is satisfied. By substituting (6.7) into (6.6), we have $\|\boldsymbol{v}_i\|\|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i = 0$, which is reduced to (2.9). Moreover, (6.1) and (6.3) verify (2.12); from this and the first equality in (2.11) we obtain (2.8).

Alternatively, assume $y_j + \ell_j^0 > \sum_{i\in\mathcal{I}_j} \|\boldsymbol{h}_i\|$; from this and (2.7), we see

$$y_j > c_j. \tag{6.8}$$

Assume $q_j > 0$ for contradiction. It follows from the last condition in (2.11) and Schwarz's inequality that

$$q_j(y_j + \ell_j^0) - \sum_{i\in\mathcal{I}_j} \boldsymbol{v}_i^\top \boldsymbol{h}_i > \sum_{i\in\mathcal{I}_j} (q_j\|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i) \geq \sum_{i\in\mathcal{I}_j} (\|\boldsymbol{v}_i\|\|\boldsymbol{h}_i\| - \boldsymbol{v}_i^\top \boldsymbol{h}_i) \geq 0,$$

which contradicts the second condition in (2.11). Hence, $q_j = 0$ holds; from this and (6.8) and the first equality in (2.11) we obtain the second expression of (2.12). It is also seen that (2.8) is satisfied. The last inequality in (2.11) and $q_j = 0$ imply $\boldsymbol{v}_i = \boldsymbol{0}$ $(i \in \mathcal{I}_j)$, which verifies (2.9). ∎

7. Concluding remarks. In this paper, we have studied the contact problems of cable networks. These problems are characterized by the nonsmooth behaviors such that (i) some members of the cable network may possibly contact each other or the rigid obstacles at the equilibrium state and (ii) each member can transmit only tension force.

By introducing the concept of the auxiliary frictionless node, we define a contact candidate as a pair of two members of the cable network, or that of a member and an edge of the obstacle. Under some practically acceptable assumptions, we formulate the complementarity problems over second-order cones, which provides the configuration and contact forces at the equilibrium state. We show that the complementarity problem presented can be equivalently rewritten as a primal-dual pair of SOCP problems, which corresponds to the pair of minimization problems of total potential energy and complementary energy.

In the examples with various contact conditions, equilibrium configurations have been obtained by using the primal-dual interior-point method for SOCP. It has been shown in the examples that the computational cost of the proposed method does not increase drastically even if we incorporate the contact conditions as additional constraints. Although the proposed method requires the assumption that the candidate nodes should exist in some regions at the equilibrium state, it does not require any assumption on either contact conditions or stress states of cable members.

In addition to these advantages, the proposed SOCP problem can be solved efficiently by using well-developed software based on primal-dual interior-point methods. Therefore, our task is only to input the geometrical and material properties of cable networks, and no effort is required to develop any analysis software. Since the primal-dual interior-point methods solve both the primal and dual SOCP problems simultaneously, we can obtain the equilibrium configurations, cable elongation, internal forces, and contact forces simultaneously.

Acknowledgment. The authors thank two anonymous referees for their careful reading of the manuscript and their suggestions for improvements.

REFERENCES

[1] A. M. Al-Fahed, G. E. Stavroulakis, and P. D. Panagiotopoulos, *A linear complementarity approach to the frictionless gripper problem*, Internat. J. Robotics Res., 11 (1992), pp. 112–122.

[2] F. Alizadeh and D. Goldfarb, *Second-order cone programming*, Math. Program., 95 (2003), pp. 3–51.

[3] M. Anitescu and F. A. Porta, *A time-stepping method for stiff multibody dynamics with contact and friction*, Internat. J. Numer. Methods Engrg., 55 (2002), pp. 753–784.

[4] J. H. Argyris and D. W. Scharpf, *Large deflection analysis of prestressed networks*, J. Engrg. Mech., 98 (1972), pp. 633–654.

[5] A. A. Atai and D. J. Steigmann, *On the nonlinear mechanics of discrete networks*, Arch. Appl. Mech., 67 (1997), pp. 303–319.

[6] A. A. Atai and D. J. Steigmann, *Coupled deformations of elastic curves and surfaces*, Internat. J. Solids Structures, 35 (1998), pp. 1915–1952.

[7] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, MPS/SIAM Ser. Optim. 2, SIAM, Philadelphia, 2001.

[8] W. Bull, ed., *Computational Modeling of Masonry, Brickwork and Blockwork Structures*, Saxe-Coburg Publications, Stirling, UK, 2001.

[9] M. Cannarozzi, *A minimum principle for tractions in the elastostatics of cable networks*, Internat. J. Solids Structures, 23 (1987), pp. 551–568.

[10] M. Cuomo and G. Ventura, *A complementary energy formulation of no tension masonry-like solids*, Comput. Methods Appl. Mech. Engrg., 189 (2000), pp. 313–339.

[11] L. Demkowicz, *On some results concerning the reciprocal formulation for the Signorini's problem*, Comput. Math. Appl., 8 (1982), pp. 57–74.

[12] M. Epstein and M. A. Forcinito, *Anisotropic membrane wrinkling: Theory and analysis*, Internat. J. Solids Structures, 38 (2001), pp. 5253–5272.

[13] M. Fukushima, Z.-Q. Luo, and P. Tseng, *Smoothing functions for second-order-cone complementarity problems*, SIAM J. Optim., 12 (2001), pp. 436–460.

[14] D. Y. Gao, R. W. Ogden, and G. E. Stavroulakis, eds., *Nonsmooth/Nonconvex Mechanics*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.

[15] J. A. Garrido, A. Foces, and F. París, *An incremental procedure for three-dimensional contact problems with friction*, Comput. Structures, 50 (1994), pp. 201–215.

[16] S. Ghosh, *Arbitrary Lagrangian–Eulerian finite element analysis of large deformation in contacting bodies*, Internat. J. Numer. Methods Engrg., 33 (1992), pp. 1891–1925.

[17] F. Jarre, M. Kočvara, and J. Zowe, *Optimal truss design by interior-point methods*, SIAM J. Optim., 8 (1998), pp. 1084–1107.

[18] Y. Kanno and M. Ohsaki, *Minimum principle of complementary energy of cable networks by using second-order cone programming*, Internat. J. Solids Structures, 40 (2003), pp. 4437–4460

[19] Y. Kanno, M. Ohsaki, and J. Ito, *Large-deformation and friction analysis of nonlinear elastic cable networks by second-order cone programming*, Internat. J. Numer. Methods Engrg., 55 (2002), pp. 1079–1114.

[20] N. Kikuchi and J. T. Oden, *Contact Problem in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM Stud. Appl. Math. 8, SIAM, Philadelphia, 1988.

[21] M. Kojima, S. Mizuno, and A. Yoshise, *A primal-dual interior point algorithm for linear programming*, in Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods, N. Megiddo, ed., Springer-Verlag, New York, 1989, pp. 29–47.

[22] T. A. Laursen, *Computational Contact and Impact Mechanics*, Springer-Verlag, Berlin, 2002.

[23] W. Lie-heng, *On the duality methods for the contact problem in elasticity*, Comput. Methods Appl. Mech. Engrg., 167 (1998), pp. 275–282.

[24] M. S. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, *Applications of second-order cone programming*, Linear Algebra Appl., 284 (1998), pp. 193–228.

[25] K. Lu, M. Accorsi, and J. Leonard, *Finite element analysis of membrane wrinkling*, Internat. J. Numer. Methods Engrg., 50 (2001), pp. 1017–1038.

[26] The MathWorks, *Using MATLAB*, Natick, MA, 1999.

[27] S. D. Mesarovic and N. A. Fleck, *Frictionless indentation of dissimilar elastic-plastic spheres*, Internat. J. Solids Structures, 37 (2000), pp. 7071–7091.

[28] A. R. Mijar and J. S. Arora, *Review of formulations for elastostatic frictional contact problems*, Struct. Multidiscip. Optim., 20 (2000), pp. 167–189.

[29] A. R. Mijar and J. S. Arora, *Study of variational inequality and equality formulations for elastostatic frictional contact problems*, Arch. Comput. Methods Engrg., 7 (2000), pp. 387–449.

[30] R. D. C. Monteiro and T. Tsuchiya, *Polynomial convergence of primal-dual algorithms for the second-order cone program based on the MZ-family of directions*, Math. Program., 88 (2000), pp. 61–83.
[31] J. Outrata, M. Kočvara, and J. Zowe, *Nonsmooth Approach to Optimization with Equilibrium Constraints*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.
[32] P. D. Panagiotopoulos, *A variational inequality approach to the inelastic stress-unilateral analysis of cable-structures*, Comput. Structures, 6 (1976), pp. 133–139.
[33] J.-S. Pang and D. E. Stewart, *A unified approach to discrete frictional contact problems*, Internat. J. Engrg. Sci., 37 (1999), pp. 1747–1768.
[34] J.-S. Pang, J. C. Trinkle, and G. Lo, *A complementarity approach to a quasistatic multi-rigid-body contact problem*, Comput. Optim. Appl., 5 (1996), pp. 139–154.
[35] A. C. Pipkin, *Relaxed energy densities for large deformations of membranes*, IMA J. Appl. Math., 52 (1994), pp. 297–308.
[36] R. T. Rockafellar, *Lagrange multipliers and optimality*, SIAM Rev., 35 (1993), pp. 183–238.
[37] T. Sasakawa and T. Tsuchiya, *Optimal magnetic shield design with second-order cone programming*, SIAM J. Sci. Comput., 24 (2003), pp. 1930–1950.
[38] D. E. Stewart, *Rigid-body dynamics with friction and impact*, SIAM Rev., 42 (2000), pp. 3–39.
[39] J. F. Sturm, *Using SeDuMi* 1.02, *a MATLAB toolbox for optimization over symmetric cones*, Optim. Methods Softw., 11/12 (1999), pp. 625–653.
[40] T. Tsuchiya, *A convergence analysis of the scaling-invariant primal-dual path-following algorithms for second-order cone programming*, Optim. Methods Softw., 11/12 (1999), pp. 141–182.
[41] J. A. Tzitzouris and J.-S. Pang, *A time-stepping complementarity approach for frictionless systems of rigid bodies*, SIAM J. Optim., 12 (2002), pp. 834–860.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2053–2076
© 2006 Society for Industrial and Applied Mathematics

ON THE REGULARIZING POWER OF MULTIGRID-TYPE ALGORITHMS*

MARCO DONATELLI† AND STEFANO SERRA-CAPIZZANO†

Abstract. We consider the deblurring problem of noisy and blurred images in the case of known space invariant point spread functions with four choices of boundary conditions. We combine an algebraic multigrid previously defined ad hoc for structured matrices related to space invariant operators (Toeplitz, circulants, trigonometric matrix algebras, etc.) and the classical geometric multigrid studied in the partial differential equations context. The resulting technique is parameterized in order to have more degrees of freedom: a simple choice of the parameters allows us to devise a quite powerful regularizing method. It defines an iterative regularizing method where the smoother itself has to be an iterative regularizing method (e.g., conjugate gradient, Landweber, conjugate gradient for normal equations, etc.). More precisely, with respect to the smoother, the regularization properties are improved and the total complexity is lower. Furthermore, in several cases, when it is directly applied to the system $A\mathbf{f} = \mathbf{g}$, the quality of the restored image is comparable with that of all the best known techniques for the normal equations $A^TA\mathbf{f} = A^T\mathbf{g}$, but the related convergence is substantially faster. Finally, the associated curves of the relative errors versus the iteration numbers are "flatter" with respect to the smoother (the estimation of the stop iteration is less crucial). Therefore, we can choose multigrid procedures which are much more efficient than classical techniques without losing accuracy in the restored image (as often occurs when using preconditioning). Several numerical experiments show the effectiveness of our proposals.

Key words. image restoration, multigrid, iterative regularization methods

AMS subject classifications. 65F22, 65N55, 65F10, 94A08

DOI. 10.1137/040605023

1. Introduction.

We consider the classical deblurring problem of noisy and blurred images in the case of space invariant point spread functions (PSFs). More precisely the problem is described by the following discrete problem (possible infinite) with shift invariant kernel, which leads to a system whose ith equation is given by

$$g(i) = \sum_{j \in \mathbf{Z}^2} h_{i-j} f(j) + \nu(i), \qquad i \in \mathbf{Z}^2. \tag{1.1}$$

Here the mask $H = (h_s)_{s \in \mathbf{Z}^2}$ represents the blurring operator; $\nu(s)$, $s \in \mathbf{Z}^2$, is the noise contribution; and $g(s)$, $s \in \mathbf{Z}^2$, is the blurred and noisy observed image; the problem is to recover the unknown true image $f(s)$ in the window of observation described by $s \in \{1, \ldots, n\}^2$. We assume that H is known, e.g., through experimental measurements.

It is clear that the system described by (1.1) is underdetermined since we have n^2 equations and $(n+m)^2$ unknowns involved when the support of the PSF is a square of size $m \times m$. In general, if the support of the PSF contains more than one point, then the number of unknowns exceeds the number of equations by at least n. In order to take care of this problem we can compute a least square solution or use appropriate boundary conditions (linear or affine relations between the unknowns outside the

*Received by the editors March 11, 2004; accepted for publication (in revised form) June 1, 2005; published electronically February 3, 2006. This work was partially supported by MIUR grant 2004015437.

http://www.siam.org/journals/sisc/27-6/60502.html

†Dipartimento di Fisica e Matematica, Università dell'Insubria - Sede di Como, Via Valleggio 11, 22100 Como, Italy (marco.donatelli@uninsubria.it, stefano.serrac@uninsubria.it, serra@mail.dm.unipi.it).

window of observation and the unknowns inside the windows of observation) leading to a square linear system. The least square solution in principle is computationally very expensive even though recently a clever trick for reducing the related cost has been devised (see [4, 31]). The use of the boundary conditions (BCs) approach is usually preferred by practitioners. Among the BCs we count zero Dirichlet, periodic, reflective or Neumann, and antireflective. As proved in [19, 25], the last two BCs provide much more precise results with respect to the classical zero Dirichlet and periodic BCs. Moreover, in the noise-free case [25], the antireflective BCs improve by one order of approximation the restoration of the true object when compared with the reflective BCs. Indeed, the reflective and antireflective BCs have been introduced in order to reduce or eliminate some artifacts located close to the borders of the image and called ringing effects. These results are confirmed in the presence of noise also when the use of regularization methods is highly recommended: in [10] we applied Tikhonov-like methods and conjugate gradient (CG) with optimal parameters for a slight modification of the linear system called reblurred linear system. When using antireflective BCs, these techniques work better than the classical Tikhonov method (see [28]) and CG for normal equations with optimal parameters, while for the remaining BCs the reblurred modification and the classical methods coincide when the PSF is symmetric.

In this paper we propose the use of multigrid methods as regularizing procedures (for the notion of regularization, see [3, 11, 28]): in order to gain in efficiency, the multigrid methods that we consider are designed ad hoc for the structured linear systems arising from the various BCs. We observe two-level Toeplitz structures when applying zero Dirichlet BCs, we have two-level circulant matrices when applying periodic BCs, and we find two-level matrices from the DCT-III algebra and from the DST-I algebra when enforcing reflective and anti-reflective BCs, respectively. The good news is that for all these structured matrices very efficient multigrid methods have been devised [13, 26, 8]. When the two-level matrices are also banded at each level, then the related overall cost is of the order of $O(n^2)$ arithmetic operations; if the matrices considered are dense, then the cost is of the order of $O(n^2 \log(n))$ arithmetic operations (for the algorithmic proposals see [13, 26, 8], and for the theoretical cost and convergence analysis see [1, 24]). In conclusion, we can assume that from the viewpoint of the complexity the multigrid proposal is a very good one. Here, taking into account the presence of noise, we discuss how to modify the method in order to introduce proper regularization features; in fact, this is the main aim of the paper.

In part of the numerical experiments we will assume that the image is located in a uniform background (but this requirement is not absolutely essential as shown by the example in section 5.1, and our proposal works essentially unchanged with general images where we are forced to use a more precise BC such as the reflective or antireflective). Under these special assumptions (often fulfilled in astronomical imaging), the four mentioned BCs are equivalent from the model viewpoint, and indeed we do not observe ringing effects. Therefore, in this ideal setting we can concentrate on checking the quality of the reconstruction with respect to the deblurring/denoising procedure.

To this end many techniques have been proposed: Tikhonov's method, Riley's method (a variant of the Tikhonov technique when the blurring matrix is symmetric and close to positive definite), Landweber, CG and its version applied to normal equations (CGNE) with early stopping, and so on. Here we propose a new iterative technique with early stopping of the iterations, which is of multigrid type. We

combine an algebraic multigrid previously defined ad hoc for structured matrices related to space invariant operators (Toeplitz, circulants, trigonometric matrix algebras, etc.) and the classical geometric multigrid designed for discretized partial differential equations (PDEs). In particular the reduction to lower dimension is made by using the Galerkin approach (without rediscretization); the structure of the prolongation operator and of the restriction operator or projector (essentially the transpose of the prolongation operator) is that of the algebraic multigrid indicated in [1] in order to maintain the structure of the problem at each level; and finally, from the viewpoint of the compact Fourier analysis [29], the prolongation operator is that of the classical geometric multigrid, i.e., the one representing the linear interpolation (a low-pass filter). The resulting technique is parameterized in order to have more degrees of freedom. A simple choice of the parameters allows one to devise a powerful regularizing method whose features are the following:

(a) It is used with early stopping (as in CG, CGNE, and the Landweber method), and its cost per iteration is about 1/3 of the cost of the method used as a smoother (CG, Landweber, CGNE);
(b) it can be adapted to work with all the BCs used in the literature (Dirichlet, periodic, Neumann/reflective or antireflective) since the basic algebraic multigrid considered in [1] is an optimally convergent method for any of the involved structures (Toeplitz, circulant, cosine-algebra, or sine-algebra) which naturally arise from the chosen BCs;
(c) the minimal relative error with respect to the true image is significantly lower with regard to all the best known techniques directly applied to the system $A\mathbf{f} = \mathbf{g}$ (Riley, CG, preconditioned CG, etc.), with optimal parameters, and the associated curve of the relative errors with respect to the iterations is "flatter" (at least in our set of experiments);
(d) when the smoothing step is applied to the normal equations $A^TA\mathbf{f} = A^T\mathbf{g}$ we observe that the minimal relative error and the total complexity are slightly lower in comparison with all the best known techniques (Tikhonov, CGNE, Landweber, etc.) with optimal parameters, and the associated curve of the relative errors versus the iterations number is, at least in our set of experiments, "flatter" (therefore, the quality of the reconstruction is not critically dependent on the choice of iteration at which to stop);
(e) when the smoothing step is applied to the system $A\mathbf{f} = \mathbf{g}$ the minimal relative error is comparable with regard to all the best known techniques for the normal equations $A^TA\mathbf{f} = A^T\mathbf{g}$, but in this case the convergence is much faster;
(f) it can be combined with nonnegativity constraints (by using a simple projection at every step); in that case we observed a substantial improvement in the total cost and in the precision when compared with the very precise but extremely slowly convergent projected CGNE and Landweber. Furthermore, in principle, it can be used in connection with edge preserving procedures such as total variation, Bayesian methods, and deterministic strategies (see, e.g., [21, 18, 14, 5]). Indeed, the nonconvex optimization (which characterizes all these quite expensive techniques) should be solved by some kind of iterative method which uses linearization, and our multigrid procedure can be applied at this level (instead of using preconditioning as suggested in [7, 2]). Finally, a priori information on the statistical nature of the noise could also be exploited for defining more appropriate interpolation and smoothing operators. All these issues will be investigated in future work.

Due to the applications in astronomy, we preferred the use of periodic BCs and therefore the version of an algebraic multigrid for two-level circulant matrices (see [26, 1]). However, as clearly discussed in sections 3 and 4, the regularizing power of our multigrid technique does not depend on the BCs (only the algebraic part of the procedure has to be conveniently adapted). Indeed, with reflective BCs the structure is related to the DCT-III algebra, with antireflective BCs the structure is related to the DST-I algebra, and optimal algebraic multigrid methods have been devised for both (see [8, 13, 1]). Extensive numerical experiments in this direction, in order to check the regularizing features of our multigrid with all the BCs considered, will be part of a future work. We tried several images and, concerning items (c) and (d), the results of our reconstruction are uniformly better with respect to the classical methods (Riley, CG, Richardson); moreover, the associated complexity is of the same order as the usual techniques. Concerning items (e) and (f), the conclusions are in some sense dual since the quality of our reconstruction is comparable to or slightly better than the best methods (Tikhonov, CGNE, Landweber with optimal parameters), but our procedure is much faster when compared with the classical iterative regularizers such as CGNE and Landweber.

We stress that multigrid procedures have already been considered in the image restoration literature by Chan, Chan, and Wan [6], Huckle and Staudacher [17], and the first author [9]. However, these works represent only numerical linear algebra contributions and do not concern regularization. The authors consider the regularized system (by using total variation and standard Tikhonov approaches, respectively), and then they use multigrid in order to obtain a fast convergence on the algebraic system (and not for regularization purposes). Our approach tries to combine numerical linear algebra requests (low complexity, exploitation of the structured matrices, etc.) and regularization issues (a good precision at the optimal iteration). No explicit comparison is made with the preconditioned conjugate gradient (PCG) and the PCG for normal equation (PCGNE) methods since the quality of their reconstructions is generally worse (see, e.g., [16, 12]) when compared with standard CG and CGNE methods; consequently, in terms of quality reconstruction and numerical complexity, a comparison with the latter two techniques is sufficient.

In summary, the main aim of this paper is to show that the proposed multigrid method (MGM) is a general framework: it can be used to improve the quality of the restored image and/or to reduce the computational cost of an iterative regularizing method which is used as a smoother in the multigrid approach. Furthermore, it inherits all the properties of the iterative regularizing methods; for instance, it can be combined with nonnegativity constraints. Finally, we emphasize that usually it is not strictly necessary to resort to a smoother applied to the normal equations since the multilevel strategy performs a further filtering.

The paper is organized as follows: in section 2, we describe the classical (geometric) multigrid, while the algebraic multigrid for shift-invariant structured matrices (belonging to trigonometric matrix algebras or to the Toeplitz class) is described in section 3. In section 4 we adapt the algebraic multigrid algorithm to the image restoration case: we essentially use projectors classically employed in the context of PDEs but now interpreted in the language of signals and images; furthermore, we give a theoretical explanation of why the technique is effective and furnish an analysis of the arithmetic cost (per iteration). In section 5 we present a wide experimentation with various choices of PSFs and signal to noise ratios (SNRs). Moreover, we compare our proposal with other iterative regularizing methods like CG, CGNE, Richardson,

Landweber, and the Tikhonov technique in terms of both resulting accuracy and complexity. In section 6 we show that the multigrid regularization is a general framework which can lead to several generalizations. Section 7 is then devoted to discussing the perspectives that such a class of multigrid-type algorithms offers in terms of regularizing features.

2. The multigrid algorithm. For solving the linear system $A_n\mathbf{x}_n = \mathbf{b}_n$, the two-grid method (TGM) refers to a smaller linear system $A_k\mathbf{x}_k = \mathbf{b}_k$ with $k < n$. In its simpler version, the TGM is the combination of two fixed-point methods: the smoother and the coarse grid correction (CGC). More precisely we have

$$\mathbf{y}_n := \mathsf{TGM}(\mathbf{x}_n, \mathbf{b}_n, S_n, \nu)$$

$$\begin{aligned}
&\texttt{If (size}(\mathbf{b}_n) \le c\texttt{) Then } \mathbf{y}_n = A_n^{-1}\mathbf{b}_n \\
&\texttt{Else } \tilde{\mathbf{x}} := \mathsf{Smooth}\big(\mathbf{x}_n, \mathbf{b}_n, S_n, \nu\big) \\
&\qquad \mathbf{b}_k := P_n^k(\mathbf{b}_n - A_n\tilde{\mathbf{x}}_n) \\
&\qquad\qquad (P_n^k\mathbf{b}_n \text{ in precomputing phase}) \\
&\qquad A_k := P_n^k A_n (P_n^k)^T \ \text{(in precomputing phase)} \\
&\qquad \mathbf{y}_k := A_k^{-1}\mathbf{b}_k \\
&\qquad \mathbf{y}_n := \tilde{\mathbf{x}}_n + (P_n^k)^T\mathbf{y}_k,
\end{aligned} \tag{2.1}$$

where c is a small constant dimension under which it is not useful to apply a two-grid strategy. By $\mathsf{Smooth}\big(\mathbf{x}_n, \mathbf{b}_n, S_n, \nu\big)$ we denote the application of ν steps of the smoother method defined by the iteration matrix S_n. In this work we do not use postsmoothing, so the presmoother is simply called the smoother. The smoother is a fixed-point iteration of the kind

$$\mathbf{x}_n^{(j+1)} = S_n\mathbf{x}_n^{(j)} + \mathbf{c}_n, \tag{2.2}$$

where $\mathbf{c}_n$ is easily computable (low cost) and is implicitly defined as $A_n(I_n - S_n)^{-1}\mathbf{c}_n = \mathbf{b}_n$ with I_n being the identity matrix of dimension n. Usually the smoother is a method like damped Jacobi, Richardson, or Gauss–Seidel, which generally are slowly convergent iterative solvers when applied to ill-conditioned linear systems. The CGC operator projects the error equation into a subspace of dimension k (usually $k \approx n/2^d$ in d dimensional problems), solves the smaller system, and then interpolates the solution to come back to the space of higher dimension n. Formally, the corresponding iteration matrix is defined as

$$CGC_n = I_n - P_k^n A_k^{-1} P_n^k A_n, \tag{2.3}$$

where the full-rank $P_\alpha^\beta \in \mathbb{R}^{\beta\times\alpha}$ projects $\mathbb{R}^\alpha$ into $\mathbb{R}^\beta$ and $A_k \in \mathbb{R}^{k\times k}$ is the projected version of A_n in the lower dimensional space. Following the Galerkin formulation, we have $A_k = P_n^k A_n P_k^n$ and $P_k^n = c(P_n^k)^T$ with c constant and where, without loss of generality, we take $c = 1$. We stress that CGC_n defined in (2.3) is a projection matrix having eigenvalues equal to zero or to one (it is a filter in the space spanned by the columns of $A_n^{-1}P_n^k$), and therefore it is not a convergent iterative method. However, its combination with the smoother is the TGM whose iteration matrix is $TGM_n = CGC_n \cdot S_n^\nu$: if we choose the projector P_k^n and/or the smoother S_k^n in such a way that CGC_n and S_n^ν have spectral complementary behavior [23], then the resulting TGM can be very effective for the solution of linear systems arising from elliptic PDEs [15] or with structured coefficient matrices related to space invariant operators [1, 13, 8]. More precisely, from an algebraic point of view [22, 23], in order to obtain

an effective method the smoother and the CGC must bring down the error in two orthogonal subspaces. Therefore, we can freeze CGC_n and then choose the smoother appropriately as in the geometric multigrid, or we can consider a fixed smoother and then choose the projector appropriately for the CGC as in [22, 1]. Now we are ready to introduce and describe a true multigrid method. In fact, the main difference with respect to the TGM is as follows. Instead of solving the linear system directly with coefficient matrix A_k, we can apply recursively the projection strategy obtaining an MGM: in this case the CGC operator CGC_n is replaced by an approximation, since the matrix A_k^{-1} is approximated by $(I_k - MGM_k^{\gamma})\, A_k^{-1}$ as implicitly described in (2.5) with $k = n_{l-1}$. Concerning notation, we will emphasize the recursion levels. If l is the maximal number of recursive calls, then MGM_i, $i = 1, \ldots, l$, denotes MGM_{n_i} with $0 < n_0 < n_1 < \cdots < n_l = n$ and n_j being the real matrix sizes. In this way $MGM_l = MGM_{n_l} = MGM_n$, $S_l = S_{n_l} = S_n$, $\mathbf{b}_l = \mathbf{b}_{n_l} = \mathbf{b}_n$, $\mathbf{x}_l^{(j)} = \mathbf{x}_{n_l}^{(j)} = \mathbf{x}_n^{(j)}$, and the smaller dimension k of the TGM method will become the dimension after the first recursive call, i.e., $k = n_{l-1}$. Let γ be the number of recursive calls at each level, let $\boldsymbol{\nu} = (*, \nu_1, \nu_2, \ldots, \nu_l)$ with ν_i being the number of smoothing steps at level i (at level zero, of course, $\nu_0 = *$ means that no smoothing is applied), and let us use the Galerkin formulation. Then the corresponding multigrid algorithm generates the approximate solution $\mathbf{x}_n^{(j+1)} = \mathsf{MGM}(l, \mathbf{x}_n^{(j)}, \mathbf{b}_n, S_n, \boldsymbol{\nu}, \gamma)$ according to the following rule:

$\mathbf{y}_i := \mathsf{MGM}(i, \mathbf{x}_i, \mathbf{b}_i, S_i, \boldsymbol{\nu}, \gamma)$

(2.4)

```
If (i = 0) Then Solve(A_0 y_0 = b_0)
           Else x̃_i    := Smooth(x_i, b_i, S_i, ν_i)
                b_{i-1} := P_i^{i-1}(b_i − A_i x̃_i)
                                  (P_i^{i-1} b_i in precomputing phase)
                A_{i-1} := P_i^{i-1} A_i (P_i^{i-1})^T  (in precomputing phase)
                y_{i-1} := 0_{i-1}
                for k = 1, γ
                    y_{i-1} := MGM(i − 1, y_{i-1}, b_{i-1}, S_{i-1}, ν, γ)
                y_i     := x̃_i + (P_i^{i-1})^T y_{i-1}.
```

Since the multigrid is again a fixed-point method, we can express $\mathbf{x}_n^{(j+1)}$ as $MGM_l \mathbf{x}_n^{(j)} + (I_n - MGM_l) A_n^{-1} \mathbf{b}_n$, where the iteration matrix MGM_l is recursively defined as follows [29]:

(2.5)
$$\begin{cases} MGM_0 = O, \\ MGM_i = \left[I_{n_i} - (P_i^{i-1})^T \left(I_{n_{i-1}} - MGM_{i-1}^{\gamma} \right) A_{i-1}^{-1} P_i^{i-1} A_i \right] \cdot S_i^{\nu_i}, \quad i = 1, \ldots, l. \end{cases}$$

For $\gamma = 1$ and $\gamma = 2$ we have the V-cycle and the W-cycle, respectively, and for $\gamma = 1$ and $l = 1$ we obtain the TGM algorithm if $\mathsf{Solve}(A_1 \mathbf{y}_1 = \mathbf{b}_1)$ is $\mathbf{y}_1 = A_1^{-1} \mathbf{b}_1$.

When considering the classical geometric multigrid, $(P_i^{i-1})^T$ is fixed and usually represents the linear interpolator, while the smoother is a simple iterative method (usually damped). For the algebraic multigrid described in [1], the smoother is a priori decided to be the damped Richardson, while P_i^{i-1} is a function of the entries of the coefficient matrix.

3. A multigrid scheme for shift invariant operators. As already mentioned in the introduction, the use of BCs leads to linear systems with coefficient matrices belonging to the two-level Toeplitz class or to two-level matrix algebras:

BCs	A_n essentially is a
Dirichlet	two-level Toeplitz
periodic	two-level circulant
reflective (Neumann)	two-level DCT-III
antireflective	two-level DST-I.

More specifically, when considering the first three choices of the BCs, the related structure is exact while, in the case of antireflective BCs, we do not find exactly a DST-I matrix. However, the solution of the associated linear system can be reduced to the solution of a linear system with a coefficient matrix belonging to the two-level DST-I algebra (see [25]).

As briefly observed in the introduction, the choice of the BCs is relevant for the precision of the restored image. In particular, when inappropriate BCs are employed, some artifacts appear close to the borders of the image and often, due to the ill-conditioning of the discrete blur operator, they spread throughout the image. More precisely, these artifacts, called ringing effects, come from artificial discontinuities introduced in the image by the imposed BCs. In this respect and with regard to computational requirements, a simple classification can be made. Zero Dirichlet BCs impose an artificial discontinuity at the borders, and the resulting structure is two-level Toeplitz (i.e., block Toeplitz with Toeplitz blocks); periodic BCs can again impose an artificial discontinuity at the borders and the resulting structure is two-level circulant (i.e., block circulant with circulant blocks). Symmetric or reflective or Neumann BCs preserve the continuity of the image but not the continuity of its normal derivative; as a consequence the ringing effects are significantly reduced (by one order of magnitude), and the resulting structure is two-level Toeplitz + Hankel. Finally, antireflective BCs preserve the continuity of the image and of its normal derivative, and the resulting structure is two-level Toeplitz + Hankel + low rank correction. From a linear algebra viewpoint, for two-level circulant matrices both solution of a linear system and matrix-vector product can be achieved in $O(n^2 \log n)$ complex operations by fast Fourier transforms (FFTs); for two-level Toeplitz or Hankel matrices the multiplication by a vector can be done in $O(n^2 \log n)$ complex operations by using FFTs, while the solution of an associated linear system is extremely costly in general. However, if the PSF is doubly symmetric for reflective and antireflective BCs, the matrix-vector multiplication and the solution of a linear system can be obtained in $O(n^2 \log n)$ real operations by fast cosine transforms (DCT-III) [19] or by fast sine transforms (DST-I) [25], respectively.

The rest of the section is devoted to the definition of the relevant matrix classes (circulants, DCT-III and DST-I algebras, Toeplitz structures) and to a brief description of the essentials of the related multigrid procedures.

3.1. Matrix algebras and Toeplitz matrices. We introduce the matrix algebras by following a unifying approach first in the one-dimensional (1D) case (signals) and then in the 2D case (images). Therefore, we denote with the multi-index n the size of the problem, which is $n = (n^{(1)}, n^{(2)})$ in the 2D case for an $n^{(1)} \times n^{(2)}$ problem with algebraic size $N = n^{(1)} \cdot n^{(2)}$, while in the 1D case it is simply $n = n^{(1)} = N$.

In the 1D case let $f : \mathbb{R} \to \mathbb{R}$ and let Q_n be the unitary matrix (i.e., $Q_n^{-1} = Q_n^H$) related to our matrix algebras. We can define the Hermitian matrix $A_n = Q_n^H \cdot \mathrm{Diag} f(\mathbf{w}^{[n]}) \cdot Q_n$, where $\mathbf{w}^{[n]}$ is a fixed vector of $\mathbb{R}^n$ and $f(\mathbf{w}^{[n]})$ denotes the

vector whose components are $f(w_i^{[n]})$, $i = 1, \ldots, n$. As a consequence $\mathbf{q}_i^{[n]} = Q_n^H \mathbf{e}_i$ is a unitary eigenvector of A_n related to the eigenvalue $f(w_i^{[n]})$. Furthermore, the generating function f defines A_n univocally, and hence we can denote A_n by $\mathcal{A}_n(f)$, where $\mathcal{A} \in \{\mathcal{C}, \mathcal{N}, \mathcal{S}\}$, which means that $\mathcal{C}_n(f)$ is circulant, $\mathcal{N}_n(f)$ is DCT-III, and $\mathcal{S}_n(f)$ is DST-I. For any of the considered algebras, the mathematical objects Q_n and $\mathbf{w}^{[n]}$ are defined in Table 3.1.

TABLE 3.1
Matrix algebras in the 1D case.

Algebra	$\mathcal{A}$	$\mathcal{I}_n$	$[\mathbf{w}^{[n]}]_{i \in \mathcal{I}_n}$	$[Q_n]_{i,j \in \mathcal{I}_n}$
Circulant	$\mathcal{C}$	$0, \ldots, n-1$	$w_i^{[n]} = \frac{2\pi i}{n}$	$\frac{1}{\sqrt{n}}\left[\mathrm{e}^{\mathbf{i}jw_i^{[n]}}\right]$
DCT-III	$\mathcal{N}$	$0, \ldots, n-1$	$w_i^{[n]} = \frac{\pi i}{n}$	$\sqrt{\frac{2-\delta_{j,1}}{n}}\left[\cos\left((j+\frac{1}{2})w_i^{[n]}\right)\right]$ $\delta_{1,1} = 1,\ \delta_{j,1} = 0$ for $j \neq 1$
DST-I	$\mathcal{S}$	$1, \ldots, n$	$w_i^{[n]} = \frac{\pi i}{n+1}$	$\sqrt{\frac{2}{n+1}}\left[\sin\left(jw_i^{[n]}\right)\right]$

In the 2D case, a two-level matrix of partial dimensions $n = (n^{(1)}, n^{(2)})$ and true dimension $N = n^{(1)} \cdot n^{(2)}$ can be described as an $n^{(1)} \times n^{(1)}$ block matrix whose elements are $n^{(2)} \times n^{(2)}$ block matrices. The two-level Toeplitz matrix $T_n(f)$ is defined as

$$T_n(f) = T_{n^{(1)},n^{(2)}}(f) = \sum_{|j_1|<n^{(1)}} \sum_{|j_2|<n^{(2)}} a_{(j_1,j_2)} J_{n^{(1)}}^{[j_1]} \otimes J_{n^{(2)}}^{[j_2]}$$

by means of the Fourier coefficients of f

$$a_{(j_1,j_2)} = \frac{1}{4\pi^2} \int_{[-\pi,\pi]^2} f(x,y) \mathrm{e}^{-\mathbf{i}(j_1 x + j_2 y)} \mathrm{d}x \mathrm{d}y, \qquad \mathbf{i}^2 = -1.$$

Here $J_m^{[j]} \in \mathbb{R}^{m \times m}$ is the matrix whose entry (s,t) equals 1 if $s - t = j$ and is 0 elsewhere. The circulant, DCT-III, and DST-I two-level matrix algebras can be defined as the matrix algebras generated from $Q_{n^{(1)}} \otimes Q_{n^{(2)}}$, with $Q_{n^{(1)}}$ and $Q_{n^{(2)}}$ selected in the same row of Table 3.1. Of course we can associate multilevel matrices $\mathcal{C}_n(f)$, $\mathcal{N}_n(f)$, and $\mathcal{S}_n(f)$ to every bivariate function $f : \mathbb{R}^2 \to \mathbb{R}$. Therefore, $\mathcal{A}_n(f) = \mathcal{A}_{n^{(1)},n^{(2)}}(f) = (Q_{n^{(1)}} \otimes Q_{n^{(2)}})^T \mathrm{Diag} f(\mathbf{w}^{[n]})(Q_{n^{(1)}} \otimes Q_{n^{(2)}})$, where $\mathbf{w}^{[n]} = \mathbf{w}^{[n^{(1)}]} \times \mathbf{w}^{[n^{(2)}]}$ and $\mathbf{w}^{[n_j]}$, $j = 1, 2$, are the same as $\mathbf{w}^{[n]}$ in Table 3.1.

It is known (see, e.g., [25, 19]) that if f is a nonnegative trigonometric polynomial, all the matrices $\mathcal{A}_n(f)$, $\mathcal{A} \in \{\mathcal{C}, \mathcal{N}, \mathcal{S}, T\}$, are (essentially) banded and, whenever f takes the zero value, they are ill conditioned. Furthermore, we require f even (with respect to each variable) in the DCT-III and DST-I cases.

3.2. A multigrid algorithm preserving the structure. We define a multigrid algorithm for two-level Toeplitz, circulant, DCT-III, and DST-I matrices using the Galerkin formulation. The algebraic requirement to define an MGM is that, for each recursion level $i = 1, \ldots, l$, the coefficient matrix A_i has to belong to the same matrix class (Toeplitz, circulant, DCT-III, or DST-I) allowing the recursive application of the algorithm. Let N_i be the dimension of the problem at the level i ($N_l = N$);

TABLE 3.2
Dimensions and cutting operators in the 1D case for $i = 1, \dots, l$.

	Circulant	Toeplitz & DST-I	DCT-III
n_l	2^k	$2^k - 1$	2^k
n_{i-1}	$\frac{n_i}{2}$	$\frac{n_i - 1}{2}$	$\frac{n_i}{2}$
$K_{n_i} \in \mathbb{R}^{n_{i-1} \times n_i}$	$\begin{bmatrix} 1 & 0 & & & \\ & 1 & 0 & & \\ & & \ddots & \ddots & \\ & & & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 & & \\ & 0 & 1 & 0 & \\ & & \ddots & \ddots & \ddots \\ & & & 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 0 & & \\ & 1 & 1 & 0 & \\ & & \ddots & \ddots & \ddots \\ & & & 0 & 1 & 1 \end{bmatrix}$

the algebraic requirement is fulfilled with an appropriate choice of the projector $P_i^{i-1} : \mathbb{R}^{N_i} \to \mathbb{R}^{N_{i-1}}$. More precisely, in the 1D case we have $P_i^{i-1} = K_{n_i} \mathcal{A}_{n_i}(p_i)$ for $i = 1, \dots, l$, where $\mathcal{A}_{n_i}(p_i)$ belongs to the same class as $\mathcal{A}_{n_i}(f_i)$ and K_{n_i} is the cutting matrix defined in Table 3.2 that preserves the same structure at each level (p_i is a trigonometric polynomial whose choice will be discussed a bit later).

We stress that in the algebraic case we have $\mathcal{A}_{n_i}(p_i)\mathcal{A}_{n_i}(f_i) = \mathcal{A}_{n_i}(p_i f_i)$, while for the Toeplitz class the structure is preserved at each level only if p_i has degree at most one (for p_i of greater degree we can change K_{n_i}, adding some zero columns at the beginning and at the end; see [1]). The choice of p_i is a technical task and, in order to obtain a fast solver which converges within a constant number of iterations independent of the size of the problem, it has to satisfy two analytic conditions for which we refer to [24, 1, 8]. In the present work we need only a simple projector, which is the classic linear interpolation used in the geometric multigrid and such that it preserves the structure of the matrix at each level. Therefore we take P_i^{i-1} associated with the related BCs generated by $p_i(x) = (1 + \cos(x))$, which maintains also the Toeplitz structure in the case of Dirichlet BCs.

In the 2D case we have $N_i = n_i^{(1)} n_i^{(2)}$, where the dimensions $n_i^{(1)}$ and $n_i^{(2)}$ evolve as in the Table 3.2. Moreover, $P_i^{i-1} = (K_{n_i^{(1)}} \otimes K_{n_i^{(2)}})\, \mathcal{A}_{n_i^{(1)}, n_i^{(2)}}(p_i)$, with p_i being a bivariate trigonometric polynomial. In this work, extending the 1D choice, we take

$$p_i(x, y) = (1 + \cos(x))(1 + \cos(y)), \tag{3.1}$$

that is, bilinear interpolation.

When damped Richardson (iteration matrix given by $S_i = I_i - \omega_i \mathcal{A}_i(f_i)$) is used as the smoother for $A_i = \mathcal{A}_i(f_i)$, the relaxation parameter ω_i has to be in $[0, 2/\|f_i\|_\infty]$. In our experimentation we take $\omega_i = 1/\|f_i\|_\infty$ (classical Richardson without damping) according to the choice performed in [1], but, of course, different values of ω_i can be explored.

From a computational cost point of view, the assembling of $A_i = \mathcal{A}_i(f_i)$, for $i = 0, \dots, l-1$, does not increase the cost of the whole iterative method since the number of nonzero entries of P_i^{i-1} in the 1D case is less than $3N_i$ and in the 2D case is less than $5N_i$ for $i = 1, \dots, l$. Therefore, A_{i-1} can be computed within a number of operations substantially lower than the matrix-vector product which involves the matrix A_i. Furthermore, all the matrices A_i can be formed in a setup phase before starting the MGM and with a total computational cost lower than the matrix-vector product with the matrix A_l. Indeed, the considered precomputing phase cost is of $O(\log(N))$ if f is a trigonometric polynomial with few nonzero entries with respect to N (A_l sparse), and it is of $O(N \log(N))$ if A_l is a dense matrix.

Concerning the arithmetic cost of one multigrid iteration, for a PSF with small support (sparse linear systems), up to constant order terms, and without considering the recursive calls, the computational cost at each level i is lower than cN_i with c constant. Therefore, the total computational cost $C(\mathsf{MGM}_n)$ of one complete multigrid cycle in 2D is

$$C(\mathsf{MGM}_n) \leq \begin{cases} \frac{4}{3}cN + O(\log(N)), & \gamma = 1, \\ 2cN + O(\log(N)), & \gamma = 2, \\ 4cN + O(\log(N)), & \gamma = 3, \\ O(N\log(N)), & \gamma = 4 \end{cases} \tag{3.2}$$

(see [29] for further details).

4. A multigrid algorithm with regularizing properties. In this section we describe a multigrid algorithm which is able to improve the regularizing properties of the iterative method used as a smoother. The considered feature derives from a particular choice of the projector which is obtained by linking the geometric multigrid with its interpretation in terms of algebraic multigrid. First we give the idea behind the regularization through projection and we explain why it works, and then we use it to define a multigrid regularizing strategy.

4.1. Regularization through projection. We give some arguments to explain why MGM can improve the regularization property of iterative methods like CG, Richardson, CGNE, or Landweber. For the sake of simplicity, the description is provided in the 1D case, but the same analysis works in the multidimensional case as well.

When the PSF is space invariant and we impose BCs the coefficient matrix is generated by a function that is zero or close to zero in a (possibly large) neighborhood of π and reaches the maximum value (which is 1, thanks to the normalization condition) at zero. For instance, in Figure 4.1 a Gaussian PSF with mask $\mathbf{a} = [\mathrm{e}^{-\mathbf{x}^2}]/\mathrm{c}$ and its generating function $z(y) = \sum_{i=-50}^{50} a_i \mathrm{e}^{-\mathrm{i}y}$ are shown. Here $\mathbf{x}$ is an equispaced sampling of 101 points in $[-10, 10]$, and c is a normalization constant such that $\sum_i a_i = 1$. Let $\mathcal{A}_n(z)$ be Toeplitz, circulant, DCT-III, or DST-I; then its eigenvalues are about an evaluation of $z(y)$ over a uniform sampling of y in $[0, \pi]$, and the eigenvectors are discrete frequency vectors whose frequency is about $\frac{yn}{2\pi}$ (see [25, 30]). Therefore, since the ill-conditioned subspace is associated with small eigenvalues, from Figure 4.1 we can see that this degenerating subspace has very large dimension (this characterizes the discretized ill-posed problems) and it essentially contains the high frequencies subspace where the noise usually lives. Classical iterative methods like CG and Richardson first reduce the error in the well-conditioned subspace, low frequencies in our case [1, 6, 26]. For instance, using the previous PSF and the original signal $\mathbf{f}$ defined as a uniform sampling of $f(x) = \sin(x)$ in $[0, \pi]$, we create the observed signal $\mathbf{g}$ as $\mathbf{g} = \mathcal{A}_n(z)\mathbf{f}$. Solving the linear system $\mathcal{A}_n(z)\mathbf{x} = \mathbf{g}$ with Richardson, taking an initial error which has components in low and high frequencies, the low frequencies are reduced after a few iterations, while the high frequencies are still almost unchanged after 10 iterations (see Figure 4.2). Therefore, the error has the usual semiconvergence property: it decreases while we are working in the low frequencies subspace, reaches a minimum, and then increases again when we arrive to work in the (unfortunately large) ill-conditioned subspace (high frequencies in our case).

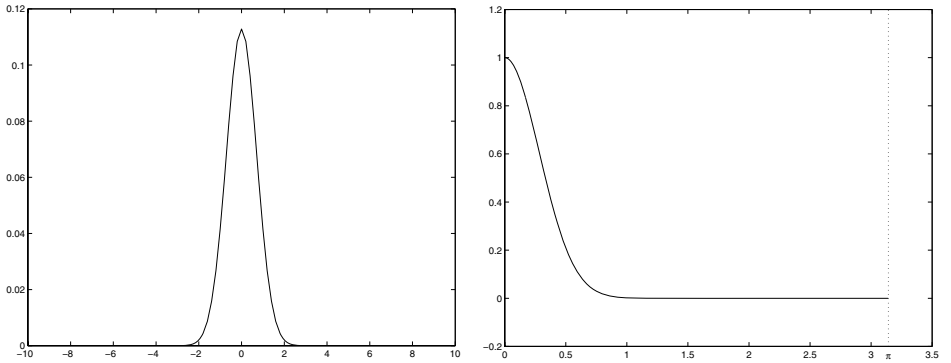


FIG. 4.1. *Gaussian PSF normalized with a constant c and its generating function.*

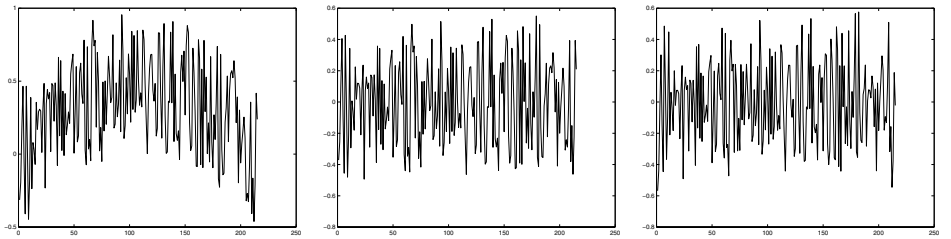


FIG. 4.2. *Components of the restoration error.*

In order to obtain an effective and fast method according to the algebraic multigrid theory [1], the algebraic multigrid used in [9] projects the system in the high frequencies subspace because this is the space where the smoother is ineffective and where we would like to obtain a better approximation. Unfortunately, the high frequencies not only contain fundamental parts of the image (e.g., the nonnegligible high frequency portion of the edges) but also a substantial part of the noise. Therefore, as already shown in [9], we obtain the noise explosion already after a few iterations, and consequently we must resort to the Tikhonov regularization and apply the algebraic multigrid to the regularized system. The good news is that we obtain a robust method which, in some cases, shows a convergence speed independent of both the matrix size and the regularization parameter (see [1, 26, 9]).

Here we would like to propose a different approach, that is, use the MGM directly as a regularizer. Instead of projecting into the high frequencies subspace, the idea is to project into a subspace where we would like to discriminate between the noise contribution and the edges which are of great importance for identifying an image. The latter important feature can be obtained via projection techniques employed in the geometric multigrid. It is well known that the MGM is an optimal solver for PDEs, where the ill-conditioned subspace is spanned by low frequencies. For instance, in the Poisson problem we have as coefficient matrix $T_n(z)$, where $z(x) = 2 - 2\cos(x)$, $x \in [0, \pi]$, that vanishes at zero and grows monotonically in $[0, \pi]$, reaching its maximum

at π. The simplest projector used for PDEs is the weighted average

$$P_i^{i-1} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & \\ & & 1 & 2 & 1 & & & \\ & & & & \ddots & \ddots & & \\ & & & & & 1 & 2 & 1 \end{bmatrix}_{n_{i-1}\times n_i},$$

while the interpolation operator is the simple linear interpolation $2(P_i^{i-1})^T$. We observe that the jth column of the interpolation operator is given by

$$0.5\left[\mathbf{e}_{j-1} + 2\mathbf{e}_j + \mathbf{e}_{j+1}\right], \quad \mathbf{e}_k = k\text{th canonical vector}.$$

These vectors, looked at as samplings of functions, have nontrivial components in lower frequencies, but their high-frequency contribution is not generic since it is associated with the necessity of preserving the jumps which characterize the edges of an image. Indeed the jth column of the interpolation operator can be seen as the sum of box functions or can be interpreted as a local linear spline: note that it is reminiscent of the basis functions in the wavelets theory. Of course in a real image restoration, 2D case, the projector will be a tensor product of a given pair of functions having the aforementioned shape.

We remark that P_i^{i-1} is the same projector used in the algebraic multigrid when considering Dirichlet BCs and with a related generating function having only one zero at the origin with order almost two. More generally, the projection into the low frequencies subspace can be obtained as $P_i^{i-1} = K_i\mathcal{A}_i(p)$ with $p(x) = 1 + \cos(x)$, where $\mathcal{A} \in \{\mathcal{C}, \mathcal{N}, \mathcal{S}, T\}$ according to the BCs used. Therefore, the latter is a good choice since it allows one to project into the low frequencies subspace where there is less noise explosion. In this way we force the smoother to better solve the problem in the subspace where there is less noise. We remark that by projecting into the low frequencies subspace we lose the optimality property of the algebraic multigrid, but now the MGM is used as a regularizer and not as a fast solver for algebraic systems. Finally, if we have a regularizing iterative method like CG, Richardson, CGNE, or Landweber, we can include it as a smoother, thereby improving the quality of the restored image.

From another point of view the effect of P_i^{i-1} on the observed image $\mathbf{g}$ is like a reblurring (see [3, 10]) obtained from a weighted average and after a down-sampling by a factor of two as a compression operation. Moreover, the effect of the linear interpolation $(P_i^{i-1})^T$ on the solution computed in the coarser grid is a smoothing effect, since the linear interpolation can reproduce well only the smooth components; it is unable to reproduce those highly oscillating components which contain the highest percentage of noise.

4.2. Two-level regularization. Using the idea described in the previous section we can define a two-level (TL) regularizing method with the following algorithm:

$$\mathbf{y}_n := \mathsf{TL}(\mathbf{x_n}, \mathbf{b}_n, S_n, \beta)$$

$$\begin{array}{l} \mathbf{b}_k := P_n^k(\mathbf{b}_n - A_n\mathbf{x_n}) \ (P_n^k\mathbf{b}_n \text{ in precomputing phase}) \\ A_k := P_n^k A_n (P_n^k)^T \qquad (\text{in precomputing phase}) \\ \mathbf{y}_k := \mathsf{Smooth}\big(\mathbf{0}_k, \mathbf{b}_k, S_k, \beta\big) \\ \mathbf{y}_n := \mathbf{x_n} + (P_n^k)^T\mathbf{y}_k. \end{array} \tag{4.1}$$

Here S_k is the same smoother as S_n but of different size (which has to be a regularizing method) applied to A_k, and β is defined in order to reach the best restoration error (instead of defining β, it is possible to apply an early stopping criterion to the coarser system $A_k\mathbf{y}_k = \mathbf{b}_k$). The TL looks like a TGM where at the finer level no smoother is applied. But perhaps the main difference between TL and TGM is that, at the coarser level, instead of solving the system directly, TL applies β iterations of the smoother.

Moreover, employing a uniform multigrid language, the above algorithm belongs to the classical multigrid scheme where, following the notation of section 2, it can be formulated as

$$\begin{aligned} &\mathsf{TL}(\mathbf{x}_n, \mathbf{b}_n, S_n, \beta) = \mathsf{MGM}(1, \mathbf{x}_n, \mathbf{b}_n, S_n, (*, 0), 1), \quad \text{where} \\ &\mathsf{Solve}(A_0\mathbf{y}_0 = \mathbf{b}_0) \to \mathbf{y}_0 := \mathsf{Smooth}\big(\mathbf{0}_0, \mathbf{b}_0, S_0, \beta\big). \end{aligned} \tag{4.2}$$

Since at the finer level we do not apply any smoother, we can also interpolate the solution obtained at each iteration and apply the early stopping criterion at the finer level. This means that performing one iteration of TL and β iterations of smoother is exactly the same as performing β iterations of TL and one of smoother.

PROPOSITION 4.1. *Let* $\mathbf{x}_n^{(j+1)} = \mathsf{TL}(\mathbf{x}_n^{(j)}, \mathbf{b}_n, S_n, 1)$, *for* $j = 0, \ldots, \beta - 1$, *and let* $\mathbf{y}_n^{(\beta)} = \mathsf{TL}(\mathbf{x}_n^{(0)}, \mathbf{b}_n, S_n, \beta)$. *Then* $\mathbf{x}_n^{(\beta)} = \mathbf{y}_n^{(\beta)}$.

Proof. It is sufficient to prove the assertion for $\beta = 2$; indeed, for a generic β the claim follows by induction. Furthermore, since TL is a fixed-point method and since the starting point $\mathbf{x}_n^{(0)}$ is the same for the two iterations, it is sufficient to show that the two methods which generate $\mathbf{x}_n^{(\beta)}$ and $\mathbf{y}_n^{(\beta)}$ have the same iteration matrix. Recalling that the smoother is defined as in (2.2), it holds that

$$\begin{aligned} \mathbf{x}_n^{(1)} &= \mathsf{TL}(\mathbf{x}_n^{(0)}, \mathbf{b}_n, S_n, 1) = (I_n - R_nA_n)\mathbf{x}_n^{(0)} + R_n\mathbf{b}_n, \\ R_n &= (P_n^k)^T(I_k - S_k)A_k^{-1}P_n^k, \end{aligned}$$

and therefore $\mathbf{x}_n^{(2)} = (I_n - R_nA_n)^2\mathbf{x}_n^{(0)} + (2I_n - R_nA_n)R_n\mathbf{b}_n$. Furthermore,

$$\begin{aligned} \mathbf{y}_n^{(2)} &= \mathsf{TL}(\mathbf{x}_n^{(0)}, \mathbf{b}_n, S_n, 2) = (I_n - Q_nA_n)\mathbf{x}_n^{(0)} + Q_n\mathbf{b}_n, \\ Q_n &= (P_n^k)^T(S_k + I_k)(I_k - S_k)A_k^{-1}P_n^k, \end{aligned}$$

and we have only to prove that $(I_n - R_nA_n)^2 = (I_n - Q_nA_n)$. Since $A_k = P_n^kA_n(P_n^k)^T$, we finally infer

$$\begin{aligned} (I_n - R_nA_n)^2 &= I_n - 2R_nA_n + R_nA_nR_nA_n \\ &= I_n - 2(P_n^k)^T(I_k - S_k)A_k^{-1}P_n^kA_n + (P_n^k)^T(I_k - S_k)^2A_k^{-1}P_n^kA_n \\ &= I_n - (P_n^k)^T(I_k - S_k)(I_k + S_k)A_k^{-1}P_n^kA_n \\ &= I_n - Q_nA_n, \end{aligned}$$

and the proof is concluded. ∎

We remark that the latter proposition could be used for analyzing the convergence features and the regularization effects of the TL. Indeed, informally speaking, if we use an iterative regularizing method as smoother (let us say Richardson, CGNE, etc.), then in our two-level algorithm we are anticipating or postponing a low-pass filter, and therefore it is not possible that the global two-level iteration amplifies the noise contributions.

4.3. Multigrid regularization. According to the description in section 2, the TGM leads to an MGM in a natural way. Similarly, the TL regularization previously proposed can be applied recursively, obtaining a multigrid regularization. To this end, the definition contained in (4.2) with $\beta = 1$ is especially useful. In terms of regularization, this means that at the coarser level we apply a smoothing iteration and then we project again into the well-conditioned subspace. This can be repeated until we reach a small grid (e.g., 8×8), where the associated linear system can be solved directly since the problem is computationally negligible, and the noise explosion is easily controllable and avoidable. If the observed image is of size $n^{(1)} \times n^{(2)}$, then we obtain a regularizing multigrid algorithm taking

1. $l = \min\{\log_2(n^{(1)} - 8), \log_2(n^{(2)} - 8)\}$,
2. $\boldsymbol{\nu} = (*, 1, \ldots, 1, 0)$,
3. $\mathsf{Solve}(A_0\mathbf{y}_0 = \mathbf{b}_0) \to \mathbf{y}_0 = A_0^{-1}\mathbf{b}_0$,
4. $S_n =$ an iterative regularizing method.

The choice of γ will be discussed in section 6: we notice that its value can be increased in order to perform more work in the well-conditioned subspace. However, the V-cycle (i.e., $\gamma = 1$) can be a good choice for every kind of problem (high or low percentage noise, good or bad BCs, etc.). Therefore, in the following experimentation we will use the notation $\mathsf{MGM}(S_n, \gamma)$ instead of $\mathsf{MGM}(\min\{\log_2(n^{(1)} - 8), \log_2(n^{(2)} - 8)\}, \mathbf{x}_n^{(j)}, \mathbf{b}_n, S_n, (*, 1, \ldots, 1, 0), \gamma)$, because with the latter choices S_n and γ are the only free parameters. For damped Richardson the relaxation parameter is chosen according to section 3.2, and so $S_n = I_n - A_n(f)/\|f\|_\infty$ is briefly denoted by $S_n =$ Rich. The Landweber method is exactly the application of Richardson to the normal equations, and hence it will be denoted by $S_n =$ RichNE. The classical MGM theory requires that the smoother be a stationary method even if better performances can be observed when nonstationary methods are used, possibly with flexible preconditioning (for the theory of flexible preconditioning, see, e.g., the very interesting paper [27]). Here the point of view is different since we are interested in the regularization features of the method, i.e., in its semiconvergence. We will denote the use of the CG method as a smoother by $S_n =$ CG, and when it is applied to the normal equations we write $S_n =$ CGNE. Since all the parameters of the problem are fixed, as for the MGM, concerning the TL method we indicate only the smoother (i.e., $\mathsf{TL}(S_n)$) since we assume $\beta = 1$ to compute the restoration error at each iteration.

When the observed image has a high percentage of noise or the PSF is not positive definite, it is necessary to resort to the normal equations, and usually the preferred iterative methods are CGNE or Landweber. These methods require only matrix-vector products and thus they can be plainly applied to the normal equations without requiring the effective computation of the matrix $A_n^T A_n$. When we use the Galerkin formulation in the MGM, the coefficient matrix at each level is obtained through projections, and hence it is necessary to assemble the coefficient matrices $A_i^T A_i$ at every level i. For Dirichlet BCs, this leads to a loss of the TL Toeplitz structure. When considering the remaining BCs, for which the related computations can be performed by convolutions, we observe again computational problems. More precisely, the resulting coefficient matrices at each level have a double block bandwidth, and each block has a double bandwidth: therefore, every matrix-vector product is increased by a factor of 4 instead of 2, as happens when applying CG to the normal equations (CGNE) or Landweber. To avoid this kind of problem and to emphasize the robustness of the proposed technique, we propose using the classic Galerkin formulation, projecting at each level i only the matrix A_i, and applying as smoother an iterative method for

normal equations like CGNE or Landweber. This choice follows the philosophy according to which the proposed MGM is a framework for improving the regularization properties of a classic regularizing method, and therefore it should be applied directly to the original system with coefficient matrix A_n.

4.4. Computational cost analysis. In this section we analyze the computational cost of one iteration of the multigrid regularizing method $\mathsf{MGM}(S_n, \gamma)$. More specifically, we will consider (3.2) by giving details on the constants involved and by comparing the resulting cost with that of a single smoothing iteration.

For the sake of simplicity, we consider square images of size $n \times n$, with $n = 2^l$, and therefore the algebraic size of the linear system at each recursion level is $N_i = n_i^2$, with $N_{i-1} = N_i/4$, $i = 0, \ldots, l$, according to Table 3.2 (in the Toeplitz and DST-I cases this is not exactly true since we work with odd dimensions, but the same considerations hold). The computational cost of $\mathsf{MGM}(S_{n_i}, \gamma)$ at a fixed level i is

$$C(\mathsf{MGM}(S_{n_i}, \gamma)) = \gamma C(\mathsf{MGM}(S_{n_{i-1}}, \gamma)) + P(n_i) + W(n_i), \qquad i = 1, \ldots, l, \tag{4.3}$$

where $P(n_i)$ is the projection cost and $W(n_i)$ is the cost of one smoother iteration S_{n_i}. Without loss of generality we assume $C(\mathsf{MGM}(S_{n_0}, \gamma)) = 1$; indeed, in a practical implementation we stop the recursion a little before, but for n large enough the direct solution at the coarsest level is negligible.

The computational cost of the projection at the level i is the sum of two matrix-vector products, one with P_i^{i-1} and the other with $(P_i^{i-1})^T$, since we take p_i as in (3.1)—the first one is $7/4N_i$ and the second one is $7/8N_i$. However, more generally (it will be useful later), we have

$$P(n_i) = bN_i, \tag{4.4}$$

and in our case $b = 21/8$.

We consider a PSF with small support with respect to the size of the image, obtaining $W(n) = aN$ with $a > b > 1$. Usually $a \gg b$, especially if the smoother is applied to the normal equations. From the analysis in [1], the bandwidth of A_{i-1} is about half of the bandwidth of A_i along both directions (at blocks and inside each block); hence

$$W(n_i) < aN_i.$$

Since $N_{i-1} = N_i/4$, $i = 1, \ldots, l$, from (4.3) we have the following general behavior:

$$C(\mathsf{MGM}(S_n, \gamma)) = \begin{cases} O(N), & \gamma < 4, \\ O(N \log(n)), & \gamma = 4, \\ O(N^{\log_4(\gamma)}), & \gamma > 4. \end{cases} \tag{4.5}$$

However, in the cases $\gamma = 1, 2, 3$ we can provide an explicit evaluation of the constants involved as functions of a and b. Indeed, by applying relation (4.3) recursively and

assuming $W(n_l) = 0$, we infer

$$\begin{aligned} C(\mathsf{MGM}(S_n,\gamma)) &= \gamma C(\mathsf{MGM}(S_{n_2},\gamma)) + P(n_1) = \cdots \\ &= \sum_{i=0}^{l-1} P(n_{l-i})\gamma^i + \sum_{i=1}^{l-1} W(n_{l-i})\gamma^i + \gamma^l \\ &< \sum_{i=0}^{l-1} P(n_{l-i})\gamma^i + \sum_{i=1}^{l} W(n_{l-i})\gamma^i \\ &< bN \sum_{i=0}^{l-1} \left(\frac{\gamma}{4}\right)^i + aN \sum_{i=1}^{l} \left(\frac{\gamma}{4}\right)^i \\ &< \frac{(4b+\gamma a)N}{4-\gamma}, \end{aligned}$$

which can be summarized as

$$C(\mathsf{MGM}(S_n,\gamma)) < \begin{cases} \left(\frac{4b}{3} + \frac{a}{3}\right)N, & \gamma = 1, \\ (2b+a)N, & \gamma = 2, \\ (4b+3a)N, & \gamma = 3. \end{cases}$$

Therefore, under the assumption $a \gg b$, which is usually satisfied in every practical application, we deduce the following relationship between the computational cost of $\mathsf{MGM}(S_n,\gamma)$ and that of the smoothing step:

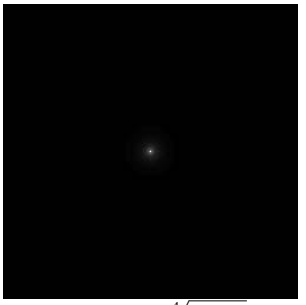
$$C(\mathsf{MGM}(S_n,\gamma)) \approx \begin{cases} \frac{1}{3}W(N), & \gamma = 1, \\ W(N), & \gamma = 2, \\ 3W(N), & \gamma = 3. \end{cases} \tag{4.6}$$

We emphasize that the case of the V-cycle ($\gamma = 1$) is very interesting since its cost is about one third of a standard smoother applied to the finest grid. Moreover, when the assumption about the small support of the PSF is not satisfied, then the advantage of the multigrid algorithms with respect to the smoothers becomes even stronger. More specifically, for a generic PSF, the matrix-vector product is computed through fast discrete transforms and then $W(n_i) = O(N_i \log(N_i))$, while the projection is identical. Therefore, the computational cost of the smoother dominates the computational cost of the projection, and the latter can be neglected. Consequently, taking into account that the smoothers considered are extremely slow (see next section), the conclusion is that in general the $\mathsf{MGM}(S_n,\gamma)$ algorithm is more efficient than the smoother at the finest level alone.

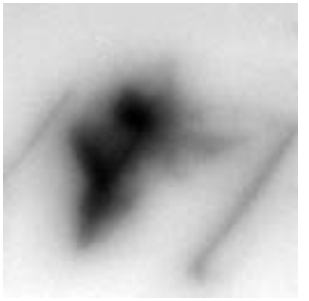
5. Numerical experiments. We present a wide experimentation in order to show the regularization properties of the proposed technique. We consider periodic BCs since they are widely used in astronomical imaging. However, similar results can also be obtained by employing other BCs like Dirichlet, reflective, or antireflective since, as discussed in section 3, similar multigrid methods have been defined also for linear systems arising from these types of BCs [1, 8]. Since we know the true image $\mathbf{x}$ at each iteration j, we can evaluate and plot the relative error norm $e_j = \|\mathbf{x} - \mathbf{x}^{(j)}\|_2 / \|\mathbf{x}\|_2$ for each iterative regularization method. The algorithms are implemented in Fortran 90 using double precision, while the images and the graphs are made by using MATLAB.



Original airplane 256×256

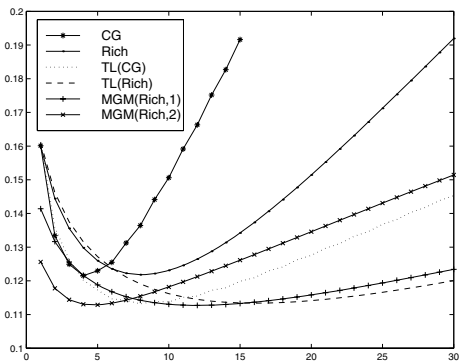


PSF $= e^{-\sqrt[4]{x^2+y^2}}$



Observed image (128×128)

FIG. 5.1. *Pictures of the airplane and of the PSF with* $SNR = 100$.



Method	$\min_{j=1,\dots}(e_j)$	$\arg\min_{j=1,\dots}(e_j)$
CG	0.1215	4
Rich	0.1218	8
TL(CG)	0.1132	8
TL(Rich)	0.1134	16
MGM(Rich,1)	0.1127	12
MGM(Rich,2)	0.1129	5
CGNE	0.1135	178
RichNE	0.1135	352

Minimum error and related number of iterations

FIG. 5.2. *Airplane with* $SNR = 100$: *relative error norm versus number of iterations.*

5.1. An airplane with periodic BCs. The PSF is created as a uniform sampling of 51 points of $e^{-\sqrt[4]{x^2+y^2}}$ in $[-20, 20] \times [-20, 20]$, while for the noise we add a random vector with uniform distribution and SNR equal to 100. The original airplane image and its blurred and noisy version are shown in Figure 5.1 (the original picture is a portion of a larger image from which the blurred one is obtained). The smallest eigenvalue of the coefficient matrix is of the order of 10^{-3}, and the matrix is positive definite. As a matter of fact, it is not strictly necessary to apply CG or Richardson to the normal equations, but it is recommended in order to obtain a good quality of the deblurred image. It is interesting to observe that the proposed multigrid, with the simple Richardson method as smoother, leads to a restoration error lower than the one obtained by CGNE or Landweber (see Figure 5.2). In this example the TL and MGM with a smoother for normal equations do not improve the quality of the restored image, because the value 0.112 is about the minimum error norm from a modelistic point of view. Indeed, by Tikhonov regularization, solving the linear system $(A_n^T A_n + \mu I_n)\mathbf{x}_n = A_n^T \mathbf{b}_n$, and choosing μ as the experimental optimal value ($\mu = 0.001$), the minimum restoration error is 0.1127, which is essentially the same value that we obtain by MGM(Rich, 1).

For higher levels of noise, e.g., SNR $= 10$, using CG or Richardson we observe noise explosion already after very few iterations, and therefore it is necessary to resort to the normal equations. In Figure 5.3, for SNR $= 10$ we can see that both the TL strategy and the MGM regularization techniques, with CGNE or Landweber as smoother,

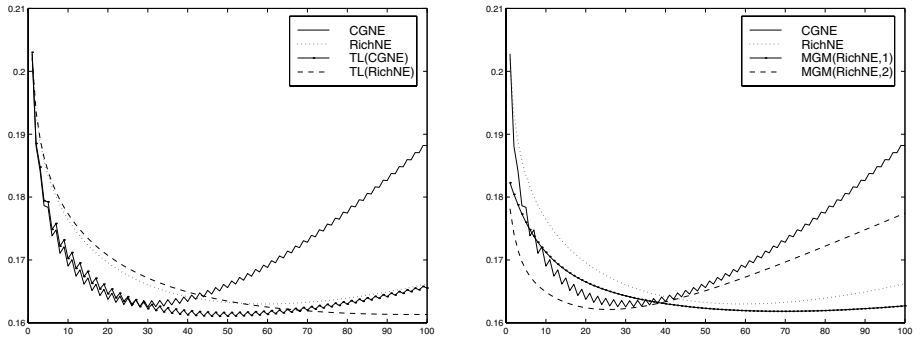


FIG. 5.3. *Airplane with $SNR = 10$: relative error norm versus number of iterations.*

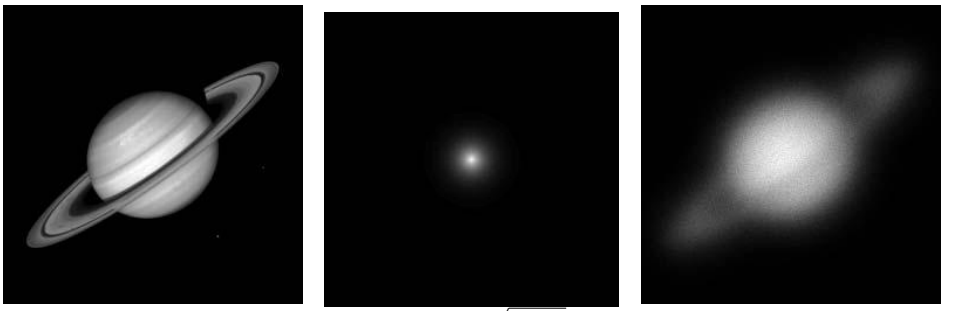
TABLE 5.1
Airplane with $SNR = 10$: minimum error and related number of iterations.

Method	$\min_{j=1,\dots}(e_j)$	$\arg\min_{j=1,\dots}(e_j)$
CGNE	0.1625	30
RichNE	0.1630	59
TL(CGNE)	0.1611	48
TL(RichNE)	0.1613	97
MGM(RichNE,1)	0.1618	69
MGM(RichNE,2)	0.1621	26
MGM(Rich,1)	0.1648	3
MGM(Rich,2)	0.1630	1

show an error curve with a lower minimum and a flatter behavior with regard to CGNE and Landweber alone. Of course, this is good news since the evaluation of the early stopping criterion becomes much easier. Moreover, for MGM with Richardson as smoother, we observe about the same restoration error norm as the one obtained by CGNE or by Landweber, again without resorting to normal equations and with a significantly smaller number of iterations (see Table 5.1).

This example clearly shows that the multigrid strategy not only improves the regularization properties of the iterative method used as smoother, but usually it does not require resorting to normal equations at all. In this way we obtain about the same (or a slightly lower) restoration error as the best iterative regularization methods applied to the normal equations, but with a measurable reduction in the arithmetic cost.

5.2. An astronomical image with nonnegativity constraints. In this example we use the PSF, the original 256×256 image of Saturn, and its blurred and noisy (with Poisson noise) version in Figure 5.4. Since this is an astronomical image, we consider a black extension outside of the image, and thus all the BCs are equivalent and do not introduce any approximation error in the model. The PSF is created as a uniform sampling of $e^{-\sqrt{x^2+y^2}}$ of 101 points in $[-5, 5] \times [-5, 5]$. Therefore, not only is the coefficient matrix ill-conditioned in a subspace of very large dimension (it happens in the case of Gaussian-like blurs, as clearly shown in the second part of Figure 4.1), but it also has some small negative eigenvalues with magnitude of order of 10^{-4}. Regarding the noise, we add Poisson noise to the blurred image with SNR = 30.



Original image of Saturn PSF = $e^{-\sqrt{x^2+y^2}}$ Blurred image + SNR = 30

FIG. 5.4. *PSF and images of Saturn (Poisson noise).*

From a practical point of view one should impose nonnegativity constraints. This task can be accomplished by projecting into the convex set of nonnegative entries, leading to a nonlinear operator. Despite its nonlinearity it can easily be implemented by setting to zero the negative entries of the current approximation for each step of an iterative regularizing method. A procedure of this kind widely used in astronomical imaging is the projected Landweber method. Usually it converges very slowly, and therefore it is often used in connection with accelerating techniques employing preconditioning strategies (see [20]). However, while our multigrid accelerates the smoother convergence by keeping the reconstruction error low, often the effect of the preconditioning is to (partially) spoil the quality of the restored image (see, e.g., [12, 16]). In this context our multigrid can be extended in two directions:

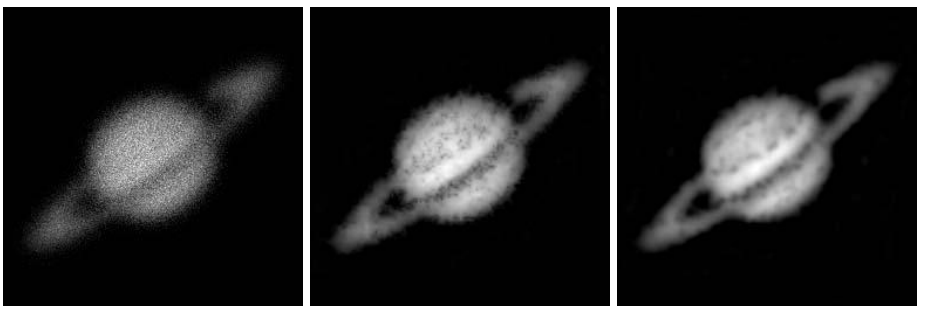
(1) projecting each approximation into the nonnegative cone (e.g., using Landweber as smoother and $\mathbf{x}^{(j+1)} = P_+\mathsf{MGM}(\mathbf{x}^{(j)})$ instead of $\mathbf{x}^{(j+1)} = \mathsf{MGM}(\mathbf{x}^{(j)})$, where P_+ projects into the nonnegative cone),
(2) using a projected method as smoother (e.g., projected Landweber).

The first choice is the classic strategy for the projection into the nonnegative cone for a generic iterative method. The second choice does not assure the nonnegativity: the projection is a weighted average or a linear interpolation, so it preserves the nonnegativity constraint of the smoother, but at the coarsest level the direct solution destroys the nonnegativity. Therefore, the latter strategy has to be combined with the previous one, or we have to project the coarsest level solution into the nonnegative cone. Experimentally, we obtain better results by using the first proposal, that is, by projecting both the smoother and every global iteration.

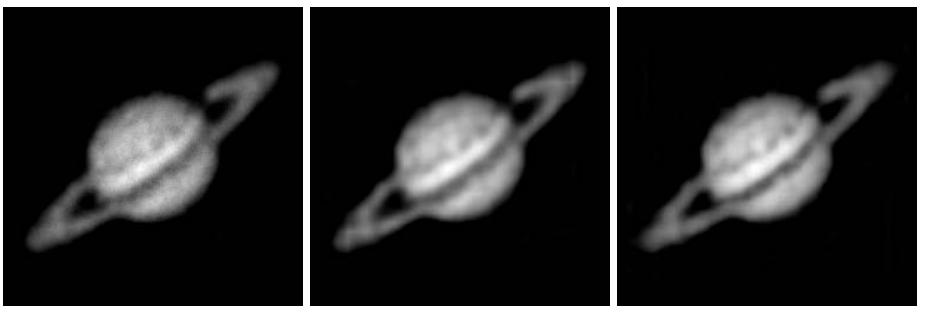
In the following the projected version of an iterative method will be denoted by the plus symbol (e.g., the projected Landweber is denoted by RichNE$^+$, and choice (1) is denoted by $\mathsf{MGM}(\cdot,\cdot)^+$). In Table 5.2 we report the minimal values reached by every method and the corresponding iteration. We can see that the first choice is uniformly better than the second (we recall that the latter does not preserve the nonnegativity), and their combination gives the best results; furthermore, all these strategies are much more effective than classic projected iterative methods (such as CG and Richardson) when applied alone to the linear system $A_n\mathbf{x} = \mathbf{b}$. Moreover, a useful property of our method (e.g., $\mathsf{MGM}(\text{Rich}^+, 2)^+$) is that the reconstruction quality is similar to the one obtained by the projected Landweber or projected CGNE but with the use of a much smaller number of iterations and without resorting to the normal equations (see Figure 5.5). However, if the main wish is to further improve the quality of the deblurred image, we can use projected Landweber or projected

TABLE 5.2
Saturn: Minimum relative error norm and its number of iterations (projected methods).

Method	$\min_{j=1,\dots}(e_j)$	$\arg\min_{j=1,\dots}(e_j)$
CG^+	0.3172	2
Rich^+	0.3176	2
$\mathsf{MGM}(\mathrm{Rich}, 1)^+$	0.2531	6
$\mathsf{MGM}(\mathrm{Rich}^+, 1)$	0.2567	9
$\mathsf{MGM}(\mathrm{Rich}^+, 1)^+$	0.2505	10
$\mathsf{MGM}(\mathrm{Rich}^+, 2)^+$	0.2473	5
RichNE^+	0.2388	676
CGNE^+	0.2385	442
$\mathsf{MGM}(\mathrm{CGNE}^+, 1)^+$	0.2351	337
$\mathsf{MGM}(\mathrm{CGNE}^+, 2)^+$	0.2351	26



CG^+ (2 iter.) | $\mathsf{MGM}(\mathrm{Rich}^+, 2)^+$ (5 iter.) | $\mathsf{MGM}(\mathrm{Rich}^+, 2)^+$ ($\alpha = 5$ in 14 iter.)



CGNE^+ (442 iter.) | $\mathsf{MGM}(\mathrm{CGNE}^+, 1)^+$ (337 iter.) | $\mathsf{MGM}(\mathrm{CGNE}^+, 2)^+$ (26 iter.)

FIG. 5.5. *Restored images at the minimum relative error norm value.*

CGNE as the smoother in the MGM with a total computational cost which is again significantly lower than for the standard projected methods. To reduce the error to 0.238 the projected CGNE requires 442 iterations and the projected Landweber requires 676 iterations, while $\mathsf{MGM}(\mathrm{CGNE}^+, 2)^+$ requires 26 iterations to reduce the error to 0.235 (see Table 5.2). We emphasize that one W-cycle iteration requires approximately the same arithmetic cost as one smoother iteration (see (4.6)): this fact results in a huge saving in time (in addition to a slightly better reconstruction).

TABLE 5.3
Minimum relative error norm for MGM(Rich$^+$, 2)$^+$ varying the degree α of the transfer grid operator ($\alpha = 1$ is the linear interpolation).

α	$\min_{j=1,\dots}(e_j)$	$\arg\min_{j=1,\dots}(e_j)$
1	0.2473	5
2	0.2421	7
3	0.2408	9
4	0.2398	12
5	0.2392	14

Finally, we study the behavior of our MGM proposal by changing a different degree of freedom. In particular we consider the choice of the multigrid transfer grid operator in order to obtain approximately the best restored image without resorting to the normal equations for the smoothing step. Here the PSF has a large support, and the periodic BCs are exact. As shown in Table 5.2, iterative methods for the normal equations require an unacceptably large number of iterations, while the MGM(Rich$^+$, 2)$^+$ loses some details (when compared to the projected CGNE) in the reconstructed image (even if its quality is still reasonably good (see Figure 5.5)). In order to overcome the problem without using a smoother for the normal equations, we can improve the filtering properties of the multigrid transfer grid operator. In fact, instead of employing linear interpolation, we choose cubic interpolation or, more generally, an operator whose generating function is equal to $(1+\cos(x))^\alpha(1+\cos(y))^\alpha$ with $\alpha \in \mathbb{N}^+$ (the linear interpolation is recovered by setting $\alpha = 1$). In Table 5.3 it is shown that, using MGM(Rich$^+$, 2)$^+$ already for $\alpha = 4, 5$, the minimum restoration error norm is slightly greater than the one obtained by CGNE$^+$. Indeed, in Figure 5.5 we can see that MGM(Rich$^+$, 2)$^+$ with $\alpha = 5$ and CGNE$^+$ produces images with about the same level of detail. Furthermore MGM(Rich$^+$, 2)$^+$ with $\alpha = 5$ is much cheaper than CGNE$^+$, since not only does it require 14 iterations instead of 442, but one of its iterations has about the same computational cost as one Richardson iteration. As a matter of fact, the value of b in (4.4) grows linearly with α. More precisely, the size of the stencil related to $\mathcal{A}_{N_i}(p_i)$ is $(2\alpha+1) \times (2\alpha+1)$, and therefore, for small values of α (let us say at most 5), a projection operation is again cheaper than the matrix-vector product with A_i and hence favorable with respect to the normal equation approach.

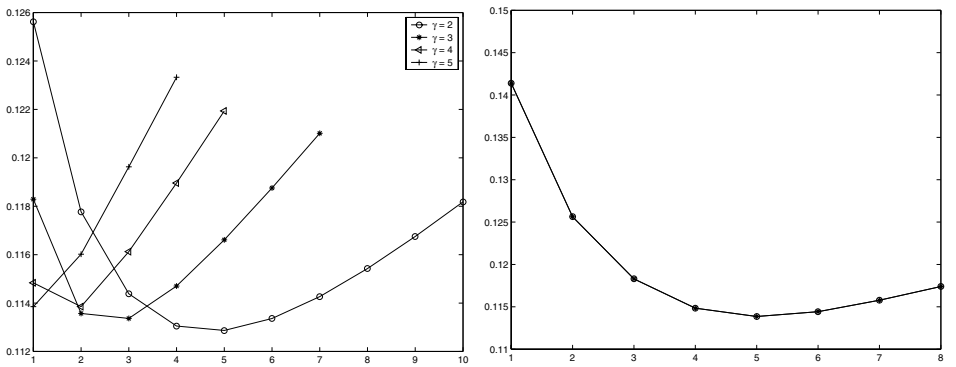
We remark that the improvement of the projector is an idea which has some analogies with the reblurring proposed in [10]. Indeed, since $A_{i-1} = P_i^{i-1} A_i (P_i^{i-1})^T$ and since $P_i^{i-1} = K_{N_i}\mathcal{A}_{N_i}(p_i)$, before the application of the cutting matrices K_{N_i} and $K_{N_i}^T$ we perform a double reblurring to the left and to the right with matrices $\mathcal{A}_{N_i}(p_i)$ and $(\mathcal{A}_{N_i}(p_i))^T$.

6. The γ regularization. As already observed, the proposed regularizing MGM leads to several generalizations which show the great flexibility of a multigrid approach. For instance, by increasing the work in the projected subspace we obtain a better deblurred image with fewer iterations. More precisely, we consider a bigger value of γ and, as a result, the associated MGM is no longer of iterative type, but it is a regularization direct method where the only parameter to be estimated is γ. We are not able to define an optimal choice of γ, but this can be the subject of future research.

As an example we consider the airplane deblurring described in section 5.1 with

TABLE 6.1
Relative error norms varying γ in the case of the airplane image with $SNR = 100$.

γ	e_1	$\min_{j=1,\dots}(e_j)$	$\arg\min_{j=1,\dots}(e_j)$
1	0.1414	0.1127	12
2	0.1256	0.1129	5
3	0.1183	0.1134	4
4	0.1148	0.1139	2
5	0.1139	0.1139	1
6	0.1144	0.1144	1
7	0.1158	0.1158	1



Error norm vs. number of iterations

Error at the first iteration e_1 vs. γ

FIG. 6.1. *Airplane with $SNR = 100$: relative error norm varying γ.*

SNR = 100 (see Figures 5.1 and 5.2). Here we study the behavior of the relative error norm after only one iteration of MGM(Rich, γ) when γ grows (as shown in Figure 6.1). By varying γ, the relative error norm at the first iteration e_1 decreases, reaches its minimum value for $\gamma = 5$, and then increases again; see also Table 6.1. We remark that the optimal restoration error obtained in one iteration with $\gamma = 5$ ($e_1 = 0.1139$) is about the same optimal value obtained by CGNE or Landweber (that is, 0.1135). However, the latter two methods require many more iterations (178 iterations for CGNE and 352 iterations for Landweber) and are applied to normal equations.

In the case of SNR = 10, in Table 5.1 it was already shown that the minimum e_1 is reached for $\gamma = 2$ and that it is the same value obtained by the Landweber method in 59 iterations.

As already observed, for a PSF with a numerically small support, i.e., a TL banded coefficient matrix with $O(N)$ nonzero entries, the computational cost of MGM(S_n, γ) grows with γ according to (4.5).

In conclusion, the considered extension of the regularization MGM is evidence that the multigrid techniques can represent a general framework which is the basis for further investigations. As an example, we should think about the optimal choice of the projector, of the smoother, of $\boldsymbol{\nu}$ (number of smoothing steps at the various levels), and of course of the number γ of recursive calls.

7. Conclusions. In this work we have presented a class of regularizing multigrid algorithms whose features are the following: if it is compared with other regularizing procedures (CG, Richardson, Riley) applied directly to the system $A\mathbf{f} = \mathbf{g}$, then the curve of relative errors is much flatter, the quality of the reconstruction is higher,

and the total arithmetic costs are similar; if it is compared with the best regularizing methods for the normal equations $A^TA\mathbf{f} = A^T\mathbf{g}$ (CGNE, Landweber, Tikhonov), then the accuracy of the restored image is similar (at most slightly better), the structure of the error curve is essentially the same, but the cost is greatly reduced. In every case, our multigrid can use normal equation methods only for the smoother, while the projection to a coarser grid is done always on the original coefficient matrix; this usually allows us to obtain a slightly better reconstruction and a lesser computational time compared with the best regularizing methods for the normal equations.

Furthermore, we stress that the presented approach can be looked at as a general framework which has the potential of leading to several extensions and improvements, and, in this respect, a lot of analysis must still be done. For instance, open questions concern the analysis of convergence and, more interestingly, the theoretical proof of the regularizing effects of the proposed method or the choice of γ in order to obtain a direct (one-step!) multigrid regularization.

Further observations could be considered. In the case where we apply a regularizing scheme directly to the original system, when the minimum reached is close to that of other regularizing procedures, this is often due to border effects. Therefore, we should also investigate a combination of our technique with recently defined BCs (e.g., reflective or antireflective BCs) or with the idea in [4, 31] in order to eliminate the artifacts due to the so-called ringing effects. With the combination of these two ingredients we have to expect that the ringing effects will also be noticeably reduced when considering generic (not necessarily with uniform background) images. Consequently, as in the cases previously considered in this paper, we believe that the proposed multigrid-type procedures will perform better than classical regularizing methods in terms of precision and/or computational cost.

Finally, we would like to investigate how our multigrid proposal can be used in connection with edge preserving procedures such as total variation, Bayesian methods, and deterministic strategies (see, e.g., [21, 18, 14, 5]). Indeed, the nonconvex optimization (which characterizes all these quite expensive techniques) should be solved by some kind of iterative method which uses linearization, and our multigrid procedure can be applied at this level (instead of using preconditioning as suggested in [7, 2]) not only for accelerating the procedures but also for regularizing purposes. Also, a priori information on the statistical nature of the noise could be exploited, specifically for defining more appropriate interpolation and smoothing operators: all these issues will be considered in future work.

Acknowledgments. Warm thanks to the referees for very pertinent and useful remarks.

REFERENCES

[1] A. Aricò, M. Donatelli, and S. Serra-Capizzano, *V-cycle optimal convergence for certain (multilevel) structured linear systems*, SIAM J. Matrix Anal. Appl., 26 (2004), pp. 186–214.

[2] L. Bedini, G. Del Corso, and A. Tonazzini, *Preconditioned edge-preserving image deblurring and denoising*, Pattern Recognition Lett., 22 (2001), pp. 1083–1101.

[3] M. Bertero and P. Boccacci, *Introduction to Inverse Problems in Imaging*, Institute of Physics Publishing, Bristol, Philadelphia, 1998.

[4] M. Bertero and P. Boccacci, *A simple method for the reduction of the boundary effects in the Richardson-Lucy approach to image deconvolution*, Astron. Astrophys., 437 (2005), pp. 369–374.

[5] A. Blake and A. Zisserman, *Visual Recognition*, MIT Press, Cambridge, MA, 1987.

[6] R.H. Chan, T.F. Chan, and W. Wan, *Multigrid for differential-convolution problems arising in image processing*, in Proceedings of the Workshop on Scientific Computing, G. Golub, S.H. Lui, F. Luk, and R. Plemmons eds., Springer, Singapore, 1997, pp. 58–72.
[7] R.H. Chan, T.F. Chan, and C.-K. Wong, *Cosine transform based preconditioners for total variation deblurring*, IEEE Trans. Image Process., 8 (1999), pp. 1472–1478.
[8] R.H. Chan, S. Serra-Capizzano, and C. Tablino Possio, *Two-grid methods for banded linear systems from DCT-*III *algebra*, Numer. Linear Algebra Appl., 12 (2005), pp. 241–249.
[9] M. Donatelli, *A multigrid for image deblurring with Tikhonov regularization*, Numer. Linear Algebra Appl., 12 (2005), pp. 715–729.
[10] M. Donatelli and S. Serra-Capizzano, *Anti-reflective boundary conditions and re-blurring*, Inverse Problems, 21 (2005), pp. 169–182.
[11] H. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1996.
[12] C. Estatico, *A class of filtering superoptimal preconditioners for highly ill-conditioned linear systems*, BIT, 42 (2002), pp. 753–778.
[13] G. Fiorentino and S. Serra, *Multigrid methods for symmetric positive definite block Toeplitz matrices with nonnegative generating functions*, SIAM J. Sci. Comput., 17 (1996), pp. 1068–1081.
[14] S. German and D. German, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Machine Intell., 6 (1984), pp. 721–740.
[15] W. Hackbush, *Multi-grid Methods and Applications*, Springer-Verlag, New York, 1979.
[16] M. Hanke, J. Nagy, and R. Plemmons, *Preconditioned iterative regularization for ill-posed problems*, in Numerical Linear Algebra (Kent, OH, 1992), de Gruyter, Berlin, 1993, pp. 141–163.
[17] T. Huckle and J. Staudacher, *Multigrid preconditioning and Toeplitz matrices*, Electron. Trans. Numer. Anal., 13 (2002), pp. 81–105.
[18] D. Mumford and J. Shah, *Optimal approximations of piecewise smooth functions and associated variational problems*, Comm. Pure Appl. Math., 42 (1989), pp. 577–685.
[19] M.K. Ng, R.H. Chan, and W.C. Tang, *A fast algorithm for deblurring models with Neumann boundary conditions*, SIAM J. Sci. Comput., 21 (1999), pp. 851–866.
[20] M. Piana and M. Bertero, *Projected Landweber method and preconditioning*, Inverse Problems, 13 (1997), pp. 441–463.
[21] L. Rudin, S. Osher, and E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Phys. D, 60 (1992), pp. 259–268.
[22] J.W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S.F. McCormick, ed., SIAM, Philadelphia, 1987, pp. 73–130.
[23] S. Serra-Capizzano, *Multi-iterative methods*, Comput. Math. Appl., 26 (1993), pp. 65–87.
[24] S. Serra-Capizzano, *Convergence analysis of two-grid methods for elliptic Toeplitz and PDEs matrix-sequences*, Numer. Math., 92 (2002), pp. 433–465.
[25] S. Serra-Capizzano, *A note on antireflective boundary conditions and fast deblurring models*, SIAM J. Sci. Comput., 25 (2003), pp. 1307–1325.
[26] S. Serra-Capizzano and C. Tablino-Possio, *Multigrid methods for multilevel circulant matrices*, SIAM J. Sci. Comput., 26 (2004), pp. 55–85.
[27] V. Simoncini and D.B. Szyld, *Flexible inner-outer Krylov subspace methods*, SIAM J. Numer. Anal., 40 (2003), pp. 2219–2239.
[28] A.N. Tikhonov and V.Y. Arsenin, *Solutions of Ill-Posed Problems*, John Wiley, New York, 1977.
[29] U. Trottenberg, C.W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.
[30] E. Tyrtyshnikov, *A unifying approach to some old and new theorems on preconditioning and clustering*, Linear Algebra Appl., 232 (1996), pp. 1–43.
[31] R. Vio, J. Bardsley, M. Donatelli, and W. Wamsteker, *Dealing with edge effects in least-squares image deconvolution problems*, Astron. Astrophys., 442 (2005), pp. 397–403.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2077–2097
© 2006 Society for Industrial and Applied Mathematics

AMGe—COARSENING STRATEGIES AND APPLICATION TO THE OSEEN EQUATIONS*

MARKUS WABRO†

Abstract. We provide some extensions to the algebraic multigrid method based on element interpolation (AMGe), concerning the agglomeration process, the application to nonconforming elements, and the application to the mixed finite element discretization of the Oseen linearized Navier–Stokes equations.

This last point, using AMGe for mixed finite elements, becomes straightforward because of the availability of coarse-level topologies. We show this exemplarily for the Crouzeix–Raviart element (including a stability result).

The numerical results show that it really pays off to take a closer look at the agglomeration strategy. A "wrong" choice can lead to insufficient convergence or even divergence of the overall multigrid method.

Key words. AMG, AMGe, convection-diffusion equations, Oseen equations, mixed problems

AMS subject classifications. 65N30, 65N55

DOI. 10.1137/040610350

1. Introduction.

The algebraic multigrid (AMG) method was introduced by Ruge and Stüben [14] and Brandt, McCormick, and Ruge [3] mainly for two reasons. First, one obtains a multilevel method without the need to perform any mesh refinements, which is helpful in situations with large problems and limited amount of computer memory. Second, it is intended to be used as "black-box" solver, e.g., as a coarse-grid solver in some geometric multigrid (GMG) methods. An overview of the technique itself and its applications can be found, e.g., in Stüben [16].

Unfortunately there are situations (especially when the operators are not represented by symmetric, positive definite M-matrices) where classical AMG might fail. For finite element (FE) discretized problems one way out was the development of the *algebraic multigrid method based on element interpolation* (AMGe).

Until now this method has undergone three important stages of development. It was introduced by Brezina et al. [4], where the classical techniques for the coarse node/fine node (C/F) splitting are used, but a new method of constructing the interpolation operator, based on the local element stiffness matrices, is introduced. In [8] the coarse grid construction is changed in a way that allows one to extract topological information on coarse levels (coarse faces, edges, and elements and their connectivity). The interpolation still depends on the element stiffness matrices, which are assembled to Neumann-type stiffness matrices on local patches (e.g., the coarse-level elements) and which have to be stored during the whole setup process. This problem is overcome by the approach in [7]. Here, an extension (extrapolation) operator (depending only on the assembled stiffness matrix) is used to construct an approximation to the local Neumann-type matrices.

*Received by the editors June 22, 2004; accepted for publication (in revised form) July 25, 2005; published electronically February 3, 2006. This work was supported by Austrian Science Foundation "Fonds zur Förderung der wissenschaftlichen Forschung (FWF)" grant P14953 "Robust Algebraic Multigrid Methods and Their Parallelization."

http://www.siam.org/journals/sisc/27-6/61035.html

†Institute of Computational Mathematics, Johannes Kepler University, Linz, A-4040 Austria (markus.wabro@ jku.at).

The full availability of the topological information on all levels opens up the possibility of constructing AMGe methods for various types of FE discretizations, especially for mixed FE discretizations of saddle point problems.

We will exploit this property when we construct a coupled AMGe (cAMGe) method for the Oseen linearized Navier–Stokes equations. "Coupled" here is meant in contrast to methods where pressure and velocity equations are iteratively decoupled (e.g., using SIMPLE or Uzawa methods) and AMG is used for the resulting scalar problems. We want to have an "all-at-once" approach, although we then have to deal with an indefinite system and with possible (inf-sup) stability problems. Some possibilities of coupled AMG (not AMGe) methods for the Oseen problem can be found in [20]; in [19] there are also first steps towards cAMGe.

In this article we will exemplarily construct a cAMGe method for the Crouzeix–Raviart (P_1^{nc}-P_0) discretization of the coupled problem and show its coarse-level stability.

Besides the development of this coupled AMGe method, the second emphasis of this work is put on the construction of the coarse levels, i.e., how to agglomerate fine-level elements to coarse-level elements. As we will see, this has a crucial influence on the performance of the method.

The article is organized as follows. After describing the main ingredients of an (element agglomerating) AMGe method, we present three new techniques for the agglomeration process, and we show how to build the interpolations for three different FE types. In the third section we apply the ideas to the Crouzeix–Raviart discretized Oseen equations and derive a stability result. The article is concluded with numerical experiments for the presented methods.

1.1. Notation. We want to construct an AMG method for a set of linear equations

$$K_1 x = b_1, \tag{1.1}$$

where K_1 is a regular $n_1 \times n_1$ matrix and b_1 is a given right-hand side. The index indicates the level; 1 is the finest level, and L will be the coarsest.

At the moment we do not specify more properties of the matrix, as we try to be as general as possible. The methods in sections 2.1 and 2.2 are in fact independent of the matrix (at least in their versions presented here), in section 2.3 we will need regularity of some submatrices (sufficient conditions will be given there), and the results in section 3 apply only for the special mixed FE discretization of the Oseen equations given there.

We create full rank prolongation matrices $P_{l+1}^l : \mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}$, $l = 1, \dots, L-1$ and $n_1 > n_2 > \cdots > n_L$, and a set of coarse-level matrices K_l with

$$K_{l+1} = (P_{l+1}^l)^T K_l P_{l+1}^l. \tag{1.2}$$

The system in (1.1) originates in an FE discretization of some partial differential equation (PDE) on a polygonal (resp., polyhedral) domain $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$. We consider FEs based on triangles (resp., tetrahedra) with straight (flat) edges (faces); thus we assume that some partitioning of Ω into triangles (resp., tetrahedra) $\Omega = \bigcup_i \tau_i$ is given, and we denote the set of elements $\mathcal{T}_h := \{\tau_1, \tau_2, \dots\}$.

2. AMGe building blocks. In this section we will present the ingredients of the coarse-level construction of an element agglomerating AMGe method:

1. the C/F splitting, which in most cases consists of

(i) the element agglomeration process and
(ii) the identification of coarse-level faces, edges, and nodes,

2. the construction of the interpolation matrix from the coarse degrees of freedom (dofs) to the fine dofs.

We want to remark that (different from some other AMG approaches) for the methods presented in this paper we will assume that total information on the fine grid is available (topology, mesh width, etc.).

2.1. Identification of the coarse-level topology. Assume that on one level (e.g., on the discretization level) we know the element-to-node connectivity, i.e., which nodes are part of a given element. Assume further that a method for the agglomeration of elements is known, satisfying the requirements that each element is part of one unique agglomerate and that each agglomerate is a connected set, meaning that for any two elements that are part of the same agglomerate there exists a path of elements of this agglomerate connecting the two elements, where each two consecutive "path-elements" are connected via edges (in 2 dimensions) or faces (in 3 dimensions). Then we can apply the following algorithm for the creation of the coarse-level topology (as presented in [8]).

ALGORITHM 1. AMGE COARSE-LEVEL TOPOLOGY.

1. Agglomerate the fine elements to coarse elements E_j (with the above properties).

2. Consider all intersections $E_j \cap E_k$ for all pairs of different agglomerated elements E_j and E_k. If such an intersection is maximal, i.e., is not contained in any other intersection, then it is called a face.

3. Consider the faces as sets of nodes. For each node n compute the intersection $\bigcap\{\text{all faces which contain } n\}$. Now the set of minimal, nonempty intersections defines the vertices.

We have formulated the algorithm for the 3D case, but it can be directly applied to 2D problems (then the "faces" correspond to edges). If (in the 3D case) one additionally wants to identify edges, then this can be done in step 3 using the set of minimal, nonempty intersections which are not already vertices.

2.2. Element agglomeration process. For the agglomeration of fine elements to coarse elements we present four different techniques, a classical one and three new possibilities. All four techniques are presented in their "standard" form; i.e., for example no anisotropies or parameter jumps which could (or should) be handled by the agglomeration process are considered.

2.2.1. Jones–Vassilevski element agglomeration. We will start with the "classical" Jones–Vassilevski algorithm [8].

ALGORITHM 2. JONES–VASSILEVSKI ELEMENT AGGLOMERATION.

Assume we have a set of finer level elements $\{e_j\}$ and faces $\{f_j\}$, and introduce an integer weight $w(f_j)$ for each face f_j.

initiate. Set $w(f) \leftarrow 0$ for all faces f;

global search. Find a face f with maximal $w(f)$, if $w(f) = -1$ we are done; set $E \leftarrow \emptyset$;

1. Set $E \leftarrow E \cup e_1 \cup e_2$, where $e_1 \cap e_2 = f$ and set $w_{\max} \leftarrow w(f)$, $w(f) \leftarrow -1$;

2. Increment $w(f_1) \leftarrow w(f_1) + 1$ for all faces f_1 such that $w(f_1) \neq -1$ and f_1 is a neighbor of f;

3. Increment $w(f_2) \leftarrow w(f_2) + 1$ for all faces f_2 such that $w(f_2) \neq -1$, f_2 is a neighbor of f, and f_2 and f are faces of a common element;

4. From the neighbors of f, choose a face g with maximal $w(g)$; if $w(g) \geq w_{\max}$ set $f \leftarrow g$ and go to step 1;

5. If all neighbors of f have smaller weight than $w_{\max}$, the agglomerated element E is complete; set $w(g) \leftarrow -1$ for all faces of the elements e contained in E;
go to **global search**.

Remark 1. In [8] also modifications to this algorithm are presented which allow some kind of semicoarsening, i.e., coarsening with the focus in one specific direction (for example determined by convection).

For the 2D case this algorithm mostly produces nice agglomerated elements, but in the 3D case some strange shapes may occur; therefore some adjustments of the algorithm seem to be necessary.

2.2.2. Red-grey-black element agglomeration. A second method which is fast and produces good-looking agglomerates but often leads to a too strong coarsening (what has a disadvantageous influence on the h-independence) is the following.

Algorithm 3. Red-grey-black element agglomeration.
At the beginning no element is colored.
repeat until all elements are colored
begin
choose an uncolored element, this is colored black;
color all uncolored or grey neighboring elements red
(where "neighboring" could be induced by faces, edges, or nodes);
color all uncolored elements neighboring red elements grey;
end
the black elements plus surrounding red elements build the agglomerated elements; each grey element is appended to the agglomerate where it "fits best" (e.g., to the agglomerate it shares the largest face with).

This algorithm has some similarities to the aggregation technique by Vanek, Mandel, and Brezina [17], with one main difference: there nodes are aggregated, but here elements are aggregated.

2.2.3. Optimization approach. Another class of methods (which originally was aimed at the coarse-level construction of multigrid methods for finite volume discretizations) was developed by Moulitas and Karypis [12] and implemented in the software package MGridGen [11]. They formulate an optimization problem where a coarse grid is generated that minimizes some objective function F (i.e., maximizes the quality of the grid in some sense), and the size of the coarse grid elements is subject to some upper and lower bounds. Examples for such objective functions are the sum or the maximum of the aspect ratios of the coarse elements.

The method for solving these optimization problems is based on a multilevel technique described in [12] and the references therein; we sketch it roughly.

Algorithm 4. MGridGen.

1. *Coarsening phase:* A sequence of approximate representations of the optimization problem is generated by constructing coarser and coarser graphs (this sequence of graphs has nothing to do with the levels in our multigrid method!).

2. *Initial solution phase:* The optimization problem is solved on the coarsest level.

3. *Uncoarsening phase:* The optimization problem is solved (approximately) on the finer levels, using the solution on the coarser levels.

As the MGridGen library is freely available, the application of the method is particularly easy.

2.2.4. Bottom-up graph partitioning. The three methods above have in common a top-down strategy; i.e., we start on the finest level and agglomerate elements to obtain the next coarser level. The following method will work the other way round, i.e., bottom-up.

ALGORITHM 5. BOTTOM-UP GRAPH PARTITIONING.

1. Build the element-element connectivity graph for the fine level mesh.
2. Utilize any mesh/graph partitioning algorithm to obtain a (small) number of subgraphs. This set of subgraphs is used as the coarsest level (each subgraph is an element).
3. Now apply this strategy recursively on the subgraphs, until only one-node-graphs remain: the finest level.

Remark 2. It may happen that the penultimate level is very close to the finest level. In this case this level should be ignored to avoid a waste of memory resources.

For the graph partitioning algorithm mentioned in step 2 of Algorithm 5 we use the METIS library by Karypis and Kumar [10]. (The underlying techniques are explained in [9].)

LEMMA 2.1. *If one of Algorithms* 2, 3, *or* 4 *is applied in the construction of an AMGe method, the number of arithmetical operations during the setup phase* $\mathcal{Q}_i^S(n_1)$, $i = 2, 3, 4$, *is* $\mathcal{O}(n_1)$. *For Algorithm* 5 *we obtain* $\mathcal{Q}_5^S(n_1) = \mathcal{O}(n_1 \log(n_1))$.

Proof. Assume for all algorithms a constant coarsening (or refinement) rate r and a minimal number of coarsest grid nodes n_c. Then the number of levels is given by

$$L = \log_r \left(\frac{n_c}{n_1} \right) + 1,$$

and the total complexity for Algorithms 2, 3, and 4 (assuming that the complexity on each level is proportional to the number of nodes) is calculated as

$$\begin{aligned} \mathcal{Q}_i^S(n_1) &= \mathcal{O}\left(n_1 + r n_1 + \cdots + r^{L-2} n_1\right) \\ &= \mathcal{O}\left(n_1 \cdot \frac{1 - r^{L-1}}{1 - r}\right) = \mathcal{O}\left(\frac{n_1 - n_c}{1 - r}\right) \\ &= \mathcal{O}(n_1) \quad \text{for } i = 2, 3, 4. \end{aligned}$$

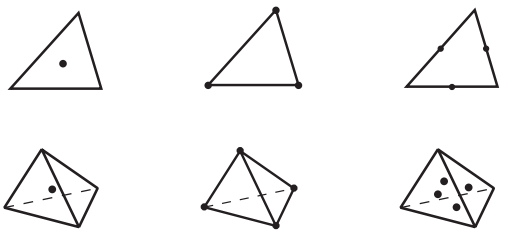
For Algorithm 5 the complexity on each level stays constant; thus the overall complexity is calculated as

$$\mathcal{Q}_5^S(n_1) = \mathcal{O}((L-1) \cdot n_1) = \mathcal{O}(n_1 \log n_1). \qquad \square$$

2.3. Interpolation. For the construction of the interpolation we can utilize the topological information we have obtained in section 2.1. We will show this exemplarily for three types of finite elements.

2.3.1. P_0-element. For the FE with piecewise constant shape functions (as in Figure 1(a)) the construction of the interpolation is particularly easy. Each fine-level element is part of exactly one coarse-level agglomerate; thus we can use identity prolongation.

2.3.2. P_1-element. For piecewise linear, continuous elements (Figure 1(b)) we apply a strategy described by Henson and Vassilevski [7], which generalizes the method used by Jones and Vassilevski [8].



(a) P_0 (b) P_1 (c) P_1^{nc}

FIG. 1. *Some mixed finite elements for triangles and tetrahedra. Dots indicate dofs.*

We first define the neighborhood of a (fine-level) node n by

$$\Omega(n) := \bigcup \{\text{all agglomerated elements that contain } n\}$$

and the minimal set by

$$\Lambda(n) := \bigcap \{\text{all agglomerated elements that contain } n\}.$$

($\Lambda(n)$ can be a node, edge, face, or element.) The set of coarse dofs in $\Omega(n)$ is denoted by $\Omega_c(n)$. Then we define the extended neighborhood

$$\overline{\Omega}(n) := \{j \notin \Omega(n) : k_{ij} \neq 0 \text{ for some } i \in \Omega(n) \setminus \Omega_c(n)\} \cup \Omega(n)$$

(where k_{ij} is the (i,j)th entry of the fine-level matrix K_l).

As the coarse nodes form a subset of the fine nodes the values there can be identically prolongated. For the edges, faces, and cells we proceed recursively as follows. Assume for a set $\Lambda(n)$ that the interpolation on the unknowns in $\partial\Lambda(n)$ has been fixed,[1] and we want to calculate the interpolation on the nodes in $\Lambda(n) \setminus \partial\Lambda(n)$. For that we build the local matrix of $\Omega(n)$ (extracted from the global matrix K_l) with the underlying partitioning $(\Omega(n) \setminus \partial\Lambda(n)) \cup \partial\Lambda(n) \cup \big(\overline{\Omega}(n) \setminus \Omega(n)\big)$

$$K_{\overline{\Omega}(n)} = \begin{pmatrix} K_{ii} & K_{ib} & K_{ie} \\ K_{bi} & K_{bb} & K_{be} \\ K_{ei} & K_{eb} & K_{ee} \end{pmatrix} \begin{matrix} \} \\ \} \\ \} \end{matrix} \begin{matrix} \Omega(n) \setminus \partial\Lambda(n) \\ \partial\Lambda(n) \\ \overline{\Omega}(n) \setminus \Omega(n) \end{matrix}$$

(i stands for interior, b for boundary, e for extended neighborhood). In [8] the small matrix

$$K_{\Omega(n)} = \begin{pmatrix} \tilde{K}_{ii} & \tilde{K}_{ib} \\ \tilde{K}_{bi} & \tilde{K}_{bb} \end{pmatrix} \tag{2.1}$$

is built using the local stiffness matrices; therefore it is a Neumann-type operator. Extracting it directly from the global stiffness matrix would lead to a Dirichlet-type

[1] The definition of the "boundary" $\partial\Lambda(n)$ is straightforward: if $\Lambda(n)$ is a face/edge, then $\partial\Lambda(n)$ are those nodes of $\Lambda(n)$ which belong to more than one face/edge; if $\Lambda(n)$ is an agglomerated element, then $\partial\Lambda(n)$ is the union of faces of this element.

operator. Thus, in [7] an extension mapping $E(n) : \Omega(n) \to \overline{\Omega}(n)$ is built, which with the above partitioning can be written as

$$E(n) = \begin{pmatrix} I & 0 \\ 0 & I \\ E_{ei} & E_{eb} \end{pmatrix}$$

and is used to get something similar to (2.1):

$$\hat{K}_{\Omega(n)} = \begin{pmatrix} \hat{K}_{ii} & \hat{K}_{ib} \\ \hat{K}_{bi} & \hat{K}_{bb} \end{pmatrix} := \begin{pmatrix} K_{ii} + K_{ie}E_{ei} & K_{ib} + K_{ie}E_{eb} \\ K_{bi} + K_{be}E_{ei} & K_{bb} + K_{be}E_{eb} \end{pmatrix}.$$

With this technique the storing of the element stiffness matrices is no longer needed (if $E(n)$ can be constructed independently of them). Now the prolongation on $\Lambda(n) \setminus \partial\Lambda(n)$ is built by extracting the corresponding entries of

$$-\hat{K}_{ii}^{-1}\hat{K}_{ib}.$$

Remark 3. In this section we have assumed that the matrices K_{ii} and $\hat{K}_{ii}$ are invertible. For symmetric positive semidefinite matrices $K_{\overline{\Omega}(n)}$ and a sufficiently rich set of boundary dofs $\partial\Lambda(n)$ this can be shown for K_{ii} (see Lemma 2.1 in [8]). Furthermore if E_{ei} is sufficiently small in norm, then $\hat{K}_{ii}$ also will be invertible (as stated in [7]).

Remark 4. For the situation in (2.1) and for symmetric positive definite K_l the setting

$$u_i = -\tilde{K}_{ii}^{-1}\tilde{K}_{ib}u_b$$

solves the minimal-energy problem: For given u_b, find u_i such that

$$(u_i^T u_b^T)K_{\Omega(n)} \begin{pmatrix} u_i \\ u_b \end{pmatrix}$$

is minimized.

What still has to be specified is the extension $E(n)$. In [7] various possibilities can be found, and we use what is called there the "A-extension" (which in our case is in fact a "K-extension"). If we want to extend a vector v to a dof $i \in \overline{\Omega}(n) \setminus \Omega(n)$, this extension is given by

$$v(i) = \frac{1}{\sum_{j\in S} |k_{ij}|} \sum_{j\in S} \left(|k_{ij}| \cdot v(j)\right),$$

where $S = \{j \in \Omega(i) : k_{ij} \neq 0\}$.

2.3.3. Nonconforming P_1-element (P_1^{nc}). For the nonconforming P_1^{nc} element with piecewise linear shape functions, continuous at the midpoints of the edges (resp., faces), illustrated in Figure 1(c), we propose the following strategy.

All dofs on fine edges/faces which are part of the same coarse edge/face are set equal to the dof value of this coarse edge/face.

For the dofs which lie in the interior of a coarse agglomerate we follow the same strategy as in section 2.3.2 (but without the necessity of either storing the element stiffness matrices or constructing some extension operators). The prolongation matrices for the interior of coarse elements are given by

$$-K_{ii}^{-1}K_{ib},$$

where K_{ii} and K_{ib} can be directly extracted from the global stiffness matrix.

3. Application to the Oseen equations. The methods above can now be directly applied to the mixed FE discretization of the Oseen equations (i.e., the fixed-point linearized stationary, incompressible Navier–Stokes equations) in $\Omega \subset \mathbb{R}^d$, $d \in \{2, 3\}$:

$$-\nu\Delta\mathbf{u} + (\mathbf{w}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \tag{3.1a}$$

$$\operatorname{div}\mathbf{u} = 0, \tag{3.1b}$$

where $\mathbf{u}(\mathbf{x})$ denotes the velocity of the fluid at point $\mathbf{x}$, $p(\mathbf{x})$ is the pressure, $\mathbf{f}(\mathbf{x})$ is some given right-hand side, ν is the given viscosity (which we assume constant in $\mathbf{x}$), and $\mathbf{w}(\mathbf{x})$ is the old approximation of the velocity.

For simplicity we assume Dirichlet boundary conditions on the whole boundary $\Gamma := \partial\Omega$:

$$\mathbf{u}|_\Gamma = \bar{\mathbf{u}}.$$

We discretize the system using a mixed FE method (for details we refer the reader, e.g., to Brezzi and Fortin [5] or Braess [1]) and obtain the following problem: Find a couple $(\mathbf{u}_h, p_h)$ in some mixed FE space $\mathbf{U}_h \times Q_h$ such that

$$\bar{a}(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) = \langle F, \mathbf{v}_h\rangle \quad \text{for all } \mathbf{v}_h \in \mathbf{U}_h, \tag{3.2a}$$

$$b(\mathbf{u}_h, q_h) = 0 \quad \text{for all } q_h \in Q_h, \tag{3.2b}$$

where

$$\bar{a}(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) = a_D(\mathbf{u}_h, \mathbf{v}_h) + a_C(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) + a_S(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h)$$

and

$$a_D(\mathbf{u}_h, \mathbf{v}_h) = \nu(\nabla\mathbf{u}_h, \nabla\mathbf{v}_h), \qquad b(\mathbf{u}_h, q_h) = -(\operatorname{div}\mathbf{u}_h, q_h),$$
$$a_C(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) = \left((\mathbf{w}_h\cdot\nabla)\mathbf{u}_h, \mathbf{v}_h\right), \qquad \langle F, .\rangle = (\mathbf{f}, .),$$

and $a_S(.; ., .)$ originates in some convection stabilization ("D" stands for diffusion, "C" for convection, and "S" for stabilization; $(., .)$ is the standard L_2 inner product). We use the *streamline upwinding Petrov Galerkin method* (SUPG) and therefore

$$a_S(\mathbf{w}_h; \mathbf{u}_h, \mathbf{v}_h) = \beta_h\left((\mathbf{w}_h\cdot\nabla)\mathbf{u}_h, (\mathbf{w}_h\cdot\nabla)\mathbf{v}_h\right),$$

with some stabilization parameter β_h (details can be found, e.g., in [13]).

We denote the *FE isomorphisms* between the discrete spaces and the spaces of coefficient vectors by $\phi_U : (\mathbb{R}^n)^d \to \mathbf{U}_h$ and $\phi_Q : \mathbb{R}^m \to Q_h$. The underlined notation is used to indicate their inverses, i.e.,

$$\phi_U\underline{\mathbf{v}}_h = \mathbf{v}_h, \qquad \underline{\phi_U\underline{\mathbf{v}}_h} = \underline{\mathbf{v}}_h, \tag{3.3a}$$

$$\phi_Q\underline{q}_h = q_h, \qquad \underline{\phi_Q\underline{q}_h} = \underline{q}_h. \tag{3.3b}$$

If it is clear from the context we omit the underlines and ϕ's and identify $\mathbf{v}_h \in \mathbf{U}_h$ and the associated $\underline{\mathbf{v}}_h \in (\mathbb{R}^n)^d$ and analogously q_h and $\underline{q}_h$.

If the mixed FE is now built of elements for which the construction of an AMGe interpolation is known, we can easily generate a cAMGe method. The coarsening is independent of the concrete element, the interpolations have to be built separately for the pressure and velocity hierarchies.

After having constructed the coarse-level hierarchies we need an appropriate smoother. Possibilities are well known smoothers for GMG methods, such as transforming smoothers (Wittum [21, 22]), the Braess–Sarazin smoother (see [2] and Zulehner [23]), or Vanka smoothers (see [18] and Schöberl and Zulehner [15]). Although developed for the GMG case, they can be directly applied to AMG.

In principal we now have everything we need for an AMG method. But it is not obvious a priori that the method works, as the (inf-sup) stability of the coarse-level systems is not automatically given (and a lack thereof would destroy the convergence properties of the method; cf. Wabro [20, 19]).

For the *Crouzeix–Raviart* element P_1^{nc}-P_0 we exemplarily show how a stability result can be obtained. The technique of the proof heavily depends on the fact that on each level a topology is available.

We first introduce the following notations. The intergrid transfer operators for pressure will be denoted by J_{l+1}^l, for velocity by $\tilde{I}_{l+1}^l$, for velocity components by I_{l+1}^l, i.e.,

$$P_{l+1}^l = \begin{pmatrix} \tilde{I}_{l+1}^l & \\ & J_{l+1}^l \end{pmatrix}, \quad \tilde{I}_{l+1}^l = \begin{pmatrix} I_{l+1}^l & & \\ & I_{l+1}^l & \\ & & I_{l+1}^l \end{pmatrix} \quad \text{(in 3 dimensions)}.$$

We denote the spaces for velocity and pressure unknown vectors at level l by $\underline{\mathbf{U}}_l := (\mathbb{R}^{n_l})^d$ and $\underline{Q}_l := \mathbb{R}^{m_l}$ and the coarse function spaces by

$$\begin{aligned} \mathbf{U}_l &:= \left\{ \mathbf{v} \in \mathbf{U}_h : \ \exists \underline{\mathbf{w}} \in \underline{\mathbf{U}}_l \text{ such that } \underline{\mathbf{v}} = \tilde{I}_2^1 \tilde{I}_3^2 \dots \tilde{I}_l^{l-1} \underline{\mathbf{w}} \right\}, \\ Q_l &:= \left\{ p \in Q_h : \ \exists \underline{q} \in \underline{Q}_l \text{ such that } \underline{p} = J_2^1 J_3^2 \dots J_l^{l-1} \underline{q} \right\}. \end{aligned}$$

We want to emphasize that because of this construction, all coarse-level function spaces are included in the finest-level function spaces.

Analogous to (3.3) we introduce the *FE-AMG isomorphisms*

$$\phi_U^l : \underline{\mathbf{U}}_l \to \mathbf{U}_l \ \text{ and } \ \phi_Q^l : \underline{Q}_l \to Q_l, \tag{3.4}$$

and we will often identify elements of $\underline{\mathbf{U}}_l$ and $\mathbf{U}_l$ and $\underline{Q}_l$ and Q_l.

The following lemma is just a translation of the well-known result of Fortin [6].

LEMMA 3.1. *Assume that for given l there exists a linear operator $\Pi_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_l$ with*

$$b(\Pi_{l-1}^l \mathbf{v}_{l-1}, q_l) = b(\mathbf{v}_{l-1}, J_l^{l-1} q_l) \quad \textit{for all } q_l \in Q_l \textit{ and } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1} \tag{3.5}$$

and that

$$\|\Pi_{l-1}^l \mathbf{v}_{l-1}\|_1 \le \delta \|\mathbf{v}_{l-1}\|_1 \quad \textit{for all } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}, \tag{3.6}$$

with δ independent of h and l.

Assume that on level $l-1$ an inf-sup condition holds, i.e.,

$$\inf_{0 \neq q_{l-1} \in Q_{l-1}} \sup_{\mathbf{0} \neq \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}} \frac{b(\mathbf{v}_{l-1}, q_{l-1})}{\|\mathbf{v}_{l-1}\|_1 \|q_{l-1}\|_0} \ge \beta_{l-1}.$$

Then an inf-sup condition holds on level l.

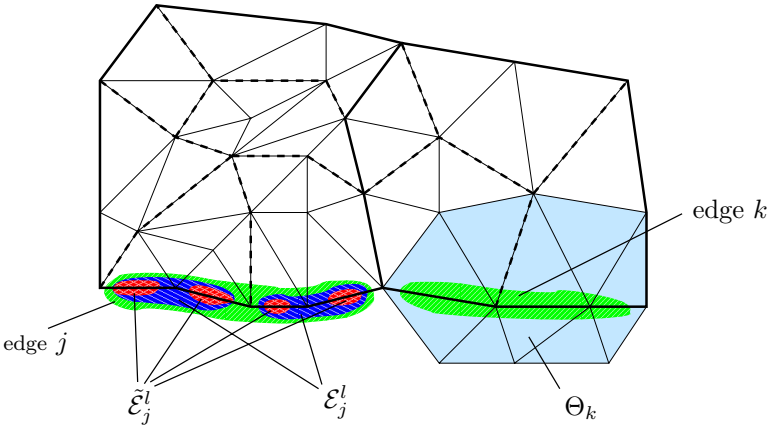


FIG. 2. *The solid thick black lines describe two level-l elements, the dashed lines level* $(l-1)$*-elements, and the solid thin lines the finest-level elements. For edge* j *this figure shows the sets* $\mathcal{E}_j^l$ *and* $\tilde{\mathcal{E}}_j^l$*, and for edge* k *it shows the tube* Θ_k*.*

Proof.

$$\begin{aligned}\inf_{0\neq q_l\in Q_l}\sup_{\mathbf{0}\neq\mathbf{v}_l\in\mathbf{U}_l}\frac{b(\mathbf{v}_l,q_l)}{\|\mathbf{v}_l\|_1\|q_l\|_0} &\geq \inf_{0\neq q_l\in Q_l}\sup_{\mathbf{0}\neq\mathbf{v}_{l-1}\in\mathbf{U}_{l-1}}\frac{b(\Pi_{l-1}^l\mathbf{v}_{l-1},q_l)}{\|\Pi_{l-1}^l\mathbf{v}_{l-1}\|_1\|q_l\|_0}\\ &= \inf_{0\neq q_l\in Q_l}\sup_{\mathbf{0}\neq\mathbf{v}_{l-1}\in\mathbf{U}_{l-1}}\frac{b(\mathbf{v}_{l-1},J_l^{l-1}q_l)}{\|\mathbf{v}_{l-1}\|_1\|q_l\|_0}\cdot\frac{\|\mathbf{v}_{l-1}\|_1}{\|\Pi_{l-1}^l\mathbf{v}_{l-1}\|_1}\\ &\geq \frac{\beta_{l-1}}{\delta}. \qquad \square\end{aligned}$$

THEOREM 3.2. *Assume that*

- *on the finest level we have*

$$\frac{h_{max}}{h_{min}}\leq c_0, \tag{3.7}$$

  where h_{max} *is the maximal element diameter,* h_{min} *is the minimal diameter, and* c_0 *is a positive constant,*
- *assume that the (finest-level) mesh is shape regular, i.e.,*

$$\frac{(\textit{length of longest edge of }\tau_j)^2}{V(\tau_j)}<c_1 \tag{3.8}$$

  for all triangles (resp., tetrahedra) τ_j*, where* $V(\tau_j)$ *is the area (resp., volume) of* τ_j *and* c_1 *is a positive constant, and*
- *assume that the coarse levels are built as described in section* 2*, and the prolongations are as in sections* 2.3.1 *and* 2.3.3 *but based on the Stokes operator (i.e.,* $\bar{a}(\mathbf{u}_h,\mathbf{v}_h)=\bar{a}(\mathbf{0},\mathbf{u}_h,\mathbf{v}_h)$*).*

Then the inf-sup condition holds on all levels.

As we want to apply Lemma 3.1 to show this result, we construct an operator Π_{l-1}^l in the following way. Consider the 2D case first (illustrated in Figure 2). Define on level l the index sets $\mathcal{E}_j^l$ of all $(l-1)$-level edges which are part of l-level edge j.

We define the length of an l-level edge recursively by

$$e_j^l := \sum_{k \in \mathcal{E}_j^l} e_k^{l-1} \quad (\text{for } l > 1).$$

For $l = 1$ it is determined by the mesh.

We now construct Π_{l-1}^l as follows. For some $(l-1)$-level function v_{l-1} the l-level function $\Pi_{l-1}^l v_{l-1}$ is determined by its values on the (l-level) edges. We set the value on a certain l-level edge to the weighted mean of the values of v_{l-1} on the $(l-1)$-level edges which are part of the edge, i.e.,

$$\left(\underline{\Pi_{l-1}^l v_{l-1}}\right)_j = \frac{1}{e_j^l} \sum_{k \in \mathcal{E}_j^l} e_k^{l-1} \left(\underline{v}_{l-1}\right)_k .$$

For a vector-valued function $\mathbf{v}_{l-1}$ the term $\Pi_{l-1}^l \mathbf{v}_{l-1}$ will denote the application of Π_{l-1}^l to the components.

LEMMA 3.3. *The operator $\Pi_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_l$, constructed as above, fulfills* (3.5)*:*

$$b(\Pi_{l-1}^l \mathbf{v}_{l-1}, q_l) = b(\mathbf{v}_{l-1}, J_l^{l-1} q_l) \quad \textit{for all } q_l \in Q_l \textit{ and } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}.$$

Proof. We want to show that

$$\sum_j \int_{\tau_j} \operatorname{div}(\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot q_l \, \mathrm{d}\mathbf{x} = \sum_j \int_{\tau_j} \operatorname{div} \mathbf{v}_{l-1} \cdot (J_l^{l-1} q_l) \, \mathrm{d}\mathbf{x}.$$

Because $q_l = J_l^{l-1} q_l$ (in functional notation), because q_l is piecewise constant on the l-level agglomerates E_j, and because both $\mathbf{v}_{l-1}$ and $\Pi_{l-1}^l \mathbf{v}_{l-1}$ are piecewise linear on the (finest-level) elements τ_j and continuous at the midpoints of their edges, we can use partial integration to derive

$$\sum_j \int_{\tau_j} \operatorname{div}(\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot q_l \, \mathrm{d}\mathbf{x} = \sum_j q_l(E_j) \int_{\partial E_j} (\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot \mathbf{n}$$

(where $q_l(E_j)$ is the constant value of q_l on E_j) and

$$\sum_j \int_{\tau_j} \operatorname{div} \mathbf{v}_{l-1} \cdot (J_l^{l-1} q_l) \, \mathrm{d}\mathbf{x} = \sum_j q_l(E_j) \int_{\partial E_j} \mathbf{v}_{l-1} \cdot \mathbf{n}.$$

By the definition of Π_{l-1}^l we see that

$$\int_{\partial E_j} (\Pi_{l-1}^l \mathbf{v}_{l-1}) \cdot \mathbf{n} = \int_{\partial E_j} \mathbf{v}_{l-1} \cdot \mathbf{n} \quad \text{for all agglomerates } E_j;$$

therefore (3.5) is shown to be true. ∎

LEMMA 3.4. *The operator Π_{l-1}^l, constructed as above, also fulfills* (3.6)*:*

$$\|\Pi_{l-1}^l \mathbf{v}_{l-1}\|_1 \le \delta \|\mathbf{v}_{l-1}\|_1 \quad \textit{for all } \mathbf{v}_{l-1} \in \mathbf{U}_{l-1}.$$

Proof. The idea of this proof is the introduction of an auxiliary operator $\tilde{\Pi}_{l-1}^l$ on the finer level $\mathbf{U}_{l-1}$, which fulfills (3.6) and which is identical to Π_{l-1}^l on the coarse edges. Because we use energy minimization for the interpolation in the interior of

agglomerates, we will then be able to estimate Π_{l-1}^l by $\tilde{\Pi}_{l-1}^l$ which will complete the proof.

We define $\tilde{\Pi}_{l-1}^l : \mathbf{U}_{l-1} \to \mathbf{U}_{l-1}$ by

$$\underline{\tilde{\Pi}}_{l-1}^l(\underline{v}_{l-1})_j := \begin{cases} (\underline{v}_{l-1})_j & \text{if } j \notin \mathcal{E}_k^l \text{ for all } k, \\ \left(\underline{\Pi_{l-1}^l v_{l-1}}\right)_k & \text{if } j \in \mathcal{E}_k^l \text{ for a certain } k. \end{cases}$$

Note that $\tilde{\Pi}_{l-1}^l v_{l-1}$ still "lives" on level $l-1$, only the values at the l-level faces are averaged.

We try to find an upper bound for $|v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}|_1$. Define $\tilde{\mathcal{E}}_j^l$ as the index set of all *finest*-level edges which lie on coarse edge j. Set $\tilde{v}_1$ (component of finest-level function $\tilde{\mathbf{v}}_1 \in \mathbf{U}_1$) equal to $v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}$ on all (finest-level) dofs in $\bigcup_j \tilde{\mathcal{E}}_j^l$ and zero on all other (finest-level) dofs. Then because of the energy minimization in the prolongation,

$$\begin{aligned} \int_\Omega \nabla\left(v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}\right) \nabla\left(v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1}\right) \mathrm{d}\mathbf{x} &\le \int_\Omega \nabla\tilde{v}_1 \nabla\tilde{v}_1 \,\mathrm{d}\mathbf{x} \\ &\le \sum_{\substack{l\text{-level} \\ \text{edges } j}} \int_{\Theta_j} \nabla\tilde{v}_1 \nabla\tilde{v}_1 \,\mathrm{d}\mathbf{x}, \end{aligned} \tag{3.9}$$

where Θ_j is the tube of (finest-level) elements which share a point or edge with l-level edge number j (cf. Figure 2).

For a (finest-level) triangle PQR and the basis function φ_{PQ}, which is equal to 1 at the midpoint of PQ and zero at the midpoints of QR and RP, one can easily calculate

$$\int_{PQR} \nabla\varphi_{PQ} \nabla\varphi_{PQ} \,\mathrm{d}\mathbf{x} = \frac{|PQ|^2}{A(PQR)},$$

where $A(PQR)$ denotes the area of the triangle PQR. Now with (3.8) we get

$$\begin{aligned} \int_{\Theta_j} \nabla\tilde{v}_1 \nabla\tilde{v}_1 \,\mathrm{d}\mathbf{x} &\le c_1 \sum_{s\in\tilde{\mathcal{E}}_j^l} \left((\underline{\tilde{v}}_1)_s\right)^2 \\ &= c_1 \sum_{s\in\tilde{\mathcal{E}}_j^l} \left[\frac{1}{\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k} \sum_{k\in\tilde{\mathcal{E}}_j^l} e_k \left(\left(\phi_U^{1\,-1} v_{l-1}\right)_s - \left(\phi_U^{1\,-1} v_{l-1}\right)_k \right) \right]^2 \\ &\le c_1 \frac{\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k^2}{\left(\sum_{k\in\tilde{\mathcal{E}}_j^l} e_k\right)^2} \sum_{\substack{s\in\tilde{\mathcal{E}}_j^l, \\ k\in\tilde{\mathcal{E}}_j^l}} \left(\left(\phi_U^{1\,-1} v_{l-1}\right)_s - \left(\phi_U^{1\,-1} v_{l-1}\right)_k \right)^2 \\ &\le c_1 \sum_{\substack{s\in\tilde{\mathcal{E}}_j^l, \\ k\in\tilde{\mathcal{E}}_j^l}} \left(\left(\phi_U^{1\,-1} v_{l-1}\right)_s - \left(\phi_U^{1\,-1} v_{l-1}\right)_k \right)^2 \\ &\le \bar{c}_1 \sum_{\substack{s\in\mathcal{E}_j^l, \\ k\in\mathcal{E}_j^l}} \left(\left(\underline{v}_{l-1}\right)_s - \left(\underline{v}_{l-1}\right)_k \right)^2, \end{aligned} \tag{3.10}$$

where $\phi_U^{1\,-1} v$ is the representation of a coarse function v on $\underline{\mathbf{U}}_1$ as in (3.4).

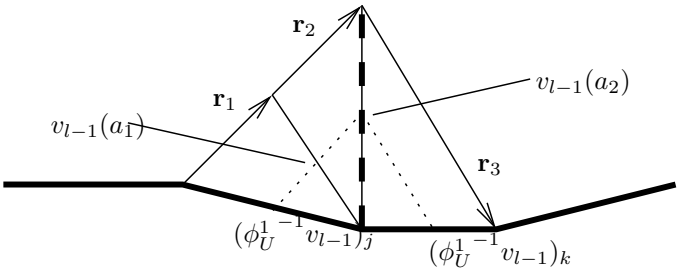


FIG. 3. *Detail of Figure* 2, *the set* $\mathcal{S}_m^{l-1}$.

We note that ∇v_{l-1} is constant on each finest-level element. Therefore we can derive the following estimate (illustrated in Figure 3). Assume that the $(l-1)$-level edges j and k share the node m. We denote the set of all finest-level elements which share the node m with $\mathcal{S}_m^{l-1}$, its index set with $\underline{\mathcal{S}}_m^{l-1}$ (where we assume without loss of generality that $\underline{\mathcal{S}}_m^{l-1} = \{1, 2, \ldots, i+1\}$). For each element τ_e in $\mathcal{S}_m^{l-1}$ we denote the edge-vector of the edge not connected to m ("in direction" $j \to k$) with $\mathbf{r}_e$. Then

$$\begin{aligned}
&(\underline{v}_{l-1})_k - (\underline{v}_{l-1})_j \\
&= ((\underline{v}_{l-1})_k - v_{l-1}(a_1)) + (v_{l-1}(a_1) - v_{l-1}(a_2)) + \cdots + (v_{l-1}(a_i) - (\underline{v}_{l-1})_j) \\
&= \frac{1}{2}\left(\nabla v_{l-1}|_{\tau_1} \cdot \mathbf{r}_1 + \cdots + \nabla v_{l-1}|_{\tau_{i+1}} \cdot \mathbf{r}_{i+1}\right) \\
&\le \frac{1}{2}\left(\sqrt{\nabla v_{l-1}|_{\tau_1} \cdot \nabla v_{l-1}|_{\tau_1}}|\mathbf{r}_1| + \cdots + \sqrt{\nabla v_{l-1}|_{\tau_{i+1}} \cdot \nabla v_{l-1}|_{\tau_{i+1}}}|\mathbf{r}_{i+1}|\right),
\end{aligned}$$

where $a_1, \ldots, a_i$ are finest-level edge midpoints as in Figure 3; thus (using the algebraic-geometric mean inequality)

$$\begin{aligned}
\left((\underline{v}_{l-1})_k - (\underline{v}_{l-1})_j\right)^2 &\le \frac{i+1}{4}\left[|\mathbf{r}_1|^2 \left(\nabla v_{l-1}|_{\tau_1}\right)^2 + \cdots + |\mathbf{r}_{i+1}|^2 \left(\nabla v_{l-1}|_{\tau_{i+1}}\right)^2\right] \\
&\le c_2 \int_{\mathcal{S}_m^{l-1}} \nabla v_{l-1} \nabla v_{l-1}\, \mathrm{d}\mathbf{x}.
\end{aligned} \tag{3.11}$$

We apply this estimate to the last term in (3.10), which is done directly for those $(l-1)$-level edges s and k which share a node. For all others we have to build a chain of connecting edges.

This leads to

$$\int_{\Theta_l} \nabla \tilde{v}_1 \nabla \tilde{v}_1\, \mathrm{d}\mathbf{x} \le c_3 \int_{\Theta_l} \nabla v_{l-1} \nabla v_{l-1}\, \mathrm{d}\mathbf{x}.$$

Now because of (3.9) we get

$$\left\| v_{l-1} - \tilde{\Pi}_{l-1}^l v_{l-1} \right\|_1 \le c_4 \|v_{l-1}\|_1;$$

thus

$$\left\| \tilde{\Pi}_{l-1}^l v_{l-1} \right\|_1 \le (1 + c_4) \|v_{l-1}\|_1.$$

Because we use energy minimization for the interpolation in the interior of coarse agglomerates, $\tilde{I}_l^{l-1} \underline{\Pi_{l-1}^l v_{l-1}}$ has minimal energy among all $(l-1)$-level functions which

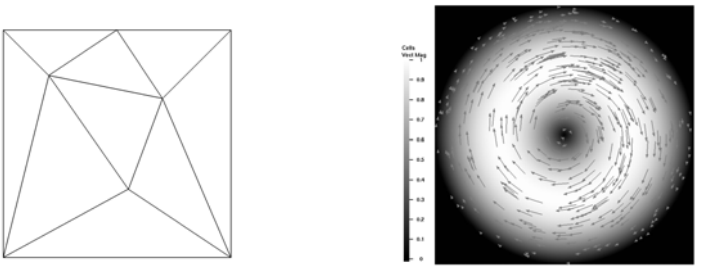


FIG. 4. *Coarsest unit-square mesh and given convection speed* $\mathbf{w}$ *for first problem.*

are identical to it on the l-level edges. Therefore

$$\left\|\Pi_{l-1}^{l} v_{l-1}\right\|_1 \le c_5 \left\|\tilde{\Pi}_{l-1}^{l} v_{l-1}\right\|_1 \le c_5(1+c_4)\|v_{l-1}\|_1. \qquad \square \tag{3.12}$$

The 3*D case.* For 3D tetrahedral elements we replace c_1 in (3.10) with $c_1 h_{j,\max}$, where $h_{j,\max}$ is the maximal element height in tube Θ_j, and c_2 in (3.11) with $c_2/h_{j,\min}$, where $h_{j,\min}$ is the minimal element height in this tube. Then because of (3.7) the argumentation remains unchanged, but the scaling argument is based on the (finest-level) tetrahedron $PQRS$:

$$\int_{PQRS} \nabla\varphi_{PQR} \nabla\varphi_{PQR} \,\mathrm{d}\mathbf{x} = \frac{A(PQR)^2}{V(PQRS)}.$$

The proof of Theorem 3.2 is completed by combining Lemmas 3.3 and 3.4 with Lemmas 3.1.

Remark 5. The constants $c_0, \ldots, c_5$ that appear in the proofs depend on the finest-level triangulation and on the agglomeration strategy (e.g., on how many fine-level edges are part of one coarse-level edge). In general they may result in $\delta > 1$ (in Lemma 3.1); thus the inf-sup constant will get worse for each additional level.

4. Numerical results. The numerical experiments were carried out using three different geometries (resp., meshes), the inevitable unit square (with meshes built by the refinement of the coarse mesh in Figure 4), the domain around a 2D symmetric valve (provided by AVL List GmbH, Graz, Austria), and the domain around two 3D valves (also provided by AVL List).

In Figures 5 and 6 we find the coarsest-level agglomerates of the valve geometries for the different techniques. One can notice that the agglomerates produced with the classical Jones–Vassilevski algorithm have frayed edges in 3D even holes. A comparison to the bottom-up approach would not be fair (as the coarsest agglomerates are the first ones, so the algorithm has full freedom to produce nice ones), but the topologies originating in the other two approaches look much better than the classical ones.

In two dimensions this is caused by the fact that the classical algorithm is the only one considered here which uses only information originating in the topology (not the geometry or the matrix). In [8] matrix-dependent variants are discussed, but they do not influence the fact that the algorithm is purely 2D and has its problems in three dimensions.

For the following tests we introduce a measure for (time) efficiency:

$$T_{0.1} := \frac{\begin{pmatrix}\text{average CPU time in minutes for the reduction}\\ \text{of the norm of the residual by a factor of 0.1}\end{pmatrix}}{\text{number of unknowns}}.$$

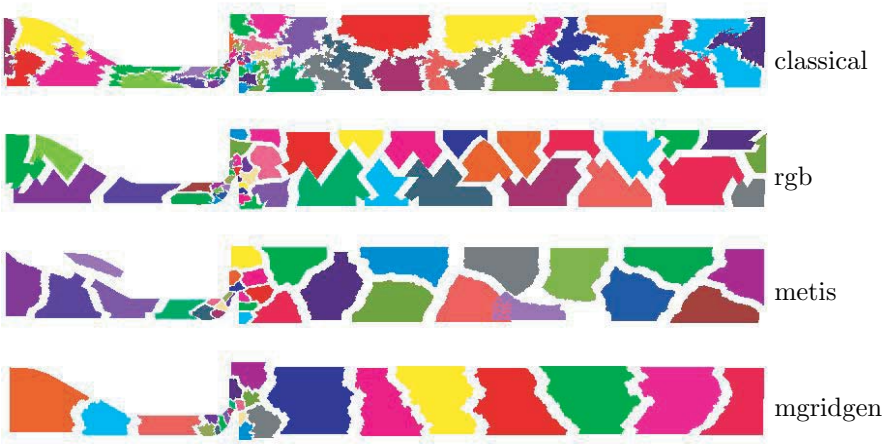


FIG. 5. *Coarsest-level agglomerates for the 2D valve. ("Classical" = Jones–Vassilevski agglomeration, "rgb" = red-gray-black, "metis" = bottom-up partitioning using metis, and "mgridgen" = optimization approach.)*



FIG. 6. *Coarsest level agglomerates for the 3D valves. ("Classical" = Jones–Vassilevski agglomeration, "rgb" = red-gray-black, "metis" = bottom-up partitioning using metis, and "mgridgen" = optimization approach; the black circles in the leftmost figure indicate holes in the agglomerates.)*

This number would be constant for different levels of refinement if we had an optimal method, i.e., if the work for a given reduction of the residual is $\mathcal{O}(n)$, where n is the number of unknowns.

All coarsening procedures are stopped at the last level with more than 20 dofs (respectively, are started there in the case of the bottom-up method). The coarse grid solver is a standard LU decomposition.

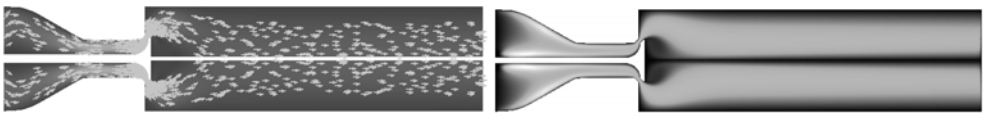


FIG. 7. *A solution of the 2D valve problem (right part of the figure) with* $\mathbf{w}$ *given as in the left part of the figure. On the symmetry plane homogeneous Dirichlet conditions are imposed.*

Remark 6. As we do not want to compare different smoothing strategies and multigrid cycles but only different coarsening strategies, we have fixed the cycle, the smoother, and the number of smoothing steps for each of the following tests.

In order to get "multigrid-like" behavior, i.e., convergence rates (mostly) independent of the refinement level, the number of smoothing steps has to be sufficiently large. For some situations this can be significantly larger as in GMG methods.

4.1. Convection-diffusion equations. For all examples in this section we use a P_1-discretization of the convection-diffusion equation in some domain $\Omega \subset \mathbb{R}^d$:

$$-\nu\Delta u + (\mathbf{w}\cdot\nabla)u = f, \tag{4.1}$$

with given (divergence-free) convective speed $\mathbf{w}$, right-hand side f, viscosity ν, and Dirichlet boundary conditions $u|_\Gamma = \bar{u}$.

The first problem we are considering lives on the unit square, with

$$\mathbf{w}(\mathbf{x}) = \begin{pmatrix} m_2 - x_2 \\ x_1 - m_1 \end{pmatrix} \cdot \begin{cases} -16\|\mathbf{m}-\mathbf{x}\| + 8 & \text{if } \|\mathbf{m}-\mathbf{x}\| < 0.5, \\ 0 & \text{otherwise,} \end{cases} \quad \mathbf{m} = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{0}-\mathbf{x}\| < 1, \\ 0 & \text{otherwise} \end{cases}$$

(cf. Figure 4), and $\bar{u} = 0$.

The second problem was the 2D valve, with $\mathbf{w}$ originating in the solution of the full Navier–Stokes equations, with a source ($f \neq 0$) near the inlet (see Figure 7 for a visualization of $\mathbf{w}$ and a solution of the problem).

For both problems we performed h-dependence tests (cf. Table 1 and Figure 8; for both we used a W-8-8 cycle with Gauss–Seidel smoothers). The values there concern only the iteration phase, not the setup. A comparison of the CPU time needed for setup (construction of the topology and of the prolongation) and the actual solution phase can be found in Figure 9(a).

The next test deals with the ν dependence (on a fixed grid). The results can be found in Table 2 and Figure 10 (cycles used are the same as before, i.e., W-8-8, Gauss–Seidel smoother).

Some h- and ν-dependence test for the 3D valves (again with $\mathbf{w}$ coming from the solution of a Navier–Stokes problem) can be found in Table 3 (W-8-8 cycles, Gauss–Seidel smoothers). The timings for the preprocessing phases and the multigrid iterations are illustrated in Figure 9(b). Here, the red-gray-black algorithm leads to a rather unbalanced method. The edges of the coarser levels are somehow joined by many neighboring elements, and thus the systems which have to be solved for the edge prolongation grow large, leading to the huge setup time.

TABLE 1

Dependence of the reduction per multigrid step (red/step) and $T_{0.1}$ on grid refinements for the unit-square problem ($\nu = 0.01$) and the 2D valve ($\nu = 0.001$).

Unit square

Ref. level	5	6	7	8
Dofs	4689	18593	74049	295553
Classical				
red/step	0.10	0.18	0.22	0.27
$T_{0.1}$	1.8e-6	2.4e-6	3.5e-6	4.8e-6
Rred-gray-black				
red/step	0.13	0.16	0.36	0.69
$T_{0.1}$	4.8e-7	4.8e-7	9.6e-7	2.8e-6
MGridGen				
red/step	0.16	0.05	0.06	0.06
$T_{0.1}$	7.1e-7	7.1e-7	8.1e-7	9.1e-7
Bottom-up, (metis, 4-fold refinement)				
red/step	0.13	0.06	0.11	0.16
$T_{0.1}$	1.3e-6	1.3e-6	1.7e-6	2.1e-6
Bottom-up, (metis, 5-fold refinement)				
red/step	0.14	0.13	0.15	0.19
$T_{0.1}$	1.2e-6	1.2e-6	1.5e-6	1.9e-6

2D valve

Ref. level	2	3	4
Dofs	11621	45865	182225
Classical			
red/step	0.06	0.08	0.16
$T_{0.1}$	1.2e-6	1.9e-6	3e-6
Red-gray-black			
red/step	0.07	0.13	0.27
$T_{0.1}$	4.7e-7	4.7e-7	7.8e-7
MGridGen			
red/step	0.05	0.05	0.07
$T_{0.1}$	7e-7	7.5e-7	8.8e-7
Bottom-up, (metis, 5-fold refinement)			
red/step	0.02	0.02	0.4
$T_{0.1}$	7.9e-7	7.9e-7	2e-6

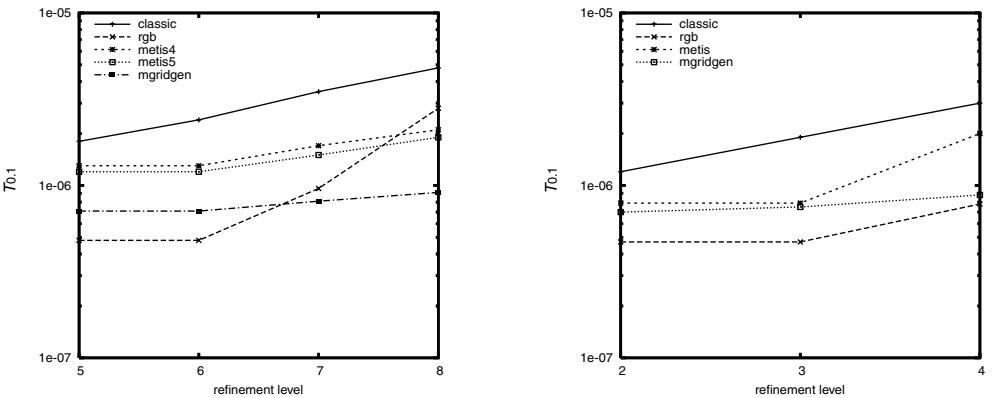
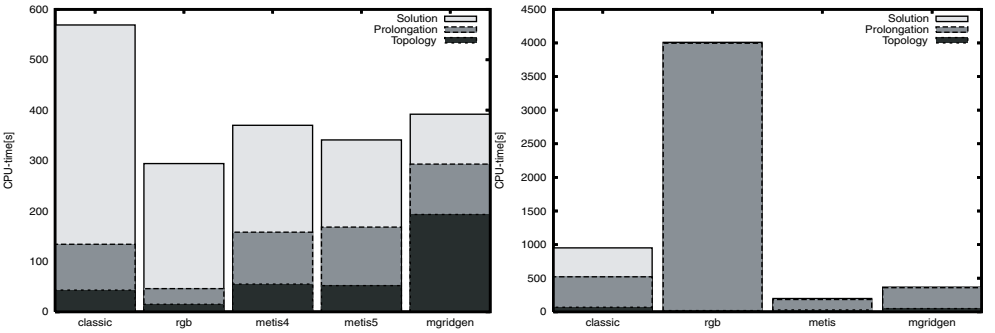


FIG. 8. *Dependence of the efficiency on the refinement level for the unit-square (left) and the 2D valve problem (right).*

In Table 4 we have collected the values for grid complexities σ^{Ω} and operator complexities σ^{A} for representative test cases.[2] In all tests the operator complexities for the classical method are higher than for the other methods, caused by more neighbors on the coarse level, and leading to the high solve times despite the reasonable reduction rates (cf. Figure 9, Table 1). In three dimensions this gets even worse, although this must not be overrated, because as already stated, the direct application of this method in three dimensions may not be sensible.

The tests show that the choice of the agglomeration strategy has a strong influence on the efficiency of the method, sometimes even on the question, if the method is converging. The general tendency which can be observed is that the optimization

[2] σ^{Ω} is the ratio of the total number of unknowns on all levels to that on the finest level, whereas σ^{A} is the ratio of the total number of nonzero entries in all matrices to that in the finest-level matrix.



(a) unit-square, 295553 dofs, $\nu = 0.01$ (b) 3D valves, 32463 dofs, $\nu = 0.001$

FIG. 9. *CPU time for the important steps of the AMGe solution process for the convection-diffusion problem ("metis4" (resp., "metis5") denotes the bottom-up strategy using metis with* 4*-fold (resp.,* 5*-fold) refinement).*

TABLE 2

Dependence of the reduction per multigrid step (red/step) and $T_{0.1}$ *on* ν *for the unit-square problem (*74049 *dofs) and the* 2*D valve (*45865 *dofs). For* $\nu < 7 \cdot 10^{-4}$ *(resp.,* $\nu < 5 \cdot 10^{-5}$*) we had no convergence. (For the red-gray-black method this occurred already for* $\nu < 4 \cdot 10^{-3}$ *(resp.,* $\nu < 2 \cdot 10^{-4}$*)).*

Unit square				
ν	1e-1	1e-2	1e-3	7e-4
Classical				
red/step	0.26	0.22	0.22	0.28
$T_{0.1}$	3.9e-6	3.5e-6	3.5e-6	4e-6
Red-gray-black				
red/step	0.74	0.36	0.22 (ν=4e-3)	
$T_{0.1}$	3.3e-6	9.6e-7	7e-7 (ν=4e-3)	
MGridGen				
red/step	0.06	0.06	0.19	0.27
$T_{0.1}$	7.9e-7	8.1e-7	1.4e-6	1.8e-6
Bottom-up, (metis, 5-fold ref.)				
red/step	0.19	0.15	0.22	0.28
$T_{0.1}$	1.6e-6	1.5e-6	1.8e-6	2.2e-6

2D valve				
ν	1e-2	1e-3	1e-4	5e-5
Classical				
red/step	0.08	0.09	0.16	0.27
$T_{0.1}$	1.9e-6	1.9e-6	2.5e-6	3.5e-6
Red-gray-black				
red/step	0.08	0.13	0.21 (ν=2e-4)	
$T_{0.1}$	4e-7	4.7e-7	6.6e-7 (ν=2e-4)	
MGridGen				
red/step	0.06	0.05	0.26	0.39
$T_{0.1}$	7.8e-7	7.5e-7	1.5e-6	2.2e-6
Bottom-up				
red/step	0.02	0.02	0.05	0.09
$T_{0.1}$	8.3e-7	7.9e-7	1.1e-6	1.4e-6

approach of MGridGen leads to the best results, although the costs for the topology construction are pretty high (but for 3D problems are low in comparison to the prolongation construction). The red-gray-black method seems rather unpredictable; sometimes it is quite good, sometimes (especially for stronger convection) it is insufficient, and sometimes it is the systems to be solved for the prolongation grow uncontrollably, leading to large setup times. The bottom-up and the classical methods lie in between; the agglomerates which are produced by the classical methods sometimes look disadvantageous (bad aspect ratio) and result in a slow convergence.

4.2. Oseen equations. To check the h-dependence of the cAMGe method for the P_1^{nc}-P_0 element we solve the Oseen equations for the driven cavity problem (unit square, $\bar{\mathbf{u}} = (1\ 0)^T$ at the top edge, $\bar{\mathbf{u}} = \mathbf{0}$ at the other edges), with $\mathbf{w}$ near the solution of the full Navier–Stokes problem and mild convection ($\nu = 0.1$). The results of the tests (with Vanka smoother and W-2-2 cycle) can be found in Table 5 and Figure 11.

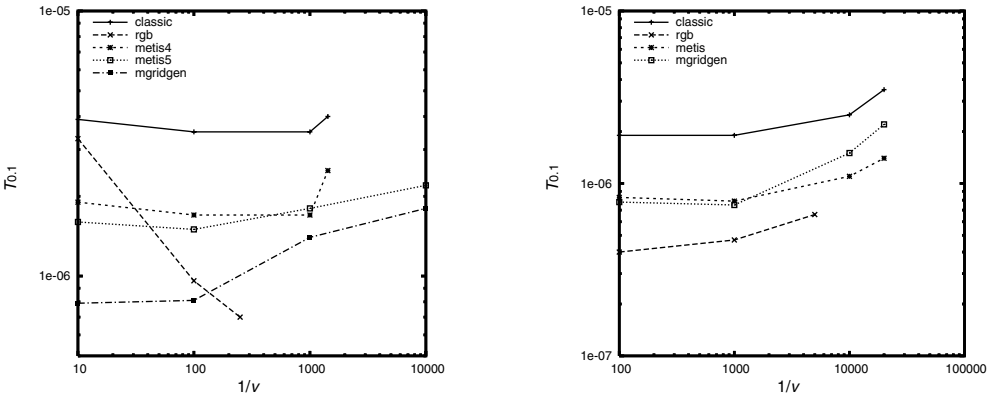


FIG. 10. *Dependence of the efficiency on decreasing ν for the unit-square (left) and the 2D valve problem (right).*

TABLE 3

Some h- and ν-dependences for the 3D valves. "tl" stands for "too long"; i.e., the results were not computable within reasonable time. A too long computing time was also the reason for not performing ν-tests for the classical and the red-gray-black topologies.

		h-dependence		ν-dependence			
	dofs	32 463	229215	1e-2	1e-3	5e-4	ν
Classical	$T_{0.1}$	3.5e-5	tl	—	—	—	
Red-gray-black	$T_{0.1}$	1.1e-6	tl	—	—	—	
MGridGen	$T_{0.1}$	6.2e-7	1.6e-6	4.4e-7	6.2e-7	7.6e-7	$T_{0.1}$
Bottom up	$T_{0.1}$	8.7e-7	1.7e-6	6.5e-7	8.7e-7	1e-6	$T_{0.1}$

TABLE 4

Grid and operator complexities for some test cases.

		Classical	Rgb	MGridGen	Bottom-up
Unit square	σ^{Ω}	2.46	1.33	1.93	2.45
(295553 dofs)	σ^{A}	2.93	1.33	2.23	2.75
2D valve	σ^{Ω}	2.46	1.34	1.92	1.70
(182225 dofs)	σ^{A}	2.93	1.34	2.05	2.23
3D valves	σ^{Ω}	5.37	2.27	1.84	2.87
(32463 dofs)	σ^{A}	14.77	4.16	2.79	4.18

TABLE 5

Results for the solution of the Oseen equations for the driven cavity problem ($\nu = 0.1$).

	Ref. level	5	6	7	8
	dofs	37,024	147,776	590,464	2,360,576
Classical	red/step	0.48	0.52	0.54	0.57
	$T_{0.1}$	2.7e-6	3.6e-6	4.3e-6	5e-6
Red-gray-black	red/step	0.64	0.67	0.69	0.70
	$T_{0.1}$	2.3e-6	1.9e-6	2.3e-6	2.4e-6
MGridGen	red/step	0.23	0.26	0.32	0.30
	$T_{0.1}$	7e-7	7.5e-7	1.1e-6	1.1e-6
Bottom-up (metis)	red/step	0.52	0.53	0.58	0.60
	$T_{0.1}$	1.6e-6	2.3e-6	2.7e-6	2.9e-6

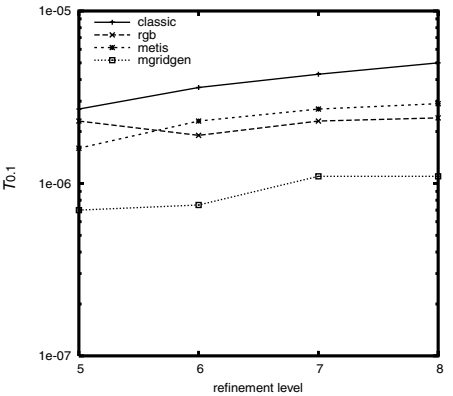


FIG. 11. *Results for the solution of the Oseen equations for the driven cavity problem ($\nu = 0.1$).*

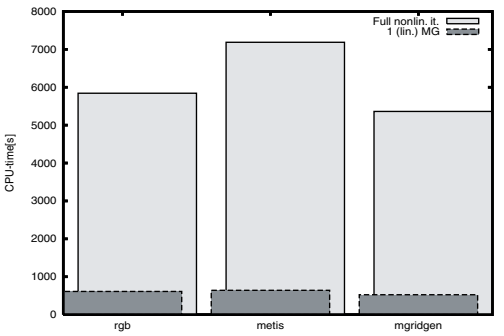


FIG. 12. *The timings for the full nonlinear problem on the 3D valves (light gray). The dark gray boxes in the foreground illustrate the time needed for the solution of one (the last) linear problem.*

What we finally want to compare is the usability of the methods for a full nonlinear Navier–Stokes problem. Here the construction of the topology has to be performed only once (except some kind of semicoarsening depending on intermediate solutions is wanted), the interpolations can also be kept for more than one nonlinear iteration. The solver itself should be fast (but it has to be kept in mind that there is no need to solve the linear problems with very high precision). We have applied the methods for the Navier–Stokes problem on the 3D valves with $\nu = 0.001$ and illustrated the timings in Figure 12. The method with the classical topology construction did not reach a solution with a reasonable number of smoothing steps and within reasonable time.

5. Conclusions. We have shown numerically that the strategy for the construction of the agglomerates has a crucial influence on the efficiency of the method. The optimization approach, which is the most sophisticated upon the methods tested (and also needs the most CPU time), in general leads to the best results.

As already mentioned in [8, 7] the AMGe method should only be used if classical AMG methods fail. The AMGe setup phase is costly, and the memory consumption can get very high, as many connectivity matrices containing the topological information are stored.

One situation where it is sensible to use AMGe is the coupled approach for a mixed FE discretization of the Oseen equations. First, it appears to be a flexible possibility for generalizations (we have demonstrated this for the P_1^{nc}-P_0 element), and second—because of the underlying topological information—it gives rise to analytic techniques which are not available for other methods (which we have exploited for a stability result). In this situation we reach the same conclusion as in [20], that it is advantageous to use knowledge from GMG methods and to give up the "black-box" character of AMG.

REFERENCES

[1] D. Braess, *Finite Elements*, 2nd ed., Cambridge University Press, Cambridge, UK, 2001.

[2] D. Braess and R. Sarazin, *An efficient smoother for the Stokes problem*, Appl. Numer. Math., 23 (1997), pp. 3–20.

[3] A. Brandt, S. McCormick, and J. Ruge, *Algebraic multigrid (AMG) for sparse matrix equations*, in Sparsity and Its Applications, D. Evans, ed., Cambridge University Press, Cambridge, UK, 1984, pp. 257–284.

[4] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. McCormick, and J. W. Ruge, *Algebraic multigrid based on element interpolation (AMGe)*, SIAM J. Sci. Comput., 22 (2000), pp. 1570–1592.

[5] F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods*, Springer Ser. Comput. Math., Springer, New York, 1991.

[6] M. Fortin, *An analysis of the convergence of mixed finite element methods*, RAIRO Anal. Numér., 11 (1977), pp. 341–354.

[7] V. Henson and P. S. Vassilevski, *Element-free AMGe: General algorithms for computing interpolation weights in AMG*, SIAM J. Sci. Comput., 23 (2001), pp. 629–650.

[8] J. E. Jones and P. S. Vassilevski, *AMGe based on element agglomeration*, SIAM J. Sci. Comput., 23 (2001), pp. 109–133.

[9] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[10] G. Karypis and V. Kumar, *METIS—A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*, University of Minnesota, 1998; also available online from www-users.cs.umn.edu/∼karypis/metis/.

[11] I. Moulitsas and G. Karypis, *MGridGen/ParMGridGen—Serial/Parallel Library for Generating Coarse Grids for Multigrid Methods*, University of Minnesota, 2001; also available online at www-users.cs.umn.edu/∼moulitsa/software.html.

[12] I. Moulitsas and G. Karypis, *Multilevel algorithms for generating coarse grids for multigrid methods*, in Conference Proceedings of Supercomputing 2001, 2001.

[13] O. Pironneau, *Finite Element Methods for Fluids*, John Wiley & Sons, Chichester, UK, 1989.

[14] J. W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, Frontiers Appl. Math. 5, S. McCormick, ed., SIAM, 1987, pp. 73–130.

[15] J. Schöberl and W. Zulehner, *On Schwarz-type smoothers for saddle point problems*, Numer. Math., 95 (2003), pp. 377–399.

[16] K. Stüben, *A review of algebraic multigrid*, Comput. Appl. Math., 128 (2001), pp. 281–309.

[17] P. Vanek, J. Mandel, and M. Brezina, *Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems*, Computing, 56 (1996), pp. 179–196.

[18] S. Vanka, *Block-implicit multigrid calculation of two-dimensional recirculating flows*, Comput. Methods Appl. Mech. Engrg., 59 (1986), pp. 29–48.

[19] M. Wabro, *Algebraic Multigrid Methods for the Numerical Solution of the Incompressible Navier-Stokes Equations*, Ph.D. thesis, University of Linz, Linz, Austria, 2003.

[20] M. Wabro, *Coupled algebraic multigrid methods for the Oseen problem*, Comput. Vis. Sci., 7 (2004), pp. 141–151.

[21] G. Wittum, *Multi-grid methods for Stokes and Navier-Stokes equations*, Numer. Math., 54 (1989), pp. 543–563.

[22] G. Wittum, *On the convergence of multi-grid methods with transforming smoothers*, Numer. Math., 57 (1990), pp. 15–38.

[23] W. Zulehner, *A class of smoothers for saddle point problems*, Computing, 65 (2000), pp. 227–246.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2098–2111
© 2006 Society for Industrial and Applied Mathematics

A STABILITY NOTION FOR LATTICE BOLTZMANN EQUATIONS*

M. K. BANDA†, W.-A. YONG‡, AND A. KLAR§

Abstract. Lattice Boltzmann equations are usually constructed to satisfy some physical requirements such as Galilean invariance and isotropy, to possess a velocity-independent pressure and no compressible effects, and so on. In this paper, a stability requirement is introduced as a new criterion for the constructions. With this requirement, we derive some relations of parameters for several lattice Boltzmann models. Interestingly, these relations are satisfied by many choices of parameters used in the literature.

Key words. lattice Boltzmann equations, incompressible Navier–Stokes equations, stability

AMS subject classifications. 76M28, 35L40, 82C35

DOI. 10.1137/040606211

1. Introduction.

Lattice Boltzmann equations provide a simple and alternative numerical method to simulate various hydrodynamic systems involving incompressible Navier–Stokes equations. A review on the method can be found in [6]. Although the lattice Boltzmann method evolved from lattice-gas automata [10, 11, 28], it can be viewed as a particular discretization of some discrete-velocity Boltzmann equations [27]. These are hyperbolic systems of partial differential equations with stiff source terms. Such equations, also called relaxation systems, have been intensively studied in the last years. To ensure a well-behaved relaxation limit, different stability criteria such as the structural stability conditions [29], the subcharacteristic condition [17], or some dissipative entropy principles [2] have been proposed for these systems.

In contrast, the lattice or discrete-velocity Boltzmann equations are constructed to satisfy some physical requirements such as Galilean invariance and isotropy, to possess a velocity-independent pressure and no compressible effects; see [21, 5, 23, 4, 32, 14, 13]. Recall that discrete-velocity lattice Boltzmann equations can also be interpreted as a discretized version of the continuum Boltzmann equation. Because the latter satisfies a dissipative entropy condition (Boltzmann's H-theorem) [3], one might expect that a stability criterion for the discrete-velocity Boltzmann equations is an analogue of the entropy condition. Unfortunately, it was proven in [30, 31] that such entropies do not exist for many widely used lattice Boltzmann models. However, the structural stability conditions in [29] can still be used here.

In this work, we adapt the structural stability conditions in [29] to the lattice Boltzmann equations. In doing so, we introduce a definition of stable lattice Boltzmann equations. For stable models thus defined, one can prove that the diffusive

*Received by the editors April 2, 2004; accepted for publication (in revised form) July 27, 2005; published electronically February 3, 2006.

http://www.siam.org/journals/sisc/27-6/60621.html

†School of Mathematical Sciences, University of KwaZulu-Natal (Pietermaritzburg), Private Bag X01, Scottsville 3209, South Africa (bandamk@ukzn.ac.za). This author's work was supported by Deutsche Forschungsgemeinschaft grant KL 1105/9.

‡Corresponding author. Zhou Pei-Yuan Center for Applied Mathematics, Tsinghua University, Beijing 100084, China (wayong@mail.tsinghua.edu.cn).

§Department of Mathematics, TU Kaiserslautern, Postfach 3049, 67653 Kaiserslautern, Germany (klar@itwm.fraunhofer.de). This author's work was supported by Deutsche Forschungsgemeinschaft grant KL 1105/9.

limit of the discrete-velocity Boltzmann equations is the incompressible Navier–Stokes equation at least in the regime of smooth flows. A complete proof can be done in exactly the same fashion as in [18] for d'Humières's generalized models [16]. Thus, it seems reasonable to consider the stability notion as a new requirement in constructing lattice Boltzmann equations for the Navier–Stokes equations. Note that the present theory is valid only for isothermal models.

To explain how the new stability requirement guides the construction of lattice Boltzmann equations, we derive some relations of parameters for the parametrized models from [27, 13] so that the models are stable when the relations are fulfilled. Interestingly, all the choices of parameters used in [23, 26, 1, 13, 15, 21] satisfy these relations. Moreover, computational experiments have been undertaken on some examples commonly used in literature. These tests confirm the usefulness of the stability notion.

Note that a general procedure with details is presented in sections 2 and 3 to derive the aforementioned relations for two specific parametrized models. It is hoped that the procedure can be adopted or revised by practitioners using the lattice Boltzmann method to choose various adjustable parameters.

We point out that our stability theory is different from the previous works [27, 22] on the stability of the lattice Boltzmann method (an explicit difference scheme). Those previous works are all based on the von Neumann stability analysis for the difference scheme and the resulting growth matrix was not treated analytically. In both [27] and [22], approximation methods for linear algebra were used to compute the eigenvalues of the growth matrix, while a perturbation technique was also employed in [22]. In contrast, our theory is based on a rigorous asymptotic analysis [18] for the lattice Boltzmann equations (partial differential equations). Thus it is useful not only for the lattice Boltzmann method but also for other discretizations of the hyperbolic systems. Moreover, our derivation of the parameter relations is purely analytic (see section 3).

The paper is organized as follows. In section 2 we present the definition and state a convergence theorem for the simplest lattice Boltzmann model. In section 3 we use the stability requirement to derive some relations of parameters for the parametrized models from [13, 27]. Finally, numerical results are presented in section 4 to verify our theory.

2. A stability notion. Lattice Boltzmann equations are of the form

$$\frac{\partial f_i}{\partial t} + c_i \cdot \nabla f_i = J_i(f), \qquad i = 1, 2, \ldots, N. \tag{2.1}$$

Here $f_i = f_i(x,t)$ are unknown (scalar) distribution functions, $(x,t) \in \mathbb{R}^d \times \mathbb{R}_+$, $c_i \cdot \nabla$ is the directional x-derivative along the mesoscopic velocity $c_i \in \mathbb{R}^d$, $J_i(f)$ stands for a collision term, and $f = (f_1, f_2, \ldots, f_N)^T$ with superscript T for the transpose.

By integrating (2.1) along the characteristics, we obtain

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \int_0^{\Delta t} J_i(f)(x + c_i\sigma, t + \sigma)d\sigma$$

with Δt a time step. Approximating the integral by the simple rectangle rule ($\sigma = 0$) yields

$$f_i(x + c_i\Delta t, t + \Delta t) = f_i(x,t) + \Delta t J_i(f)(x,t). \tag{2.2}$$

This is the usual lattice Boltzmann method and is widely used in a *long-time large-scale* limit to simulate incompressible flows. The latter are traditionally described with the Navier–Stokes equations

$$\begin{aligned} \text{div}\mathbf{u} &= 0, \\ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &= \nu\Delta\mathbf{u} \end{aligned} \tag{2.3}$$

with initial and boundary conditions. Here $\mathbf{u}$, p, and ν stand for the fluid velocity, pressure, and viscosity coefficient (a given positive constant), respectively.

Quantitatively, the long-time large-scale limit is characterized by the diffusive scaling

$$t \longrightarrow t/\epsilon^2, \qquad x \longrightarrow x/\epsilon$$

with ϵ a small positive parameter. In this scaling, (2.1) becomes

$$\frac{\partial f_i}{\partial t} + \frac{c_i}{\epsilon}\cdot\nabla f_i = \frac{J_i(f)}{\epsilon^2}, \qquad i = 1, 2, \ldots, N. \tag{2.4}$$

For the lattice Boltzmann models used in the simulation of incompressible flows, the c_i's and $J_i(f)$'s are always chosen such that the Chapman–Enskog expansion of the scaled system (2.4) yields the Navier–Stokes equations. Such lattice Boltzmann models will be called *consistent* (with the Navier–Stokes equations).

Now we introduce our *stability* notion for the lattice Boltzmann equations (2.1).

DEFINITION 2.1. *Let f_* be a constant state satisfying $J(f_*) = 0$. The lattice Boltzmann model* (2.1) *is called* stable *at $f = f_*$ if there is an invertible matrix $P \in \mathbb{R}^{N\times N}$ such that P^TP is diagonal and $PJ_f(f_*) = -diag(\lambda_1, \lambda_2, \ldots, \lambda_N)P$ with $\lambda_i = 0$ for $i \le d+1$ and $\lambda_i > 0$ for $i > d+1$. Here $J_f(f) \in \mathbb{R}^{N\times N}$ is the Jacobian of $J(f) = (J_1(f), J_2(f), \ldots, J_N(f))^T$.*

REMARK 2.1. *This definition is based on the stability conditions* [29] *for hyperbolic systems with source terms. Note that the usual entropy structure is not valid for many widely used lattice Boltzmann equations* [30, 31].

REMARK 2.2. *In* [18]*, the generalized lattice Boltzmann models from* [16] *were shown to be stable provided that some parameters satisfy a constraint. With such a stability property, it was further proven there that the generalized models converge in the diffusive limit $\epsilon \to 0$ towards the Navier–Stokes equations. In general, for consistent and stable lattice Boltzmann models, similar convergence results can be proven with the approach in* [18].

Next we check the above stability definition for a simple example and state a convergence theorem for the following lattice Boltzmann model:

$$\begin{aligned} &d = 2, \qquad N = 6, \\ &c_i = \big(\cos\big(\pi(i-1)/3\big), \sin\big(\pi(i-1)/3\big)\big)^T, \\ &J_i(f) = \frac{f_i^{eq}(n,\mathbf{u}) - f_i}{\tau}. \end{aligned} \tag{2.5}$$

In (2.5), τ is a positive parameter (relaxation time) and

$$\begin{aligned} &f_i^{eq}(n,\mathbf{u}) = n/6 + \mathbf{u}c_i/3 + 2(\mathbf{u}c_i)^2/3 - |\mathbf{u}|^2/3, \\ &n = \sum_{i=1}^{6} f_i(x,t), \qquad \mathbf{u} = \sum_{i=1}^{6} c_i f_i(x,t). \end{aligned} \tag{2.6}$$

The consistency of this model is well known (see, e.g., [20]). To show its stability, we note that

$$\begin{aligned}\sum_{i=1}^{6} f_i^{eq}(n,\mathbf{u}) &= n = \sum_{i=1}^{6} f_i(x,t),\\ \sum_{i=1}^{6} f_i^{eq}(n,\mathbf{u})c_i &= \mathbf{u} = \sum_{i=1}^{6} f_i(x,t)c_i.\end{aligned} \tag{2.7}$$

By computing

$$\frac{\partial f_i^{eq}(n,\mathbf{u})}{\partial f_j} = \frac{1}{6} + \frac{c_i c_j}{3} + \frac{4(\mathbf{u}c_i)c_j c_i}{3} - \frac{2\mathbf{u}c_j}{3} \tag{2.8}$$

and using (2.7), we deduce from (2.8) that

$$\left[f_{if_j}^{eq}(n,\mathbf{u})\right]^2 = \left[f_{if_j}^{eq}(n,\mathbf{u})\right],$$

that is, the Jacobian $[f_{if_j}^{eq}(n,\mathbf{u})]$ is a projection matrix. Thus, the eigenvalues of $J_f(f) := ([f_{if_j}^{eq}(n,\mathbf{u})] - I_6)/\tau$ are 0 and $-1/\tau$.

Take $f_* = f^{eq}(1,0)$. Set $e = (1,1,\ldots,1) \in \mathbb{R}^6$ and $\mathbf{c} = (c_1,\ldots,c_6)$. From (2.8) we see that

$$\left[f_{if_j}^{eq}(n,\mathbf{u})\right]\big|_{\mathbf{u}=0} = \mathbf{c}^T\mathbf{c}/3 + e^T e/6 \tag{2.9}$$

is symmetric. Note that

$$e\mathbf{c}^T = 0 \quad \text{and} \quad \mathbf{c}\mathbf{c}^T = 3I_2.$$

The relation (2.9) indicates that the rank of $[f_{if_j}^{eq}(n,\mathbf{u})]|_{\mathbf{u}=0}$ is 3. Since $[f_{if_j}^{eq}(n,\mathbf{u})]$ is a projection matrix and

$$\tau J_f(f_*) = [f_{if_j}^{eq}(n,\mathbf{u})]|_{\mathbf{u}=0} - I_6,$$

the rank of $J_f(f_*)$ is 3.

On the other hand, since $\left[f_{if_j}^{eq}(n,\mathbf{u})\right]\big|_{\mathbf{u}=0}$ is symmetric, there is a unitary matrix P such that

$$J_f(f_*) = P^T \Lambda P/\tau$$

with Λ a diagonal matrix. Since Λ is similar to $\tau J_f(f_*)$, we may as well assume that

$$\Lambda = -\mathrm{diag}(0,0,0,1,1,1).$$

Thus, we have shown the following.

PROPOSITION 2.1. *The 2-dimensional 6-velocity model* (2.5)–(2.6) *is stable at* $f_* = f^{eq}(1,0)$.

Based on the stability and consistency results, one can prove a convergence theorem for the model (2.5)–(2.6). For this purpose, we need the following existence result [19] for the Navier–Stokes equations on the d-dimensional torus $\mathbb{T}^d$.

LEMMA 2.2 (see [19]). *Let* $s \geq [d/2]+1$ *be an integer and* $\bar{\mathbf{u}} \in H^s$ *with* $\mathrm{div}\bar{\mathbf{u}} = 0$. *Then there exists* $T_* \in (0,\infty)$, *depending only on* $\|\bar{\mathbf{u}}\|_s$, *such that the Navier–Stokes*

equations (2.3) *with* $\mathbf{u}(x,0)=\bar{\mathbf{u}}(x)$ *and* $\int p(x,t)dx=0$ *have a unique solution* $(\mathbf{u},p)$ *satisfying*

$$\mathbf{u}\in C(0,T_*;H^s)\cap L^1(0,T_*;H^{s+1}),\qquad p\in L^1(0,T_*;H^{s+1}).$$

Here and below, H^s is the Sobolev space of functions which, together with their distributional x-derivatives of order $\le s$, are square integrable on $\mathbb{T}^d$. Its norm is denoted by $\|\cdot\|_s$. The integral $\int$ is taken over a period, say, $(0,1]^d$. In addition, $C(0,T;H^s)$ and $L^1(0,T;H^s)$ denote the Banach space of H^s-valued continuous and integrable functions on $[0,T]$, respectively.

Now we are in a position to state the following convergence result, which is an analogue of Theorem 4.1 in [18] and can be proven similarly.

THEOREM 2.3. *Let* $s\ge 2$ *be an integer,* $\bar{\mathbf{u}}\in H^{s+5}$ *with* $\operatorname{div}\bar{\mathbf{u}}=0$*, and* $\bar{p}$ *the solution of* $\Delta p=-\operatorname{div}(\bar{\mathbf{u}}\cdot\nabla\bar{\mathbf{u}})$ *with* $2\int\bar{p}=-\int|\bar{\mathbf{u}}|^2$*. Then the lattice Boltzmann equations* (2.4) *with* (2.5), (2.6)*,* $\tau=4\nu$*, and initial data*

$$f_i^\epsilon(x,t)=f_i^{eq}(1,\epsilon\bar{\mathbf{u}})+\epsilon^2(2\bar{p}+|\bar{\mathbf{u}}|^2-8\nu c_i\cdot\nabla\bar{\mathbf{u}}c_i)/6$$

have a unique solution $f^\epsilon\in C(0,T_*;H^s)$ *for* $\epsilon>0$ *sufficiently small. Moreover, there is a constant* K*, independent of* ϵ*, such that*

$$\Big\|\mathbf{u}(\cdot,t)-\frac{\mathbf{u}^\epsilon(\cdot,t)}{\epsilon}\Big\|_s\le K\epsilon^2,$$

$$\Big\|p(\cdot,t)-\frac{1}{2\epsilon^2}\Big[n^\epsilon(\cdot,t)-|\mathbf{u}^\epsilon|^2+\int|\mathbf{u}^\epsilon|^2-1\Big]\Big\|_s\le K\epsilon$$

for all $t\le T_*$*. Here* $\mathbf{u}^\epsilon=\sum_i c_i f_i^\epsilon$*,* $n^\epsilon=\sum_i f_i^\epsilon$*, and* $(\mathbf{u},p)$ *denotes the above solution of the Navier–Stokes equations.*

Similar theorems can be proven for other consistent and stable lattice Boltzmann models. On the other hand, the requirement that lattice Boltzmann models are stable may be used as a guideline to construct such models. Consistent models chosen in such a way will automatically converge to the Navier–Stokes equations in the diffusive limit.

3. Determination of parameters. In this section, we fix some parameters for several parametrized lattice Boltzmann models by requiring that they are stable. These models are taken from [13, 27] and their consistency is already shown there.

3.1. 2-dimensional 8-velocity models (D2Q8). We begin with an incompressible model from [13], which is (2.1) with $d=2$, $N=8$,

$$\{c_i: i=1,2,3,4\}=\{(\pm1,0)^T,(0,\pm1)^T\},$$
$$\{c_i: i=5,6,7,8\}=\{(\pm1,\pm1)^T\},$$

and

$$J_i(f)=\frac{f_i^{eq}(\mathbf{u},p)-f_i}{\tau}.$$

Here τ is a positive constant (relaxation time) and

$$f_i^{eq}(\mathbf{u},p)=\begin{cases}\big(3\mathbf{u}c_i+(3\mathbf{u}c_i)^2/2-3|\mathbf{u}|^2/2\big)/9+\lambda p, & i=1,2,3,4,\\ \big(3\mathbf{u}c_i+(3\mathbf{u}c_i)^2/2-3|\mathbf{u}|^2/2\big)/36+\gamma p, & i=5,6,7,8,\end{cases}$$

with two parameters λ and γ, where

$$\mathbf{u} = \sum_{i=1}^{8} c_i f_i(x,t), \qquad p = \frac{1}{4(\lambda+\gamma)}\left[\sum_{i=1}^{8} f_i(x,t) - \frac{2|\mathbf{u}|^2}{3}\right].$$

It can be directly verified that

$$\begin{aligned} \sum_{i=1}^{8} f_i^{eq}(\mathbf{u},p) &= \sum_{i=1}^{8} f_i(x,t), \\ \sum_{i=1}^{8} f_i^{eq}(\mathbf{u},p)c_i &= \mathbf{u} = \sum_{i=1}^{8} f_i(x,t)c_i. \end{aligned} \tag{3.1}$$

As for the 2-dimensional 6-velocity model, by computing

$$\frac{\partial f_i^{eq}(\mathbf{u},p)}{\partial f_j} = \begin{cases} \dfrac{\left(3c_jc_i + 9(\mathbf{u}c_i)c_jc_i - 3\mathbf{u}c_j\right)}{9} + \dfrac{1-4\mathbf{u}c_j/3}{4(\lambda+\gamma)}\lambda, & i = 1,2,3,4, \\ \dfrac{\left(3c_jc_i + 9(\mathbf{u}c_i)c_jc_i - 3\mathbf{u}c_j\right)}{36} + \dfrac{1-4\mathbf{u}c_j/3}{4(\lambda+\gamma)}\gamma, & i = 5,6,7,8, \end{cases} \tag{3.2}$$

and using (3.1), we deduce from (3.2) that

$$\left[f_{if_j}^{eq}(\mathbf{u},p)\right]^2 = \left[f_{if_j}^{eq}(\mathbf{u},p)\right];$$

that is, the Jacobian $[f_{if_j}^{eq}(\mathbf{u},p)]$ is a projection matrix. Thus, the eigenvalues of $J_f(f) := ([f_{if_j}^{eq}(\mathbf{u},p)] - I_8)/\tau$ are 0 and $-1/\tau$.

In addition, take $f_* = f^{eq}(0,1)$ and set $e = (1,1,\ldots,1) \in \mathbb{R}^8$, $\mathbf{c} = (c_1,\ldots,c_8)$, and

$$A_0 = 9\ \mathrm{diag}\big(I_4, 4I_4\big). \tag{3.3}$$

From (3.2) and (3.3) we see that

$$A_0\left[f_{if_j}^{eq}(\mathbf{u},p)\right]\big|_{\mathbf{u}=0} = 3\mathbf{c}^T\mathbf{c} + A_0(\lambda,\lambda,\lambda,\lambda,\gamma,\gamma,\gamma,\gamma)^T e/(4(\lambda+\gamma)) \tag{3.4}$$

is symmetric if

$$\lambda = 4\gamma. \tag{3.5}$$

Note that

$$e\mathbf{c}^T = 0 \qquad \text{and} \qquad \mathbf{c}\mathbf{c}^T = 6I_2.$$

The relation (3.4) indicates that the rank of $[f_{if_j}^{eq}(\mathbf{u},p)]|_{\mathbf{u}=0}$ is 3. Since $[f_{if_j}^{eq}(\mathbf{u},p)]$ is a projection matrix and $\tau J_f(f_*) = [f_{if_j}^{eq}(\mathbf{u},p)]|_{\mathbf{u}=0} - I_8$, the rank of $J_f(f_*)$ is 5.

On the other hand, since $A_0\left[f_{if_j}^{eq}(\mathbf{u},p)\right]\big|_{\mathbf{u}=0}$ is symmetric and A_0 is symmetric positive definite, it is well known that there is an invertible matrix P such that

$$A_0 = P^TP \qquad \text{and} \qquad A_0J_f(f_*) = P^T\Lambda P/\tau$$

with Λ a diagonal matrix. Note that Λ is similar to $\tau J_f(f_*)$. We may as well assume that

$$\Lambda = -\mathrm{diag}(0,0,0,1,1,1,1,1).$$

Thus we have proven the following.

PROPOSITION 3.1. *If $\lambda = 4\gamma$, then the model in* [13] *is stable at $f_* = f^{eq}(0,1)$.*

REMARK 3.1. *In* [13]*, the authors derived a consistency condition $\lambda + 2\gamma = 1/2$. This condition alone allows infinite many choices of the parameters. However, in the numerical tests reported in* [13]*, only $\lambda = 1/3$ was used. The authors neither explained how to get this λ nor mentioned numerical results with other choices of λ. Note that the tested choice is exactly that given by our stability condition* (3.5) *and the above consistency condition.*

REMARK 3.2. *A similar convergence theorem as in section* 2 *can be established for the D2Q8 model with the above consistency and stability constraints.*

REMARK 3.3. *Proposition* 3.1 *is proven with almost the same procedure as that for Proposition* 2.1*. Similar procedures can be applied to other models. We state several results in the following subsections but omit the detailed proofs, leaving them as exercises for the interested reader.*

3.2. 2-dimensional 9-velocity models (D2Q9). Next we consider a D2Q9 parametrized model [27], i.e., (2.1) with $c_9 = 0$, other c_i equal to that of the above incompressible D2Q8 model, and

$$J_i(f) = \frac{f_i^{eq}(n,\mathbf{u}) - f_i}{\tau}.$$

Here τ is a relaxation time, $f_9^{eq}(n,\mathbf{u}) = \alpha n - \dfrac{2}{3}n|\mathbf{u}|^2$,

$$\begin{aligned} f_i^{eq}(n,\mathbf{u}) =& \beta n + \frac{n}{3}\mathbf{u}c_i + \frac{n}{2}(\mathbf{u}c_i)^2 - \frac{n}{6}|\mathbf{u}|^2, \qquad 1 \le i \le 4, \\ f_i^{eq}(n,\mathbf{u}) =& \frac{(1-\alpha-4\beta)}{4}n + \frac{n}{12}\mathbf{u}c_i + \frac{n}{8}(\mathbf{u}c_i)^2 - \frac{n}{24}|\mathbf{u}|^2, \qquad 5 \le i \le 8, \end{aligned}$$

with

$$n = \sum_{i=1}^{9} f_i(x,t), \qquad n\mathbf{u} = \sum_{i=1}^{9} c_i f_i(x,t).$$

For this parametrized model, we have the following.

PROPOSITION 3.2. *If $\alpha \in (0,1)$ and $\alpha + 5\beta = 1$, then the D2Q9 model is stable at $f_* = f^{eq}(1,0)$.*

It is remarkable that the condition of Proposition 3.2 is satisfied by both $(\alpha,\beta) = (4/9, 1/9)$ used in the well-known D2Q9 model and $(\alpha,\beta) = (2/7, 1/7)$ used in [26].

3.3. 3-dimensional 15-velocity models (D3Q15). A D3Q15 parametrized model from [27] is (2.1) with $c_{15} = 0$,

$$\begin{aligned} \{c_i : i = 1,\dots,6\} &= \{(\pm1,0,0)^T, (0,\pm1,0)^T, (0,0,\pm1)^T\}, \\ \{c_i : i = 7,\dots,14\} &= \{(\pm1,\pm1,\pm1)^T\}, \end{aligned}$$

and

$$J_i(f) = \frac{f_i^{eq}(n,\mathbf{u}) - f_i}{\tau}.$$

Here τ is a relaxation time, $f_{15}^{eq}(n,\mathbf{u}) = \alpha n - \frac{n}{3}|\mathbf{u}|^2$,

$$f_i^{eq}(n,\mathbf{u}) = \beta n + \frac{n}{3}\mathbf{u}c_i + \frac{n}{2}(\mathbf{u}c_i)^2 - \frac{n}{6}|\mathbf{u}|^2, \qquad 1 \le i \le 6,$$
$$f_i^{eq}(n,\mathbf{u}) = \frac{(1-\alpha-6\beta)}{8}n + \frac{n}{24}\mathbf{u}c_i + \frac{n}{16}(\mathbf{u}c_i)^2 - \frac{n}{48}|\mathbf{u}|^2, \qquad 7 \le i \le 14,$$

with

$$n = \sum_{i=1}^{15} f_i(x,t), \qquad n\mathbf{u} = \sum_{i=1}^{15} c_i f_i(x,t).$$

For this parametrized model, we have the following.

PROPOSITION 3.3. *If $\alpha \in (0,1)$ and $\alpha + 7\beta = 1$, then the $D3Q15$ model is stable at $f_* = f^{eq}(1,0)$.*

It is remarkable that the condition of Proposition 3.3 is satisfied by both $(\alpha,\beta) = (1/8,1/8)$ used in [1] and $(\alpha,\beta) = (2/9,1/9)$ used in [23].

Following the arguments above, we expect that the stability requirement leads to a good choice of the parameters. Moreover, choosing parameters outside the range required by the stability condition should in general yield unstable methods. These statements will be further verified with numerical tests presented in the next section.

4. Numerical results. In this section we report a number of numerical experiments to verify that the stability criteria presented in Propositions 3.1, 3.2, and 3.3 for the different models give reasonable ranges for the corresponding parameters. The goal is to show that the choices given in the propositions always work well. On the other hand, choices outside the ranges suggested in the propositions lead in general to unreliable or unstable results. The numerical experiments in this section use the standard lattice Boltzmann discretization (2.2) on a few examples commonly used in the literature.

4.1. Test examples.

4.1.1. Taylor vortex flow. The first example models the decay of a Taylor vortex in a 2π periodic domain. We refer to [24] and [8] for details. In two dimensions the initial conditions are

$$u(x,y,0) = \cos x \sin y,$$
$$v(x,y,0) = \sin x \cos y,$$

where u (resp., v) is the flow velocity in the x- (resp., y-) direction. The tests are performed at $\mathrm{Re} = 100$. In the initial step, the pressure field is generated from the velocity field by taking the divergence of the momentum equation in the Navier–Stokes system and then solving for pressure. Since the velocity field $\mathbf{u} = (u,v)$ is divergence free, the pressure p solves

$$\nabla^2 p = -\nabla \cdot (\mathbf{u} \cdot \nabla \mathbf{u}).$$

In three dimensions we choose the initial velocity distribution as follows:

$$u(x,0) = U_0 \cos x \sin y \cos z,$$
$$v(x,0) = -U_0 \sin x \cos y \cos z,$$
$$w(x,0) = 0,$$

where $U_0 = 0.2$. We consider $\mathrm{Re} = 200$.

4.1.2. Poiseuille flow. This example is taken from [13] or [7]. The flow is restricted in the region $\{(x,y)|0 \le x \le 2,\ 0 \le y \le 1\}$ with initial and boundary conditions

$$\begin{aligned} u(x,y,0) = v(x,y,0) = 0, \quad & p(x,y,0) = p_0, \\ u(x,0,t) = u(x,1,t) = v(x,0,t) = v(x,1,t) = 0, & \\ p(0,y,t) = p_{\text{in}}, p(2,y,t) = p_{\text{out}}, & \end{aligned}$$

where $p_0 = 0.5(p_{\text{in}} + p_{\text{out}})$ with $p_{\text{in}} = 1.1$ and $p_{\text{out}} = 1.0$. To determine the steady state we apply the following criterion, as in [13]:

$$\frac{\sum_{ij} |u_{ij}^{(n+1)} - u_{ij}^{(n)}|}{\sum_{ij} |u_{ij}^{(n+1)}|} \le 5.0 \times 10^{-9},$$

where u_{ij}^n is the discrete flow velocity in the x-direction.

4.1.3. Driven cavity flow. The last example is a driven cavity flow. In two dimensions we consider a square cavity whose top plate moves from left to right with a constant velocity, while the other three boundaries are fixed. Numerical simulations of the driven cavity have been studied by many authors using traditional schemes such as finite difference [9, 25] and multigrid methods [12]. The lattice Boltzmann simulation for the D2Q9 model was performed in [15] with a detailed analysis and for the D2Q8 model in [13]. The Reynolds number is determined by $\text{Re} = LU/\nu$, where L is the height or width of the cavity, U is the uniform velocity of the top plate, and ν is the kinematic viscosity. We set $U = 0.1$ and ν depends on the Reynolds number. Numerical simulations are performed with $\text{Re} = 400, 1000, 2000$ on a 257×257 lattice. The initial condition is $u = v = 0$ and $p = 1$. A steady state is reached when the difference between the velocities at the center of the cavity for successive 1000 steps is less than 5×10^{-6}, as in [13].

In three dimensions the fluid is bounded by a cubic enclosure and the flow is driven by a uniform translation of the top in the x-direction. The top boundary moves in the x-direction with velocity U. The cavity has a $65 \times 65 \times 65$ lattice. Initially the velocities at all nodes, except the top ones, are set to zero. The x-velocity of the top is $U = 0.01$ for $\text{Re} = 100$ and $U = 0.1$ for higher Re. The y- and z-velocities are zero. Uniform fluid density $\rho = 2.7$ is imposed initially. The Reynolds number used in the cavity is $\text{Re} = UL_N/\nu$, where U is the uniform velocity at the top plate, L_N is the number of lattice units along one side of the cavity, and ν is the kinematic viscosity. The scheme and the boundary condition implemented in this example are the same as in [15].

4.2. Results for the different models.

4.2.1. The D2Q8 model. For details of the computation of the Taylor vortex flow we refer to [24]. We perform tests on different ratios of λ and γ: $\lambda = 4\gamma$, 2γ, and 6γ, which also satisfy the consistency condition $\lambda + 2\gamma = 1/2$. All of them give stable results for this test case. In Figure 1, the results obtained with the different parameters are compared to the exact solution. In addition, an extra test is made with $\lambda = 4\gamma$ and $\gamma = 1/3$. In this case the consistency condition does not hold. Figure 1 (right-hand side) shows that the model leads to a stable but unacceptable result.

For the Poiseuille flow example we consider the same parameters. The results obtained here lead to the same conclusions as those for the Taylor vortex flow. We

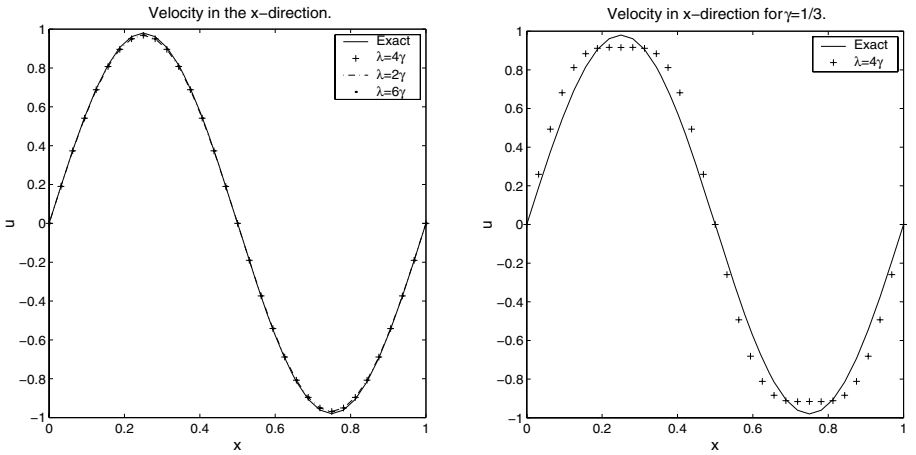


FIG. 1. *Taylor vortex flow with different parameter ratios of* λ *and* γ. $t = 1$, $y = \pi$.

also test the inconsistent choice of parameters $\lambda = 4\gamma$ and $\gamma = 1/3$ for the present flow model.

The last example for the D2Q8 model is the driven cavity flow. To verify the stability criterion we choose λ as in the two examples above. In all cases $\lambda = 4\gamma$ and $\lambda = 6\gamma$ yield stable results, while $\lambda = 2\gamma$ is unstable. Figure 2 presents the stable results for $\mathrm{Re} = 1000$.

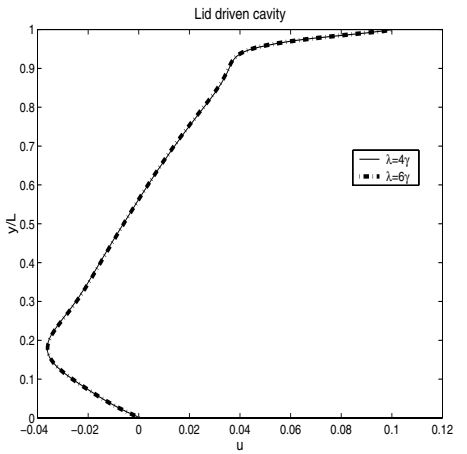


FIG. 2. *Lid driven cavity flow (velocity in the* x*-direction) with different parameter ratios of* λ *and* γ*, and* $Re = 1000$. *A cross-section taken in the middle of the cavity:* $x/L = 0.5$.

In conclusion we obtain the following results for the D2Q8 model.

1. The ratio $\lambda = 4\gamma$ gives stable results for all examples, even when the consistency condition is violated. This is backed by the theory.
2. The ratio $\lambda = 2\gamma$ gives stable results for the first two flow models, but unstable ones for the driven cavity problem, in accordance with the theory.
3. The ratio $\lambda = 6\gamma$ gives stable results for all examples. This does not necessarily contradict the stability theory. The ratio $\lambda = 6\gamma$ seems to be a good

choice for the examples considered here—although not backed by the stability theory—and thus remains to be tested on more complicated flows.

4.2.2. The D2Q9 model. We test the same configurations as in the last section. Tests are made with $\alpha = 0, 2/7, 4/9, 0.5, 1.0, 1.5, 2.0, 10.0$ and β satisfying the stability requirement given in Proposition 3.2.

For Taylor vortex flow we perform tests with the above values of α and see that $\alpha = 1.0$ still works but $\alpha > 1$ gives unacceptable results, as illustrated in Figure 3.

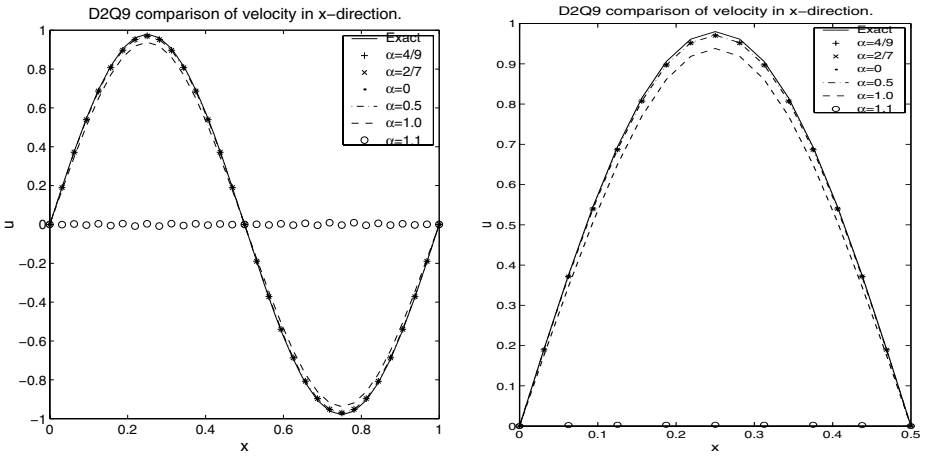


FIG. 3. *Taylor vortex flow with different values of α and β with $\alpha + 5\beta = 1$. $t = 1$, $y = \pi$.*

Further tests are made to investigate the behavior of the D2Q9 model for the Taylor vortex flow problem when $\alpha \in (0, 1)$ and $\alpha + 5\beta \neq 1$. We observe, for example, that the model leads to unacceptable results when $\alpha = 0.5$, $\beta = 0.5$, i.e., $\alpha + 5\beta = 3$.

For the Poiseuille flow problem, one obtains stable results for $\alpha = 0, 1/2, 4/9, 2/7$. For all other cases where $\alpha \in (0, 1)$ and $\alpha + 5\beta \neq 1$, the model gives unacceptable results. We observe, for example, that the model leads to unacceptable results when $\alpha = 0.5$, $\beta = 0.5$, i.e., $\alpha + 5\beta = 3$.

Finally, we consider the driven cavity flow. The results for $\mathrm{Re} = 1000$ are displayed in Figure 4. We obtain stable results for parameters $\alpha = 2/7, 4/9, 0.5$ and $\alpha + 5\beta = 1$. Parameters $\alpha = 0, 1.0, 1.5, 2.0, 10.0$ which are outside the range given by the theory give unstable results.

For this flow problem we also test $\alpha \in (0, 1)$ with $\alpha + 5\beta \neq 1$ for $\mathrm{Re} = 100$ and $U = 0.01$. The results are unacceptable.

In conclusion, we have investigated cases with the parameters $\alpha \in (0, 1)$ with $\alpha + 5\beta = 1$ which satisfies the stability requirement. Moreover, cases with $\alpha \notin (0, 1)$ with $\alpha + 5\beta = 1$ and $\alpha \in (0, 1)$ with $\alpha + 5\beta \neq 1$ are considered.

1. For $\alpha \in (0, 1)$ with $\alpha + 5\beta = 1$ all results are reasonable, which supports the theory.
2. For $\alpha \notin (0, 1)$ and $\alpha + 5\beta = 1$ all cases tested for the driven cavity problem give unstable or unacceptable results. The same is the case for $\alpha \in (0, 1)$ with $\alpha + 5\beta \neq 1$.

4.2.3. The D3Q15 model. For the Taylor–Green vortex flow we use parameter values of $\alpha = 0.0$, 1/8, 2/9, 0.5, 0.8, and 1.0. All of them give stable results, although

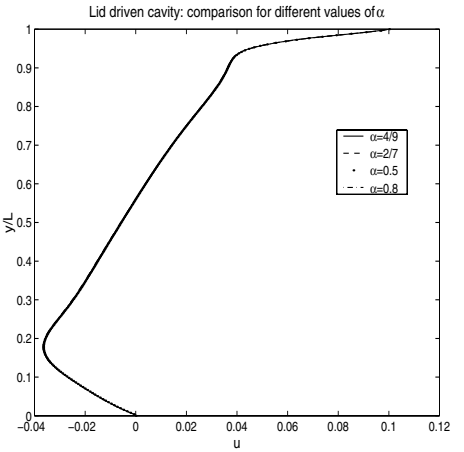


FIG. 4. *Comparison of the velocity in the x-direction for different parameters* α, $\alpha + 5\beta = 1$ *for the lid driven cavity flow problem,* $Re = 1000$. *A cross-section taken in the middle of the cavity:* $x/L = 0.5$.

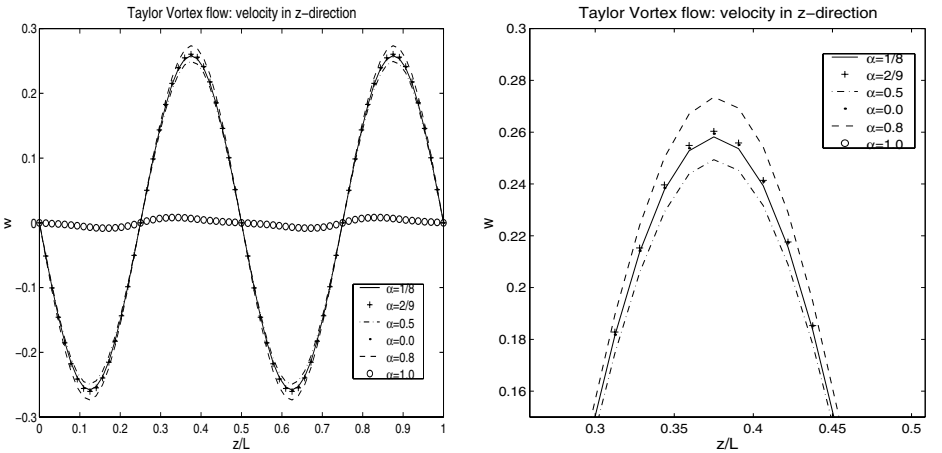


FIG. 5. *Comparison for different parameters* α, $\alpha + 7\beta = 1$ *for the Taylor–Green vortex flow problem. Profiles are taken at middle cross-sectional planes of the cube for velocity in the z-direction.*

$\alpha = 1.0$ gives unacceptable ones. The results are displayed in Figure 5, where the velocity in the z-direction is shown.

For the three-dimensional driven cavity flow we test the cases Re = 100, 400. Stable results for both cases are found only for $\alpha = 1/8$, $2/9$, 0.5, 0.8 as predicted by the theory. The results are displayed in Figure 6 for Re = 100, 400, respectively. In particular, for the case Re = 400 the cases $\alpha = 0$ and $\alpha = 1$ are unstable.

For Re = 400 and $U = 0.1$ we also test cases with $\alpha \in (0, 1)$ and $\alpha + 7\beta \neq 1$. We used $\alpha = 2/9$, $1/8$ with $\alpha + 7\beta = 0$ and $\alpha + 7\beta = 2$. In all these cases no reasonable results are obtained.

In conclusion, the D3Q15 model has been tested on two examples to verify the stability theory. We considered $\alpha \in (0, 1)$ with $\alpha + 7\beta = 1$, $\alpha \notin (0, 1)$ with $\alpha + 7\beta = 1$, and $\alpha \in (0, 1)$ with $\alpha + 7\beta \neq 1$.

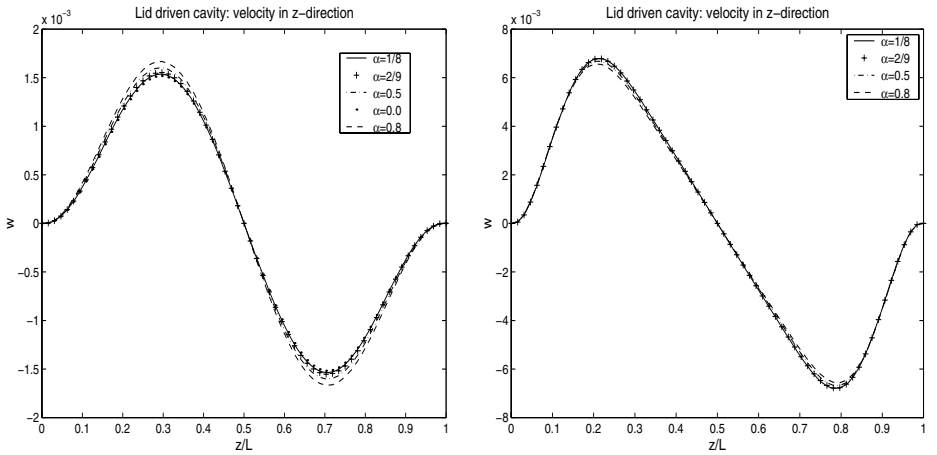


FIG. 6. *Comparison for different parameters* α, $\alpha+7\beta=1$ *for the lid driven cavity flow problem using the* $D3Q15$ *model,* $Re=100$ *and* $Re=400$. *Profiles are taken at middle cross-sectional planes of the cube for velocity in the z-direction.*

1. For all tests, $\alpha \in (0,1)$ with $\alpha+7\beta=1$ produces reasonable results.
2. $\alpha \notin (0,1)$ with $\alpha+7\beta=1$ is unstable or unacceptable for all tested parameter values for the driven cavity problem with $\mathrm{Re}=400$, and so is $\alpha \in (0,1)$ with $\alpha+7\beta \neq 1$.

These results are in perfect agreement with the theory developed in section 2.

5. Conclusions.

- In this paper we introduce a new stability notion for constructing lattice Boltzmann equations. With the stability requirement, new relations of parameters have been derived for several well-known parametrized models.
- Computational results are presented to verify the stability theory for a few examples commonly used in the literature. They show that the stability requirement gives some reliable choices of parameters.

REFERENCES

[1] F. J. ALEXANDER, S. CHEN, AND D. W. GRUNAU, *Hydrodynamic spinodal decomposition: Growth kinetics and scaling functions*, Phys. Rev. B, 48 (1993), pp. 634–637.

[2] F. BOUCHUT, *Construction of BGK models with a family of kinetic entropies for a given system of conservation laws*, J. Statist. Phys., 95 (1999), pp. 113–170.

[3] C. CERCIGNANI, R. ILLNER, AND M. PULVIRENTI, *The Mathematical Theory of Dilute Gases*, Appl. Math. Sci. 106, Springer-Verlag, New York, 1994.

[4] H. CHEN, S. CHEN, AND W. H. MATTHAEUS, *Recovery of the Navier-Stokes equations using a lattice-gas Boltzmann method*, Phys. Rev. A, 45 (1992), pp. 5339–5342.

[5] S. CHEN, H. CHEN, D. MARTINEZ, AND W. H. MATTHAEUS, *Lattice Boltzmann model for simulation of magnetohydrodynamics*, Phys. Rev. Lett., 67 (1991), pp. 3776–3779.

[6] S. CHEN AND G. D. DOOLEN, *Lattice Boltzmann method for fluid flows*, Ann. Rev. Fluid Mech., 30 (1998), pp. 329–364.

[7] S. CHEN, D. MARTINEZ, AND R. MEI, *On boundary conditions in lattice Boltzmann methods*, Phys. Fluids, 8 (1996), pp. 2527–2536.

[8] S. CHEN, Z. WANG, X. SHAN, AND G. D. DOOLEN, *Lattice Boltzmann computational fluid dynamics in three dimensions*, J. Statist. Phys., 68 (1992), pp. 379–400.

[9] W. E AND J. LIU, *Essentially compact schemes for unsteady viscous incompressible flows*, J. Comput. Phys., 126 (1996), pp. 122–138.

[10] U. Frisch, B. Hasslacher, and Y. Pomeau, *Lattice-gas automata for the Navier-Stokes equations*, Phys. Rev. Lett., 56 (1986), pp. 1505–1508.
[11] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J. P. Rivet, *Lattice gas hydrodynamics in two and three dimensions*, Complex Systems, 1 (1987), pp. 649–707.
[12] U. Ghia, K. N. Ghia, and C. T. Shin, *High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method*, J. Comput. Phys., 48 (1982), pp. 387–411.
[13] Z. Guo, B. Shi, and N. Wang, *Lattice BGK model for incompressible Navier-Stokes equation*, J. Comput. Phys., 165 (2000), pp. 288–306.
[14] X. He and L.-S. Luo, *Lattice Boltzmann model for the incompressible Navier-Stokes equation*, J. Statist. Phys., 88 (1997), pp. 927–944.
[15] S. Hou, Q. Zou, S. Chen, G. Doolen, and A. C. Cogley, *Simulation of cavity flow by the lattice Boltzmann method*, J. Comput. Phys., 118 (1995), pp. 329–347.
[16] D. d'Humières, *Generalized lattice Boltzmann equations*, in Rarefied Gas Dynamics: Theory and Simulations, Progress in Astronautics and Aeronautics 159, B. D. Shizgal and D. P. Weaver, eds., American Institute of Aeronautics and Astronautics, Washington, D.C., 1992, pp. 450–458.
[17] S. Jin and Z. Xin, *The relaxation schemes for systems of conservation laws in arbitrary space dimensions*, Comm. Pure Appl. Math., 48 (1995), pp. 235–277.
[18] M. Junk and W.-A. Yong, *Rigorous Navier-Stokes limit of the lattice Boltzmann equation*, Asymptot. Anal., 35 (2003), pp. 165–185.
[19] T. Kato, *Nonstationary flows of viscous and ideal fluids in* $\mathbf{R}^3$, J. Funct. Anal., 9 (1972), pp. 296–305.
[20] A. Klar, *Relaxation scheme for a lattice-Boltzmann-type discrete velocity model and numerical Navier-Stokes limit*, J. Comput. Phys., 148 (1999), pp. 416–432.
[21] J. M. V. A. Koelman, *A simple lattice Boltzmann scheme for Navier-Stokes fluid flow*, Europhys. Lett., 15 (1991), pp. 603–607.
[22] P. Lallemand and L.-S. Luo, *Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability*, Phys. Rev. E, 61 (2000), pp. 6546–6562.
[23] Y. H. Qian, D. d'Humières, and P. Lallemand, *Lattice BGK models for Navier-Stokes equation*, Europhys. Lett., 17 (1992), pp. 479–484.
[24] M. B. Reider and J. D. Sterling, *Accuracy of discrete-velocity BGK models for the simulation of the incompressible Navier-Stokes equations*, Comput. & Fluids, 24 (1995), pp. 459–467.
[25] R. Schreiber and H. Keller, *Driven cavity flows by efficient numerical techniques*, J. Comput. Phys., 49 (1983), pp. 310–333.
[26] P. A. Skordos, *Initial and boundary conditions for the lattice Boltzmann method*, Phys. Rev. E, 48 (1993), pp. 4823–4842.
[27] J. D. Sterling and S. Chen, *Stability analysis of lattice Boltzmann methods*, J. Comput. Phys., 123 (1996), pp. 196–206.
[28] S. Wolfram, *Cellular automaton fluids* I: *Basic theory*, J. Statist. Phys., 45 (1986), pp. 471–526.
[29] W.-A. Yong, *Basic aspects of hyperbolic relaxation systems*, in Advances in the Theory of Shock Waves, Progr. Nonlinear Differential Equations Appl. 47, H. Freistühler and A. Szepessy, eds., Birkhäuser Boston, Boston, MA, 2001, pp. 259–305.
[30] W.-A. Yong and L.-S. Luo, *Nonexistence of H theorems for athermal lattice Boltzmann models with polynomial equilibria*, Phys. Rev. E, 67 (2003), 051105.
[31] W.-A. Yong and L.-S. Luo, *Nonexistence of H theorem for some lattice Boltzmann models*, J. Statist. Phys., 121 (2005), pp. 91–103.
[32] Q. Zou, S. Hou, S. Chen, and G. Doolen, *An improved incompressible lattice Boltzmann model for time-independent flows*, J. Statist. Phys., 81 (1995), pp. 35–49.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2112–2120
© 2006 Society for Industrial and Applied Mathematics

A REORDERING FOR THE PAGERANK PROBLEM*

AMY N. LANGVILLE† AND CARL D. MEYER‡

Abstract. We describe a reordering particularly suited to the PageRank problem, which reduces the computation of the PageRank vector to that of solving a much smaller system and then using forward substitution to get the full solution vector. We compare the theoretical rates of convergence of the original PageRank algorithm to that of the new reordered PageRank algorithm, showing that the new algorithm can do no worse than the original algorithm. We present results of an experimental comparison on five datasets, which demonstrate that the reordered PageRank algorithm can provide a speedup of as much as a factor of 6. We also note potential additional benefits that result from the proposed reordering.

Key words. Markov chains, PageRank, reorderings, power method, convergence, stationary vector, dangling nodes

AMS subject classifications. 65B99, 65F10, 65C40, 60J22, 65F15, 65F50

DOI. 10.1137/040607551

1. Introduction. It is well known that many subsets of the web contain a large proportion of dangling nodes, i.e., webpages with no outlinks. Dangling nodes can result from many sources: a page containing an image or a PostScript or pdf file; a page of data tables; or a page whose links have yet to be crawled by the search engine's spider. These dangling nodes can present philosophical, storage, and computational issues for a search engine such as Google that employs a ranking system for ordering retrieved webpages.

Let us introduce some notation. The hyperlink structure of the web defines a directed graph, which can be expressed as a sparse matrix. The founders of Google, Brin and Page, defined the elements of a hyperlink matrix $\mathbf{H}$ as $h_{ij} = 1/|O_i|$ if there exists a hyperlink from page i to page j, and 0 otherwise. The scalar $|O_i|$ is the number of outlinks from page i. Thus, the nonzero rows of $\mathbf{H}$ sum to 1. The matrix $\mathbf{H}$ contains a $\mathbf{0}^T$ row for each dangling node. Brin and Page define the PageRank vector, a vector holding the global measure of importance for each page, to be the stationary vector for a Markov chain related to $\mathbf{H}$ [3, 4]. This definition is intuitive, as the stationary vector gives the long-run proportion of time the chain will spend in each state. For Brin and Page's application, the stationary elements represent the long-run proportion of time a random web surfer will spend on a page in the web, and thus provide a measure of a page's importance or popularity. One problem with the hyperlink matrix $\mathbf{H}$ created strictly from the web's structure is that it is not stochastic, and a Markov chain is defined only for stochastic matrices. To remedy this and create the stochastic matrix called $\mathbf{S}$, Brin and Page suggest replacing each $\mathbf{0}^T$ row (corresponding to a dangling node) of the sparse hyperlink matrix with a dense vector. The original fix for the dangling node problem used a uniform vector $\mathbf{e}^T/n$ ($\mathbf{e}$ is the vector of all ones), which was later replaced by the more general probability vector

*Received by the editors April 29, 2004; accepted for publication (in revised form) April 20, 2005; published electronically February 21, 2006.
http://www.siam.org/journals/sisc/27-6/60755.html
†Department of Mathematics, College of Charleston, Charleston, SC 29424 (langvillea@cofc.edu).
‡Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (meyer@math.ncsu.edu). This author's research was supported in part by NSF CCR-ITR-0113121 and NSF DMS 9714811.

$\mathbf{v}^T$, known as the personalization or teleportation vector. Of course, if this suggestion were to be implemented explicitly, storage requirements would increase dramatically. Instead, the stochasticity fix can be modeled implicitly with the construction of one vector denoted $\mathbf{a}$. Element $a_i = 1$ if row i of $\mathbf{H}$ corresponds to a dangling node, and 0 otherwise. Then $\mathbf{S}$ can be written as a rank-one update of $\mathbf{H}$; $\mathbf{S} = \mathbf{H} + \mathbf{a}\mathbf{v}^T$.

Before the existence of the stationary PageRank vector could be guaranteed, Brin and Page had to make one final adjustment to the hyperlink matrix. Because the web is reducible, their original PageRank algorithm would often get caught in a rank sink, causing convergence problems. To remedy this, and thereby guarantee the existence and uniqueness of the PageRank vector, Brin and Page added another rank-one update, this time an irreducibility adjustment in the form of a dense perturbation matrix $\mathbf{e}\mathbf{v}^T$ that creates direct connections between each page. The stochastic, irreducible matrix called the Google matrix $\mathbf{G}$ is given by

$$\begin{aligned}\mathbf{G} &= \alpha\,\mathbf{S} + (1-\alpha)\,\mathbf{e}\mathbf{v}^T \\ &= \alpha\,\mathbf{H} + \alpha\,\mathbf{a}\mathbf{v}^T + (1-\alpha)\,\mathbf{e}\mathbf{v}^T \\ &= \alpha\,\mathbf{H} + (\alpha\,\mathbf{a} + (1-\alpha)\,\mathbf{e})\mathbf{v}^T,\end{aligned}$$

where $0 < \alpha < 1$. Brin and Page, and consequently Google, then use the power method to compute the unique stationary vector of this enormous Markov chain, thereby producing their famous PageRank vector.

We now turn to the philosophical issue of the presence of dangling nodes. In one of their early papers [2], Brin et al. report that they "often remove dangling nodes during the computation of PageRank, then add them back in after the PageRanks have converged." From this vague statement it is hard to say exactly how Brin and Page compute PageRank for the dangling nodes. However, the removal of dangling nodes at any time during the power method does not make intuitive sense. Some dangling nodes should receive high PageRank. For example, a very authoritative pdf file could have many inlinks from respected sources and thus should receive a high PageRank. Simply removing the dangling nodes biases the PageRank vector unjustly. (See [5] and [13] for additional arguments against removal of dangling nodes.) Further, incorporating dangling nodes into the PageRank power method is very simple and inexpensive. The power method treats PageRank as an eigenvector problem and follows the iterative formula

$$\begin{aligned}\mathbf{x}^{(k)T} &= \mathbf{x}^{(k-1)T}\mathbf{G} \\ &= \alpha\mathbf{x}^{(k-1)T}\mathbf{S} + (1-\alpha)\mathbf{x}^{(k-1)T}\mathbf{e}\mathbf{v}^T \\ &= \alpha\mathbf{x}^{(k-1)T}\mathbf{S} + (1-\alpha)\mathbf{v}^T \\ &= \alpha\mathbf{x}^{(k-1)T}\mathbf{H} + \alpha\mathbf{x}^{(k-1)T}\mathbf{a}\mathbf{v}^T + (1-\alpha)\mathbf{v}^T, \qquad (1)\end{aligned}$$

since $x^{(k-1)T}$ is a probability vector, and thus, $\mathbf{x}^{(k-1)T}\mathbf{e} = 1$. This shows that the power method applied to $\mathbf{G}$ can be implemented with vector-matrix multiplications on the extremely sparse $\mathbf{H}$, and $\mathbf{G}$ and $\mathbf{S}$ are never formed or stored. Since the vector-matrix multiplications involve enormous entities, an important question to ask is how many iterations can we expect until the power method converges to the PageRank vector. It has been proven that the asymptotic rate of convergence of the power method applied to the Google matrix (after the stochasticity and irreducibility adjustments) is the rate at which $\alpha^k \to 0$. Since Google uses $\alpha = .85$, one can expect roughly 114 power iterations to give a convergence tolerance (as measured by the norm

of the difference of successive iterates) of less than 10^{-8}. Further, this means that on the order of $O(114 \cdot nnz(\mathbf{H}))$ operations must be performed to compute the PageRank vector with that prescribed tolerance, where $nnz(\mathbf{H})$ is the number of nonzeros in $\mathbf{H}$. Since $\mathbf{H}$ is so sparse, this is not a prohibitive cost, but it does nevertheless take days of computation when n, the number of pages in the index, is billions.

At this point, we have arrived at the computational issue associated with the dangling nodes. Since the dangling nodes form identical rows in the $\mathbf{H}$ matrix, they can be conveniently lumped into one class and the theory of lumpable and aggregated Markov chains can be used to compute the PageRank vector quickly. Lee, Golub, and Zenios were the first to notice and exploit the structure provided by the dangling nodes [13]. Their iterative power method implementation on aggregated Markov chains led to drastic improvements in the computation of PageRank. On some sample datasets dangling nodes make up over 80% of the webpages, meaning the Lee–Golub–Zenios algorithm can compute PageRank with a factor of 5 speedup, since the aggregated chain is roughly 1/5 the size of the full chain.

In this paper, we present an analogous formulation of the Lee–Golub–Zenios algorithm (section 2). Our formulation uses a linear system formulation rather than a Markov chain formulation. Then we extend the idea of exploiting the dangling nodes to reduce computation, thereby producing a convenient reordering of the Google matrix, which has several advantageous properties (section 3). In section 4, we present the results of our reordering algorithm on several sample datasets.

2. A linear system formulation for exploiting dangling nodes. Although the size of the Google problem makes the power method with its eigenvector formulation ($\boldsymbol{\pi}^T = \boldsymbol{\pi}^T\mathbf{G}$ and $\boldsymbol{\pi}^T\mathbf{e} = 1$) one of the few practical solution methods for obtaining the PageRank vector $\boldsymbol{\pi}^T$, there are other formulations of the problem that are theoretically possible. The following theorem shows that a related linear system exists.

THEOREM 2.1 (linear system for Google problem). *Solving the linear system*

$$\mathbf{x}^T(\mathbf{I} - \alpha\mathbf{H}) = \mathbf{v}^T \tag{2}$$

and letting $\boldsymbol{\pi}^T = \mathbf{x}^T/\mathbf{x}^T\mathbf{e}$ *produces the PageRank vector.*

Proof. $\boldsymbol{\pi}^T$ is the PageRank vector if it satisfies $\boldsymbol{\pi}^T\mathbf{G} = \boldsymbol{\pi}^T$ and $\boldsymbol{\pi}^T\mathbf{e} = 1$. Clearly, $\boldsymbol{\pi}^T\mathbf{e} = 1$. Showing $\boldsymbol{\pi}^T\mathbf{G} = \boldsymbol{\pi}^T$ is equivalent to showing $\boldsymbol{\pi}^T(\mathbf{I} - \mathbf{G}) = \mathbf{0}^T$, which is equivalent to showing $\mathbf{x}^T(\mathbf{I} - \mathbf{G}) = \mathbf{0}^T$:

$$\begin{aligned}\mathbf{x}^T(\mathbf{I} - \mathbf{G}) &= \mathbf{x}^T(\mathbf{I} - \alpha\mathbf{H} - \alpha\mathbf{a}\mathbf{v}^T - (1-\alpha)\mathbf{e}\mathbf{v}^T)\\ &= \mathbf{x}^T(\mathbf{I} - \alpha\mathbf{H}) - \mathbf{x}^T(\alpha\mathbf{a} + (1-\alpha)\mathbf{e})\mathbf{v}^T\\ &= \mathbf{v}^T - \mathbf{v}^T = \mathbf{0}^T.\end{aligned}$$

The above line results from the fact that $\mathbf{x}^T(\alpha\mathbf{a} + (1-\alpha)\mathbf{e})\mathbf{v}^T = 1$ because

$$\begin{aligned}1 &= \mathbf{v}^T\mathbf{e}\\ &= \mathbf{x}^T(\mathbf{I} - \alpha\mathbf{H})\mathbf{e}\\ &= \mathbf{x}^T\mathbf{e} - \alpha\mathbf{x}^T\mathbf{H}\mathbf{e}\\ &= \mathbf{x}^T\mathbf{e} - \alpha\mathbf{x}^T(\mathbf{e} - \mathbf{a})\\ &= (1-\alpha)\mathbf{x}^T\mathbf{e} + \alpha\mathbf{x}^T\mathbf{a}. \qquad \square\end{aligned}$$

Thus, PageRank is both the stationary distribution of a Markov chain and the solution of a linear system. While these two views are inherently linked, they suggest

different methods for finding the PageRank vector. Future advances in computation will likely require a thorough understanding and use of both views.

As a result, we examine the linear system view, which has received much less attention. The coefficient matrix $(\mathbf{I}-\alpha\mathbf{H})$ of the linear system has many nice properties, which were proven in [12]. Some that are relevant for this paper are as follows:

- $(\mathbf{I}-\alpha\mathbf{H})$ is nonsingular.
- The row sums of $(\mathbf{I}-\alpha\mathbf{H})^{-1}$ are equal to 1 for the dangling nodes and less than or equal to $1/(1-\alpha)$ for the nondangling nodes.
- The row of $(\mathbf{I}-\alpha\mathbf{H})^{-1}$ corresponding to dangling node i is $\mathbf{e}_i^T$, where $\mathbf{e}_i$ is the ith column of the identity matrix.

The last property makes the computation of the PageRank vector especially efficient. Suppose the rows and columns of $\mathbf{H}$ are permuted (i.e., the indices are reordered) so that the rows corresponding to dangling nodes are at the bottom of the matrix:

$$\mathbf{H} = \begin{matrix} \\ ND \\ D \end{matrix} \begin{matrix} \begin{matrix} ND & \quad D \end{matrix} \\ \begin{pmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \end{matrix}, \tag{3}$$

where ND is the set of nondangling nodes and D is the set of dangling nodes. Then the coefficient matrix in the sparse linear system formulation becomes

$$(\mathbf{I}-\alpha\mathbf{H}) = \begin{pmatrix} \mathbf{I}-\alpha\mathbf{H}_{11} & -\alpha\mathbf{H}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix},$$

and the inverse of this matrix is

$$(\mathbf{I}-\alpha\mathbf{H})^{-1} = \begin{pmatrix} (\mathbf{I}-\alpha\mathbf{H}_{11})^{-1} & \alpha(\mathbf{I}-\alpha\mathbf{H}_{11})^{-1}\mathbf{H}_{12} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}.$$

Therefore, the unnormalized PageRank vector $\mathbf{x}^T = \mathbf{v}^T(\mathbf{I}-\alpha\mathbf{H})^{-1}$ can be written as

$$\mathbf{x}^T = \begin{pmatrix} \mathbf{v}_1^T(\mathbf{I}-\alpha\mathbf{H}_{11})^{-1} & | & \alpha\mathbf{v}_1^T(\mathbf{I}-\alpha\mathbf{H}_{11})^{-1}\mathbf{H}_{12} + \mathbf{v}_2^T \end{pmatrix},$$

where the personalization vector $\mathbf{v}^T$ has been partitioned into nondangling ($\mathbf{v}_1^T$) and dangling ($\mathbf{v}_2^T$) sections. In summary, we now have an algorithm that computes the PageRank vector using only the nondangling portion of the web, exploiting the rank-one structure of the dangling node fix.

ALGORITHM 1.

1. Reorder the states of the Markov chain so that the reordered matrix has the structure of (3).
2. Solve for $\mathbf{x}_1^T$ in $\boldsymbol{\pi}_1^T(\mathbf{I}-\alpha\mathbf{H}_{11}) = \mathbf{v}_1^T$.
3. Compute $\mathbf{x}_2^T = \alpha\boldsymbol{\pi}_1^T\mathbf{H}_{12} + \mathbf{v}_2^T$.
4. Normalize $\boldsymbol{\pi}^T = [\mathbf{x}_1^T \;\; \mathbf{x}_2^T]/\|[\mathbf{x}_1^T \;\; \mathbf{x}_2^T]\|_1$.

This algorithm is much simpler and cleaner than, but equivalent to, the specialized iterative method proposed by Lee, Golub, and Zenios [13], which exploits the dangling nodes to reduce computation of the PageRank vector, sometimes by a factor of 5.

3. A PageRank algorithm based on a reordering of the Google matrix.

3.1. The reordered PageRank algorithm. The linear system formulation of section 2 leads to a deeper examination of the structure of the Google matrix $\mathbf{H}$. Since the presence of zero rows in the matrix is so advantageous, we hope to find more zero rows in the submatrix $\mathbf{H}_{11}$, which is needed to solve the system in step 1 of Algorithm 1. This process of locating zero rows can be repeated recursively on smaller and smaller submatrices of $\mathbf{H}$, continuing until a submatrix is created that has no zero rows. The result of this process is a decomposition of the $\mathbf{H}$ matrix that looks like Figure 1. This process amounts to a simple reordering of the indices of the Markov chain. The top part shows the original $\mathbf{H}$ matrix, and the bottom part is the reordered matrix according to the recursive dangling node idea. The dataset `CA.dat` (available from http://www.cs.cornell.edu/Courses/cs685/2002fa/) is a typical subset of the web. It contains 9,664 nodes and 16,773 links pertaining to the query topic of "california."

In general, after this symmetric reordering, the hyperlink matrix $\mathbf{H}$ has the following structure:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{11} & \mathbf{H}_{12} & \mathbf{H}_{13} & \cdots & \mathbf{H}_{1b} \\ & \mathbf{0} & \mathbf{H}_{23} & \cdots & \mathbf{H}_{2b} \\ & & \mathbf{0} & \cdots & \mathbf{H}_{3b} \\ & & & \ddots & \\ & & & & \mathbf{0} \end{pmatrix}, \tag{4}$$

where $b \geq 2$ is the number of square diagonal blocks in the reordered matrix. Therefore, the coefficient matrix of the linear system formulation of the PageRank problem (2) has the following structure:

$$(\mathbf{I} - \alpha\mathbf{H}) = \begin{pmatrix} \mathbf{I} - \alpha\mathbf{H}_{11} & -\alpha\mathbf{H}_{12} & -\alpha\mathbf{H}_{13} & \cdots & -\alpha\mathbf{H}_{1b} \\ & \mathbf{I} & -\alpha\mathbf{H}_{23} & \cdots & -\alpha\mathbf{H}_{2b} \\ & & \mathbf{I} & \cdots & -\alpha\mathbf{H}_{3b} \\ & & & \ddots & \\ & & & & \mathbf{I} \end{pmatrix}. \tag{5}$$

As a result, the PageRank system in (2) after reordering can be solved by forward substitution. The only system that must be solved directly is the first subsystem, $\mathbf{x}_1^T(\mathbf{I} - \alpha\mathbf{H}_{11}) = \mathbf{v}_1^T$, where $\mathbf{x}^T$ and $\mathbf{v}^T$ have also been partitioned according to the number and size of the blocks. The remaining subvectors of $\mathbf{x}^T$ are computed quickly and efficiently by forward substitution. In the `CA.dat` example, a $2{,}622 \times 2{,}622$ system can be solved instead of the full $9{,}664 \times 9{,}664$ system, or even the once-reduced $5{,}132 \times 5{,}132$ system of Algorithm 1. The reordered PageRank algorithm is an extension of the dangling node method of section 2. This reordered PageRank idea can also be written in a Markov chain formulation, thereby extending the ideas in [13]. The steps of the reordered PageRank (in its linear system formulation) are enumerated in Algorithm 2.

ALGORITHM 2.

1. Reorder the states of the Markov chain so that the reordered matrix has the structure of (4).
2. Solve for $\mathbf{x}_1^T$ in $\mathbf{x}_1^T(\mathbf{I} - \alpha\mathbf{H}_{11}) = \mathbf{v}_1^T$.
3. For $i = 2$ to b, compute $\mathbf{x}_i^T = \alpha \sum_{j=1}^{i-1} \mathbf{x}_j^T \mathbf{H}_{ji} + \mathbf{v}_i^T$.
4. Normalize $\boldsymbol{\pi}^T = [\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T]/\|[\mathbf{x}_1^T \ \mathbf{x}_2^T \ \cdots \ \mathbf{x}_b^T]\|_1$.

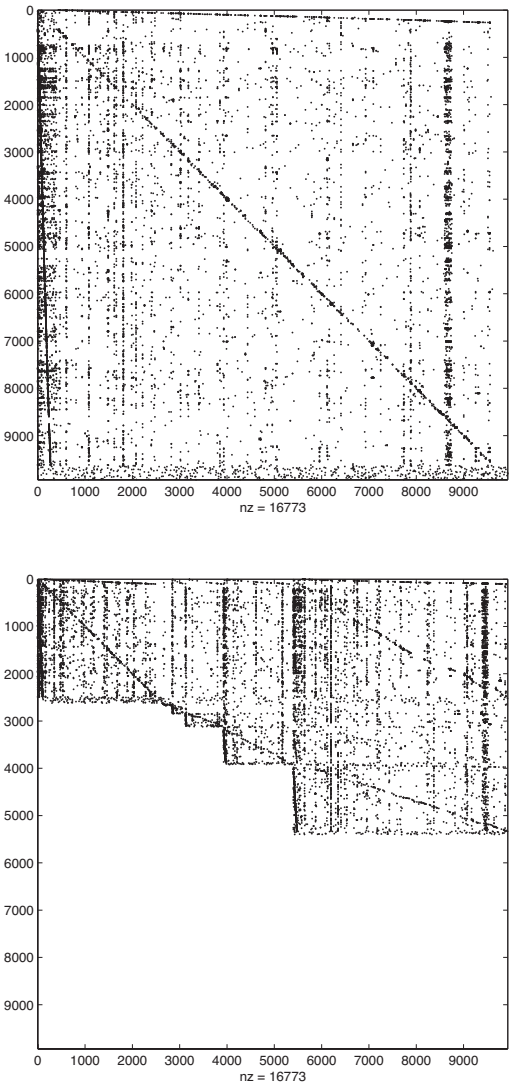


FIG. 1. *Original* **H** *(top) and reordered* **H** *matrix (bottom) for* `CA.dat`.

3.2. Analysis of the reordered PageRank algorithm. We discuss each step of Algorithm 2. The reordering, step 1 of Algorithm 2, can be accomplished with a simple recursive scheme that successively locates zero rows of submatrices within **H**, and thus amounts to the work of a few power iterations on the full matrix. However, the memory manipulation required to successively reorder the submatrices can make this step time consuming and potentially of less value than Algorithm 1 (see the experiments of section 4). The forward substitution step of Algorithm 2, step 3, amounts to slightly less than the work of one power iteration. Of course, step 4 is trivial. Thus, the small system solve of step 2 is the most time-consuming step. Table 1 in section 4 gives an enlightening numerical comparison of the amount of work spent on each step.

Some hyperlink matrices $\mathbf{H}$ can be particularly suited to the reordered PageRank algorithm. In this case, large sets of zero rows are consecutively extracted, and the remaining $\mathbf{H}_{11}$ block is very small. In the worst case, $b = 2$ and $nnz(\mathbf{H}_{11})/nnz(\mathbf{H})$ is close to 1. Regardless, the smaller system $\mathbf{x}_1^T(\mathbf{I} - \alpha\mathbf{H}_{11}) = \mathbf{v}_1^T$ can be solved by any appropriate direct or iterative method. However, we have found that the sparsity of the $\mathbf{H}_{11}$ matrix makes a simple iterative method very attractive. The Jacobi method [14] is especially simple due to the properties of the $\mathbf{H}_{11}$ matrix. The diagonal elements of $\mathbf{H}_{11}$ are 0, since by convention no webpage links to itself. This means that the diagonal matrix $\mathbf{D}$ involved in the splitting of the coefficient matrix $(\mathbf{I} - \alpha\mathbf{H}_{11})$ is the identity matrix $\mathbf{I}$. Therefore, the Jacobi iterates are

$$\mathbf{x}^{(k)T} = \alpha\mathbf{x}^{(k-1)T}\mathbf{H}_{11} + \mathbf{v}_1^T. \tag{6}$$

This stationary iterative process converges to a unique solution for all starting iterates because $\rho(\alpha\mathbf{H}_{11}) \leq 1$. ($\mathbf{H}_{11}$ is substochastic.) The asymptotic rate of convergence of the iterative reordered PageRank process is the same as or better than the asymptotic rate of convergence of the standard PageRank power method. In fact, the reordered PageRank method will require about $O(\frac{\log(\tau)}{\log(\alpha)} \cdot nnz(\mathbf{H}_{11}))$ operations to reach a convergence tolerance of τ. In summary, Algorithm 2, the reordered PageRank method, requires roughly $\frac{nnz(\mathbf{H}_{11})}{nnz(\mathbf{H})}\%$ of the time that the original PageRank method requires. The Jacobi iterates of (6), like the power iterates of the original PageRank algorithm, also allow for easy parallelization, which is essential due to the scale of the matrices involved in practical applications. Further, acceleration methods [1, 6, 8, 9, 10], such as extrapolation and preconditioners, can be applied to the small $\mathbf{H}_{11}$ system to achieve even greater speedups.

4. Experiments with the reordering method. We compare the reordered PageRank algorithm to the original PageRank algorithm, experimenting with four datasets. The first two datasets are small, containing fewer than 10,000 pages. The remaining two datasets are much larger, with one containing over 325,000 pages and the other over 450,000 pages. Each dataset is a different type of subset of the web. The `CA.dat` dataset contains 9,664 pages pertaining to the query topic of "california." `NCSU.dat` is a crawl that started at the North Carolina State homepage and stopped once 10,000 pages were reached. Most of the pages are internal to the NCSU domain, but some are external. The dataset `ND.dat` contains 325,729 pages within the Notre Dame domain. Finally, `SU450k.dat` contains 451,237 pages from a 2001 crawl of the Stanford domain by the Stanford WebBase project.

The experiments in this section were conducted on a 867 MHz Mac G4 with 1.5 GB of RAM. We used $\alpha = .9$ as the scaling factor and $\tau = 10^{-10}$ as the convergence tolerance. The original PageRank algorithm and both reordered PageRank algorithms (Algorithms 1 and 2) were run on each dataset. The results are summarized in Table 1.

The reordered PageRank algorithm beats the original PageRank algorithm by a factor of almost 6 on some datasets. The speedup depends on the ratio $\frac{nnz(\mathbf{H})}{nnz(\mathbf{H}_{11})}$, which depends on the properties of a particular dataset and the number of zero rows that can be successively squeezed out of the submatrices. The reordered PageRank algorithm is guaranteed to outperform the original PageRank method as long as some dangling nodes are present, i.e., $b \geq 2$.

Unfortunately, Table 1 clearly shows that Algorithm 2 is not necessarily an improvement over Algorithm 1 because the cost of memory management required by the reordering step often offsets the decrease in the size of the system solve. Nevertheless,

TABLE 1
Comparison of original PageRank and reordered PageRank algorithms.

			CA.dat	NCSU.dat	ND.dat	SU450k.dat
Original PageRank	$n(\mathbf{H})$		10K	10K	325K	450K
	$nnz(\mathbf{H})$		16K	101K	1,497K	1,082K
	Iterations		178	162	166	164
	Time (sec.)		3.30	5.28	126.71	175.55
Reordered PR Algorithm 1 ($b = 2$)	b		2	2	2	2
	$n(\mathbf{H}_{11})$		5K	7K	138K	137K
	$nnz(\mathbf{H}_{11})$		8K	81K	1,208K	307K
	Iterations		170	162	166	145
	Time:	Step 1	.14	.32	2.37	2.84
		Step 2	.78	3.96	62.47	26.84
		Step 3	.10	.13	.66	1.16
		Total	1.02	4.41	65.50	30.89
	Speedup		**3.2**	**1.2**	**1.9**	**5.7**
Reordered PR Algorithm 2 ($b > 2$)	b		7	5	18	12
	$n(\mathbf{H}_{11})$		2.6K	6.8K	127K	84K
	$nnz(\mathbf{H}_{11})$		5K	79K	1,191K	267K
	Iterations		169	160	170	145
	Time:	Step 1	.24	.58	22.74	12.42
		Step 2	.57	4.17	64.62	20.76
		Step 3	.11	.13	2.53	2.16
		Total	.92	4.88	89.89	35.34
	Speedup		**3.6**	**1.1**	**1.4**	**5.0**

compared to the original PageRank algorithm, both algorithms do drastically improve the time required to compute the PageRank vector.

5. Future work. The relationship between the spectrum of $\mathbf{G}$ and that of $\mathbf{S}$ is known [7, 12]. However, their relationship to the spectrum of $\mathbf{H}$ is not known. Recall that $\mathbf{S}$ and $\mathbf{G}$ are web matrices that have been artificially enhanced to generate advantageous mathematical properties in the matrices. The true web matrix is $\mathbf{H}$, yet little is known about how far $\mathbf{S}$ and $\mathbf{G}$ and their corresponding ranking vectors stray from the web's true nature. The reordering presented here may illuminate this unknown relationship. The symmetric reordering shows that the eigenvalues of $\mathbf{H}$ are the eigenvalues of $\mathbf{H}_{11}$ plus many 0 eigenvalues for the $\mathbf{0}$ diagonal blocks. The size of $\mathbf{H}_{11}$ may be small enough that its subdominant eigenvalues and eigenvectors can be computed. These subdominant eigenvectors can provide beneficial information about the community structure of a subset of the web [11]. Further, since $\mathbf{S}$ is created from a rank-one update to $\mathbf{H}$, perturbation theory for eigenvalues and eigenvectors can provide additional information about the PageRank vector with respect to its original hyperlink structure ($\mathbf{H}$, not $\mathbf{S}$ or $\mathbf{G}$).

6. Conclusions. Reorderings for linear systems have been well studied. Some popular reorderings, such as the minimum degree reordering, the reverse Cuthill–McKee reordering, and the Dulmage–Mendelson reordering, do not give the nice structure of (5), which requires only one small system solve plus a quick forward substitution that amounts to the work of one power iteration. This reordering naturally produces an efficient alternative to the original PageRank algorithm, which we have called the reordered PageRank algorithm. The reordered algorithm has an asymptotic rate of convergence that is as good as or better than the asymptotic rate of convergence of the original method. The reordered algorithm produces a dataset-dependent speedup over the original algorithm. We presented experimental results

comparing the reordered PageRank algorithm to the original PageRank algorithm on five datasets, showing a factor of nearly 6 speedup on one dataset. This reordering and its linear system formulation also open the door for new PageRank acceleration techniques beyond the usual power method acceleration techniques.

Acknowledgments. The paper [13] by Chris Lee of Stanford University and discussions with Michele Benzi of Emory University sparked the reordering idea presented in this paper. We thank Cleve Moler (Mathworks), Ronny Lempel (IBM Research Labs), and Jon Kleinberg (Cornell) for providing us with several small datasets that we used for testing. Our large dataset from the Stanford web crawl was provided by Chris Lee.

REFERENCES

[1] A. Arasu, J. Novak, A. Tomkins, and J. Tomlin, *PageRank computation and the structure of the Web: Experiments and algorithms*, in The Eleventh International WWW Conference, ACM Press, New York, 2002.

[2] S. Brin, R. Motwani, L. Page, and T. Winograd, *What can you do with a Web in your pocket?*, Data Engrg. Bull., 21 (1998), pp. 37–47.

[3] S. Brin and L. Page, *The anatomy of a large-scale hypertextual Web search engine*, Comput. Networks and ISDN Syst., 33 (1998), pp. 107–117.

[4] S. Brin, L. Page, R. Motwami, and T. Winograd, *The PageRank Citation Ranking: Bringing Order to the Web*, Technical Report 1999-0120, Computer Science Department, Stanford University, Stanford, CA, 1999.

[5] N. Eiron, K. S. McCurley, and J. A. Tomlin, *Ranking the Web frontier*, in The Thirteenth International World Wide Web Conference, ACM Press, New York, 2004.

[6] G. H. Golub and C. Greif, *Arnoldi-Type Algorithms for Computing Stationary Distribution Vectors, with Application to PageRank*, Technical Report SCCM-2004-15, Scientific Computation and Computational Mathematics, Stanford University, Stanford, CA, 2004.

[7] T. H. Haveliwala and S. D. Kamvar, *The Second Eigenvalue of the Google Matrix*, Technical Report 2003-20, Stanford University, Stanford, CA, 2003.

[8] S. D. Kamvar, T. H. Haveliwala, and G. H. Golub, *Adaptive Methods for the Computation of PageRank*, Technical Report 2003-26, Stanford University, Stanford, CA, 2003.

[9] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, *Exploiting the Block Structure of the Web for Computing PageRank*, Technical Report 2003-17, Stanford University, Stanford, CA, 2003.

[10] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, *Extrapolation methods for accelerating PageRank computations*, in The Twelfth International World Wide Web Conference, ACM Press, New York, 2003.

[11] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, J. ACM, 46 (1999), pp. 604–632.

[12] A. N. Langville and C. D. Meyer, *Deeper inside PageRank*, Internet Math. J., 1 (2005), pp. 335–380.

[13] C. P.-C. Lee, G. H. Golub, and S. A. Zenios, *A Fast Two-Stage Algorithm for Computing PageRank and Its Extensions*, Technical Report SCCM-2003-15, Scientific Computation and Computational Mathematics, Stanford University, Stanford, CA, 2003.

[14] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2121–2139
© 2006 Society for Industrial and Applied Mathematics

NUMERICAL STOCHASTIC INTEGRATION FOR QUASI-SYMPLECTIC FLOWS*

R. MANNELLA†

Abstract. Algorithms for the numerical integration of Langevin equations obeying detailed balance are introduced. The algorithms are derived fulfilling the requirements that they should become symplectic in the deterministic frictionless limit, and should reproduce the equilibrium distributions to some higher order in the integration time step when the equilibrium distribution exists. Extensions to the case when the system volume or pressure are kept constant are discussed. Comparisons with other integration schemes are carried out both for static and dynamical quantities.

Key words. numerical integration, stochastic differential equations, symplectic integration

AMS subject classifications. 65C30, 65Z05

DOI. 10.1137/040620965

1. Introduction.

Stochastic processes are well known to be at the heart of many physical systems.[1] Several approaches have therefore been developed to understand the dynamics realized by specific models: among others, one of techniques which is widely used is Monte Carlo integration of the equations of motion.

The literature on the numerical integration of stochastic differential equations (SDEs) is huge; we will limit ourselves to mentioning a couple of classical citations widely used in the physics community [2, 3, 4]. Additional comments and references can be found in [5]. The numerical algorithms presented in these works are very general ones and can be applied to basically any stochastic flow; however, they might not be the optimal ones for cases when additional information about the system under study is available.

An important class for which specialized algorithms can be derived is given by the equations of motion

$$\begin{aligned} \dot{x} &= v, \\ \dot{v} &= -\gamma v + F(x) + \xi(t), \end{aligned} \tag{1.1}$$

where $\xi(t)$ is a random Gaussian noise, with zero average and standard deviation

$$\langle \xi(t)\xi(s) \rangle = 2\gamma T \delta(t-s). \tag{1.2}$$

In the following, we will also use $V(x)$, defined as $F(x) \equiv -V'(x)$. Note that although for simplicity we will first deal with only one Brownian walker, we will then extend the algorithms to the case of many walkers, where x and v are vectors, and $F_i(x)$, the force acting on the ith walker, is a function of all other walkers and/or an external field.

The above equation is commonly found in the liquid state literature (for numerical schemes appropriate in the integration of the Brownian dynamics of a liquid, see among others [6, 7, 8, 9, 10, 11, 12, 13]), and some algorithms have been proposed,

*Received by the editors December 16, 2004; accepted for publication (in revised form) July 7, 2005; published electronically February 21, 2006.
http://www.siam.org/journals/sisc/27-6/62096.html

†Dipartimento di Fisica and INFM, UdR Pisa, Università di Pisa, Largo Bruno Pontecorvo 3, I-56127 Pisa, Italy (mannella@df.unipi.it).

[1] For a classical introduction, see [1].

over the years, for its numerical integration. Note also that the dynamics implicit in (1.1) is of a system in thermal equilibrium with a thermostat at a temperature T. In the case of many walkers in contact with a thermostat, other approaches have been developed which attempt to reproduce the correct equilibrium properties without using stochastic terms (the reason being the possibility of using algorithms for ordinary differential equations; see below). We will show that the algorithms derived here are a viable alternative to these methods.

To date, perhaps the most widely used algorithms for the integration of (1.1) are the ones derived in [6], where a couple of algorithms have been proposed (see also references therein). We will benchmark against one of them, and to this end we will briefly review it below. Note that the system of (1.1) becomes symplectic when $\gamma \to 0$ and, until some recent works [14, 15, 16], the symplectic nature was not really exploited in deriving numerical schemes. Algorithms which also use low order symplectic schemes as base schemes are the ones in [11, 12, 13]. Among them, the best known is the so-called Brunger, Brooks, and Karplus (BBK) scheme, which we will review in the next section and which will also be used as a benchmark.

In [17] we reported already on quasi-symplectic integrators, focusing on one Brownian walker and studying the behavior of the various schemes for large and small dampings γ. Here, we will limit ourselves to the small damping regime, extending, first, the comparison to include schemes not considered in [17]. We will then turn to the case of many interacting walkers, develop higher order symplectic schemes, and finally apply the algorithms to the integration of the equations of motion of many interacting Brownian walkers in contact with a thermostat and for fixed volume or fixed pressure. There is going to be, hence, some overlap between the first two sections and the material in [17], repeated here to make this paper clearer and as self-contained as possible.

The approach we will follow is to derive numerical algorithms having in mind two requirements: (i) the algorithm should become symplectic in the deterministic, frictionless limit ($T = \gamma = 0$); and (ii) the numerical algorithm should reproduce as closely as possible the equilibrium distribution, when this is defined (i.e., for $V(x)$ bounded from below; see next section), of the system given by (1.1). The requirement $T = 0$ seems redundant once $\gamma = 0$ is imposed. It is, however, possible to have symplectic stochastic dynamics, see for instance [14, 15, 18], i.e., with $\gamma = 0$. Looking at the structure of (1.1) and at the condition of detailed balance (1.2), this would imply an infinite T, and we need to explicitly exclude this situation. As we will see below, to the best of our knowledge either the former or the latter requirements have been enforced in the derivation of numerical schemes, but never both of them, before [17].

2. Some definitions and a brief review of the benchmark algorithms. To assess how well each algorithm is performing, we start from the knowledge that for $V(x)$ which are bounded from below, (1.1) leads to an equilibrium distribution $P(x, v)$ for the variables x and v of the form

$$P(x, v) = N \exp\left\{-\left(v^2/2 + V(x)\right)/T\right\}, \tag{2.1}$$

where N is a normalization constant. Other approaches can be used, but here we are going to compare the exact theoretical equilibrium distribution to the equilibrium distribution obtained from the simulations. It is possible, in principle, to check theoretically what is the equilibrium distribution which is expected when integrating using a given numerical scheme, following [19]. Suppose we have a numerical scheme

of the form

$$x_i(t+h) = x_i(t) + hW_i(x_i, \xi).$$

Then the probability distribution of x_i satisfies the Kramers–Moyal expansion

$$P(x_i, t+h) - P(x_i, t) = \sum_{n=1}^{\infty} \sum_{x_i} \frac{\partial}{\partial x_i}, \ldots, \frac{\partial}{\partial x_n} K_{1,\ldots,n} P(x_i, t) \tag{2.2}$$

with

$$K_{1,\ldots,n} \equiv (-1)^n \frac{1}{n!} \langle W_{x_1} \ldots W_{x_n} \rangle_\xi,$$

where $\langle \ldots \rangle_\xi$ means averaging over the noise realizations and the $K_{1,\ldots,n}$ depend explicitly on the integration time step h and its powers. Equation (2.2) is obtained [19] from the evolution equation for $P(x,t)$,

$$P(x_i, t+h) = \left\langle \int dx_i(t) P(x_i, t) \times \Pi_i \delta(x_i(t+h) - x_i(t) - hW_i(x_i, \xi)) \right\rangle_\xi,$$

by Taylor expanding the δ function in powers of W.

A system in detailed balance at temperature T, like the one given by (1.1), has some theoretical equilibrium distribution (for $V(x)$ bounded from below). A numerical scheme, however, will in general generate an equilibrium distribution which differs from the theoretical one. Following [19], we expect that

$$P(x_i, t=\infty)_{sim} = P(x_i, \infty)_{true} \times \exp\left(\sum_{n=1}^{\infty} h^n S_n / T \right),$$

where P_{sim} is the equilibrium distribution generated in the simulations, and P_{true} is the theoretical equilibrium distribution. Given the explicit form of the numerical scheme, the various $K_{1,\ldots,n}$ can be computed: applying (2.2) with P_{sim} on the right-hand side (R.H.S.) as $P(x_i, t)$ and expanding the R.H.S. of (2.2) in the small parameter h, assuming (equilibrium) that $P(x_i, t+h) - P(x_i, t) = 0$, we can derive the equations satisfied by the S_i annihilating the coefficients in a h power series. Note that the lowest order in h is the "standard" Fokker–Planck equation for the system under study. The first nonzero S_i yields the correction to the true equilibrium distribution generated by the numerical scheme.

The algebra to derive the correction to the equilibrium distribution induced by the numerical scheme in the general case and for a given high order integration scheme can be formidable. Therefore, we are going to show how to use (2.2) taking as an example a simple case, the Euler integration scheme. The Euler scheme integrates (1.1) using the prescription

$$\begin{aligned} x(t+h) &= x(t) + hv(t), \\ v(t+h) &= v(t) + hF(x(t)) - h\gamma v(t) + \eta, \end{aligned} \tag{2.3}$$

where η is a random Gaussian variable with zero average and moment

$$\langle \eta^2 \rangle = 2\gamma T h.$$

We compute first some of the lowest K's, up to $\langle W^4 \rangle$, i.e., up to terms like K_{ijkl}. To give a glimpse of the procedure followed, we have, for instance,

$$K_x = -hv(t),$$

$$K_{vv} = \frac{1}{2}\left[hF(x(t) - h\gamma v(t))^2 + \langle \eta^2 \rangle\right],$$

$$K_{xxv} = -\frac{1}{6}h^3 v^2(t)(F(x(t)) - \gamma v(t)),$$

and so on. Assuming that (see (2.1), with $S \equiv S_1$)

$$P(x,v) = N \exp\left\{-\left[v^2/2 + V(x) + hS(x,v)\right]/T\right\},$$

plugging the scheme of (2.3) into (2.2), and using the computed expressions of K's, we have that $S(x,v)$ satisfies the PDE

$$2\gamma T \frac{\partial^2 S(x,v)}{\partial v^2} - 2v\frac{\partial S(x,v)}{\partial x} - 2\left(F(x) + \gamma v\right)\frac{\partial S(x,v)}{\partial v}$$

$$+ T\gamma^2 - v^2\gamma^2 - F(x)^2 - F'(x)(2T - v^2) - 2v\gamma F(x) = 0.$$

It is clear that $S(x,v) = 0$ is not a solution of this PDE, and this implies that this algorithm fails to reproduce the correct equilibrium distribution at $O(h)$ in the exponent.

Before deriving the quasi-symplectic schemes, let us briefly mention some of the most-used algorithms for the integration of (1.1). In the limit of small γ, in [6] an algorithm is proposed, which reads

$$\begin{aligned} x(t+h) &= x(t) + c_1 h v(t) + c_2 h^2 F(x(t)) + \eta_1, \\ v(t+h) &= c_0 v(t) + (c_1 - c_2)hF(x(t)) + c_2 hF(x(t+h)) + \eta_2, \end{aligned} \tag{2.4}$$

where

$$c_0 = e^{-\gamma h}, \quad c_1 = \frac{1 - c_0}{\gamma h}, \quad c_2 = \frac{1 - c_1}{\gamma h},$$

and η_1 and η_2 are two random Gaussian variables with zero average and moments

$$\langle \eta_1^2 \rangle = \frac{Th}{\gamma}\left(2 - \frac{3 - 4e^{-\gamma h} + e^{-2\gamma h}}{\gamma h}\right),$$

$$\langle \eta_2^2 \rangle = T\left(1 - e^{-2\gamma h}\right),$$

$$\langle \eta_1 \eta_2 \rangle = \frac{T}{\gamma}\left(1 - e^{-\gamma h}\right)^2.$$

Using (2.2) to evaluate the correction to the true equilibrium distribution generated by this algorithm, we find that the contribution S_1 vanishes, and we are left with the term S_2. In other words, this algorithm reproduces the correct equilibrium distribution at $O(h)$, but there are still corrections $O(h^2)$ in the exponent. The algorithm given in (2.4) is one of reference algorithms which we propose to improve in the next section. We will refer to this algorithm as "Li" (from *li*quid) in the following.

A second widely used algorithm is the BBK scheme [11, 12, 13]. The BBK scheme integrates (1.1) as

$$\tilde{v} = v(t) + \frac{h}{2}F(x(t)),$$
$$x(t+h) = x(t) + \frac{[1-\exp(-\gamma h)]}{\gamma}\tilde{v} + \sqrt{2T/\gamma}\eta_2,$$
$$v(t+h) = \exp(-\gamma h)\tilde{v} + \frac{h}{2}F(x(t+h)) + \sqrt{2T\gamma}\eta_1.$$

For the stochastic variables η_1, η_2 one has [11, 12, 13]

$$\eta_1 = \sqrt{\tau_2}w_1,$$

$$\eta_2 = \frac{\tau_1 - \tau_2}{\sqrt{\tau_2}}w_1 + \sqrt{h - \frac{\tau_1^2}{\tau_2}}w_2,$$

where w_1 and w_2 are two uncorrelated Gaussian variables with average zero and standard deviation one, $\tau_1 = (1/\gamma)\left[1 - e^{-\gamma h}\right]$, and $\tau_2 = (1/2\gamma)\left[1 - e^{-2\gamma h}\right]$. The BBK algorithm reproduces the equilibrium distribution at order $o(h^2)$ in the exponent; the correction obeys the equation ($S \equiv S_2$):

$$0 = -\gamma F(x)^2 - (2\gamma T + 3\gamma v^2)F'(x) - v^3F''(x)12\gamma v\frac{\partial S(x,v)}{\partial v}$$
$$- F(x)\left[\gamma^2 v + 3vF'(x) + 12\frac{\partial S(x,v)}{\partial v}\right] + 12\gamma T\frac{\partial^2 S(x,v)}{\partial v^2} - 12v\frac{\partial S(x,v)}{\partial x}.$$

In the following, we will use also the Heun algorithm to carry out the various comparisons. To make this paper as self-contained as possible, we recall here that the Heun algorithm for a system like the one in (1.1) is given by the following prescription:

$$x_1 = x(t) + hv(t),$$
$$v_1 = v(t) - h\gamma v(t) + hF(x(t)) + \sqrt{2\gamma Th}\eta,$$
$$x_2 = x(t) + hv_1,$$
$$v_2 = v(t) - h\gamma v_1 + hF(x_1) + \sqrt{2\gamma Th}\eta,$$
$$x(t+h) = \frac{x_1 + x_2}{2}, \quad v(t+h) = \frac{v_1 + v_2}{2},$$

where η is a Gaussian random variable of zero average and standard deviation 1. The Heun algorithm, which is a fairly widely used algorithm for the integration of generic stochastic equations, does not make use of the (quasi-) symplectic nature of the flow; we expect that it will not fare too well in the limit of small γ. We recall that it is known [5] that the equilibrium distribution generated by the Heun algorithm is accurate up to $o(h^2)$ in the exponent. We will refer to the Heun scheme as "He."

3. Quasi-symplectic algorithms. A symplectic algorithm is a numerical scheme which aims to preserve the 2-forms $dq_i \times dp_i$ during the integration of a Hamiltonian flow. The quantity q_i is a generalized coordinate, and p_i is the corresponding conjugate momentum. A nice introduction to the symplectic integration

can be found in [20, 21]; in particular, we will use one of the results of [20] in what follows. Given the Hamiltonian $H(q_i, p_i)$ and the equations of motion

$$\dot{q}_i = \{q_i, H\}, \qquad \dot{p}_i = \{p_i, H\},$$

a symplectic integrator will conserve some quantity $\hat{H}$, which in general reads

$$\hat{H} = H + h^n G(p_i, q_i),$$

where h is the integration time step, and G is a function which depends on the numerical scheme (read the symplectic scheme) used for the integration. The problem of Hamiltonian flows in the presence of fluctuations has been addressed also in [14, 15], whereas quasi-symplectic schemes were derived in [16] (see also below, where various comparisons are carried out).

Given that we are interested here in the integration of (1.1), the starting point will be the symplectic integration of Hamiltonians which are separable and quadratic in the velocities. There are very many different possible symplectic schemes; however, having in mind that we are seeking a scheme which should be used in the integration of a SDE, we restrict ourselves to considering schemes in the form

$$\begin{aligned} p(i) &= p(i-1) + hb_i F(q(i-1)), \\ q(i) &= q(i-1) + ha_i p(i) \end{aligned} \tag{3.1}$$

for i between 1 and N, where $q(0) = q(t = 0)$, $q(N) = q(t = h)$, etc., and h is the integration time step in the simulations. The coefficients $a(i)$ and $b(i)$ are chosen as to minimize, in some sense, the quantity $G(p, q)$.

The lowest possible symplectic algorithm one can write to integrate (1.1) following this approach reads, when $\gamma = T = 0$ (this scheme is also known as "leap frog"),

$$\begin{aligned} \tilde{x} &= x(t) + \frac{h}{2} v(t), \\ v(t+h) &= v(t) + hF(\tilde{x}), \\ x(t+h) &= \tilde{x} + \frac{h}{2} v(t+h), \end{aligned} \tag{3.2}$$

where x is the position and v is the velocity. This scheme conserves the quantity $H - h^3(vFF' + v^3F''/6)/4$, where $H \equiv v^2/2 + V(x)$. Higher order schemes can be found in the quoted references [20, 21]. Paralleling the procedure of [17] (see also [22]), it is then possible to reintroduce both the dissipation $(-\gamma v)$ and the noise, writing the tentative scheme

$$\begin{aligned} \tilde{x} &= x(t) + \frac{h}{2} v(t), \\ v(t+h) &= c_2 \left[c_1 v(t) + hF(\tilde{x}) + d_1 \eta \right], \\ x(t+h) &= \tilde{x} + \frac{h}{2} v(t+h), \end{aligned} \tag{3.3}$$

where η is a Gaussian variable with standard deviation one and average zero. We use again (2.2), and, imposing that the term $O(h)$ in the exponent (i.e., the term hS_1) vanishes, we find that the unknown arbitrary quantities c_1, c_2, and d_1 read

$$\begin{aligned} c_1 &= 1 - \frac{\gamma h}{2}, \\ c_2 &= \frac{1}{1 + \gamma h/2}, \\ d_1 &= \sqrt{2T\gamma h}. \end{aligned} \tag{3.4}$$

Although we will carry out more extensive comparisons further down, let us briefly compare this scheme to the scheme of (2.4). The present scheme is by construction well behaved in the limit of $\gamma \to 0$; it has the same accuracy in computing the equilibrium distribution as (2.4), but it requires only one random deviate per integration time step (as opposed to two deviates for the scheme of (2.4)), so it will run faster. In the following, we will refer to the algorithm of (3.3) as symplectic low order (SLO).

Looking at the structure of the previous algorithm, we can try to derive an algorithm of higher order. In the derivation of (3.3), when we applied (2.2), given the number of unknown quantities we could only impose that the term $O(h)$ in the exponent disappeared. If we could somehow increase the number of unknown quantities when applying (2.2), while keeping the algorithm simple, we might be able to make the terms $O(h^2)$ in the exponent disappear. We start combining two steps, each one of the form of (3.3), done with an integration time step $h/2$,

$$\begin{aligned} x_1 &= x(t) + \frac{h}{4} v(t), \\ v_1 &= c_2 \left[c_1 v(t) + \frac{h}{2} F(x_1) + \sqrt{\gamma T h} (a_1 \eta_1 + a_2 \eta_2) \right], \\ x_2 &= x_1 + \frac{h}{2} v_1, \\ v(t+h) &= c_2 \left[c_1 v_1 + \frac{h}{2} F(x_2) + \sqrt{\gamma T h} (b_1 \eta_1 + b_2 \eta_2) \right], \\ x(t+h) &= x_2 + \frac{h}{4} v(t+h), \end{aligned} \tag{3.5}$$

where $c_1 = (1 - \gamma h/4)$ and $c_2 = 1/(1 + \gamma h/4)$. Here, η_1 and η_2 are two random Gaussian variables of standard deviation one and average zero. The idea is now to choose the coefficients a's and b's in such a way as to annihilate S_1, and possibly minimize S_2. This is done using (3.5) in (2.2), which results in a number of algebraic equations for a_i and b_i. The algebra, although straightforward, is cumbersome and we will simply report here the results. For a given a_1, the following choice for the other three parameters will ensure that S_1 vanishes identically:

$$b_1 = -\frac{a_1}{7} + \frac{2\sqrt{14 - 12a_1^2}}{7},$$

$$b_2 = -\frac{\sqrt{7 + 282a_1^2 + 24a_1\sqrt{14 - 16a_1^2}}}{\sqrt{42}},$$

$$a_2 = \frac{\sqrt{42} b_2 (-7\sqrt{2} + 6\sqrt{2}a_1^2 + 24a_1\sqrt{7 - 6a_1^2})}{\sqrt{3}(-14 + 588a_1^2)}.$$

As a function of a_1, we can now write the equations satisfied by S_2: we find that S_2 vanishes for a particular choice of the parameter a_1. Summarizing the numerics, the set of a's and b's which simultaneously makes S_1 and S_2 vanish are

$$\begin{aligned} a_1 &= -1.0691860043307065\ldots, \\ a_2 &= -0.1533230407019893\ldots, \\ b_1 &= 0.3044913128854065\ldots, \\ b_2 &= -1.0363164126095790\ldots. \end{aligned} \tag{3.6}$$

Note that there is a symmetry: quantities with indexes 1 and 2 can be exchanged. The conclusion is that the algorithm given by (3.5) and (3.6) is symplectic in the limit $\gamma, T \to 0$ (conserving the quantity $H - h^3(vFF' + v^3F''/6)/16$), whereas for finite γ and T it reproduces the correct equilibrium distribution with an error $O(h^3)$ in the exponent. We will refer to this algorithm as to symplectic high order (SHO).

We want to compare our algorithms to the quasi-symplectic algorithms of [16]. We should mention here that really the latter are weak integration schemes (for a definition of weak and strong integration schemes, see [2]); hence they are bound to give worse results than the other schemes when, as we do, average quantities are considered, and in particular for mean first passage times. The two algorithms used in the numerical simulations of [16] integrate with the prescriptions ([16] should be consulted for more details and for higher order algorithms; here we are using only the ones explicitly considered by the authors of [16]):

MT1:

$$\begin{aligned} x(t+h) &= x(t) + hv(t+h)\,, \\ v(t+h) &= v(t) - hV'(x(t+h)) - h\gamma v(t+h) \\ &\quad + \sqrt{2Th\gamma}\eta\,, \end{aligned} \tag{3.7}$$

where $v(t+h)$ and $x(t+h)$ should be found recursively.

MT2:

$$\begin{aligned} v(t+h) &= (1-\gamma h)(v(t) - hV'(x(t)) + \sqrt{2Th\gamma}\eta), \\ x(t+h) &= x(t) + h(v(t) - hV'(x(t))). \end{aligned} \tag{3.8}$$

The random variables η take the values ± 1. These variables are faster to generate than a Gaussian variable, hence these algorithms will run faster, allowing for a smaller integration time step to compensate for less accuracy when averaged quantities are considered. However, having said this, if we used (2.2) to assess these two algorithms, we would find that they both have a correction to the equilibrium distribution $O(h)$ in the exponent. These will reflect in the numerical experiments, as we will comment below.

Finally, it should be noted that given the structure of the Hamiltonian in the limit $T \to 0$, $\gamma \to 0$, which is $H = v^2/2 + V(x)$, the equilibrium distribution for (1.1) can also be written as

$$P(x, v) \propto \exp -H/T.$$

At first sight, it may appear that the request of a symplectic integration scheme is redundant, once we made sure that the "correct" equilibrium distribution is generated in the numerical integration. This is not right: the limit $T \to 0$ is singular, hence a symplectic form for the numerical scheme when $T = 0$ can (and should) be imposed as an additional condition.

We proceed now to compare the algorithms taking as benchmarks two very simple dynamical systems. We consider first how the different algorithms reproduce the equilibrium properties: we integrate the equations of motion and compute some momenta. For the systems we have in mind, it is possible to derive theoretically the momenta considered in the simulations, for a better comparison. The equilibrium distribution is computed from the integrated trajectories, sampled at time intervals larger than the typical timescales, which corresponds to $1/\gamma$ for the parameter ranges

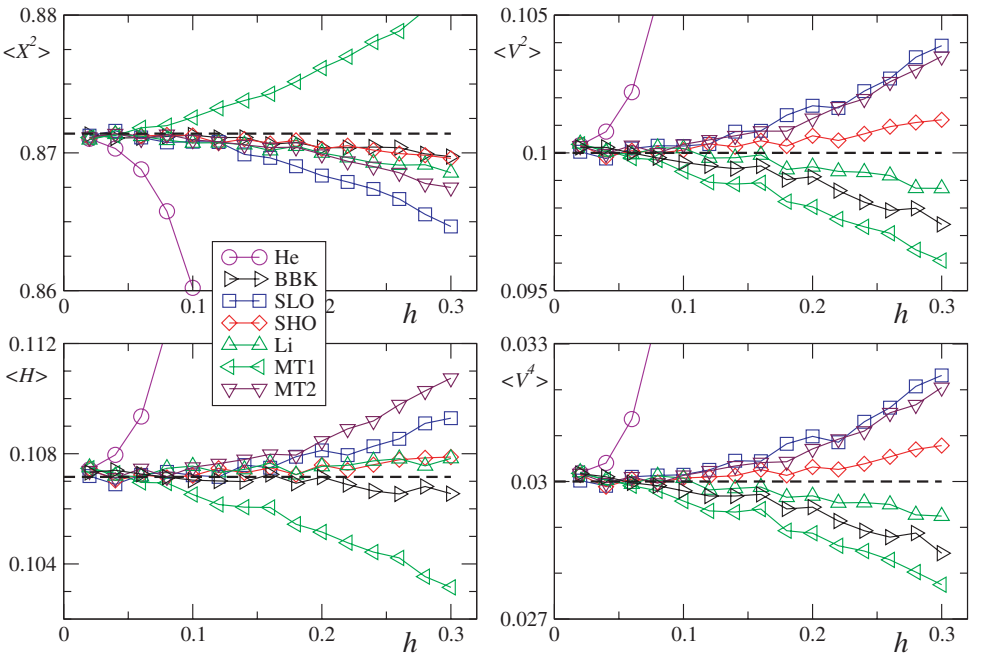


FIG. 3.1. *Result of simulations for different integration schemes, as a function of the integration time step h, for the system in* (3.9). *Various moments are considered (see text for details), taking* $T = 0.1$ *and* $\gamma = 10^{-2}$. *The number of averages is* 4×10^5.

considered. For the random number generation we used the algorithms of [23, 24, 25]. The systems considered will be

$$\dot{x} = v,$$
$$\dot{v} = -\gamma v - V'(x) + \sqrt{2\gamma T}\eta,$$

with the potentials

$$V(x) = x^4/4 - x^2/2 \tag{3.9}$$

and

$$V(x) = x^4/4 + x^2/2. \tag{3.10}$$

The first case is a double well anharmonic oscillator with minima located at $x \pm 1$, and the second one is a single well oscillator, with a minimum located at $x = 0$. In the simulations, $T = 0.1$. The results of the simulations are summarized in Figures 3.1 and 3.2, respectively. In both figures, the results of the digital simulations are shown by symbols with a grey line as a guide for the eye; the bold dashed line is the (expected) theoretical value for the quantity considered. For the number of averages considered, the statistical error is of the order of the symbol size. Also, $\langle H \rangle \equiv \langle v^2/2 + V(x) \rangle$.

Let us comment on the results. It is evident that the Heun method (He in figures) is not very appropriate, given the small dampings considered, and it is outperformed by the SLO scheme (3.3); note that the SLO is somewhat faster than He, given that it requires only one evaluation of the deterministic force for each integration time step.

The algorithms MT1 and MT2 are reasonably accurate; it should be noted that MT2 seems to be more accurate than MT1 for a given integration time step. Considering that MT2 is much faster than MT1, the conclusion seems to be that MT2

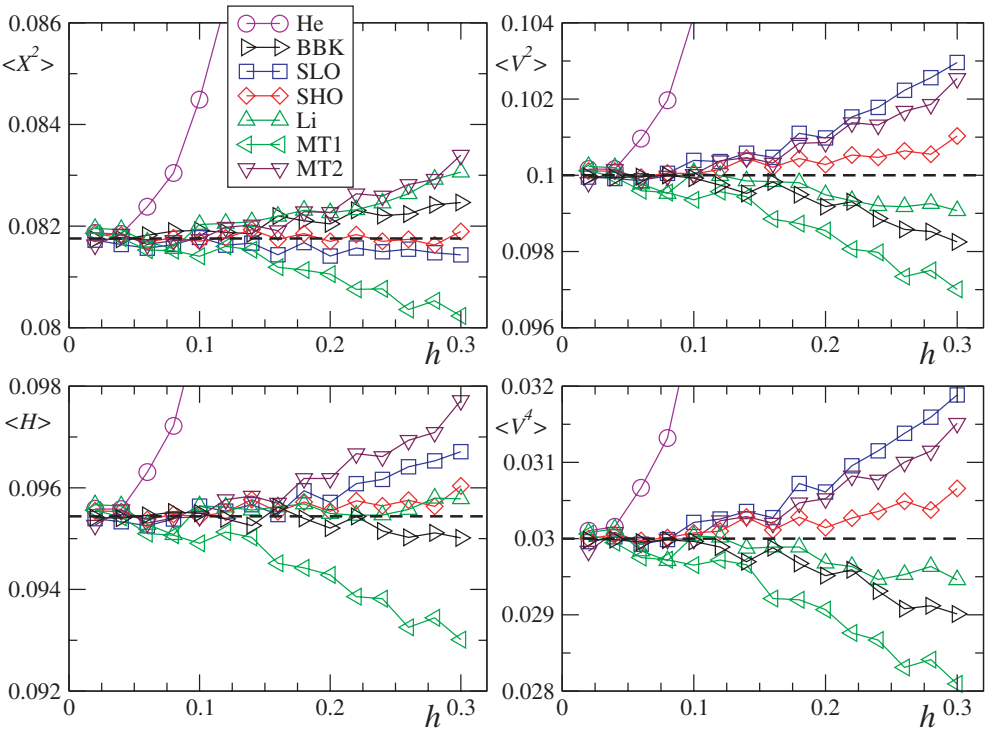


FIG. 3.2. *Result of simulations for different integration schemes, as a function of the integration time step h, for the system in* (3.10). *Various moments are considered (see text for details), taking* $T = 0.1$ *and* $\gamma = 10^{-2}$. *The number of averages is* 4×10^5.

ought to be preferred between these two schemes. Note also that the error on the moments for these two schemes seems to grow linearly with the integration time step h, which is related to the $O(h)$ error in the exponent which was mentioned in the previous section.

The Li and BBK algorithms are less accurate than SHO when the x^2 moments are considered. The SHO is the algorithm which gives the most accurate results for the x^2 moment, with results comparable to or better than those obtained with Li of BBK for the v^2 and v^4 moments. It is only when $\langle H \rangle$ is considered that Li and BBK seem to be more accurate than SHO. However, care is necessary in drawing conclusions from $\langle H \rangle$: looking, for instance, at Figure 3.1, we note that Li and BBK underestimate both v^2 and x^2 (in particular, of these three algorithms, BBK shows a fairly large departure from the correct $\langle v^2 \rangle$). Recalling the structure of the potential $V(x) = -x^2/2 + x^4/4$, it is clear that these two underestimates tend to cancel out, leading to a $\langle H \rangle$ closer to the theoretical one, but only by virtue of a coincidental cancellation.

It is also interesting and important to assess some dynamical quantities: similar calculations, for instance, would be carried out in multistable systems to compute escape rates. In most cases these quantities are not known theoretically, and stochastic simulations are one of the tools to determine them. Using the system of (3.9), we computed the mean first passage times (MFPTs) to go from one of the minima to the other one (the minima are located at $x = \pm 1$). The results of the simulations are summarized in Figure 3.3, after averaging a suitable number of passage times between

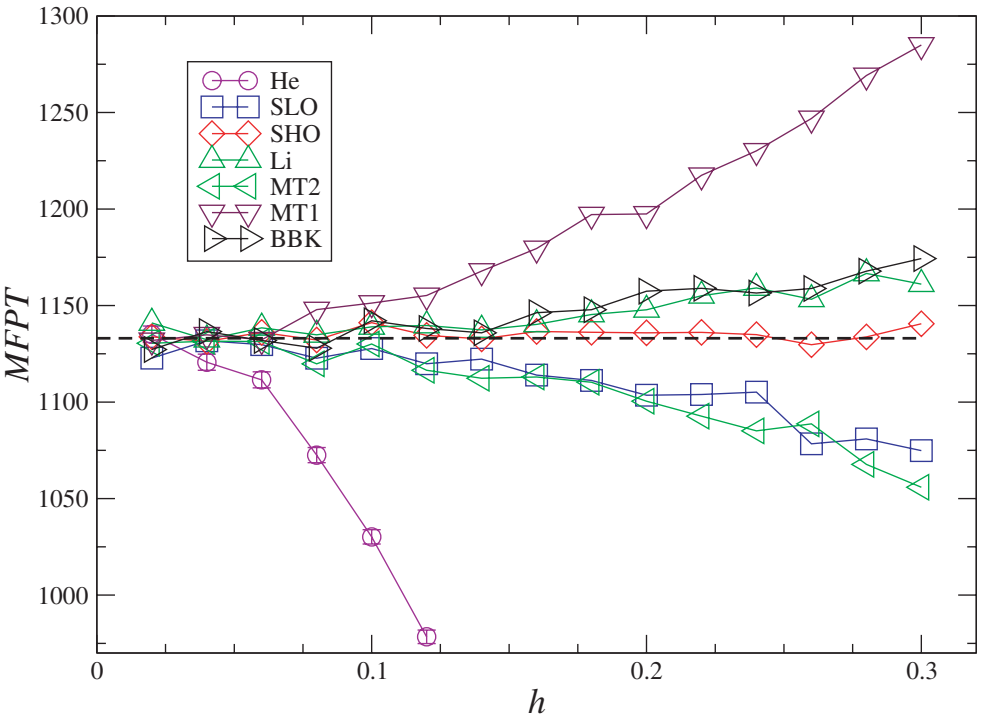


FIG. 3.3. *Result of simulations for different integration schemes, as a function of the integration time step h, for the system in* (3.9). *The MFPT between the minima is considered (see text for details), taking $T = 0.1$ and $\gamma = 10^{-2}$. The number of averages used to compute each point in the figure is in excess of 7×10^3.*

the minima. In the caption we quote a lower bound on the number of averages taken; each point was actually computed using a slightly different number of averages. Given the fairly large value of the noise intensity ($T = 0.1$), there is no theory available to compute an "exact" MFPT for comparison with the simulated MFPTs. To have some reference, we took as reference value the average of the MFPTs obtained via the algorithms Li and SHO for the three smallest h values used in the simulations and drew the dashed line at this value. While showing an error which grows only quadratically in h, clearly He is the algorithm which performs worst. MT1 and MT2 give more reasonable results, and they yield MFPTs comparable to the ones obtained using SLO. SHO outperforms Li and BBK, giving MFPTs closer to the "exact" ones over the whole h range considered. Li and BBK, while close to each other, on the other hand, seem to give results which are roughly equivalent to the ones obtained using MT2 or SLO. For all simulations the statistical error due to the sample finiteness is of the size of the symbols used.

4. Higher order quasi-symplectic schemes Nosé–Hoover dynamics. In real situations, the SDEs one has to integrate are typically multidimensional, with (very) different timescales; there are also typically many Brownian walkers interacting with each other. The quasi-symplectic schemes derived above, in the deterministic frictionless limit, are relatively low order algorithms. As such, they might have problems (read they might require a too small integration time step) when used for the integration of the dynamics of real systems. The solution could be to use symplectic

TABLE 4.1
Coefficients used in our quasi-symplectic fourth order integrator.

i	a_i	b_i
1	0.515 352 837 431 123	0.134 496 199 277 431
2	-0.085 782 019 412 974	-0.224 819 803 079 421
3	0.441 583 023 616 467	0.756 320 000 515 668
4	0.128 846 158 365 384	0.334 003 603 286 321

schemes which, in the deterministic frictionless limit, are of higher order than the leap frog algorithm considered above and which are the building block of SLO and SHO.

Indeed, it is possible to obtain quasi-symplectic algorithms of arbitrarily high order in the frictionless deterministic limit which reproduce the correct equilibrium distribution to $O(h)$ in the exponent (i.e., at the same order of SLO above). Given a generic deterministic symplectic scheme of order N (see (3.1)), there is a corresponding quasi-symplectic algorithm which reads

$$\begin{aligned} p(i) &= c_{2,i}\left[c_{1,i}p(i-1) + b_i\left(hF(q(i-1)) + \eta\right)\right], \\ q(i) &= q(i-1) + ha_i p(i), \end{aligned} \tag{4.1}$$

$i = 1, N$, where

$$c_{1,i} \equiv 1 - \frac{h}{2}\gamma b_i, \qquad c_{2,i} \equiv \frac{1}{1 + \gamma h b_i/2}$$

and where η is a random Gaussian variable with average zero and standard deviation $2T\gamma h$. It is important to stress that the variable η is the *same* for each of the steps' i's within the integration time step h. Of course, η will be changed once the time h is reached, and so on. The statement that the algorithm of (4.1) has the correct frictionless deterministic properties follows trivially taking the limit $\gamma, T \to 0$ in the scheme; the stochastic properties, i.e., the fact that it reproduces the correct equilibrium distribution at $o(h^2)$ in the exponent, follow by direct substitution in (2.2) and after carrying out the necessary algebra. It should be appreciated that this implies that better quasi-symplectic algorithms could be obtained from the research on symplectic schemes. In the following, besides SLO we will also use a fourth order quasi-symplectic scheme for which we took the coefficients summarized in Table 4.1 (from one of the fourth order symplectic schemes proposed in [20]).

We turn now to possible realistic applications of the algorithms discussed so far. An important class of simulations described by (1.1) where quasi-symplectic schemes could be used is molecular dynamics simulations of many interacting particles, done keeping fixed either temperature and volume or temperature and pressure throughout the simulation. The systems simulated in this way are relevant for studying chemical kinetics.

As we mentioned in the introduction, the evolution given by (1.1) corresponds to the evolution of a system in contact with a thermostat. The evolution of dynamical systems in contact with a thermostat has been extensively studied via the Nosé–Hoover dynamics, rather than via integration of (1.1). This means that if we were interested in the integration of (1.1), we could also proceed replacing the true dynamics with the Nosé–Hoover one, integrating the latter rather than directly (1.1). We refer the interested reader to [26, 27, 28, 29, 30] for a deeper discussion of the Nosé–Hoover approach (see also [6]); here, we will briefly recall how a simulation for constant temperature and volume is typically done using the Nosé–Hoover dynamics.

Starting with the N particles Hamiltonian ($m = 1$)

$$H(\vec{p}, \vec{x}) = \sum_{i=1}^{N} \vec{p}_i^2/2 + V(\vec{x}),$$

the Nosé–Hoover dynamics prescribes that the equations of motion one should integrate are

$$\begin{aligned} \ddot{x}_i &= -z\dot{x}_i - \frac{\partial V(\vec{x})}{\partial x_i}, \\ \dot{z} &= \sum_{i}^{N} \dot{x}_i^2 - NT, \end{aligned} \tag{4.2}$$

where T is the temperature which is meant to stay constant, and z is an auxiliary variable. From the structure of the above equations, it is clear that when the kinetic energy of the system is larger than $NT/2$, z increases: this means that the particles become more and more damped. On the other hand, when the kinetic energy is smaller than $NT/2$, z decreases, to the point that it could be negative; when negative, it acts as a pushing force on the particles, increasing their energy. The finite volume is implemented through the boundary conditions. The idea behind the Nosé–Hoover dynamics is a clever one: it should be appreciated that it does not require any stochastic force. Hence one has available, while integrating (4.2), all the theoretical results and the numerical tricks normally used in the integration of ordinary differential equations. The Nosé–Hoover dynamics is of course not "real": in fact, it should be used only to reach thermal equilibrium, at which point the simulations should proceed without any reference to z.

The quasi-symplectic algorithms introduced here are a possible, very interesting alternative to the Nosé–Hoover dynamics. The requirement of constant temperature is in fact automatically built into the quasi-symplectic structure of the equations. Even more important, one could ask the question as to what is responsible for the evolution towards thermal equilibrium. Typically, this equilibrium is reached thanks also to the coupling with the thermal bath, which does not simply give rise to temperature (fluctuations), but also to some dissipation (damping). And if some guesses are available as to the strength and speed of these processes, the quasi-symplectic algorithms would be capable of not simply generating the statics (equilibrium) but also of tracking the actual relaxation towards equilibrium.

We also note that the Nosé–Hoover dynamics controls *only* the overall energy of the system, relying on the internal dynamics to get equilibrium. When this internal dynamics is not strong enough (for instance, in the case of diluted particles) we might even experience problems. We now turn again to some numerical experiments.

The system looked at is an ensemble of particles interacting pairwise via a Lennard–Jones (LJ) potential

$$V(r) = 4\epsilon \left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6} \right],$$

where r is the distance between two particles. The simulations were done in three dimensions. We recall that the minimum of the LJ potential is located at

$$r_{LJ} = \sigma 2^{1/6}.$$

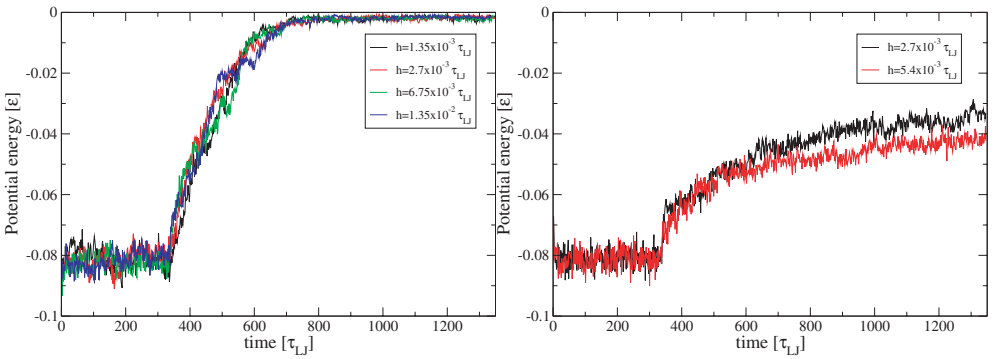


FIG. 4.1. *Result of simulations for different integration time step h's for* 125 *particles interacting via a LJ potential. The quantity plotted is the average potential energy vs. time (see text for details about the simulation); $T = 0.8\epsilon$. Left: SLO scheme, $1/\gamma = 100\tau_{LJ}$. Right: Nosé–Hoover dynamics.*

We took the mass of the particles to be 40. In the following, times will be scaled on τ_{LJ}, the period of small oscillations around the LJ potential minimum for the given mass, and energies and distances will be measured in units of ϵ and σ, respectively. Roughly speaking, the chosen mass is Argon's: if the other LJ parameters were taken to be the Argon's ones, τ_{LJ} would be in the range of ps.

We took 125 particles and, for the first case, fixed the volume: particles hitting the boundaries were reflected assuming elastic scattering from a rigid wall. Typically, the system was prepared in a small cubic box with sides equal to $6r_{LJ}$. After waiting until equilibrium was reached, the box was suddenly deformed, keeping constant the size along z but increasing the size along x and y to be $60r_{LJ}$. For the sake of comparison, we decided to look at the average potential energy in our system. For these simulations, we superimposed a static potential of the form

$$V(x, y) = \epsilon \left[\cos(x/\sigma) + \cos(y/\sigma)\right], \tag{4.3}$$

where $\{x, y, z\}$ is the coordinate of each particle, with the superimposed potential slowing down the x, y diffusion. The result of the simulations for $T = 0.8\epsilon$ and different time steps is shown in Figure 4.1. The system is initially allowed to relax within the small box. Then at times $337.5\tau_{LJ}$ the box is suddenly increased as described above, and the potential energy monitored. On the left plate of Figure 4.1 we show the result of using a quasi-symplectic SLO scheme: we observe no dependence on the integration time step h in the relaxation towards equilibrium. The equilibrium is reached around $800\tau_{LJ}$, and, in case we were interested in computing equilibrium quantities, from that time onward we would be in position to start the averaging.

It is striking to compare this with the results obtained integrating using the Nosé–Hoover dynamics (right plate of Figure 4.1). We observe that (a) the relaxation depends on the integration time step, and (b) even waiting for times as large as $1300\tau_{LJ}$, we are still very far from the equilibrium state. Hence, we are still not in position to start averaging over the equilibrium distribution. The Nosé–Hoover dynamics was integrated using a fourth order Adams–Bashforth predictor, followed by a fourth order Adams–Moulton corrector.

There is no doubt that the relaxation towards equilibrium in the SLO integration is dictated by the value of γ. But as mentioned, at worst the SLO can be used as the Nosé–Hoover dynamics is used—to reach the new equilibrium state—at which

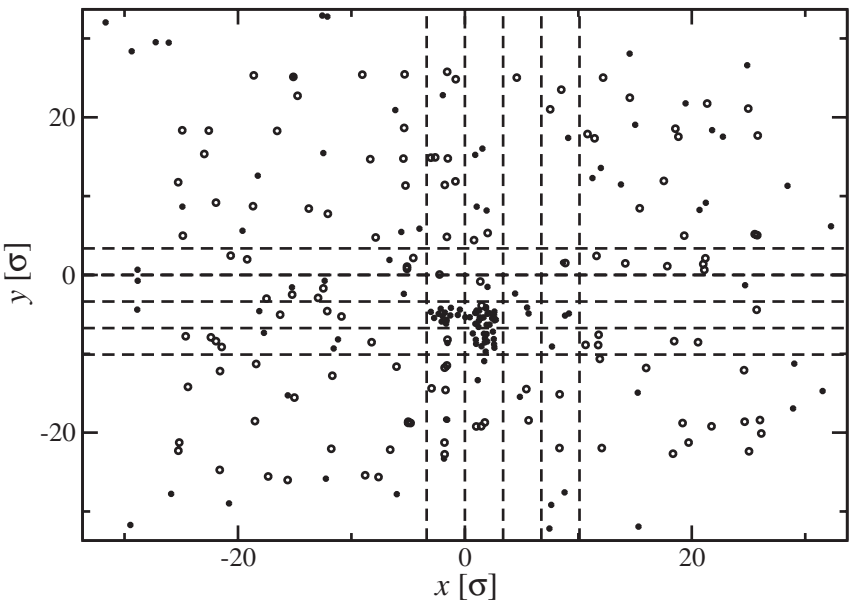


FIG. 4.2. *Position of* 125 *particles interacting through a LJ potential projected onto the* x, y *plane. The projection was done (see Figure* 4.1*) at times equal to* $1200\tau_{LJ}$*. The symbols are the particles (full circles, Nosé–Hoover dynamics; empty circles, quasi-symplectic algorithm); the dashed lines mark some of the maxima of the periodic potential of* (4.3) *lying closer to the origin.* $h = 2.7 \times 10^{-3}\tau_{LJ}$.

point nothing prevents us from simply switching off the damping, taking $\gamma = 0$: we will be left with a symplectic integrator and an evolving system where averages can be taken. But, on the other hand, if γ is chosen to resemble the actual interactions between particles and thermal bath, we would be simulating even the off equilibrium properties (i.e., the relaxation towards equilibrium), something which is not possible with the Nosé–Hoover dynamics.

The lack of equilibrium in the Nosé–Hoover dynamics is easily understood looking at the position of the particles at larger times (Figure 4.2). In the Nosé–Hoover dynamics even for the larger times considered many particles are stuck at the bottom of the potential minima. For the configuration in Figure 4.2 we have that $\sum_i p_i^2/2 \approx NT/2$, and $z \approx 0$ in (4.2). This implies that the only possibility to increase the energy of the low energy particles (the ones in the wells) is via scattering with the particles already outside, a process which is clearly inefficient given that the typical distances over which the LJ acts are around σ (i.e., of the order of the distance between the dashed lines in Figure 4.2).

We note, furthermore, that the SLO, requiring only one evaluation of the deterministic force per integration time step, is approximatively twice as fast as methods based on a predictor corrector, for the same time step. Also, in practice we observed that we could not increase the integration time step in the Nosé–Hoover integration much further than what is shown in the legend of Figure 4.1: when we tried to make the integration time step larger, the integration broke down. To summarize, SLO proved to be faster and allowed for larger integration time steps.

We mentioned that the Nosé–Hoover dynamics is used also for the integration at fixed temperature and pressure. It is possible to carry out similar integrations using the quasi-symplectic schemes proposed here. The approach to follow is to also actually

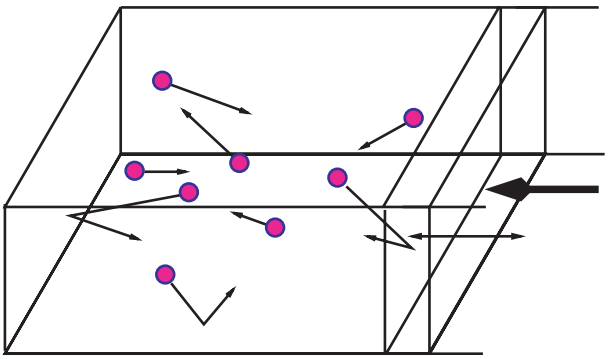


FIG. 4.3. *Scheme of how simulations at constant pressure and temperature could be carried out with the quasi-symplectic schemes. The dynamical system is enlarged to include a moving set (shown on the right) with an applied external constant force (the bold arrow on the right). The other walls are rigid, and particles scatter elastically from them.*

simulate what should keep the pressure constant, like a moving set with a constant applied force. It will be the scattering with the moving set which will yield the constant pressure. This is exemplified in Figure 4.3: the figure shows some interacting particles, the moving set (on the right), and the applied external constant force.

We tested the algorithms, simulating both single particles and dymers (couples of elastically bounded particles), interacting via a LJ potential, in three dimensions. The integration was carried out using a fourth order quasi-symplectic algorithm. Before turning to a quantitative comparison, we think that it could be interesting for the reader to look at some short movies showing the dynamics generated. The movies, available at http://www.df.unipi.it/~mannella/simulations.html, show clearly the formation of droplets for smaller temperatures and the evaporation of the droplets as the temperature is increased; they also show the moving set interacting with the particles. We found that in practice rather than using a hard moving set, it proved numerically simpler to have a "soft" moving set. In the simulations, assuming that the moving set closes the box from the right (like in the movies and in Figure 4.3), we took an interacting potential between the moving set and particles of the form

$$V(x_s, x_p) = \theta(x_p - x_s)(x_p - x_s)\frac{\epsilon}{\sigma},$$

where x_s and x_p are the x coordinates of, respectively, the moving set and of a particle, and $\theta(x)$ is 0 (1) for $x < 0$ (> 0). This leads to a smooth transition into the region where the moving set is, making the numerical integration simpler.

We turn now to some quantitative assessment of the quasi-symplectic algorithms for the integration at fixed temperature and pressure. The simulations are again done in the three-dimensional case, using a soft moving set and taking 125 particles interacting via a pairwise LJ potential. The equations of motion were integrated in a box with the y size equal to $60r_{LJ}$ and the z size equal to $6r_{LJ}$. The moving set was free to move along the x direction.

In the first numerical experiment, we fixed the value of the temperature and, changing the force applied to the moving set, we computed the average distance between the moving set and the opposite wall, once equilibrium was reached: this is proportional to the volume taken by the particles. It should be appreciated that in our simulations the moving set does move around some average position, as shown in

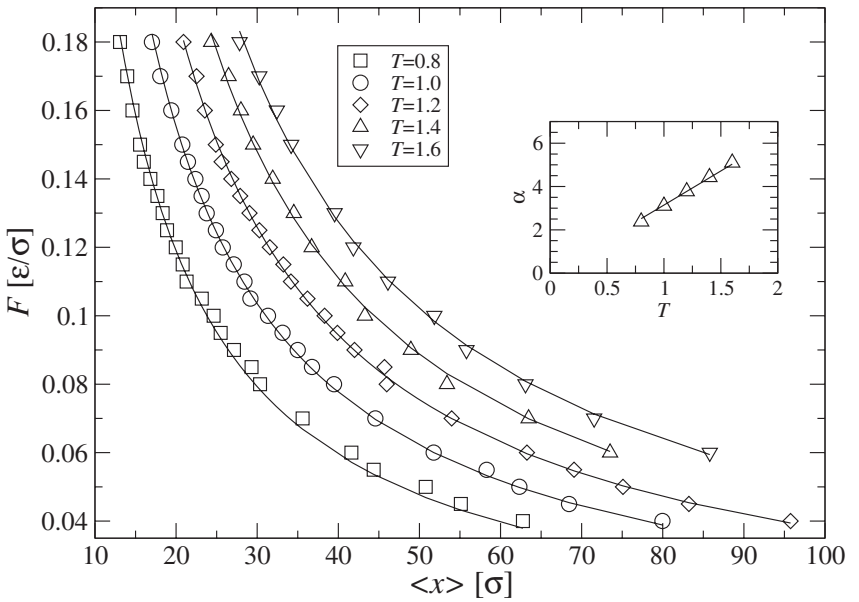


FIG. 4.4. *Relation between applied force F and average position of the moving set $\langle x\rangle$. The latter is proportional to the volume occupied by the* 125 *particles interacting via a LJ potential. Integration is done using a fourth order quasi-symplectic algorithm, with* $1/\gamma = 10^4\tau_{LJ}$ *and* $h = 1.35\times10^{-2}\tau_{LJ}$. *The symbols are simulations; the solid line is best fit with* $F = \alpha/\langle x\rangle$, *where* α *is the (only) fitting parameter. Inset, relation between* α *and temperature* T: *symbols are obtained from the fit; the solid line is the result of a linear fit using the function* $\alpha = \beta T$, *with* β *fitting parameter.*

the movies. We recall that there is proportionality between the force and the pressure felt by the particles, of the form, for our geometry,

$$P[\epsilon/\sigma^3] = F[\epsilon/\sigma]/(360 \times 2^{1/3}\sigma^2).$$

We should recover Boyle's law for ideal gases in the simulations. The result of the numerics is summarized in Figure 4.4. The symbols are the results for different temperatures, showing the relation between applied force, F, and average position of the set, $\langle x\rangle$. The solid lines are fits done with the function $F = \alpha/\langle x\rangle$, where α is the fitting parameter. The functional form of the fit is obviously suggested by Boyle's law. We note that the simulations follow closely the fitting formula. The inset of Figure 4.4 shows a plot of α as a function of the system temperature: if Boyle's law applied, we should recover a linear dependence of α on T, which is indeed the case.

Given the attractive nature of the LJ for a range of distances, we expect that for high enough pressures and/or small temperatures we should observe a phase transition to the liquid state. This is indeed the case, as seen in Figure 4.5, where we plotted the volume taken by the particles as a function of temperature. We applied a constant force $F = 0.1\epsilon/\sigma$, which would imply a pressure of $P = 1/(3600 \times 2^{1/3})\epsilon/\sigma^3$, and changed temperature. For larger temperatures, we again observe Boyle's law: the solid line in the figure is a fit done with a linear dependence of volume on temperature. However, as we decreased the temperature, we observed a sharp drop in the volume occupied by the particles. By direct inspection of the state reached by the particles in the simulations, we observed the formation of a droplet with all particles attached together—the transition to the liquid state.

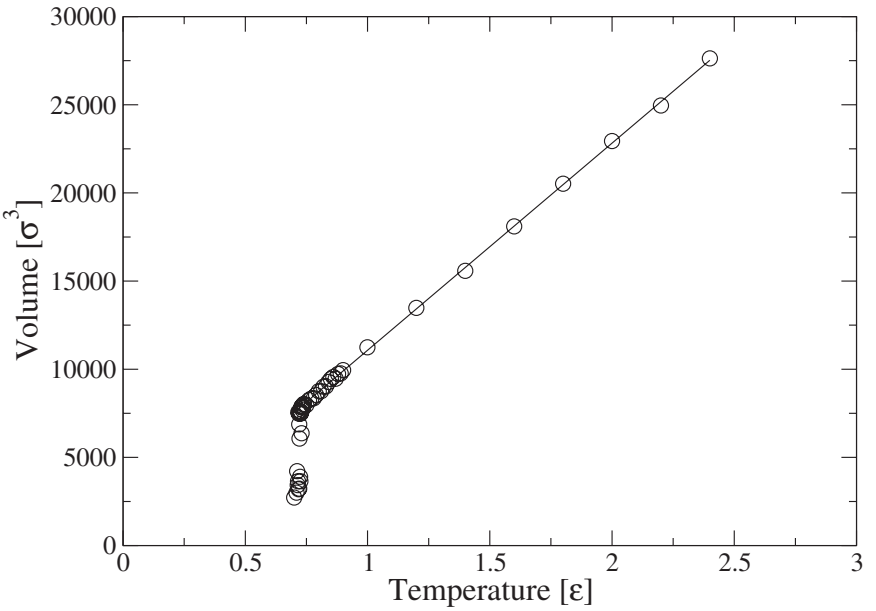


FIG. 4.5. *Average volume occupied by* 125 *particles interacting via a LJ potential for different temperatures and constant pressure. Integration is done using a fourth order quasi-symplectic algorithm, with* $1/\gamma = 10^4 \tau_{LJ}$ *and* $h = 1.35 \times 10^{-2} \tau_{LJ}$. *The symbols are simulations; the solid line is best linear fit.*

We conclude this section noting that indeed the quasi-symplectic algorithms are capable of correctly integrating the dynamics of systems for which we are required to keep either temperature and volume or temperature and pressure constant.

5. Conclusions. We introduced quasi-symplectic algorithms for the numerical integration of Langevin equations obeying detailed balance. The features of these algorithms are that they become symplectic when the damping and the temperature in the Langevin equation are taken to be zero, and give the correct equilibrium distribution to some higher order in the integration time steps for a finite damping and temperature. This, in turn, leads to more accuracy when dynamical quantities are considered (like the MFPT) than when leading existing integration schemes are used. We showed that possible applications of these algorithms are in the integration of the dynamics in the liquid state, where the speed up provided by algorithms which are stable for fairly large time steps may help in improving current simulations. The algorithms make it possible to perform simulations for both constant temperature and volume and constant temperature and pressure. They appear to be a viable alternative to approaches like the Nosé–Hoover dynamics, currently the leading method used to handle the simulation of systems in contact with a thermal bath. Finally, the approach followed here can be used to derive higher order quasi-symplectic schemes by simply exploring the possibilities of symplectic schemes.

REFERENCES

[1] F. MOSS AND P. V. E. MCCLINTOCK, EDS., *Noise in Nonlinear Dynamical Systems*, Vols. 1–3, Cambridge University Press, Cambridge, UK, 1989.

[2] P. E. KLOEDEN AND E. PLATEN, *Numerical Simulation of Stochastic Differential Equations*, Springer-Verlag, Berlin, 1992.

[3] J. Garcia-Ojalvo and J. Sancho, *Noise in Spatially Extended Systems*, Springer-Verlag, Berlin, 1999.
[4] G. N. Milstein and M. V. Tret'yakov, *Numerical methods in the weak sense for stochastic differential equations with small noise*, SIAM J. Numer. Anal., 34 (1997), pp. 2142–2167.
[5] R. Mannella, *Integration of stochastic differential equations on a computer*, Internat. J. Modern Phys. C, 13 (2002), pp. 1177–1194.
[6] M. P. Allen and D. J. Tildesley, *Computer Simulation of Liquids*, Oxford University Press, Oxford, UK, 2001.
[7] A. C. Brańka and D. M. Heyes, *Algorithms for Brownian dynamics computer simulations: Multivariable case*, Phys. Rev. E (3), 60 (1999), pp. 2381–2388.
[8] H. A. Forbert and S. A. Chin, *Fourth order algorithms for solving the multivariable Langevin equation and the Kramers equation*, Phys. Rev. E (3), 63 (2001), 016703.
[9] J. Qiang and S. Habib, *Second-order stochastic leapfrog algorithm for multiplicative noise Brownian motion*, Phys. Rev. E (3), 62 (2000), pp. 7430–7437.
[10] W. Paul and D. Y. Yoon, *Stochastic phase space dynamics with constraints for molecular systems*, Phys. Rev. E (3), 52 (1995), pp. 2076–2083.
[11] J. A. Izaguirre, D. P. Catarello, J. M. Wozniak, and R. D. Skeel, *Langevin stabilization of molecular dynamics*, J. Chem. Phys., 114 (2001), pp. 2090–2098.
[12] R. D. Skeel and J. A. Izaguirre, *An impulse integrator for Langevin dynamics*, Mol. Phys., 100 (2002), pp. 3885–3891.
[13] W. Wang and R. D. Skeel, *Analysis of a few numerical integration methods for the Langevin equation*, Mol. Phys., 101 (2003), pp. 2149–2156.
[14] G. N. Milstein and M. V. Tretyakov, *Numerical Methods for Langevin Type Equations Based on Symplectic Integrators*, Preprint 727, Weierstrab-Institut für Angewandte Analysis und Stochastik, Berlin, 2002.
[15] G. N. Milstein, Yu. M. Repin, and M. V. Tretyakov, *Numerical methods for stochastic systems preserving symplectic structure*, SIAM J. Numer. Anal., 40 (2002), pp. 1583–1604.
[16] G. N. Milstein and M. V. Tretyakov, *Quasi-symplectic methods for Langevin-type equations*, IMA J. Numer. Anal., 23 (2003), pp. 593–626.
[17] R. Mannella, *Quasisymplectic integrators for stochastic differential equations*, Phys. Rev. E (3), 69 (2004), 041107.
[18] M. Freidlin and A. D. Wentzell, *Random Perturbations in Dynamical Systems*, Springer-Verlag, New York, 1984.
[19] G. G. Batrouni, G. R. Katz, A. S. Kronfeld, G. P. Lepage, B. Svetitsky, and K. G. Wilson, *Langevin simulations of lattice field theories*, Phys. Rev. D (3), 32 (1985), pp. 2736–2747.
[20] H. Yoshida, *Construction of higher order symplectic integrators*, Phys. Lett. A, 150 (1990), pp. 262–268.
[21] J. M. Sanz-Serna, *Symplectic integrators for Hamiltonian problems: An overview*, Acta Numer., 1 (1992), pp. 243–286.
[22] R. Mannella, *Numerical integration of stochastic differential equations*, in Supercomputation in Nonlinear and Disordered Systems, L. Vasquez, F. Tirado, and I. Martin, eds., World Scientific, River Edge, NJ, 1997, pp. 100–129.
[23] C. M. Luscher, *A portable high-quality random number generator for lattice field theory simulations*, Comput. Phys. Comm., 79 (1994), pp. 100–110.
[24] F. James, *RANLUX: A FORTRAN implementation of the high-quality pseudorandom number generator of Luscher*. Comput. Phys. Comm., 79 (1994), pp. 111–114.
[25] G. Marsaglia and W. W. Tsang, *A fast, easily implemented method for sampling from decreasing or symmetric unimodal density functions*, SIAM J. Sci. Statist. Comput., 5 (1984), pp. 349–359.
[26] S. Nosé and M. Klein, *Constant pressure molecular dynamics for molecular systems*, Mol. Phys., 50 (1983), pp. 1055–1076.
[27] S. Nosé, *A molecular dynamics method for simulations in the canonical ensemble*, Mol. Phys., 52 (1984), pp. 255–268.
[28] W. Hoover, *Canonical dynamics: Equilibrium phase-space distributions*, Phys. Rev. A (3), 31 (1985), pp. 1695–1697.
[29] S. Nosé, *A unified formulation of the constant temperature molecular dynamics methods*, J. Chem. Phys., 81 (1984), pp. 511–519.
[30] M. Parrinello and A. Rahman, *Crystal structure and pair potentials: A molecular-dynamics study*, Phys. Rev. Lett., 45 (1980), pp. 1196–1199.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2140–2166
© 2006 Society for Industrial and Applied Mathematics

RECYCLING SUBSPACE INFORMATION FOR DIFFUSE OPTICAL TOMOGRAPHY*

MISHA E. KILMER† AND ERIC DE STURLER‡

Abstract. We discuss the efficient solution of a long sequence of slowly varying linear systems arising in computations for diffuse optical tomographic imaging.

The reconstruction of three-dimensional absorption and scattering information by matching computed solutions from a parameterized model to measured data leads to a nonlinear least squares problem that we solve using the Gauss–Newton method with a line search. This algorithm requires the solution of a long sequence of linear systems. Each choice of parameters in the nonlinear least squares algorithm results in a different matrix describing the optical properties of the medium. These matrices change slowly from one step to the next, but may change significantly over many steps. For each matrix we must solve a set of linear systems with multiple shifts and multiple right-hand sides.

For this problem, we derive strategies for recycling Krylov subspace information that exploit properties of the application and the nonlinear optimization algorithm to significantly reduce the total number of iterations over all linear systems. Furthermore, we introduce variants of GCRO that exploit symmetry and that allow simultaneous solution of multiple shifted systems using a single Krylov subspace in combination with recycling. Although we focus on a particular application and optimization algorithm, our approach is applicable generally to problems where sequences of linear systems must be solved. This may guide other researchers to exploit the opportunities of tunable solvers.

We provide results for two sets of numerical experiments to demonstrate the effectiveness of the resulting method.

Key words. Krylov subspace, GCRO, recycle, MINRES, eigenvalue, invariant subspace

AMS subject classifications. 65F10, 65N22

DOI. 10.1137/040610271

1. Introduction.

In diffuse optical tomography (DOT), data is obtained by transmitting near-infrared light into a highly absorbing and scattering medium and then recording the photon flux. The goal is to use the diffuse optical data measured on the surface to reconstruct three-dimensional (3D) images of the absorption and "reduced scattering" functions in the medium. In the case of breast tissue imaging, differences in the absorption and scattering may indicate the presence of a tumor or other anomaly.

The *forward problem* is the determination of synthetic data (photon flux) for given absorption and scattering functions from some mathematical model. A number of mathematical models have been proposed in the literature [2]. We focus on the frequency-domain diffusion model in which the data is a nonlinear function of the absorption and scattering functions. In order to solve the imaging problem—the determination of the absorption and reduced scattering functions—one must solve many instances of the forward problem. This fact implies a huge computational

*Received by the editors June 21, 2004; accepted for publication (in revised form) July 8, 2005; published electronically February 21, 2006. This work was supported by NSF grants 0208548 and 0139968 and CenSSIS (the Center for Subsurface Sensing and Imaging Systems) under the ERC Program of the NSF (EEC-9986821), grant DOE LLNL B341494, and grants NSF DMR-9976550 and NSF DMR-0325939.

http://www.siam.org/journals/sisc/27-6/61027.html

†Department of Mathematics, Tufts University, 503 Boston Ave., Medford, MA, 02155 (misha.kilmer@tufts.edu).

‡The Thomas M. Siebel Center for Computer Science, University of Illinois at Urbana-Champaign, 201 N. Goodwin, Urbana, IL 61801-2302 (sturler@cs.uiuc.edu).

bottleneck for the imaging problem. The goal of this paper is to discuss techniques for reducing the computational complexity of forward solves, thereby greatly improving the execution time of the nonlinear imaging problem.

In [26], we recycle the subspaces that are generated automatically by the methods used. In the present paper, we analyze the application, its solutions, and the nonlinear optimization algorithm to derive strategies for selecting subspaces for recycling that enhance convergence even further. In particular, we explore which information to keep and which to save as the damped Gauss–Newton (GN) method progresses. Furthermore, we present a variant of GCRO [10] that exploits the symmetry of the matrix and that can solve multiple shifted complex systems simultaneously with a single Krylov subspace in combination with recycling. The symmetry means that there is no need to restart for a single linear system to save on storage.

We consider the solution of a sequence of linear systems of the form

$$(A(p_j) + i\gamma I)x_{s,\omega}^{(j)} = b_s \tag{1.1}$$

that arise in a 3D imaging algorithm for DOT. Here, $i = \sqrt{-1}$, γ is a positive constant that depends on the frequency, ω, and the vector p_j denotes the vector of parameters that define the diffusion and absorption in tissue at the jth step of a GN iteration to solve the nonlinear least squares problem (defined below) for an optimal set of parameters. The sparse, symmetric positive definite matrix A depends on the parameters p_j and arises from the discretization of the diffusion-absorption equation with mixed boundary conditions and some further matrix manipulation (see section 3). Therefore, Krylov iterative solvers such as MINRES [25, 4] are good candidates for solving these linear systems.

We observe that in our application the matrices $A(p_j)$ vary slowly from one GN step or line search step to the next, but they change significantly over multiple GN steps. In addition, we need to solve for multiple (complex) shifts and multiple right-hand sides for each matrix. In order to solve this problem in the most efficient way we must exploit all these features. For each of these features separately, suggestions have been made to reduce the overall cost (for example, see [19, 18, 22] for solving for a group of matrices that differ by a constant times the identity and [17, 31, 9, 26, 21] for solving for multiple right-hand sides). However, the various methods have not been combined to address all these features at once, and this is not a trivial issue.

The problem of solving a sequence of systems where the matrix changes slowly is the most complicated feature to exploit. In [26] we propose to *recycle* from one linear system to the next the Krylov subspaces that solvers like GCRO [10], GCROT [11], and GMRESDR [24] retain to improve the convergence for a single linear system. GMRESDR cannot recycle a subspace for a subsequent linear system, as it requires a Krylov space to work with; therefore, we introduced the variant GCRODR [26].

Other approaches have been proposed as well. If all matrices in a set of symmetric positive definite linear systems are pairwise close to each other and all right-hand sides are available simultaneously, the methods proposed by Chan and Ng [7] can be used. However, this is not the case for our application. For the Hermitian positive definite case, Rey and Risler have proposed to reduce the effective condition number by augmenting the search space either with all converged Ritz vectors from previous CG iterations or with all previously generated Krylov spaces for a few iterations to implicitly approximate dominant eigenvectors [27, 28]. Clearly, both in memory and floating point operations these methods are very expensive. Moreover, in their approach, although they use CG, full recurrences are required. Finally, both approaches

lack the possibility to gradually improve the recycled space and to adapt it for a sequence of linear systems that change gradually but substantially over many steps. For this reason, Rey and Risler also make the assumption that all matrices remain close. Fischer has proposed to project right-hand sides from subsequent time steps onto the space of previous right-hand sides and to solve only for the remainder [16]. This leads to a better starting guess if the right-hand sides are correlated, but his algorithm does not maintain orthogonality to this subspace and, in general, does not improve the rate of convergence.

Multigrid methods might provide an alternative for the methods proposed here, as they have lower complexity than Krylov methods for diffusion type problems, and several papers have discussed how to deal effectively with large jumps in the diffusion and absorption coefficients [1, 5, 34]. However, it is not clear that the convergence rate of multigrid methods can be significantly improved for a sequence of matrices and/or right-hand sides when subsequent solutions differ significantly, as is accomplished by the methods proposed here (see also [26]). Adaptations of multigrid methods for sequences of problems mainly involve the judicious choice on which grids to update an old solution [5]. This approach is not suitable if the new solution differs significantly from the previous solution because it does not improve convergence in general. It just reduces the work in updating a slightly changed solution. In addition, our algorithms compute solutions for multiple shifts more or less for free; in a multigrid setting this is not possible. Finally, we can improve the convergence of the algorithm over multiple right-hand sides, which is not straightforward with multigrid methods.

As mentioned above, our problem includes the solution of a small set of right-hand sides for each matrix (with fixed p_j and γ). As was shown in [9, 26], subspace recycling is quite effective for this problem. However, other approaches or variations have been successful as well, in particular block methods [17, 31] and seed methods [8, 21]. Subspace recycling is the easiest to implement, as it can solve the right-hand sides one by one as individual linear systems, simply recycling the selected search space [26]. This avoids changes to the program to deal with varying block sizes and deflation if the vectors inside a block become dependent.

Finally, we also want to solve for multiple shifts using a single Krylov space. This is not complicated in itself, as the Krylov space for a shifted problem equals the Krylov space for the original problem. However, it is not easily combined with recycling Krylov subspaces, because the images of recycled spaces under matrices with different shifts are not the same. We derive an extension to the GCRO method in section 5 to deal with this problem.

The paper is organized as follows. Section 2 gives some background on GCRO and subspace recycling. In section 3, we give background information for the imaging problem in DOT, and we derive the sequence of linear systems of the form (1.1) that we wish to solve. In section 4, we discuss characteristics of the system that favor the use of recycling. We describe our algorithm in section 5, and give numerical results in section 6. Conclusions and future work are the subject of section 7.

2. Recycling Krylov subspaces. The ideas we exploit in this paper find their origin in attempts to improve the convergence of restarted and truncated Krylov subspace methods for a single, nonsymmetric, linear system. Restarting GMRES [30] may lead to poor convergence and even stagnation. Therefore, recent research has focused on truncated methods that improve convergence by retaining a selected subspace when they restart [3, 10, 11, 24, 26, 29]. We refer to this subspace as the *recycled* subspace. These methods aim to maintain convergence close to that of full

GMRES while significantly reducing memory and CPU costs. A taxonomy of popular choices is given in [15], and approaches to convergence theory for GMRES that are relevant here can be found in [32, 35].

An important choice for the present paper is to recycle an approximate invariant subspace [23, 24]. An alternative method that adapts the recycled space based on its effectiveness is given in [11]. In [10] the updates to the solution (residual) are recycled. This is also proposed in [3], but the recycled space is used slightly differently. In this paper, we exploit knowledge of the physics underlying DOT, the solutions in a general sense, and the nonlinear optimization algorithm to combine the strategies from [10, 24, 16, 26] for a sequence of linear systems.

The GCRO method provides a general mechanism for Krylov methods to include arbitrary additional subspaces in the search space. We explain briefly how GCRO combines the recycled space with a newly generated Krylov subspace to obtain an optimal approximation over the sum of these spaces. To prepare for the extension to the simultaneous solution of a set of linear systems with matrices that differ only by a constant times the identity using a single Krylov subspace, we present the basic steps from [10] in a slightly different way.

Consider the linear system $Ax = b$, where $A \in \mathbb{R}^{N\times N}$ and $b \in \mathbb{R}^N$. Assume we have the matrix $U \in \mathbb{R}^{N\times n_c}$, such that $AU = C$ and $C^TC = I$. There are no other restrictions on the matrix U. The approximate solution in range(U) that minimizes the 2-norm of the residual yields the residual $r = b - CC^Tb$ that is orthogonal to range(C). If this solution is not adequate, we expand the subspace as follows [10]. Let $v_1 = (I-CC^T)b/\|(I-CC^T)b\|_2$. We use an Arnoldi recurrence with $(I-CC^T)A$ and v_1 to generate the recurrence relation

$$\begin{aligned}(I - CC^T)AV_m &= V_{m+1}\underline{\mathrm{H}}_m \Leftrightarrow\\ AV_m &= CC^TAV_m + V_{m+1}\underline{\mathrm{H}}_m.\end{aligned} \tag{2.1}$$

Next, we compute the approximate solution in range$([V_m\ U])$ that minimizes the 2-norm of the residual, $\|b - A(V_my + Uz)\|_2$, as follows:

$$\begin{aligned}&\min_{y,z}\left\|b - A[U\ V_m]\begin{bmatrix} z\\ y\end{bmatrix}\right\|_2\\ &= \min_{y,z}\left\|b - [C\ V_{m+1}]\begin{bmatrix} I & C^TAV_m\\ 0 & \underline{\mathrm{H}}_m\end{bmatrix}\begin{bmatrix} z\\ y\end{bmatrix}\right\|_2\\ &= \min_{y,z}\left\|\begin{bmatrix} C^Tb\\ \xi e_1\end{bmatrix} - \begin{bmatrix} I & C^TAV_m\\ 0 & \underline{\mathrm{H}}_m\end{bmatrix}\begin{bmatrix} z\\ y\end{bmatrix}\right\|_2,\end{aligned} \tag{2.2}$$

where e_1 denotes the first Cartesian basis vector in $\mathbb{R}^{m+1}$ and $\xi = \|(I-CC^T)b\|_2$. The minimization in (2.2) corresponds to a small least squares problem, whose solution requires only the QR decomposition of the submatrix $\underline{\mathrm{H}}_m$ using Givens rotations as in [30]. We will show below that this approach is also extended easily to a set of matrices $A + i\gamma I$. In this case, we need to deal with the problem that range$((A + i\gamma I)U) \not\subset$ range(C), and of course range$((A + i\gamma I)U)$ depends on γ.

If we have found a matrix U that speeds up the convergence significantly, we can recycle this matrix for the next right-hand side (with the same A and γ) and possibly improve it. This does not require any changes to the algorithm, and it allows the algorithm to learn as it computes the solutions for subsequent right-hand sides which spaces are best to recycle. This improves the convergence for multiple right-hand

sides and a constant matrix without requiring extra storage. This algorithm can also be extended to a block method [37].

Now, consider a sequence of systems with coefficient matrices $A^{(j)}$. If we have found a matrix U that speeds up convergence significantly for $A^{(j)}$, and $A^{(j+1)} - A^{(j)}$ is small, it makes sense to reuse the matrix U, possibly extended with other search directions, for the linear systems with $A^{(j+1)}$. In this case, we need to update C to reflect the new operator. In many cases this can be done very cheaply. This process is not complicated, and we refer to [26] for details.

3. The DOT imaging application. In this section, we introduce the image reconstruction problem for DOT. Here, we derive the systems of the form (1.1) that must be solved at each step of the nonlinear reconstruction algorithm.

3.1. The forward and inverse problems. We assume that the region to be imaged is a box with a limited number of N_s sources on the top and a limited number of N_d detectors on either the top or bottom or both. We use the diffusion model [2] for photon flux/fluence $\phi_{s,\omega}(r)$ given input $f_s(r)$:

$$\begin{aligned} -\nabla D(r)\nabla\phi_{s,\omega}(r) + \mu_a(r)\phi_{s,\omega}(r) + i\frac{\omega}{\nu}\phi_{s,\omega}(r) &= f_s(r) \\ \text{for } r = (x,y,z) \text{ and } -a < x < a,\ -b < y < b,\ 0 < z < c, & \\ \phi_{s,\omega}(r) = 0 \quad \text{if } 0 \le z \le c \text{ and either } x = -a,\ x = a,\ y = -b, \text{ or } y = b, & \\ .25\phi_{s,\omega}(r) + \frac{D(r)}{2}\frac{\partial\phi_{s,\omega}(r)}{\partial\eta} &= 0 \text{ for } z = 0 \text{ or } z = c. \end{aligned}$$

Here, $D(r)$ denotes the diffusion, which is related to the "reduced scattering" function $\mu_s'(r)$ by $D = 1/(3\mu_s'(r))$, and $\mu_a(r)$ denotes absorption [2]. We have used $i = \sqrt{-1}$, while ω represents the frequency modulation of light, and ν is the speed of light in the medium. The integer subscript s indicates the model with a single source at a known position. The function $f_s(r)$ is the source and $\phi_{s,\omega}(r)$ is the photon flux/fluence due to the source at frequency ω, given the functions $\mu_a(r)$ and $D(r)$. The function $\phi_{s,\omega}(r)$ is complex-valued if ω is nonzero. Knowing the source and the functions $\mu_a(r), D(r)$, we can compute the corresponding $\phi_s(r)$ everywhere, in particular at the detectors (i.e., at a subset of gridpoints where $z = 0$ or $z = c$).

We discretize the PDE using finite differences [6] on a uniform grid in such a way as to achieve second-order accuracy away from the boundary. The meshwidth in each direction is h centimeters. We use first-order forward or backward differences, as appropriate, on the boundary. The unknowns become $\phi_{s,\omega}(x_l, y_j, z_k)$ for $l = 1 : N_x, j = 1 : N_y, k = 1 : N_z$. We will order the unknowns so that $\phi_{s,\omega}$ values at points on the top of the box come first (i.e., let $k = 1$ and loop over all l, j), then the $\phi_{s,\omega}$ corresponding to points on the bottom (i.e., let $k = N_z$ and loop over all l, j), followed by the rest of the values by ordering in increasing l, then j, then k. We denote the corresponding vector with entries $\phi_{s,\omega}(x_l, y_j, z_k)$ as $\boldsymbol{\phi_{s,\omega}}$. Likewise, we denote the vector with entries $\mu_a(x_l, y_j, z_k)$ as $\boldsymbol{\mu_a}$. The vector $\boldsymbol{D}$ is comprised from sampling D at both whole and half integer gridpoints, because of the particular discretization we are using.

The corresponding matrix equation, after multiplication by h^2, has the following block structure:

$$\begin{bmatrix} B & D_1 \\ D_2 & (F + i\frac{\omega h^2}{\nu} I) \end{bmatrix} \begin{bmatrix} w_{s,\omega} \\ x_{s,\omega} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_s}^{(1)} \\ \boldsymbol{f_s}^{(2)} \end{bmatrix}, \tag{3.1}$$

where $w_{s,\omega}$ and $x_{s,\omega}$ denote the discretization of $\phi_{s,\omega}(r)$ on the boundary and at internal points, respectively. The measured data *due to source s* **predicted** by this forward model is a subsampled version of the subvector $w_{s,\omega}$, which we call $\boldsymbol{\psi_{s,\omega}}$. It is important to note that in our application $\boldsymbol{f_s}^{(2)} = 0$ and $\boldsymbol{f_s}^{(1)} = h^2 e_i$, where e_i is the ith Cartesian unit vector and the position of the 1 corresponds to the location of the source.

Let $y_{s,\omega}$ denote the data subvector **measured** at all the detectors for a fixed source s and frequency ω, and let $W_{s,\omega}$ denote the diagonal weighting matrix whose diagonal entries are the inverses of the standard deviations of the noise in the data. Recall that p is a vector of parameters that describe the diffusion and absorption at all points in the region of interest. We will briefly discuss the choice of p below and refer the interested reader to [20] for more details. The 3D imaging problem then becomes one of finding the parameters that minimize the difference between the data predicted by the model and the measured data. That is, we wish to solve

$$\min_p \sum_{s,\omega} \|W_{s,\omega}(y_{s,\omega} - \boldsymbol{\psi_{s,\omega}}(p))\|_2^2 \equiv \min_p \|W(\mathbf{y} - \boldsymbol{\psi}(p))\|_2^2, \tag{3.2}$$

where $\mathbf{y}$ denotes the vector obtained by stacking the subvectors $y_{s,\omega}$, $\boldsymbol{\psi}(p)$ is the vector obtained by stacking the subvectors $\boldsymbol{\psi_{s,\omega}}(p)$, and W is the block matrix whose diagonal blocks are the matrices $W_{s,\omega}$. Let $\epsilon(p)$ denote the residual vector $W(\mathbf{y} - \boldsymbol{\psi}(p))$. We use the following damped GN iteration to solve this nonlinear least squares problem [14], where $J(p)$ denotes the Jacobian of the residual, evaluated at p:

1. compute $\epsilon(p_k)$, $J(p_k)$,
2. solve $J(p_k)^T J(p_k) s_k = -J(p_k)^T \epsilon(p_k)$,
3. $p_{k+1} = p_k + \lambda_k s_k$,

with λ_k chosen using a backtracking line search [14].

If the number of parameters used to define diffusion and absorption is small, the Jacobian will only have a small number of columns, and therefore step 2, solving for the search direction, is not computationally intensive. However, to compute $\epsilon(p_k)$ and to compute the entries in the Jacobian using an adjoint-type approach (called a "co-state" method in [36]) requires solutions of the matrix equation in (3.1) for every source and every frequency. Furthermore, each line search step requires evaluation of residual and Jacobian at a new point. Therefore, the rest of this paper is devoted to analyzing the systems (3.1) and methods for computing their solutions efficiently.

3.2. The matrix revisited. Here, we describe properties of the matrices and submatrices involved in solving for each $\boldsymbol{\phi_{s,\omega}}$. First, regarding the blocks in (3.1),

- B is an invertible diagonal matrix,
- D_1 has at most one nonzero per row, and these occur only in the first $N_x N_y$ and last $N_x N_y$ columns,
- D_2, although it has different entries, has the same sparsity pattern as D_1^T.

A MATLAB sparsity plot of the matrix in (3.1) is given in Figure 1 to give the reader a visual interpretation of the structure just mentioned.

To solve systems involving this matrix, we consider the decomposition $G = LU$, where L is block unit lower triangular, U is block upper triangular, and G represents the block matrix in (3.1). It can readily be shown that

$$G = \begin{bmatrix} I & 0 \\ D_2 B^{-1} & I \end{bmatrix} \begin{bmatrix} B & D_1 \\ 0 & F - D_2 B^{-1} D_1 + i\gamma I \end{bmatrix},$$

where $\gamma = h^2\omega/\nu$. Hence, the system $G \begin{bmatrix} w_{s,\omega} \\ x_{s,\omega} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f_s}^{(1)} \\ 0 \end{bmatrix}$ can be solved as follows.

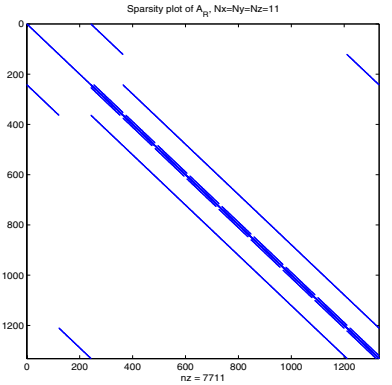


FIG. 1. *Sparsity plot of matrix G.*

Step 1. Solve the equation $L \begin{bmatrix} a_s \\ b_s \end{bmatrix} = R \begin{bmatrix} \boldsymbol{f_s}^{(1)} \\ 0 \end{bmatrix}$:

(a) $a_s = \boldsymbol{f_s}^{(1)}$,

(b) $b_s = -D_2(B^{-1}a_s)$.

Step 2. Solve the equation $U \begin{bmatrix} w_{s,\omega} \\ x_{s,\omega} \end{bmatrix} = \begin{bmatrix} a_s \\ b_s \end{bmatrix}$:

(a) solve $(F - D_2B^{-1}D_1 + i\gamma I)x_{s,\omega} = b_s$,

(b) $w_{s,\omega} = B^{-1}(a_s - D_1x_{s,\omega})$.

Multiplying with B^{-1} can be done very cheaply because B is diagonal, and D_1 and D_2 have only $(2N_xN_y)$ nonzero entries each. The computationally intensive part of this procedure is Step 2(a). Note that the system in Step 2(a) is exactly the system in (1.1).

We conclude this section with the proof that $F - D_2B^{-1}D_1$ is symmetric and positive definite. The matrix F is symmetric and positive definite because it corresponds to the finite difference discretization of the operator $-\nabla D(r)\nabla I + \mu_a(r)I$ at the internal points on the box assuming zero boundary conditions. B is a diagonal matrix of size $2N_xN_y \times 2N_xN_y$ with entries $.25h^2 + \frac{h}{2}\boldsymbol{D}_{l,j,m}$ with $m = 1$ or $m = N_z$. Due to the lexicographical ordering of the internal nodes, D_1 is $2N_xN_y \times (N_xN_y(N_z - 2))$ with only one nonzero per row and D_2 is $(N_xN_y(N_z-2))\times 2N_xN_y$ with only one nonzero per column. The nonzero entries in D_1 are of the form $-\frac{h}{2}\boldsymbol{D}_{l,j,m}$ with $m = 1$ or $m = N_z$. The nonzero entries in D_2 are $-\boldsymbol{D}_{l,j,\frac{3}{2}}$ or $-\boldsymbol{D}_{l,j,N_z-\frac{1}{2}}$. Therefore, $D_2B^{-1}D_1$ is a diagonal matrix with nonzero (positive) entries only in the first and last N_xN_y positions. We now use the previous two facts to prove the following theorem.

THEOREM 3.1. *The matrix $F - D_2B^{-1}D_1$ is symmetric and positive definite.*

Proof. Symmetry follows from the symmetry of F and $D_2B^{-1}D_1$. Matrices $F-D_2B^{-1}D_1$ and F differ only in the first and last N_xN_y components on the diagonal. Since the nonzeros in the second matrix are strictly positive, the first and last N_xN_y diagonal entries of $F - D_2B^{-1}D_1$ are smaller than the corresponding diagonal entries of F. Therefore, it is sufficient to consider the Gershgorin disks corresponding to these rows. Consider the first N_xN_y rows (the argument for the last N_xN_y is analogous). From Gershgorin's theorem we observe

$$\left(-\frac{\boldsymbol{D}_{i,j,1.5}\boldsymbol{D}_{i,j,1}}{0.5h + \boldsymbol{D}_{i,j,1}} + \boldsymbol{D}_{i,j,1.5}\right) + h^2\boldsymbol{\mu_a}_{i,j,1} \leq \lambda,$$

where λ is an eigenvalue. Since the first term on the left in parentheses is positive, together with lower bounds from all other Gershgorin disks, it follows that the eigenvalues are greater than zero. ∎

4. Summary of system properties. In the previous section, we observed that solving the forward problem efficiently boils down to solving the systems in Step 2(a) efficiently. The remainder of this paper is therefore devoted to this cause.

In this section, we discuss those system properties that can be exploited to develop efficient recycling Krylov methods for solving the systems in the previous section. We repeat the form of these systems here for convenience:

$$(\underbrace{F - D_2 B^{-1} D_1}_{A^{(j)}} + i\gamma I) x_{s,\omega}^{(j)} = b_s. \tag{4.1}$$

The superscript is used to denote dependence on the parameter vector, p_j. To simplify notation, we use $x_s^{(j)}$ for the case when $\omega = 0$.

The parametric model. As in [20], we use a piecewise continuous model for both the diffusion and absorption. In particular, we have

$$\boldsymbol{D} = \alpha_1 \Xi_1 + (\mathbf{1} - \Xi_1) B_1 \beta_1 \qquad \text{and} \qquad \boldsymbol{\mu_a} = \alpha_2 \Xi_2 + (\mathbf{1} - \Xi_2) B_2 \beta_2.$$

The vectors Ξ_i are discrete characteristic functions, having a "1" in a position corresponding to an anomaly and a 0 otherwise. The matrices B_1, B_2 are known and contain "basis" vectors while β_1 and β_2 are the unknown expansion coefficients. In a piecewise constant model, for instance, B_1 and B_2 would be vectors of all ones and the β_i would give the background value of diffusion and absorption, respectively. A more realistic model, however, accounts for the fact that tissue is not homogeneous, and in this case B_1, B_2 would correspond to vectorized "images" of a lumpy background. The vectors Ξ_i are unknown, but we assume that anomalies are modeled by ellipsoids. In this case, the entries in the vectors are determined by the parameters specifying center location, rotation, and axis lengths. Therefore, the list of unknown parameters includes $\alpha_i, \beta_i, i = 1, 2$, and up to 6 length-3 vectors specifying the locations of the 2 ellipsoids. For more details, see [20].

Diffusion, absorption, and matrix updates. Typical values for μ_a in our application range from .005 to .3 cm^{-1} whereas typical values for D range from 1/6 to 1/45 cm^{-1}. Usually, we have or can obtain good approximations to the average background values of diffusion and absorption and use these as starting guesses for the GN iteration [20]. As the GN iteration progresses, we begin to localize and characterize anomalous regions of absorption and diffusion whereas the background values become well resolved early on. This is due to the fact that the anomalies are so small relative to the size of the background that the data contains primarily information about the average background values. This means that during a single line search or when moving from one GN step to the next, $A^{(j+1)} = A^{(j)} + E_1 + E_2$, where $\|E_1\|$ is small (corresponding to a slight change in the background parameters) and E_2 has small relative rank (corresponding to a change in the shape of the object and values inside the object).

Eigenvalues and invariant subspaces. We are interested most in the smallest eigenvalues of the $A^{(j)}$. It appears that, for our examples, the matrices $A^{(j)}$ have a number of small eigenvalues occurring in clusters that remain disjoint from one system to the next, even if the eigenvalues themselves differ from one matrix to the next (see Figure 4, for example). This suggests that the corresponding invariant subspaces for subsequent matrices remain close. We discuss this further in the next subsection.

Similarity among right-hand sides. Recall that $\boldsymbol{f_s}$ has only one nonzero coefficient in, say, position m_s. It follows that $b_s^{(j)} = -\frac{(D_2)_{m_s,m_s}h^2}{B_{m_s,m_s}} e_{m_s}$, and so the $b_s^{(j)}$ are the same up to a scalar factor. The latter has no influence on the solver. Therefore, in the remaining discussion we study the convergence of iterative methods applied to the systems

$$(A^{(j)} + i\gamma I)x_{s,\omega}^{(j)} = e_{m_s}. \tag{4.2}$$

Similarity among solutions. From (4.2), it is easy to see that the solutions to any pair of (nonshifted) systems, say systems k and j, during the GN iteration are related by

$$x_s^{(k)} = (A^{(k)})^{-1}A^{(j)}x_s^{(j)}.$$

Thus, the solutions do not change much as long as the matrices $A^{(j)}$ and $A^{(k)}$ remain close, especially with respect to the smallest eigenvalues and corresponding eigenvectors; see also the following discussion on smooth solutions. The reason why we expect, in particular, the smooth components of the matrices to remain close is discussed in section 4.1. This expectation is borne out in the numerical experiments. It is important to note that the indices j and k need not be consecutive, nor must they refer to systems from within the same line search. See section 5 for details.

Smooth solutions. The vector e_{m_s} is comprised mostly of high frequency Fourier components. Furthermore, the eigenvectors corresponding to the smallest eigenvalues of $A^{(j)}$ represent low frequency (smooth) modes while the eigenvectors corresponding to the largest eigenvalues represent high frequency modes. The matrix $(A^{(j)})^{-1}$ corresponds to an integration operator that acts as a blurring operator. Therefore, we expect $x_s^{(j)}$ to be well represented in terms of the eigenvectors of $A^{(j)}$ that correspond to the smallest eigenvalues.

Multiple frequencies. The Krylov vectors generated to solve $A^{(j)}x_s^{(j)} = e_{m_s}$ also span the Krylov subspace generated by the shifted matrix $(A^{(j)} + i\gamma I)$ and e_{m_s} (see [22] and the references cited therein). When subspace recycling is used, however, it is not straightforward to produce solutions to the complex shifted systems. We discuss the extension of the algorithm to this case in section 5.4.

4.1. Invariant subspaces. In deciding whether or not it is worthwhile to recycle an approximate invariant subspace corresponding to small eigenvalues obtained from previous matrices, we must explore the relationships among these invariant subspaces of a sequence of matrices. In the previous discussion, we noted that from experiments it appears that the smallest eigenvalues of the $A^{(j)}$ do not change much and remain in more or less disjoint clusters. For a small enough perturbation $E = A^{(k)} - A^{(j)}$, this shows that the corresponding invariant subspaces from these matrices must remain close. Unfortunately, $\|E\|$ is too large to assume this without considering further details. However, recall the observation above that $E = E_1 + E_2$, where E_1 corresponds to very small changes in the background parameters and E_2 corresponds to a small rank update describing the shape of the object. From this we conjecture that most of the changes in the matrix correspond to the high frequency components and larger eigenvalues. Given this conjecture, we now show under which conditions the invariant subspaces corresponding to the smallest eigenvalues remain about the same even if the corresponding eigenvalues are not very well separated from the remaining eigenvalues.

To simplify notation, we remove all subscript and superscript notation and deal specifically with a symmetric and positive definite matrix A and a corresponding symmetric perturbation E.

Let A be a symmetric positive definite matrix, and let A have the eigendecomposition

$$A = [Q_1\ Q_2\ Q_3]\,\mathrm{diag}(\Lambda_1, \Lambda_2, \Lambda_3)\,[Q_1\ Q_2\ Q_3]^T, \tag{4.3}$$

where $Q = [Q_1\ Q_2\ Q_3]$ is an orthogonal matrix, $\Lambda_1 = \mathrm{diag}(\lambda_1^{(1)}, \ldots, \lambda_{k1}^{(1)})$, and Λ_2 and Λ_3 are defined analogously. Furthermore,

$$\lambda_1^{(1)} \le \cdots \le \lambda_{k1}^{(1)} < \lambda_1^{(2)} \le \cdots \le \lambda_{k2}^{(2)} < \lambda_1^{(3)} \le \cdots \le \lambda_{k3}^{(3)}.$$

In relation to our algorithm, Q_1 corresponds to the recycled invariant subspace (or a subspace of that), but Q_2 and Q_3 can be chosen to fit the theorem. Now we consider the changes in the invariant subspace $\mathrm{range}(Q_1)$ and the eigenvalues $\lambda_i^{(1)}$ under a symmetric perturbation E of A, where E is not small, but the projection of E onto the subspace $\mathrm{range}([Q_1\ Q_2])$ is small, say $\|E[Q_1\ Q_2]\|_F \le \varepsilon$, and $\|EQ_3\|_F = \eta \approx \|E\|_F$. We also assume that $\|E\|_F$ is small relative to $\mathrm{sep}(\Lambda_1, \Lambda_3) = \lambda_1^{(3)} - \lambda_{k1}^{(1)}$ and that ε is small relative to $\mathrm{sep}(\Lambda_1, \Lambda_2) = \lambda_1^{(2)} - \lambda_{k1}^{(1)}$. However, we do not need to assume that $\mathrm{sep}(\Lambda_1, \Lambda_2)$ is large. We now prove that the matrix $A + E$ has an invariant subspace $\mathrm{range}(\hat{Q}_1)$ such that the canonical angles between $\mathrm{range}(Q_1)$ and $\mathrm{range}(\hat{Q}_1)$ are small. This result shows that an invariant subspace whose associated eigenvalues are not well separated from the remaining eigenvalues is still insensitive to perturbations that are concentrated in an invariant subspace whose eigenvalues are sufficiently far removed.

We define the following notation. For two matrices $Y, Z \in \mathbb{R}^{N\times m}$, where $N \ge m$, $\Theta(\mathrm{range}(Y), \mathrm{range}(Z))$ denotes the diagonal matrix with the canonical angles between $\mathrm{range}(Z)$ and $\mathrm{range}(Y)$ as coefficients, and $\theta_1(\mathrm{range}(Y), \mathrm{range}(Z))$ denotes the largest canonical angle between $\mathrm{range}(Z)$ and $\mathrm{range}(Y)$. We use $\mathcal{L}(A)$ to denote the set of eigenvalues of A, and $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ to denote $\max \mathcal{L}(A)$ and $\min \mathcal{L}(A)$, respectively.

Furthermore, we assume that

$$\delta \equiv \min(\lambda_1^{(2)} - \varepsilon, \lambda_1^{(3)} - \eta) - 2\varepsilon - (\lambda_{k1}^{(1)} + \varepsilon) \gg \varepsilon, \tag{4.4}$$

$$\hat{\delta} = \delta\left(1 - \frac{2\varepsilon^2}{\delta^2}\right) + \lambda_{k1}^{(1)} + \varepsilon, \tag{4.5}$$

and as a consequence of (4.4) that $\delta > 2\varepsilon$.

THEOREM 4.1. *Let A be s.p.d. and have the eigendecomposition given in* (4.3)*, and let E, ε, η, δ, and $\hat{\delta}$ be defined as above. Then, there exists a matrix $\hat{Q}_1$ conforming to Q_1 such that* $\mathrm{range}(\hat{Q}_1)$ *is a simple invariant subspace of $A + E$, and*

$$\tan\theta_1\left(\mathrm{range}(Q_1), \mathrm{range}(\hat{Q}_1)\right) \le \frac{\varepsilon}{\hat{\delta}}.$$

Furthermore, for each eigenvalue $\hat{\lambda}_j^{(1)}$ of $A+E$ corresponding to the invariant subspace $\mathrm{range}(\hat{Q}_1)$*, there exists a value $\lambda_i^{(1)}$ such that*

$$|\hat{\lambda}_j^{(1)} - \lambda_i^{(1)}| \le \varepsilon + \frac{2\varepsilon^2}{\delta},$$

and in particular,

$$\lambda_{\max}(\,\hat{Q}_1^T(A+E)\hat{Q}_1\,) \le \lambda_{k1}^{(1)} + \varepsilon + \frac{2\varepsilon^2}{\delta}.$$

Proof. We consider the perturbation E, such that

$$Q^T(A+E)Q = \begin{pmatrix} \Lambda_1+\mathcal{E}_{11} & 0 & 0 \\ 0 & \Lambda_2+\mathcal{E}_{22} & \mathcal{E}_{32}^T \\ 0 & \mathcal{E}_{32} & \Lambda_3+\mathcal{E}_{33} \end{pmatrix} + \begin{pmatrix} 0 & \mathcal{E}_{21}^T & \mathcal{E}_{31}^T \\ \mathcal{E}_{21} & 0 & 0 \\ \mathcal{E}_{31} & 0 & 0 \end{pmatrix}. \tag{4.6}$$

By the assumptions above we also have

$$\left\| \begin{pmatrix} \mathcal{E}_{21} \\ \mathcal{E}_{31} \end{pmatrix} \right\|_F \le \varepsilon,$$

$\|\mathcal{E}_{11}\|_F \le \varepsilon$, $\|\mathcal{E}_{22}\|_F \le \varepsilon$, and $\|\mathcal{E}_{33}\|_F \le \eta$. From (4.6) we see that

$$\begin{aligned} L_1 \;&\equiv\; Q_1^T(A+E)Q_1 = \Lambda_1 + \mathcal{E}_{11}, \\ L_{23} \;&\equiv\; [Q_2\; Q_3]^T(A+E)[Q_2\; Q_3] = \begin{pmatrix} \Lambda_2+\mathcal{E}_{22} & \mathcal{E}_{32}^T \\ \mathcal{E}_{32} & \Lambda_3+\mathcal{E}_{33} \end{pmatrix}. \end{aligned}$$

From [33, Corollary IV.3.4] it follows that

$$\lambda_{\max}(\Lambda_1+\mathcal{E}_{11}) \le \lambda_{k1}^{(1)} + \|\mathcal{E}_{11}\| \le \lambda_{k1}^{(1)} + \varepsilon, \tag{4.7}$$

$$\lambda_{\min}(\Lambda_2+\mathcal{E}_{22}) \ge \lambda_1^{(2)} - \|\mathcal{E}_{22}\| \ge \lambda_1^{(2)} - \varepsilon, \tag{4.8}$$

$$\lambda_{\min}(\Lambda_3+\mathcal{E}_{33}) \ge \lambda_1^{(3)} - \|\mathcal{E}_{33}\| \ge \lambda_1^{(3)} - \eta. \tag{4.9}$$

Now we can apply [33, Corollary IV.3.4] once more to obtain

$$\lambda_{\min}(L_{23}) \ge \min(\lambda_1^{(2)} - \varepsilon, \lambda_1^{(3)} - \eta) - 2\varepsilon. \tag{4.10}$$

From (4.7)–(4.10) we have $\mathrm{sep}(L_1, L_{23}) > \delta$. Furthermore, let $R \equiv (A+E)Q_1 - Q_1L_1 = Q_2\mathcal{E}_{21} + Q_3\mathcal{E}_{31}$. Then, from symmetry it follows that $Q_1^T(A+E) - L_1Q_1^T = R^T$, and we have $\|R\|_F = \|R^T\|_F \le \varepsilon$. Finally, we have

$$\frac{\|R\|_F\|R^T\|_F}{\mathrm{sep}(L_1, L_{23})^2} \le \frac{\varepsilon^2}{\delta^2} < \frac{1}{4},$$

and by [33, Corollary V.2.2] we know there exists a matrix $\hat{Q}_1$ conforming to Q_1 such that $\mathrm{range}(\hat{Q}_1)$ is a simple invariant subspace of $A+E$, and

$$\tan\theta_1\left(\mathrm{range}(Q_1), \mathrm{range}(\hat{Q}_1)\right) \;\le\; \|\tan\Theta\left(\mathrm{range}(Q_1), \mathrm{range}(\hat{Q}_1)\right)\|_F \;\le\; 2\frac{\varepsilon}{\delta}. \tag{4.11}$$

Since A and $A+E$ are symmetric and we have established the existence of $\hat{Q}_1$, we can obtain a better bound using [33, Theorem V.3.10]. This theorem conforms nicely to our special case. However, we need to establish the minimal distance between eigenvalues of L_1 and the eigenvalues of $\hat{L}_{23} \equiv \hat{Q}_{23}^T(A+E)\hat{Q}_{23}$, where $\mathrm{range}(\hat{Q}_{23}) = \mathrm{range}(\hat{Q}_1)^\perp$ and $\hat{Q}_{23}$ has orthonormal columns. From [33, Theorem V.2.1] specialized to the symmetric case, we know there exists a matrix P, such that $\|P\|_F \le 2\varepsilon/\delta$ and

$\mathcal{L}(\hat{L}_{23}) = \mathcal{L}(L_{23} - P[\mathcal{E}_{21}^T\ \mathcal{E}_{31}^T])$. From $\|P[\mathcal{E}_{21}^T\ \mathcal{E}_{31}^T]\| \le 2\varepsilon^2/\delta$ and [33, Corollary IV.3.4], we have the following bound (with $\hat{\delta}$ defined in (4.5)):

$$\lambda_{\min}(\hat{L}_{23}) \ge \min(\lambda_1^{(2)} - \varepsilon, \lambda_1^{(3)} - \eta) - 2\varepsilon - \frac{2\varepsilon^2}{\delta} = \hat{\delta}. \tag{4.12}$$

Finally, we obtain from [33, Theorem V.3.10]

$$\tan\theta_1(\mathrm{range}(Q_1), \mathrm{range}(\hat{Q}_1)) \le \|\tan\Theta(\mathrm{range}(Q_1), \mathrm{range}(\hat{Q}_1))\|_F \le \frac{\varepsilon}{\hat{\delta}},$$

which is better than (4.11) by about a factor 2 or more. Analogously to (4.12) we observe from [33, Theorem V.3.10] that for each eigenvalue $\hat{\lambda}_j^{(1)}$ of $\hat{Q}_1^T(A+E)\hat{Q}_1$ there exists a number, $\lambda_i^{(1)}$, such that

$$|\hat{\lambda}_j^{(1)} - \lambda_i^{(1)}| \le \varepsilon + \frac{2\varepsilon^2}{\delta}.$$

In particular, this gives

$$\lambda_{\max}(\hat{Q}_1^T(A+E)\hat{Q}_1) \le \lambda_{k1}^{(1)} + \varepsilon + \frac{2\varepsilon^2}{\delta}. \qquad \square$$

Our numerical experiments confirm our conjecture except for a few GN steps, when the new matrix is quite far from previous ones. In those cases, the projection of E on the (smooth) invariant subspace corresponding to the smallest eigenvalues is still small, but $\|E\|$ is large enough that the invariant subspaces corresponding to large eigenvalues might perturb those corresponding to the smallest ones. Because of the problem size we cannot check the projection of E on invariant subspaces corresponding to medium or larger eigenvalues. However, Figures 4 and 9 include examples with relatively large canonical angles between invariant subspaces associated with the smallest eigenvalues for systems at the start of a line search.

5. Algorithm. Let us outline some aspects related to the optimization algorithm. We combine the GN algorithm with a line search. As we will see in the numerical results section, the final steps in each line search tend to be small. Therefore, the solutions obtained toward the end of each line search are not too different for a few GN iterations. However, over many GN steps they tend to differ more significantly; see Figure 2. An obvious way to exploit this is by using the solution of the previous step as a starting guess. However, this will work better or worse depending on whether the previous step was the first step of a line search, towards the start or the end of a line search, and so on. In practice, it is not easy to choose the best among several previous solutions as the best choice is governed by the progression of the algorithm (it is not necessarily the latest one). However, recycling a small subspace of previous solutions for the search space relieves this problem. In fact, we can vary this additional subspace depending on whether we are at the start of a line search or near the end and on how large the line search parameter is. We can also update this additional search space as we go. Strategies based on this approach turn out to be very effective; we will give more details later in this section when we discuss the application and the optimization algorithm. The idea to use previous solutions was also proposed in [16], although in [16] they were only used to provide a better initial guess.

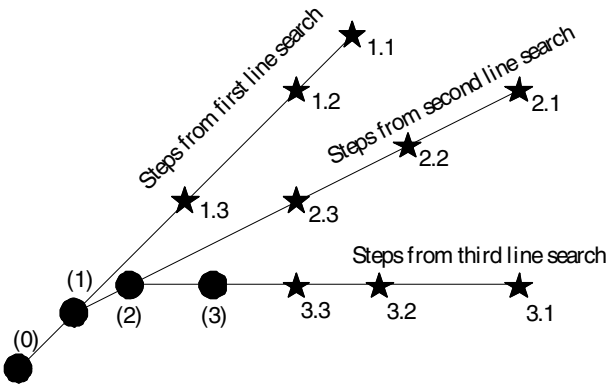


FIG. 2. *Conceptual overview of three complete steps of the GN with line search algorithm. The black circle marked* (0) *indicates the start of the overall procedure. The other black circles denote the end of a line search (the end of the first line search is marked* (1)*, etc.). The black stars denote the first and intermediate steps from each line search as indicated (the first step of line search* 1 *is indicated by* (1.1)*, etc.).*

The recycling strategy that we propose is based on the observations in the previous section. In particular, we use the proximity of certain invariant subspaces and tune which old solutions to recycle to the phase of the GN algorithm with line search.

5.1. Recycled GCRO for DOT. Consider the sequence of systems

$$A^{(j)}x_1^{(j)} = e_{m_1},$$

for the source $s = 1$. We will always recycle, in a matrix U, the most recent solution that occurred at the end of a line search. We may recycle previous solutions from within a particular line search. We do not know whether a line search is complete until after the corresponding system has been solved. However, before we solve a system, we can test whether we are "close" to the end of a line search. We assume this is the case if the relative residual norm of the current system for the solution at the end of the previous line search is below a certain threshold. This information helps to keep the smallest possible recycling space. It may not be necessary to keep a large approximate invariant subspace to reduce the initial residual over that part of the spectrum when we keep solutions in the recycle space that have the effect of reducing the residual over the same part of spectrum anyway (see the discussion in section 5.2).

To keep the notation as simple as possible, we begin by reviewing a basic GCRO algorithm that implements the recycling strategy mentioned above; then we discuss modifications that allow us to solve for multiple right-hand sides and multiple frequencies. Below, the MATLAB notation $[C, R] = qr(C, 0)$ is used to mean that C is overwritten with the orthogonal matrix from its compact QR factorization and R is the upper triangular factor.

ALGORITHM 1.

1. *Solve* $A^{(1)}x_1^{(1)} = e_{m_1}$ *with MINRES. Set* $x^{(curr)} = x_1^{(1)}, x^{(beg)} = []$.
2. *Form approximate eigenvectors for* $A^{(1)}$ *from information generated from the MINRES run. Save these eigenvectors in the matrix* W.

3. ***For*** $j = 2, \ldots$
   (a) ***If*** $(\|A^{(j)}x^{(curr)} - e_{m_1}\|/\|e_{m_1}\| \leq tol)$ ***and not at beginning of LS***
   $U = [W_{ind}, x^{(curr)}]$,
   Else
   $U = [W, x^{(curr)}, x^{(beg)}]$.
   (b) $A^{(j)}U = C$, $[C, R] = qr(C, 0)$, *set*[1] $U = UR^{-1}$.
   (c) *Compute* $x_0 = U(C^T e_{m_1})$. *(Ensures* x_0 *is optimal in the sense that the residual is minimized over all solutions in* range(U)*.)*
   (d) *Set* $P = (I - CC^T)$. *Compute* $r_0 = Pb$.
   (e) *Solve* $PA^{(j)}Pv = r_0$ *by MINRES.*[2]
   (f) *Update* x, r.
   (g) ***If*** *(at end of LS),* $x^{(curr)} = x_1^{(j)}$.
   (h) ***If*** *(at beginning of LS),* $x^{(beg)} = x_1^{(j)}$.

The index j is the index for the system within the GN algorithm, and therefore the loop on j terminates when GN converges for (3.2). Here, W_{ind} indicates that we may wish to keep fewer approximate eigenvectors according to the discussion preceding the algorithm (see also the discussion in section 5.2 and numerical example 1). The logic for tailoring the choice of the columns of U to the GN process comes from the discussion in the previous section. First, based on our observations for this application, we expect the approximate invariant subspace corresponding to the smallest eigenvalues of the first matrix to be close to an invariant subspace of other matrices in the GN sequence corresponding to the smallest eigenvalues. If the GN iteration is converging, we expect that at the end of two consecutive line searches the corresponding matrices will be related since the purpose of the line search is to produce a parameter update vector that allows the GN process to converge. Likewise, if we are at the beginning of a line search sequence, matrices from that sequence should be related, too. An analysis of the effect of these choices on the convergence of the MINRES steps is provided in the next subsection.

5.2. Algorithm analysis. Given that U always contains the matrix W, which we assume is a good approximation for the invariant subspace corresponding to the smallest eigenvalues for all the systems, we expect the systems in 3(e) to converge as if the smallest eigenvalues have been deflated from $A^{(j)}$. This is the case, in fact, as Theorem 4.1 from [26] (which closely follows Theorem 2.1 in [32]) implies.

Next, we consider the effect of keeping a previous solution, $x^{(curr)}$, in U. Let the current matrix U have n_c columns. For ease of discussion, we may assume that the first column of U is $x^{(curr)}$, since this does not change the orthogonal projector. Now, at the end of 3(b), we observe that $A^{(j)}x^{(curr)} = \rho_1 c_1$, where c_1 is the first column of C and ρ_1 an appropriate scaling. However, $x^{(curr)} = (A^{(k)})^{-1}e_{m_s}$ for some previous index k, and hence $A^{(j)}(A^{(k)})^{-1}e_{m_s} = \rho_1 c_1$. We have $A^{(j)}(A^{(k)})^{-1} = I + \tilde{E}$ for error matrix $\tilde{E} = -E_1(A^{(k)})^{-1} - E_2(A^{(k)})^{-1}$, with E_1, E_2 as the small norm and relatively small rank terms defined previously. Therefore,

$$\tilde{v} \equiv (I - c_1 c_1^T)e_{m_s} = e_{m_s} - c_1 c_1^T(\rho_1 c_1 - \tilde{E}e_{m_s}) = e_{m_s} - \rho_1 c_1 + c_1(c_1^T \tilde{E} e_{m_s}).$$

[1] From an implementation standpoint, we would not perform the matrix-matrix product UR^{-1}. Rather, we would keep R around, and when we needed to perform UR^{-1} times a vector, as in the next step, we would do backward substitution with R followed by multiplication with U. However, this explicit notation simplifies the introduction of the algorithm.

[2] Since all the Krylov vectors in the MINRES iteration are orthogonal to C, we do not need to apply P on the right. This choice of notation serves only to illustrate the symmetry of the matrix operator and therefore the applicability of MINRES.

Then the initial residual is $r_0 = (I - C_{n_c-1}C_{n_c-1}^T)\tilde{v}$, where C_{n_c-1} contains the remaining columns in C. Thus we have

$$\begin{aligned} r_0 &= (I - C_{n_c-1}C_{n_c-1}^T)(e_{m_s} - \rho_1 c_1) + (c_1^T \tilde{E} e_{m_s})c_1 \\ &= (I - C_{n_c-1}C_{n_c-1}^T)(A^{(k)} - A^{(j)})x^{(curr)} + (c_1^T \tilde{E} e_{m_s})c_1. \end{aligned}$$

The solution $x^{(curr)}$ is smooth and, as seen earlier, $A^{(k)} - A^{(j)}$ is small over the invariant subspace of $A^{(j)}$ corresponding to the smallest eigenvalues. Thus, the vector $e_{m_s} - \rho_1 c_1$ should already be small in norm. Since C_{n_c-1} contains approximate eigenvectors corresponding to the smallest (and smoothest) eigenvectors, then clearly r_0 will be even smaller in norm, particularly if $(e_{m_s} - \rho_1 c_1)$ lies predominately in the direction of these eigenvectors anyway. Furthermore, we have $c_1^T \tilde{E} e_{m_s} = -c_1^T E_1 x^{(curr)} - c_1^T E_2 x^{(curr)}$. The first term in this expression is small in norm. The second term must also be small, since E_2 lies predominantly in the direction corresponding to larger magnitude eigenvalues whereas $x^{(curr)}$ is smooth. In summary, not only do we observe that the norm of r_0 is small, but the smoothness properties ensure it is smallest in directions corresponding to the larger magnitude eigenvalues and that it has been reduced in directions corresponding to the smallest magnitude eigenvalues. This is demonstrated clearly by the reduction of the norm of the initial residual as the GN iteration progresses in Figures 7 and 12 and also by the eigenvector components of $r_0^{(j)}$ given in Figure 5. Hence, corrections to the residual occur primarily over the remaining subspace, which accounts for the convergence behavior observed in our numerical examples.

5.3. Multiple right-hand sides. The next consideration is the solution for multiple right-hand sides. First, we expand step 2 in Algorithm 1 to accommodate the right-hand sides. Once the first system for the first source has been solved, we have approximate eigenvector information. We can use this eigenvector information when we solve for the remaining right-hand sides, and we can also collect additional eigenvector information. Thus, the new step 2 becomes the following:

- Form approximate eigenvectors for $A^{(1)}$ from information generated from the MINRES run. Save these eigenvectors in the matrix W.
- **For** $s = 2, \ldots, N_s$
  - Set $U = W$.
  - Perform steps 3b–3e of Algorithm 1 (only updates of U and C).
  - Update $x_s^{(1)}$ with this information.
  - Add columns to W if desired.

Additional (approximate) invariant subspace information might be computed either from range(V_m) or from range($[U\, V_m]$). Then, to solve for $x_s^{(j)}$ for the remaining sources, we insert a loop over the remaining sources just after step 3 which would run over all s and contain steps 3(a)–3(h).

In this way, all the right-hand sides use a different last column (or two) in the U matrix, depending on the source. However, for every right-hand side, the first several columns of the U matrix are comprised of the (final) approximate eigenvector matrix W. Therefore, step 3(b) is cheaper for sources $s \geq 2$, since all but the last column (or two) of C were determined during the run on the first source.

5.4. Multiple shifts γ. To account for multiple frequencies, in step 3(e), we have $v_1 = (I - CC)^T e_{m_s}/\|(I - CC^T)e_{m_s}\|_2$ and the matrix recurrence (cf. (2.1))

$$AV_m = CB_m + V_{m+1}\underline{T}_m, \tag{5.1}$$

where $B_m = C^T A V_m$ and the leading $m \times m$ submatrix of $\underline{\mathrm{T}}_m$ is symmetric and positive definite. From this recurrence, we obtain

$$\begin{aligned}(A + i\gamma I)[V_m\ U] &= [V_{m+1}(\underline{\mathrm{T}}_m + i\gamma \underline{\mathrm{I}}_m) + CB_m\ \ C + i\gamma U] \\ &= [V_{m+1}\ \ C\ \ U] \begin{bmatrix} \underline{\mathrm{T}}_m + i\gamma \underline{\mathrm{I}}_m & 0 \\ B_m & I \\ 0 & i\gamma I \end{bmatrix} \\ &= [V_{m+1}\ \ C\ \ \hat{U}] \begin{bmatrix} I & 0 & V_{m+1}^T \hat{U} \\ 0 & I & C^T U \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} \underline{\mathrm{T}}_m + i\gamma \underline{\mathrm{I}}_m & 0 \\ B_m & I \\ 0 & i\gamma I \end{bmatrix},\end{aligned}$$

where the last step involves the reduced QR decomposition of $[V_{m+1}\ C\ U]$, so that $[V_{m+1}\ C\ \hat{U}]$ has orthonormal columns. Notice that $[V_{m+1}\ C]$ is already an orthogonal matrix. Restricting our approximate solutions to be in range$(U) \oplus$ range(V_m), we need to solve the small least squares problem

$$\begin{bmatrix} I & 0 & V_{m+1}^T U \\ 0 & I & C^T U \\ 0 & 0 & N \end{bmatrix} \begin{bmatrix} \underline{\mathrm{T}}_m + \gamma \underline{\mathrm{I}}_m & 0 \\ B_m & I \\ 0 & \gamma I \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} \approx \begin{bmatrix} \xi e_1 \\ C^T e_{m_s} \\ 0 \end{bmatrix}$$

for every choice of γ, and put $x_{s,\omega}^{(j)} = V_m y + U z$. This is equivalent to solving

$$\begin{bmatrix} \underline{\mathrm{T}}_m + i\gamma \underline{\mathrm{I}}_m & i\gamma V_{m+1}^T U \\ B_m & I + i\gamma C^T U \\ 0 & i\gamma N \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} \approx \begin{bmatrix} \xi e_1 \\ C^T e_{m_s} \\ 0 \end{bmatrix}. \tag{5.2}$$

This equation leads to an algorithm for updating the solutions to systems in which $\gamma \neq 0$. After step 3(e) in Algorithm 1, we insert the following piece of code:

ALGORITHM 2a: based on solving (5.2)

For each γ

- Solve (5.2) for $\begin{bmatrix} y \\ z \end{bmatrix}$.
- Set $x_{s,\omega}^{(j)} = V_m y + U z$.

Care must be taken in solving (5.2) if a subspace of range(U) is very close to an invariant subspace of $A^{(j)}$. In particular, if range(U) is an invariant subspace of $A^{(j)}$, then $U = C\Delta$, for some Δ. In this case, the least squares problems simplify considerably as both $V_{m+1}^T U = 0$ and $N = 0$. Hence, we need to solve only the following least squares problems (compare to (2.2)):

$$\begin{bmatrix} I + i\gamma\Delta & B_m \\ 0 & \underline{\mathrm{T}}_m + i\gamma \underline{\mathrm{I}}_m \end{bmatrix} \begin{bmatrix} z \\ y \end{bmatrix} \approx \begin{bmatrix} C^T e_{m_s} \\ \xi e_1 \end{bmatrix}. \tag{5.3}$$

Having to solve only (5.3) is the ideal situation because this problem can be solved as two separate minimization problems: the first to compute y, the second to compute z. However, we do not need y explicitly, only the products $V_m y$ and $B_m y = C^T A V_m y$, which can be obtained from short-term recurrences using a MINRES-type approach without keeping the vectors V_m around. The short-term recurrences are similar to the MINRES algorithm and therefore their derivation is left out for brevity.

ALGORITHM 2b: based on solving (5.3)

For each γ

- Determine $V_m y$ using short-term recurrences.

- Determine z.
- Set $x_{s,\omega}^{(j)} = V_m y + Uz$.

On the other hand, if the columns of U do not span an invariant subspace of $A^{(j)}$, we should solve (5.2). Unfortunately, there exists no short-term recurrence for $V_m y$ in this case, and thus we are forced to save the V_m in order to form $x_{s,\omega}^{(j)}$. However, for a fixed source, only one set of vectors V_m needs to be saved from which solutions at all other frequencies can be computed. Moreover, the generation of V_m itself requires only short-term recurrences, so no extra work is required.

We advocate Algorithm 2b when storage is at a premium and/or when we know that U is a good approximate invariant subspace, and Algorithm 2a otherwise.

We make one further modification to Algorithm 1 to solve these shifted systems simultaneously with the nonshifted systems. We change step 3(a) to append to U also the column $\text{imag}(x_{s,\omega}^{(curr)})$ and possibly $\text{imag}(x_{s,\omega}^{(beg)})$ depending on which part of the conditional statement is executed. The justification is as follows. Let $E = A^{(j)} - A^{(k)}$, and therefore $E = A_\omega^{(j)} - A_\omega^{(k)}$ as well. One can show that $A^{(j)}(x_s^{(j)} - x_s^{(k)}) = Ex_s^{(k)}$ and $A_\omega^{(j)}(x_{s,\omega}^{(j)} - x_{s,\omega}^{(k)}) = Ex_{s,\omega}^{(k)}$. Given the eigendecomposition $A^{(j)} = Q\Lambda Q^T$, we have

$$x_{s,\omega}^{(k)} = (A^{(k)} + i\gamma I)^{-1} e_{m_s} = \sum_\ell q_\ell \frac{q_\ell^T e_{m_s}}{\lambda_\ell + i\gamma},$$

where γ is reasonably small compared to the (real) eigenvalues λ_ℓ. Hence, the real part of $x_{s,\omega}^{(k)}$ is close to $x_s^{(k)}$, and the difference between $Ex_s^{(k)}$ and $Ex_{s,\omega}^{(k)}$ is primarily due to the (small) imaginary part of $x_{s,\omega}^{(k)}$. Since we look for solutions to the complex system in $\text{range}(U) \oplus \text{range}(V_m)$, it makes sense to include $\text{imag}(x_{s,\omega}^{(k)})$ in U.

We expect either algorithm to perform sufficiently well when $\gamma = h^2\omega/\nu$ is not too large in an absolute sense. In our application, γ will typically be less than or equal to $O(10^{-4})$, and we will not be solving the system for very many values of γ. This means that if MINRES had been applied directly to the shifted system, the number of iterations required for convergence would be about the same as the number of iterations required for the unshifted system. However, if γ is much larger, neither algorithm will necessarily produce solutions with a small relative residual norm. This stems from the fact that in solving the projected problem, we may leave out directions from the Krylov subspace in which the solutions to the complex systems have large components. On the other hand, if γ is very large relative to the smallest eigenvalue of the $A^{(j)}$ (which is not the case in our application), the eigenvalues of the shifted matrix will be well clustered. In the latter case, MINRES applied to this shifted problem would converge quickly, and recycling may not be necessary.

5.5. Computational issues. In this subsection, we briefly outline the computational complexity and storage issues associated with Algorithm 1 and its variants. We focus on overhead introduced by our algorithmic changes compared with MINRES. This overhead is offset by the reduction in iterations. Clearly, due to storage limitations we cannot have too many columns for U if the dimension of the problem is large. In addition, we must balance the cost of orthogonalizing the Krylov vectors against the columns of C with the reduction in the total number of iterations times the cost per iteration. If the cost of the matrix-vector product and preconditioning is large we can invest more time in orthogonalizations to reduce the number of iterations. In general, the idea is to find U with as few columns as possible such that the number of MINRES iterations is significantly reduced.

Computational cost. We consider only the complexity of computational steps that are linear in the size of the system matrix, i.e., $O(N)$. The costs of the various computations on small matrices, such as the solution of least squares problems and the computation of (selected) eigenvalues and eigenvectors, are negligible.

The main overhead introduced by our method compared with standard MINRES is the orthogonalization of each Krylov vector against the matrix C, indicated by the left multiplication by P in step 3(e) in Algorithm 1. Let the number of columns in C be n_c. Then the orthogonalizations introduce a cost of $4n_cN$ floating point operations (flops) per iteration. For each linear system, the initialization of x_0 and r_0 in 3(c) and 3(d) takes about $4n_cN$ flops, and the correction to the solution with the term $U(B_my)$, $x = (x_0 + V_my) - UB_my$ (see [10]), after the MINRES iteration, costs an additional $2n_cN$ flops. For each new matrix (but not for each right-hand side) we have the cost of n_c matrix-vector products and the cost of the QR decomposition of an N-by-n_c matrix (about $2n_c^2N$ flops) in 3(b). If k indicates the average number of nonzeros per row in the matrix, the cost for the matrix-vector products from which C is formed is $2n_ckN$ flops. However, these matrix-vector products have been accounted for in Figures 6 and 11, whereas they do not correspond to iterations. In that respect they actually represent a reduction in computational cost compared with MINRES. Note that, in general, the multiplication of the matrix times a block of vectors takes less time than multiplying each vector separately. Since we vary one or two columns in C for each right-hand side there are an additional one or two matrix-vector products per right-hand side.

Finally, we use the information generated by the solver for the first system matrix and a few right-hand sides to approximate invariant subspace information. If we approximate eigenvectors over range(V_m), the only $O(N)$ cost arises from constructing approximate eigenvectors for $A^{(1)}$ from the eigenvectors of the $m \times m$ leading principal submatrix of $\underline{T}_m$ (5.1) generated in the MINRES iteration. This costs approximately $2mN$ flops per approximate eigenvector, where m is the length of the MINRES run at the time the invariant subspace is computed. This may vary and need not be at the end of the MINRES run. Approximating eigenvectors over range($[V_m\ U]$) takes approximately $2(m + n_c)N$ flops per eigenvector plus $2(n_c^2 + mn_c)N$ flops to set up the generalized eigenvalue problem. Since the invariant subspace associated with the smallest eigenvalues changes only little in our application and high accuracy is not needed, this is done typically only for the first few eigenvectors for one of the matrices early in the GN iteration. Thus, these costs are incurred only once.

For Algorithm 2a, the only additional cost of $O(N)$ arises from the orthogonalization of the U matrix against the matrices V_{m+1} and C and an additional $2n_cN$ flops for updating the residual with the $\hat{U}$ component. For the solution there is no change in cost with respect to Algorithm 1. The additional cost of the orthogonalization is $2n_c(2m+3n_c)N$. Note that there is significant overlap with the computation of eigenvectors. Thus, in steps where eigenvectors are computed or updated the additional cost is negligible. Finally, note that the significant reduction in the number of iterations per right-hand side helps in reducing the actual overhead. For Algorithm 2b no additional costs arise. The matrix Δ is available from the computation of the invariant subspace.

Parallel implementation. Realistic problems from DOT are very large and therefore well suited to straightforward parallelization by distributing each matrix row-wise following a good domain partitioning. The special structure of the matrix (see section 4) allows this in spite of the factored form of the matrix. Therefore, the

parallel implementation of the algorithms outlined above can follow the discussion in, for example, [13, 12]. In addition, we could parallelize over the right-hand sides. This can be done with minor variations of the algorithms presented here. Since we can solve for multiple shifts with a single Krylov subspace, there is little benefit in parallelizing over the shifts.

6. Numerical results. In this section, we give the results of our proposed algorithm on two sequences of matrices generated from two different runs of the parametric nonlinear inversion scheme outlined previously. In the first experiment, a piecewise constant model for diffusion and absorption was used. In the second, we used a nonhomogenous model for the diffusion and absorption in the background whereas the diffusion and absorption inside the anomalies were constant. There were 16 sources and 32 detectors. In both experiments, the nonlinear inversion scheme was run using only data for the 0 frequency case; however, we apply our algorithm to both the 0 frequency case and shifted systems at 5 MHz in order to test our algorithm. The region was discretized[3] into $31 \times 31 \times 21$ voxels of volume h^3, with $h = .2$ cm. The sizes of the matrices in both experiments were $18{,}259 \times 18{,}259$. The sources and detectors were located in a 3 cm $\times$ 3 cm plane over the center of the grid. The starting guesses for the ellipsoids describing the anomaly were the largest possible ellipsoids fitting in the 3 cm $\times$ 3 cm $\times$ 4 cm region under the sources. Starting guesses for the other parameters were then obtained by fixing the shape parameters and using 1–5 GN steps to find the best values for those starting ellipsoids.

All experiments were conducted in MATLAB using IEEE double precision floating point arithmetic.

6.1. Experiment 1. We ran our algorithm on the first 40 systems that were generated by a damped GN run trying to reconstruct piecewise constant absorption and diffusion images. Systems numbered 2, 5, 7, 10, 13–19 (odd), 22–40 (even) correspond to the beginning of a line search; systems numbered 4, 6, 9, 12, 14–18 (even), 21–41 (odd) correspond to the end of a line search; and the remaining systems correspond to the middle of a line search.

First, we test our hypothesis that the invariant subspaces corresponding to the smallest eigenvalues of these matrices do not change much, whether we compare within a line search or across line searches. Recall that if the columns of $W^{(j)}$ form an orthonormal basis for the eigenspace associated with the smallest M eigenvalues for matrix $A^{(j)}$, and the same holds for $W^{(k)}$ and $A^{(k)}$, then the cosines of the canonical angles between range$(W^{(k)})$ and range$(W^{(j)})$ are given by [33, Corollary I.5.4]

$$\cos\Theta[\text{range}(W^{(j)}), \text{range}(W^{(k)})] = \Sigma[(W^{(j)})^T W^{(k)}],$$

where $\Sigma[V]$ denotes the singular values of the argument V. The sines of the canonical angles are therefore $\sqrt{1-\sigma_i^2}$, where the σ_i denote the cosines of the canonical angles. In Figure 3, we display the sines of the canonical angles between pairs of subspaces corresponding to the smallest eigenvalues of matrices. We observe that the invariant subspaces corresponding to the smallest eigenvalues are in fact relatively close, as predicted by Theorem 4.1. The plots for the sines with $M = 3$ and $M = 12$ illustrate the relative insensitivity of the invariant subspaces corresponding to the smallest 3 and smallest 12 eigenvalues, respectively. The indices have no correspondence with

[3]These experiments represent small test cases designed to test the regularization scheme itself. Ideal practical implementations of the inversion routine, which are not feasible without fast forward solvers such as those we present here, will require voxelations giving millions of unknowns.

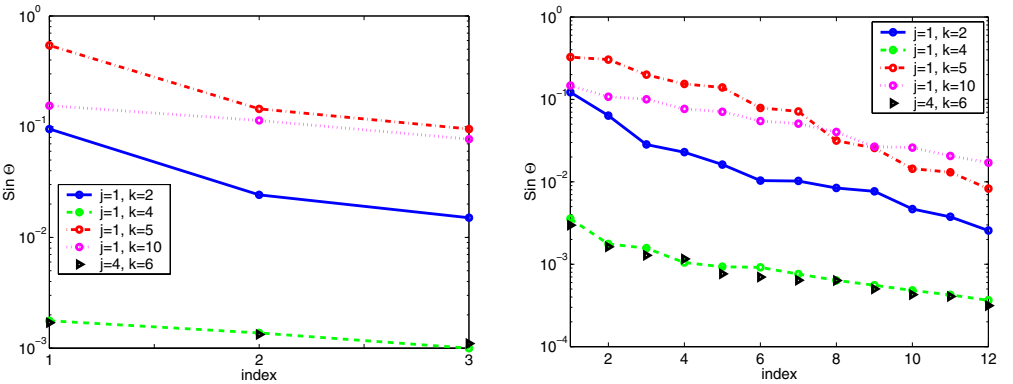


FIG. 3. *Experiment* 1. *Plots of* $\sin\Theta[\text{range}(W^{(j)}), \text{range}(W^{(k)})]$ *for various* (j,k) *for subspaces of dimension* $M = 3$ *(left) and* $M = 12$ *(right).*

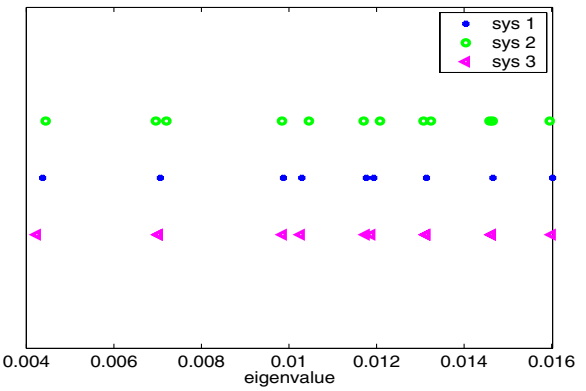


FIG. 4. *Experiment* 1. *Smallest* 12 *eigenvalues of* $A^{(j)}$, $j = 1, 2, 3$.

the eigenvectors themselves, nor do the canonical angles reflect the angles between corresponding eigenvectors. The values of j and k in the pictures were selected to illustrate the fact that the relevant invariant subspaces of matrices corresponding to (final) updates in the GN process remain fairly close to each other (e.g., 1 and 4, 4 and 6), whereas those from matrices from distinct phases in the line search differ more (e.g., 1 and 5, 1 and 10). Nevertheless, even these do not differ that much, particularly if a larger dimensional invariant subspace is used. The smallest 12 eigenvalues of $A^{(j)}$ for $j = 1, 2, 3$ are given in Figure 4.

We tested Algorithm 1, adjusted according to section 5.3 for the multiple right-hand side problem for zero frequency, and also adjusted for an additional nonzero frequency of $\omega = 5$ MHz using Algorithms 2a and 2b. For the first 6 right-hand sides in the initial phase of Algorithm 1, we saved 2 harmonic Ritz vectors at the end of each MINRES run to approximate the invariant subspace of $A^{(1)}$ corresponding to the 12 smallest eigenvalues. We used a threshold value of 10^{-3}, derived by trial and error, to distinguish between the beginning of a line search step and steps near the end. The left plot in Figure 5 shows the magnitudes of the spectral coefficients of the initial residuals for systems 2 through 5 which correspond to the 50 smallest eigenvalues, while the right plot gives the magnitudes corresponding to the 50 largest eigenvalues. Systems 2, 3, and 4 are all from the same line search, and we observe that the spectral

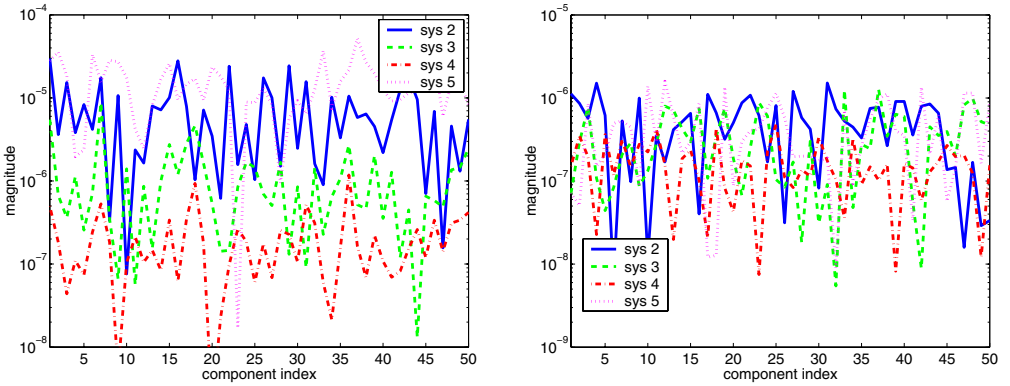


FIG. 5. *Experiment* 1. *Left: Plots of* $(W^{(j)})^T r_0$ *for* $j = 2, 3, 4, 5$ *for source* 1*, where* $W^{(j)}$ *corresponds to the eigenvector matrix associated with the* 50 *smallest eigenvalues of* $A^{(j)}$*. Right: Same, except for* 50 *largest eigenvalues of* $A^{(j)}$*.*

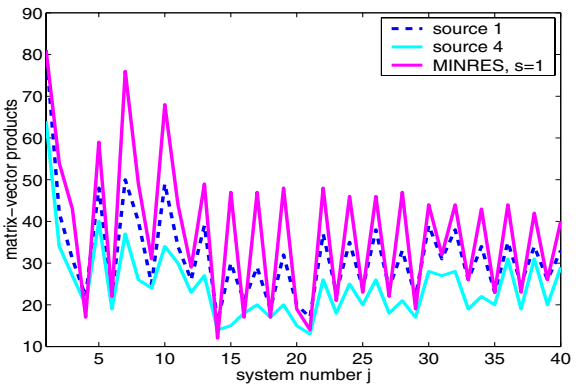


FIG. 6. *Experiment* 1. *Number of matrix-vector products per system solve for systems* 1:40 *and sources* 1 *and* 4*. The number of matrix-vector products for source* 4 *is representative of all sources after source* 1*. For comparison we give the number of matrix-vector products required for MINRES on the systems for source* 1*, where the starting guesses for each system were the solutions at the end of the most recent line search. Without subspace recycling, these numbers for source* 1 *are representative for all sources.*

components decrease by roughly one order of magnitude with each system. System 5 corresponds to the beginning of a new line search whose matrix and solution are not as close to those from the first line search. We observe a corresponding increase in the magnitudes of the spectral coefficients over the small eigenvalues, whereas a comparison of the figures shows that the initial residual is typically smaller over the subspace corresponding to the largest eigenvalues. This behavior is consistent both with the analysis of the initial residual in section 5.2 and with the observed convergence behavior for system 5 in the sense that the solver must work harder to reduce the residual significantly over these components.

Figure 6 gives the total number of matrix-vector products to solve each system using our recycling algorithm. This number includes the matrix-vector products required to compute the columns of C. Note that these represent a lower computational cost than actual iterations. The residuals for all the real systems were required to have a relative norm of 10^{-6}. Note the savings in matrix-vector products for the

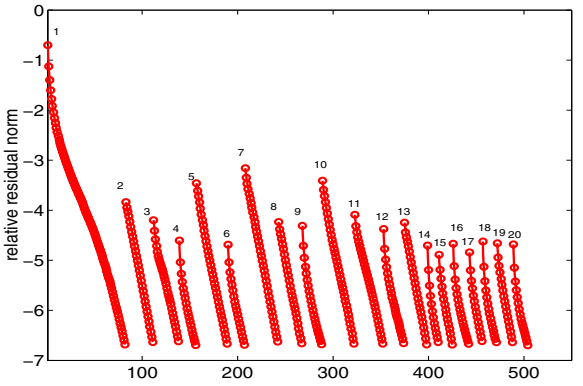


FIG. 7. *Experiment* 1. *Relative residual norms are displayed for the first* 20 *systems, source* 1.

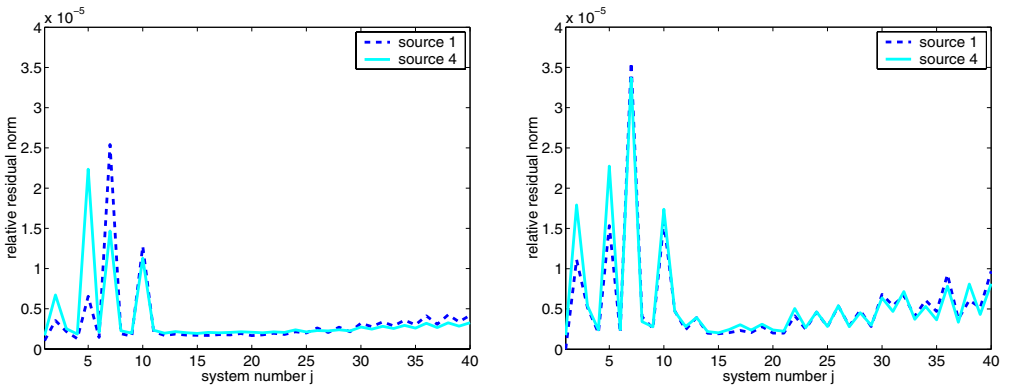


FIG. 8. *Experiment* 1. *Left: Relative residual norm per system solve, systems* 1:40*, sources* 1 *and* 4*, for* $\omega = 5$ *MHz, results computed using Algorithm* 2a. *Right: Relative residual norm per system solve, systems* 1:40*, sources* 1 *and* 4*, for* $\omega = 5$ *MHz computed using Algorithm* 2b.

right-hand sides other than the first of a single system, because most of the columns of C are computed only once for each s. Moreover, the projection has the desired effect of reducing the total number of iterations required for the projected system.

For comparison purposes, we note that if we used MINRES with a zero starting guess for every system and right-hand side, the number of matrix-vector products would be roughly constant at about 81 iterations for each system. Even MINRES with the solution at the end of the most recent line search as a starting guess could not achieve the reduction in the number of iterations we achieve with our algorithm, as demonstrated in Figure 6. Additionally, the matrix-matrix product $(A^{(j)}U)$ performed prior to running MINRES on the projected system runs faster than the equivalent number of matrix-vector products performed inside (unprojected) MINRES.

The relative residual norms for the first 20 systems, for source 1, are given in Figure 7. Note the convergence rate becomes higher and the initial relative residual norm becomes smaller as we move through one sequence of systems in a line search.

The plot in Figure 8 illustrates the relative residual norms that are achieved when solving the complex system with $\omega = 5$ MHz when Algorithms 2a and 2b are used to update the complex solution vectors. In neither case do we exactly attain a relative residual norm of 10^{-6}, the stopping criterion for the corresponding real

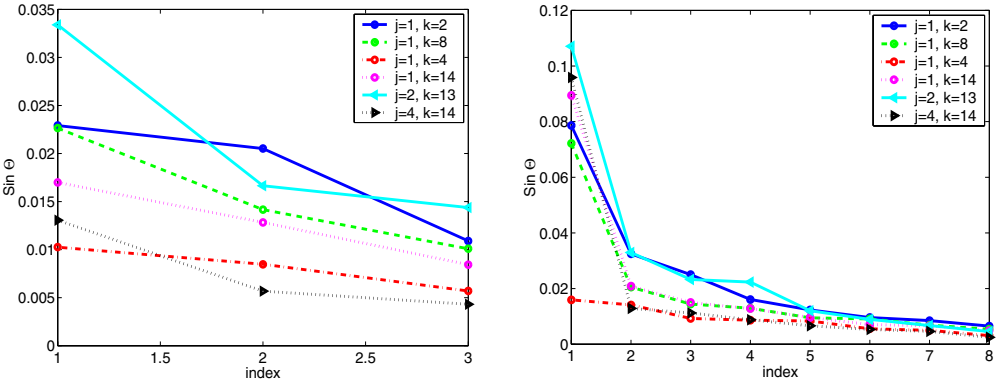


FIG. 9. *Experiment* 2. *Left: Plots of* $\sin\Theta[\text{range}(W^{(j)}), \text{range}(W^{(k)})]$ *for various* (j,k) *assuming a subspace dimension of* 3. *Right: Assuming a subspace dimension of* 8.

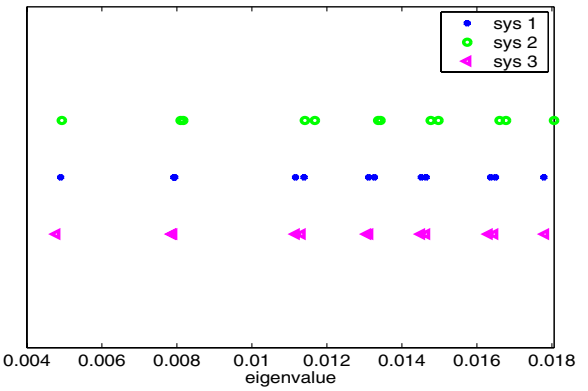


FIG. 10. *Experiment* 2. *Plot of smallest* 12 *eigenvalues for* $A^{(j)}$, $j = 1, 2, 3$. *Note that the smallest magnitude eigenvalues remain in clusters and do not cross clusters.*

system. However, for our application we feel this is sufficient, and perhaps more than sufficient since the measured data that defines the least squares problem contains noise at a level several orders of magnitude larger than 10^{-6}. On the other hand, we need to take the (unknown) conditioning of the algebraic system into account. Since previous solutions occur in the U matrix, this accounts for the slight upward creep of the graphs toward the end of the sequence of systems.

6.2. Experiment 2. In this experiment, the background diffusion and absorption were generated to have a "lumpy" variation [20], and thus the matrices that were generated correspond to piecewise continuous, rather than piecewise constant, absorption and diffusion. The total number of GN steps was 24 and the total number of system matrices was 51. For the first 40 systems the indices corresponding to the beginning of a line search are 2, 5, 8, 10, 13–39 (odd), while indices corresponding to the end of a line search are 4 ,7, 9, 12, 14–40 (even).

The sines of the canonical angles between different pairs of invariant subspaces associated with the smallest eigenvalues and for different subspace dimensions are given in Figure 9. In Figure 10 the smallest 12 eigenvalues are displayed. Consistent with Theorem 4.1 and our conjecture that the changes in the matrices are concentrated

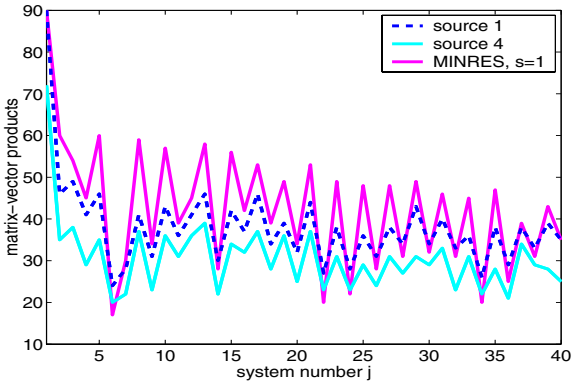


FIG. 11. *Experiment* 2. *Number of matrix-vector products per system solve for systems* 1:40 *and sources* 1 *and* 4. *The number of matrix-vector products for source* 4 *is representative of all sources after source* 1. *For comparison we give the number of matrix-vector products required for MINRES on the systems for source* 1 *where the starting guesses for each system were the solutions at the end of the most recent line search. Without subspace recycling, these numbers for source* 1 *are representative for all sources.*

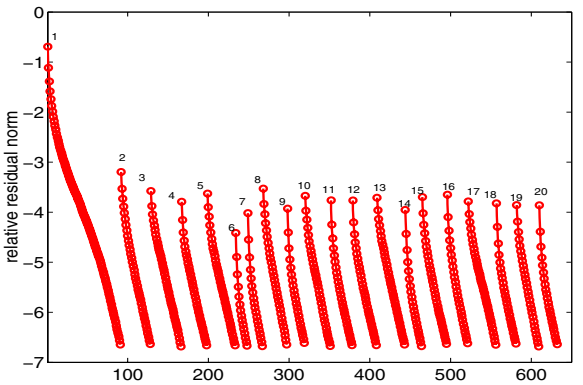


FIG. 12. *Experiment* 2. *Relative residual norms for systems* 1 *to* 20, *source* 1.

in the invariant subspaces corresponding to higher frequencies (larger eigenvalues), the smallest eigenvalues remain in disjoint clusters.

Figures 9 and 10 seem to indicate that the smallest 8 or so eigenvalues correspond to an invariant subspace that remains well separated from its orthogonal complement. Therefore, in the initialization phase of Algorithm 1, we added two vectors to W corresponding to the smallest harmonic Ritz values of $\tilde{T}_m$ (i.e., the matrix in (5.3) with $\gamma = 0$) for each of the first 4 sources. In Figure 11, we see the effect of keeping these 8 columns plus the other vectors proposed in section 5.3. In this experiment, $W_{ind} = W$, and a threshold value was used to distinguish the phase within the line search. Just as for the first experiment, note the savings in matrix-vector products for the right-hand sides after the first for a single system. Again, we compare our results to MINRES with $x^{(curr)}$ from Algorithm 1 as the starting guess.

Relative residual norms for Algorithm 1 for the first 20 systems for source 1 are given in Figure 12. We observe an increased rate of convergence and smaller initial residuals in the course of a line search.

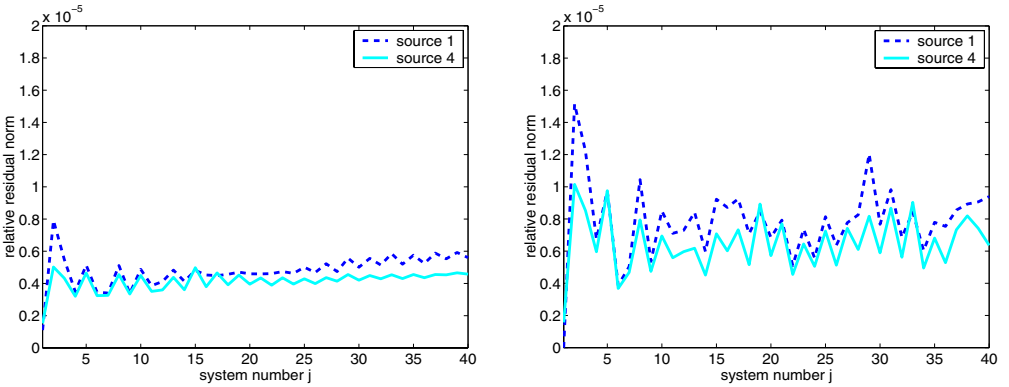


FIG. 13. *Experiment* 2. *Left: Relative residual norms for systems* 1 *to* 40*, sources* 1 *and* 4*,* $\omega = 5$ *MHz using Algorithm* 2a. *Right: Relative residual norms for systems* 1 *to* 40*, sources* 1 *and* 4*,* $\omega = 5$ *MHz using Algorithm* 2b.

Finally, the relative residual norms for the complex systems at $\omega = 5$ MHz are given in Figure 13 for Algorithm 2a and 2b. In general, the behavior is more uniform for these systems versus those in the first experiment since the invariant subspaces corresponding to the small eigenvalues among the matrices are more closely related.

7. Conclusions and future work. We have discussed various strategies for Krylov subspace recycling to improve the convergence of linear solvers for a sequence of slowly changing linear systems arising in computations for optical tomography. We have combined strategies of recycling approximate invariant subspaces and strategies of recycling subspaces from previous solutions. Furthermore, our algorithms are based on a careful analysis of which strategy is most useful at each stage of the optimization algorithm. This analysis also takes the underlying application, DOT, and matrix symmetry into account. Additionally, we have adapted the GCRO algorithm to combine subspace recycling with solving for multiple shifted systems using a single Krylov subspace. Our numerical results, based on two model problems for DOT, show that our strategies are quite effective. Although we have focused on a particular application and optimization algorithm, we feel that this approach to tuning the linear solver is applicable generally to problems where many linear systems must be solved.

Important future work includes the study of how characteristics of the matrices arising in DOT, such as invariant subspaces and eigenvalues, change for small changes in model parameters. This may lead to further improvements for linear solvers and also improved line search strategies for the nonlinear solver. This issue is, of course, equally important for other applications where we must solve a long sequence of slowly changing problems, such as crack propagation [26]. Future work will combine modeling aspects from applications with matrix theory. Other useful extensions that we plan to research are the use of GCROT-like techniques [11] to measure the effectiveness of the recycled subspace and update the space accordingly, to combine our current algorithms with a block approach [37], and to further tune the nonlinear algorithm, in particular the line search.

Acknowledgment. We would like to thank the referees for their comments, which helped us to improve the presentation of this paper.

REFERENCES

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, Jr., and J. W. Painter, *The multi-grid method for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 430–454.

[2] S. R. Arridge, *Optical tomography in medical imaging*, Inverse Problems, 16 (1999), pp. R41–R93.

[3] A. H. Baker, E. R. Jessup, and T. Manteuffel, *A technique for accelerating the convergence of restarted GMRES*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 962–984.

[4] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1993.

[5] A. Brandt, *Multigrid Techniques:* 1984 *Guide, with Applications to Fluid Dynamics*, GMD-Stud. 85, St. Augustin, Germany, 1984.

[6] M. Celia and W. Gray, *Numerical Methods for Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1992.

[7] T. F. Chan and M. K. Ng, *Galerkin projection methods for solving multiple linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 836–850.

[8] T. F. Chan and W. L. Wan, *Analysis of projection methods for solving linear systems with multiple right-hand sides*, SIAM J. Sci. Comput., 18 (1997), pp. 1698–1721.

[9] E. de Sturler, *Inner-outer methods with deflation for linear systems with multiple right-hand sides*, in Householder Symposium XIII, Proceedings of the Householder Symposium on Numerical Algebra, Pontresina, Switzerland, 1996, pp. 193–196.

[10] E. de Sturler, *Nested Krylov methods based on GCR*, J. Comput. Appl. Math., 67 (1996), pp. 15–41.

[11] E. de Sturler, *Truncation strategies for optimal Krylov subspace methods*, SIAM J. Numer. Anal., 36 (1999), pp. 864–889.

[12] E. De Sturler and H. A. Van der Vorst, *Communication cost reduction for Krylov methods on parallel computers*, in High-Performance Computing and Networking, Lecture Notes in Comput. Sci. 797, W. Gentzsch and U. Harms, eds., Springer-Verlag, Berlin, 1994, pp. 190–195.

[13] E. De Sturler and H. A. Van der Vorst, *Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers*, Appl. Numer. Math., 18 (1995), pp. 441–459.

[14] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics Appl. Math. 16, SIAM, Philadelphia, 1996.

[15] M. Eiermann, O. G. Ernst, and O. Schneider, *Analysis of acceleration strategies for restarted minimal residual methods*, J. Comput. Appl. Math., 123 (2000), pp. 261–292.

[16] P. F. Fischer, *Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 193–204.

[17] R. Freund and M. Malhotra, *A block QMR algorithm for non-Hermitian linear systems with multiple right-hand sides*, Linear Algebra Appl., 254 (1997), pp. 197–257.

[18] A. Frommer and U. Glässner, *Restarted GMRES for shifted linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 15–26.

[19] A. Frommer and P. Maass, *Fast CG-based methods for Tikhonov–Phillips regularization*, SIAM J. Sci. Comput., 20 (1999), pp. 1831–1850.

[20] M. E. Kilmer, E. Miller, A. Barbaro, and D. Boas, *3D shape-based imaging of absorption perturbation for diffuse optical tomography*, Appl. Optics, 42 (2003), pp. 3129–3144.

[21] M. Kilmer, E. Miller, and C. Rappaport, *QMR-based projection techniques for the solution of non-Hermitian systems with multiple right-hand sides*, SIAM J. Sci. Comput., 23 (2001), pp. 761–780.

[22] M. E. Kilmer and D. P. O'Leary, *Choosing regularization parameters in iterative methods for ill-posed problems*, SIAM J. Matrix Anal. Appl., 22 (2001), pp. 1204–1221.

[23] R. B. Morgan, *Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations*, SIAM J. Matrix Anal. Appl., 21 (2000), pp. 1112–1135.

[24] R. B. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[25] C. C. Paige and M. A. Saunders, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[26] M. L. Parks, E. de Sturler, G. Mackey, D. D. Johnson, and S. Maiti, *Recycling Krylov Subspaces for Sequences of Linear Systems*, Technical report UIUCDCS-R-2004-2421/UILU-ENG-2004-1722, Department of Computer Science, University of Illinois at Urbana-Champaign, 2004. SIAM J. Sci. Comput., submitted.

[27] C. Rey and F. Risler, *A Rayleigh-Ritz preconditioner for the iterative solution to large scale nonlinear problems*, Numer. Algorithms, 17 (1998), pp. 279–311.
[28] F. Risler and C. Rey, *Iterative accelerating algorithms with Krylov subspaces for the solution to large-scale nonlinear problems*, Numer. Algorithms, 23 (2000), pp. 1–30.
[29] Y. Saad, *Analysis of augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 435–449.
[30] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
[31] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput., 16 (1995), pp. 917–933.
[32] V. Simoncini and D. Szyld, *On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods*, SIAM Rev., 47 (2005), pp. 247–272.
[33] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, 1990.
[34] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2000.
[35] H. A. Van der Vorst and C. Vuik, *The superlinear convergence behaviour of GMRES*, J. Comput. Appl. Math., 48 (1993), pp. 327–341.
[36] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers Appl. Math. 23, SIAM, Philadelphia, 2002.
[37] R. Yu, E. de Sturler, and D. D. Johnson, *A Block Iterative Solver for Complex Non-Hermitian Systems Applied to Large-Scale Electronic-Structure Calculations*, Technical report UIUCDCS-R-2002-2299 and UILU-ENG-2002-1742, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

SIAM J. SCI. COMPUT.
Vol. 27, No. 6, pp. 2167–2182
© 2006 Society for Industrial and Applied Mathematics

BOUNDING THE SOLUTIONS OF PARAMETER DEPENDENT NONLINEAR ORDINARY DIFFERENTIAL EQUATIONS*

ADAM B. SINGER† AND PAUL I. BARTON†

Abstract. This paper presents two techniques for generating rigorous bounds on the solution of parameter dependent nonlinear ordinary differential equations. The first technique is an extension of differential inequalities that enables the construction of tight time varying state bounds by utilizing prior knowledge of the solution of the differential equations. The second technique provides a method for constructing pointwise in time convex and concave relaxations of the image of the solution of a system of parameter dependent differential equations on subsets of a Euclidean space. Two examples are presented to demonstrate the construction of the bounds. The examples include a brief discussion of a computer program written to automatically generate the bounds.

Key words. differential inequalities, dynamic optimization, convex relaxations

AMS subject classifications. 34A40, 90C26

DOI. 10.1137/040604388

1. Introduction.

Many applications involve the study of nonlinear ordinary differential equations (ODEs) dependent on parameters:

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{p}), \quad \mathbf{x}(t_0) = \mathbf{x}_0(\mathbf{p}), \tag{1.1}$$

where $\mathbf{x}(t, \mathbf{p}) \in \mathbb{R}^{n_x}$ and $\mathbf{p} \in P \subset \mathbb{R}^{n_p}$. This paper studies two distinct problems concerning bounds on the solutions $\mathbf{x}(t, \mathbf{p})$ for $\mathbf{p} \in P$. The first problem concerns computing tight time varying state bounds $\mathbf{x}^L(t)$ and $\mathbf{x}^U(t)$ containing the image of $\mathbf{p}$ under the solution of (1.1). The second issue concerns deriving pointwise in time convex underestimators and concave overestimators on P of the solution of (1.1). The original motivation for studying both issues stems from the development of algorithms for global optimization with ODEs embedded [8]; for linear systems, we have already addressed both issues [8]. This paper focuses exclusively on the theory for generating bounds for nonlinear parameter dependent differential equations; application of this theory to global optimization problems is found in [7].

The majority of research concerning the bounding of differential equations originated with the study of differential inequalities. While the roots of differential inequalities may be traced to the early part of the 20th century, the book by Walter [11] provides an excellent overview of the salient concepts. One important application of differential inequalities is computing time varying lower and upper bounds on the solution of systems of ODEs. Essentially, the analysis demonstrates that solving differential equations whose right-hand sides underestimate the right-hand sides of the original system at all points including the initial condition provides rigorous lower bounds for the original state variables. An analogous result applies to constructing upper bounds. Harrison [2] extended these results from Walter to parameter embedded ODEs. Unfortunately, except under stringent conditions of quasimonotonicity,

*Received by the editors February 23, 2004; accepted for publication (in revised form) August 17, 2005; published electronically February 21, 2006. This material is based upon work supported by the National Science Foundation under grant 0120441.

http://www.siam.org/journals/sisc/27-6/60438.html

†Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 (absinger@alum.mit.edu, pib@mit.edu).

the bounds generated via differential inequalities are typically weak and often suffer from bound explosion, a situation in which the solution of the bounding differential equations tends rapidly toward infinity, hence rendering the resulting bounds effectively useless on some time scales of interest.

The remainder of the paper is divided into five sections. In the first section, a method is presented for computing tight time varying state bounds for a system of ODEs, given prior information concerning the evolution of the solution to the ODEs. Utilizing the tight state bounds, the second section details a technique for generating pointwise in time parameter dependent convex and concave relaxations for the solution of parameter dependent ODEs. Next, an illustrative example is presented demonstrating the application of the state bounding theory. As will be evidenced, the application of the theory is complex even for small problems. The next section briefly discusses a program, GDOC, that was developed to automate the application of the theory; the program utilizes code generation. Finally, a short section applies GDOC to an analytically intractable nonlinear problem to generate a convex and concave relaxation.

2. Time varying state bounds. As previously mentioned, the theory of differential inequalities provides a method for computing time varying bounds on the solution of a system of differential equations. In particular, we are interested in computing a superset of the image of the solution of a system of ODEs on a parameter set. The traditional techniques such as Harrison's technique depend exclusively on information obtained from the mathematical statement of the differential equations and their corresponding initial conditions. However, differential equations of practical interest to scientists and engineers are derived from physical systems for which more information is known about the behavior of the system dynamics than the information embodied by the differential equations alone. For example, for a system undergoing decay, the state never exceeds its initial condition. In another example, differential equations modeling mass and heat transfer obey conservation principles. In this section, we demonstrate how these natural bounds and differential invariants may be applied in conjunction with differential inequalities to produce tight bounds on the solution of ODEs. We make one very important note before beginning this discussion, which is that this technique is not applicable to bounds imposed upon the system such as path constraints or bounds on a control variable. The technique is only applicable given additional information that naturally governs the evolution of the solution of the ODEs.

For the duration of this paper, two relatively mild assumptions are made concerning the solution of the ODEs. First, we assume that a solution exists and is unique for each $\mathbf{p} \in P$. This assumption permits the use of regular inequalities rather than strict inequalities in the results. Second, we assume that the ODEs possess a solution in the sense of Carathéodory. That is, an admissible solution is one that satisfies the ODEs almost everywhere (a.e.). Therefore, this assumption immediately implies that our derivative exists everywhere except possibly on a set of measure zero. This exception poses no difficulty, for at points where the derivative does not exist, we can either replace the derivative with the Dini derivative or simply remove the points of nonexistence by extending the function by continuation without altering the solution of the ODEs. We begin by presenting a slight variant of a fundamental lemma from Walter [11].

LEMMA 2.1. *Suppose the vector functions $\boldsymbol{\varphi}(t)$ and $\boldsymbol{\psi}(t)$ are differentiable a.e. in $(t_0, t_f]$. For some index i and fixed $\underline{t} \in (t_0, t_f]$, if $\dot{\varphi}_i(\underline{t}) < \dot{\psi}_i(\underline{t})$ when $\boldsymbol{\varphi}(\underline{t}) \leq \boldsymbol{\psi}(\underline{t})$,*

$\varphi_i(\underline{t}) = \psi_i(\underline{t})$, then we have precisely one of the following two cases:

(i) $\boldsymbol{\varphi} < \boldsymbol{\psi}$ *in* $(t_0, t_f]$,

(ii) $\boldsymbol{\varphi}(t_{0^+}) < \boldsymbol{\psi}(t_{0^+})$ *does not hold; i.e., there exists an arbitrary small* $\bar{t} \in (t_0, t_f]$ *such that* $\varphi_i(\bar{t}) \geq \psi_i(\bar{t})$ *for at least one index* i.

Proof. For the proof, see Lemma 12.I in [11]. □

Remark 2.2. Clearly, a symmetric result for $\boldsymbol{\varphi} > \boldsymbol{\psi}$ holds.

We wish to extend Lemma 2.1 to a method for bounding the solution of (1.1) given prior knowledge concerning some set $\overline{\mathcal{X}}(t, \mathbf{p})$ containing the solution of (1.1). For example, given an ODE that describes the consumption of a chemical species, the set $\overline{\mathcal{X}}(t, \mathbf{p})$ could be defined as $[0, x(t_0, \mathbf{p})]$, where x is the concentration of a chemical species. The derivation is performed in several steps. First, we present an extension of Walter's fundamental result concerning the construction of superfunctions and subfunctions, given the set $\overline{\mathcal{X}}(t, \mathbf{p})$. From this fundamental theorem, two corollaries are constructed. The first corollary extends the result to differential equations with embedded parameters and yields a parameter dependent solution for which the inequality holds pointwise in the parameter. A second corollary follows, which illustrates a method for constructing a solution valid for all parameter values in a given set. This second corollary yields the desired result which bounds the image of the solution to a parameter embedded ODE on a subset of a Euclidean space.

THEOREM 2.3. *Let* $\mathbf{x}(t)$ *be a solution of*

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}), \quad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.1}$$

in $(t_0, t_f]$, *and further suppose that* $\mathbf{x}(t) \in \overline{\mathcal{X}}(t)$, *where* $\overline{\mathcal{X}}(t)$ *is known independently from the solution of* (2.1). *For* $i = 1, \ldots, n_x$, *if*

(i) $\mathbf{v}(t_0) < \mathbf{x}(t_0) < \mathbf{w}(t_0)$

and if $\forall\ \mathbf{v}(t), \mathbf{w}(t) \in G(t)$,

(ii) $\dot{v}_i = g_i(t, \mathbf{v}, \mathbf{w}) < \inf_{\substack{\mathbf{z} \in \overline{\mathcal{X}}(t) \cap G(t) \\ z_i = v_i(t)}} f_i(t, \mathbf{z})$,

(iii) $\dot{w}_i = h_i(t, \mathbf{v}, \mathbf{w}) > \sup_{\substack{\mathbf{z} \in \overline{\mathcal{X}}(t) \cap G(t) \\ z_i = w_i(t)}} f_i(t, \mathbf{z})$,

where $G(t) = \{\mathbf{z} \mid \mathbf{v}(t) \leq \mathbf{z} \leq \mathbf{w}(t)\}$, *then*

$$\mathbf{v}(t) < \mathbf{x}(t) < \mathbf{w}(t) \quad \forall\ t \in (t_0, t_f].$$

Proof. Hypotheses (i), (ii), and (iii) guarantee that if there exists some $\underline{t} \in (t_0, t_f]$ such that $\mathbf{v}(\underline{t}) \leq \mathbf{x}(\underline{t}) \leq \mathbf{w}(\underline{t})$, $v_i(\underline{t}) = x_i(\underline{t}) = w_i(\underline{t})$ for some index i, we have

$$\dot{v}_i = g_i(\underline{t}, \mathbf{v}, \mathbf{w}) < \inf_{\substack{\mathbf{z} \in \overline{\mathcal{X}}(t) \cap G(t) \\ z_i = v_i(t)}} f_i(\underline{t}, \mathbf{z}) \leq f_i(\underline{t}, \mathbf{x}) = \dot{x}_i,$$

$$\dot{w}_i = h_i(\underline{t}, \mathbf{v}, \mathbf{w}) > \sup_{\substack{\mathbf{z} \in \overline{\mathcal{X}}(t) \cap G(t) \\ z_i = w_i(t)}} f_i(\underline{t}, \mathbf{z}) \geq f_i(\underline{t}, \mathbf{x}) = \dot{x}_i,$$

where the last inequality in each equation above holds because we have already established $\mathbf{x}(\underline{t}) \in G(\underline{t})$, and $\mathbf{x}(\underline{t}) \in \overline{\mathcal{X}}(\underline{t})$ by hypothesis. We have now established the hypothesis of Lemma 2.1 and therefore know that either Lemma 2.1(i) or 2.1(ii) holds. However, (i) is inconsistent with 2.1(ii). □

Remark 2.4. By asserting uniqueness of the solution of the differential equations, the hypotheses of the above theorem may be relaxed to

(i) $\mathbf{v}(t_0) \leq \mathbf{x}(t_0) \leq \mathbf{w}(t_0)$,

(ii) $\dot{v}_i = g_i(t,\mathbf{v},\mathbf{w}) \leq \inf\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t)\cap G(t)\\ z_i=v_i(t)}} f_i(t,\mathbf{z})$,

(iii) $\dot{w}_i = h_i(t,\mathbf{v},\mathbf{w}) \geq \sup\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t)\cap G(t)\\ z_i=w_i(t)}} f_i(t,\mathbf{z})$

(see Remarks 12.X in [11]). Furthermore, by asserting regular inequalities above, the result of the theorem also permits $\mathbf{v}(t) \leq \mathbf{x}(t) \leq \mathbf{w}(t)\ \forall\ t \in (t_0, t_f]$. For the remainder of this paper, since uniqueness of the solution of the differential equations has been assumed, it is understood that reference to Theorem 2.3 refers also to the regular inequalities in this remark.

The following corollary extends Theorem 2.3 to pointwise in $\mathbf{p}$ bounds for (1.1). While not needed for bounding the solution of a parameter dependent differential equation, Corollary 2.5 is needed directly for proving the results on convex bounding in the subsequent section.

COROLLARY 2.5. *Let* $\mathbf{x}(t,\mathbf{p})$ *be a solution of* (1.1)*, and let* $\mathbf{x}(t,\mathbf{p}) \in \overline{\mathcal{X}}(t,\mathbf{p})$ *for each* $\mathbf{p} \in P$*, where* $\overline{\mathcal{X}}(t,\mathbf{p})$ *is known independently from the solution of* (1.1)*. If for* $i = 1,\ldots,n_x$ *and for each* $\mathbf{p} \in P$

(i) $\mathbf{v}(t_0,\mathbf{p}) \leq \mathbf{x}(t_0,\mathbf{p}) \leq \mathbf{w}(t_0,\mathbf{p})$,

and if $\forall\ \mathbf{v}(t,\mathbf{p}), \mathbf{w}(t,\mathbf{p}) \in \overline{\mathcal{X}}(t,\mathbf{p}) \cap G(t,\mathbf{p})$,

(ii) $\dot{v}_i = g_i(t,\mathbf{v},\mathbf{w},\mathbf{p}) \leq \inf\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t,\mathbf{p})\cap G(t,\mathbf{p})\\ z_i=v_i(t,\mathbf{p})}} f_i(t,\mathbf{z},\mathbf{p})$,

(iii) $\dot{w}_i = h_i(t,\mathbf{v},\mathbf{w},\mathbf{p}) \geq \sup\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t,\mathbf{p})\cap G(t,\mathbf{p})\\ z_i=w_i(t,\mathbf{p})}} f_i(t,\mathbf{z},\mathbf{p})$,

where $G(t,\mathbf{p}) = \{\mathbf{z} \mid \mathbf{v}(t,\mathbf{p}) \leq \mathbf{z} \leq \mathbf{w}(t,\mathbf{p})\}$*, then for each fixed* $\mathbf{p} \in P$*,*

$$\mathbf{v}(t,\mathbf{p}) \leq \mathbf{x}(t,\mathbf{p}) \leq \mathbf{w}(t,\mathbf{p}) \qquad \forall\ t \in (t_0, t_f].$$

Proof. Theorem 2.3 holds for any fixed $\mathbf{p} \in P$. □

The following corollary is the fundamental result for bounding the image of a parameter embedded differential equation on the parameter set P. The resulting bounding equations are not necessarily exact. That is, the derived bounding set may actually be a superset of the true image of the solution on P.

COROLLARY 2.6. *Let* $\mathbf{x}(t,\mathbf{p})$ *be a solution of* (1.1)*, and let* $\mathbf{x}(t,\mathbf{p}) \in \overline{\mathcal{X}}(t,\mathbf{p})$ *for each* $\mathbf{p} \in P$*, where* $\overline{\mathcal{X}}(t,\mathbf{p})$ *is known independently from the solution of* (1.1)*. Furthermore, let* $\overline{\mathcal{X}}(\underline{t})$ *be defined pointwise in time by*

$$\overline{\mathcal{X}}(\underline{t}) = \left[\inf_{\mathbf{q}\in P} \overline{\mathcal{X}}(\underline{t},\mathbf{q}),\ \sup_{\mathbf{q}\in P} \overline{\mathcal{X}}(\underline{t},\mathbf{q})\right].$$

If for $i = 1,\ldots,n_x$*,*

(i) $\mathbf{v}(t_0) \leq \inf\limits_{\mathbf{q}\in P} \mathbf{x}(t_0,\mathbf{q})$,

(ii) $\mathbf{w}(t_0) \geq \sup\limits_{\mathbf{q}\in P} \mathbf{x}(t_0,\mathbf{q})$,

and if $\forall\ \mathbf{v}(t), \mathbf{w}(t) \in G(t)$

(iii) $\dot{v}_i = g_i(t,\mathbf{v},\mathbf{w}) \leq \inf\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t)\cap G(t),\ \mathbf{q}\in P\\ z_i=v_i(t)}} f_i(t,\mathbf{z},\mathbf{q})$,

(iv) $\dot{w}_i = h_i(t,\mathbf{v},\mathbf{w}) \geq \sup\limits_{\substack{\mathbf{z}\in\overline{\mathcal{X}}(t)\cap G(t),\ \mathbf{q}\in P\\ z_i=w_i(t)}} f_i(t,\mathbf{z},\mathbf{q})$,

then

$$\mathbf{v}(t) \leq \mathbf{x}(t, \mathbf{p}) \leq \mathbf{w}(t) \qquad \forall\ (t, \mathbf{p}) \in (t_0, t_f] \times P.$$

Proof. The proof is a trivial extension of the proof of Theorem 2.3. □

Remark 2.7. Because the functions $\mathbf{v}$ and $\mathbf{w}$ from the above corollary bound $\mathbf{x}(\cdot, \mathbf{p})$ for all values of $\mathbf{p} \in P$, when used in the above context, these state bounds are labeled $\mathbf{x}^L$ and $\mathbf{x}^U$, respectively. Pointwise in time, we label the interval $[\mathbf{x}^L(\underline{t}), \mathbf{x}^U(\underline{t})]$ as $\mathcal{X}(\underline{t})$ for each fixed $\underline{t} \in (t_0, t_f]$.

From inspection of Corollary 2.6, the most difficult aspect of applying the corollary appears to be bounding the solutions of the parametric optimization problems defining the right-hand sides of the bounding differential equations. While computing the exact solution to the optimization problem would yield the tightest bounds possible from this method, actually solving the optimization problems at each integration step in a numerical integration would be a prohibitively expensive task. Instead, the solution of the optimization problem on the right-hand side of the differential equation is estimated by interval arithmetic [5] pointwise in time. To apply interval arithmetic, the set $\overline{\mathcal{X}}(\underline{t}) \cap G(\underline{t})$ must be representable as an interval pointwise in time. For some fixed $\underline{t} \in (t_0, t_f]$, let $\overline{\mathcal{X}}(\underline{t}) = [\overline{\mathbf{x}}^L(\underline{t}), \overline{\mathbf{x}}^U(\underline{t})]$; then pointwise in time, the set $\overline{\mathcal{X}}(\underline{t}) \cap G(\underline{t})$ is given by

$$[\max\{\overline{x}_1^L(\underline{t}), x_1^L(\underline{t})\}, \min\{\overline{x}_1^U(\underline{t}), x_1^U(\underline{t})\}] \times \cdots \times [\max\{\overline{x}_{n_x}^L(\underline{t}), x_{n_x}^L(\underline{t})\}, \min\{\overline{x}_{n_x}^U(\underline{t}), x_{n_x}^U(\underline{t})\}].$$

We note that, given an inclusion monotonic interval function [5] for the right-hand sides of the differential equation, the bounds approach the original function when the parameter space approaches degeneracy. Therefore, satisfying the right-hand side of the bounding differential equations is a relatively straightforward task.

Another interesting aspect of formulating the bounding differential equations as constrained optimization problems is that these optimization problems may have no feasible point. Assume that such an infeasibility occurs in the bounding problem for the ith variable. Such infeasibilities usually arise from the equality constraint on the ith variable. This situation immediately implies that the ith bound lies outside the set $\overline{\mathcal{X}}(t)$. In this case, any finite value for the right-hand side of the differential equation is valid, for any finite instantaneous rate of change in the ith bounding variable ensures that the bound remains outside $\overline{\mathcal{X}}(t)$. In practice, when employing interval techniques, the interval computation provides a finite value for the right-hand side of the differential equation, regardless of the feasibility of the optimization problem.

In some special cases, the optimization problems of Corollary 2.6 are trivially satisfied by the structure of the right-hand side of (1.1). In particular, vast simplifications can be applied under certain monotonicity assumptions. Due to space limitations, however, the interested reader is referred to [6] to see these results.

3. Pointwise in time convex and concave bounds. As described in the introduction, one motivation for the results presented in this paper is algorithms for global optimization with ODEs embedded. In regular optimization problems, the objective function is expressed in terms of compositions of explicitly represented functions. In dynamic optimization problems, however, the objective function is typically defined as a function of the states and parameters, where the state variables' dependence on the optimization parameters is not known explicitly but can be evaluated via numerical solution of the ODEs. Thus, the crux of the global optimization algorithm lies in computing a convex relaxation for a composition of functions where the

functional form of the composition is unknown a priori. In his seminal paper on computing global solutions of nonconvex optimization problems [4], McCormick outlined a technique for deriving convex relaxations of functions defined by composition. In order to apply the technique, one needs to derive convex underestimators and concave overestimators for the inner functions in the composition. For dynamic optimization problems, this problem reduces to deriving pointwise in time convex underestimators and concave overestimators for the image of the solution of a nonlinear parameter dependent ODE on a parameter space. Before presenting this novel result, we define a notation for the pointwise in time linearization of a function at a reference trajectory.

DEFINITION 3.1. *Let* $\mathcal{L}_{\mathbf{f}}(t,\mathbf{y},\mathbf{q})\big|_{\mathbf{y}^*(t),\mathbf{q}^*}$ *be a linearization of* $\mathbf{f}(t,\mathbf{x},\mathbf{p})$ *at* $(\mathbf{y}^*(t),\mathbf{q}^*)$. *The linearization is given for* $i=1,\ldots,n_x$ *by the following:*

$$\mathcal{L}_{f_i}(t,\mathbf{y},\mathbf{q})\big|_{\mathbf{y}^*(t),\mathbf{q}^*} = f_i(t,\mathbf{y}^*(t),\mathbf{q}^*) + \sum_{j=1}^{n_x} \left.\frac{\partial f_i}{\partial x_j}\right|_{\mathbf{y}^*(t),\mathbf{q}^*} \left(y_j - y_j^*(t)\right) + \sum_{j=1}^{n_p} \left.\frac{\partial f_i}{\partial p_j}\right|_{\mathbf{y}^*(t),\mathbf{q}^*} \left(q_j - q_j^*\right). \tag{3.1}$$

We now present the following theorem utilized for constructing convex underestimators and concave overestimators for the solution of a parameter dependent ODE.

THEOREM 3.2. *Consider differential equation* (1.1) *and the set* $\mathcal{X}(t)$ *as defined in Remark* 2.7. *Let* P *be convex. For* $i=1,\ldots,n_x$ *and each fixed* $\underline{t}\in(t_0,t_f]$, *let* $u_i(\underline{t},\mathbf{x},\mathbf{p})$ *be a convex underestimator for* f_i *on* $\mathcal{X}(\underline{t})\times P$, *and let* $o_i(\underline{t},\mathbf{x},\mathbf{p})$ *be a concave overestimator for* f_i *on* $\mathcal{X}(\underline{t})\times P$. *For* $i=1,\ldots,n_x$, *consider the differential equations*

$$\dot{c}_i = g_{c,i}(t,\mathbf{c},\mathbf{C},\mathbf{p}) = \inf_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=c_i(t)}} \mathcal{L}_{u_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*},$$

$$\dot{C}_i = g_{C,i}(t,\mathbf{c},\mathbf{C},\mathbf{p}) = \sup_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=C_i(t)}} \mathcal{L}_{o_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*}$$

for some reference trajectory $(\mathbf{x}^*(\underline{t}),\mathbf{p}^*)\in\mathcal{X}(\underline{t})\times P$ *(differentiability of* u_i *and* o_i *is assumed along the reference trajectory) with initial conditions*

$$\mathbf{c}(t_0)\le\mathbf{x}(t_0,\mathbf{p})\le\mathbf{C}(t_0)\qquad\forall\ \mathbf{p}\in P, \tag{3.2}$$

where $\mathcal{C}(t,\mathbf{p})=\{\mathbf{z}\mid\mathbf{c}(t,\mathbf{p})\le\mathbf{z}\le\mathbf{C}(t,\mathbf{p})\}$. *Then for each fixed* $\underline{t}\in(t_0,t_f]$, $\mathbf{c}(\underline{t},\mathbf{p})$ *is a convex underestimator for* $\mathbf{x}(\underline{t},\mathbf{p})$ *on* P, *and* $\mathbf{C}(\underline{t},\mathbf{p})$ *is a concave overestimator for* $\mathbf{x}(\underline{t},\mathbf{p})$ *on* P.

Proof. Because for each fixed $\underline{t}\in(t_0,t_f]$, $u_i(\underline{t},\mathbf{x},\mathbf{p})$ is a convex underestimator for f_i on $\mathcal{X}(\underline{t})\times P$, the linearization $\mathcal{L}_{u_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(\underline{t}),\mathbf{p}^*}$ validly supports f_i on $\mathcal{X}(\underline{t})\times P$ from below. Analogously, $\mathcal{L}_{o_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(\underline{t}),\mathbf{p}^*}$ supports f_i on $\mathcal{X}(\underline{t})\times P$ from above. For each fixed $\underline{t}\in(t_0,t_f]$ and all $(\mathbf{z},\mathbf{p})\in\mathcal{X}(\underline{t})\times P$, the pointwise inequalities

$$\mathcal{L}_{u_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(\underline{t}),\mathbf{p}^*}\le u_i(\underline{t},\mathbf{z},\mathbf{p})\le f_i(\underline{t},\mathbf{z},\mathbf{p}),$$

$$\mathcal{L}_{o_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(\underline{t}),\mathbf{p}^*}\ge o_i(\underline{t},\mathbf{z},\mathbf{p})\ge f_i(\underline{t},\mathbf{z},\mathbf{p})$$

are therefore established for $i=1,\ldots,n_x$, which immediately implies

$$\dot{c}_i = \inf_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=c_i(t)}} \mathcal{L}_{u_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*}\le\inf_{\substack{\mathbf{z}\in\mathcal{X}(t)\cap\mathcal{C}(t,\mathbf{p})\\ z_i=c_i(t)}} f_i(t,\mathbf{z},\mathbf{p}), \tag{3.3a}$$

$$\dot{C}_i = \sup_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=C_i(t)}} \mathcal{L}_{o_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*} \geq \sup_{\substack{\mathbf{z}\in\mathcal{X}(t)\cap\mathcal{C}(t,\mathbf{p})\\ z_i=C_i(t)}} f_i(t,\mathbf{z},\mathbf{p}) \tag{3.3b}$$

for $i = 1,\ldots,n_x$. We note that the infimum (supremum) over $\mathcal{C}(t,\mathbf{p})$ is less (greater) than the infimum (supremum) over $\mathcal{X}(t)\cap\mathcal{C}(t,\mathbf{p})$, for the infimum (supremum) over a larger set cannot increase the tightness of the bound. Given initial condition (3.2) and (3.3a) and (3.3b), the hypotheses of Corollary 2.5 are satisfied, and we have that for each $\mathbf{p}\in P$,

$$\mathbf{c}(\underline{t},\mathbf{p}) \leq \mathbf{x}(\underline{t},\mathbf{p}) \leq \mathbf{C}(\underline{t},\mathbf{p})$$

in $(t_0,t_f]$. Because $\mathcal{L}_{u_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*}$ and $\mathcal{L}_{o_i}(\underline{t},\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*}$ are both affine in $\mathbf{z}$ for fixed $\underline{t}\in(t_0,t_f]$, the infimum and supremum are attained at the vertices of the set $\mathcal{C}(t,\mathbf{p})$ ($z_i = c_i(t)$) and are respectively given by

$$\begin{aligned}\dot{c}_i = \inf_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=c_i(t)}} \mathcal{L}_{u_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*} &= u_i(t,\mathbf{x}^*(t),\mathbf{p}^*) + \left.\frac{\partial u_i}{\partial x_i}\right|_{\mathbf{x}^*(t),\mathbf{p}^*}(c_i - x_i^*(t))\\ &+ \sum_{j\neq i}\min\left\{\left.\frac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j,\ \left.\frac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j\right\} - \left.\frac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} x_j^*(t)\\ &+ \sum_{j=1}^{n_p}\left.\frac{\partial u_i}{\partial p_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*}\left(p_j - p_j^*\right)\end{aligned} \tag{3.4}$$

and

$$\begin{aligned}\dot{C}_i = \sup_{\substack{\mathbf{z}\in\mathcal{C}(t,\mathbf{p})\\ z_i=C_i(t)}} \mathcal{L}_{o_i}(t,\mathbf{z},\mathbf{p})\big|_{\mathbf{x}^*(t),\mathbf{p}^*} &= o_i(t,\mathbf{x}^*(t),\mathbf{p}^*) + \left.\frac{\partial o_i}{\partial x_i}\right|_{\mathbf{x}^*(t),\mathbf{p}^*}(C_i - x_i^*(t))\\ &+ \sum_{j\neq i}\max\left\{\left.\frac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j,\ \left.\frac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j\right\} - \left.\frac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} x_j^*(t)\\ &+ \sum_{j=1}^{n_p}\left.\frac{\partial o_i}{\partial p_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*}\left(p_j - p_j^*\right)\end{aligned} \tag{3.5}$$

for $i = 1,\ldots,n_x$. By construction, $\mathbf{c}\leq\mathbf{C}$. Therefore,

$$\min\left\{\left.\frac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j,\ \left.\frac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j\right\} = \begin{cases}\left.\dfrac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j & \text{if } \left.\dfrac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*}\geq 0,\\ \left.\dfrac{\partial u_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j & \text{otherwise}\end{cases} \tag{3.6}$$

and

(3.7)

$$\max\left\{\left.\frac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j,\ \left.\frac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j\right\} = \begin{cases} \left.\dfrac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} c_j & \text{if } \left.\dfrac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} \leq 0, \\ \left.\dfrac{\partial o_i}{\partial x_j}\right|_{\mathbf{x}^*(t),\mathbf{p}^*} C_j & \text{otherwise.} \end{cases}$$

Thus, (3.4) and (3.5) reduce to linear differential equations with a finite number of discontinuities fixed in time in the coefficient matrix multiplying the states. The assumption that the number of discontinuities is finite is valid, for it is not unreasonable to assume that $u_i(t, \mathbf{x}, \mathbf{p})$ and $o_i(t, \mathbf{x}, \mathbf{p})$ are chosen such that their derivatives do not switch sign infinitely in a finite interval. That the discontinuities are fixed in time is clear because the discontinuities are defined by functions of the reference trajectories and are hence independent of the parameter. The solution to such a system of differential equations has been shown in [3] to be affine in $\mathbf{p}$ for fixed time; therefore, $\mathbf{c}(t, \mathbf{p})$ and $\mathbf{C}(t, \mathbf{p})$ are both concave and convex on P for each fixed $\underline{t} \in (t_0, t_f]$. □

Remark 3.3. Differentiability of u_i and o_i along the reference trajectory is assumed in order to employ Definition 3.1 to establish a formulaic representation of the linearization. Theorem 3.2 still holds for nondifferentiable convex functions u_i and nondifferentiable concave functions o_i, provided that any suitable subgradient is chosen for construction. That is, the linearization employed may be any valid hyperplane supporting the relaxation along the reference trajectory. In practice, nondifferentiability typically arises from taking the maximum or minimum of two valid supporting hyperplanes (e.g., the relaxation of a bilinear function). In this case, arbitrarily choosing either of the two hyperplanes yields a valid relaxation; this selection process is discussed in detail in [6].

Remark 3.4. If the initial condition depends on $\mathbf{p}$, then $\mathbf{c}_0$ and $\mathbf{C}_0$ may also depend on $\mathbf{p}$, provided that they are both affine in $\mathbf{p}$ and underestimate and overestimate $\mathbf{x}(t_0, \mathbf{p})$ for each $\mathbf{p} \in P$, respectively. This condition can also be met via a linearization on a convex/concave relaxation of the initial condition.

Following Corollary 2.5, we noted that simplifications to the theory were possible under the stronger assumptions of monotonicity. Such simplifications are also possible for Theorem 3.2. Again, the interested reader is referred to [6] for these omitted details.

4. A simple bounds example. The following section presents a small example demonstrating the state bounding techniques developed earlier in the paper. The example bounds a bimolecular reversible chemical kinetics problem. The system is not quasi-monotone and exhibits bounds explosion when Corollary 2.6 is applied without utilizing natural bounds.

Example 4.1. Consider the following bimolecular reversible chemical reaction:

$$A + B \rightleftharpoons C$$

with forward rate constant k_f and reverse rate constant k_r. Given a bounding set $[k_f^L, k_f^U] \times [k_r^L, k_r^U]$ for the rate constants, we wish to compute bounds for the concentrations of the three chemical species over a specified time interval, given the following

differential equations describing the evolution of the species concentrations:

$$\begin{aligned}\dot{x}_A &= -k_f x_A x_B + k_r x_C,\\ \dot{x}_B &= -k_f x_A x_B + k_r x_C,\\ \dot{x}_C &= k_f x_A x_B - k_r x_C,\\ x_A(0) = x_{A0}, \quad x_B(0) &= x_{B0}, \quad x_C(0) = x_{C0}.\end{aligned}$$

Without loss of generality, we will assume that $x_{B0} \geq x_{A0}$.

In order to compute state bounds for Example 4.1, a trade-off exists between the ease of computing bounds and the quality of the computed bounds. For this example, we choose to satisfy Corollary 2.6 via interval arithmetic. The bounds are first derived without employing a priori information concerning the behavior of the states using Harrison's theorem [2]. This analysis yields the following differential equations for $\dot{x}_A^L$ and $\dot{x}_A^U$ (see [6] for the complete system of ODEs):

$$\begin{aligned}\dot{x}_A^L = &-\max\{\min\{k_f^L x_A^L,\, k_f^U x_A^L\}x_B^L,\, \min\{k_f^L x_A^L,\, k_f^U x_A^L\}x_B^U,\, \max\{k_f^L x_A^L,\\ &k_f^U x_A^L\}x_B^L,\, \max\{k_f^L x_A^L,\, k_f^U x_A^L\}x_B^U\} + \min\{k_r^L x_C^L,\, k_r^L x_C^U,\, k_r^U x_C^L,\, k_r^U x_C^U\},\end{aligned} \tag{4.1a}$$

$$\begin{aligned}\dot{x}_A^U = &-\min\{\min\{k_f^L x_A^U,\, k_f^U x_A^U\}x_B^L,\, \min\{k_f^L x_A^U,\, k_f^U x_A^U\}x_B^U,\, \max\{k_f^L x_A^U,\\ &k_f^U x_A^U\}x_B^L,\, \max\{k_f^L x_A^U,\, k_f^U x_A^U\}x_B^U\} + \max\{k_r^L x_C^L,\, k_r^L x_C^U,\, k_r^U x_C^L,\, k_r^U x_C^U\},\end{aligned} \tag{4.1b}$$

subject to the initial condition $x_A^L(0) = x_A^U(0) = x_{A0}$.

Despite their unwieldy appearance, because the right-hand sides of the above differential equations are Lipschitz continuous, numerically integrating the differential equations is not difficult. In order to compute a numerical solution, the following values are assigned to the constants:

$$k_f^L = 100,\ k_f^U = 500,\ k_r^L = 0.001,\ k_r^U = 0.01,\ x_{A0} = 1,\ x_{B0} = 1.5,\ x_{C0} = 0.5,$$

where the units for concentration are mol/L, the units for the forward rate constant are L/(mol·min), and the units for the reverse rate constant are min^{-1}. The time interval of interest is set to 1.5 minutes. A plot of the upper bound for species B is shown in Figure 4.1. Clearly, from the figure, the bounding method is not acceptable. From physical considerations, we know that the upper bound should not exceed the sum of initial concentrations of species B and C in the system. However, because the system is not quasi-monotone, the upper bound for species B explodes (in fact, the upper bound for the concentrations of all species explodes). For Figure 4.1, the integration was terminated when the bound exceeded 10^3.

In order to correct the bounds explosion, Corollary 2.6 is now applied to the system in Example 4.1, utilizing the a priori information we know from considering species mole balances on the system. Given the assumption that x_{B0} is greater than x_{A0}, from the kinetics of Example 4.1, we know that the following inequalities hold:

$$\begin{aligned}0 \leq x_A(t) &\leq x_{A0} + x_{C0},\\ x_{B0} - x_{A0} \leq x_B(t) &\leq x_{B0} + x_{C0},\\ 0 \leq x_C(t) &\leq x_{A0} + x_{C0}.\end{aligned}$$

Corollary 2.6 may now be implemented by several different methods; three methods are presented here. The first method is a trivial extension of applying interval arithmetic. Recalling that the optimization is now performed over the intersection of the physical and differential bounds, if we set $\overline{\mathcal{X}}(t) = [\overline{\mathbf{x}}^L(t), \overline{\mathbf{x}}^U(t)]$ and

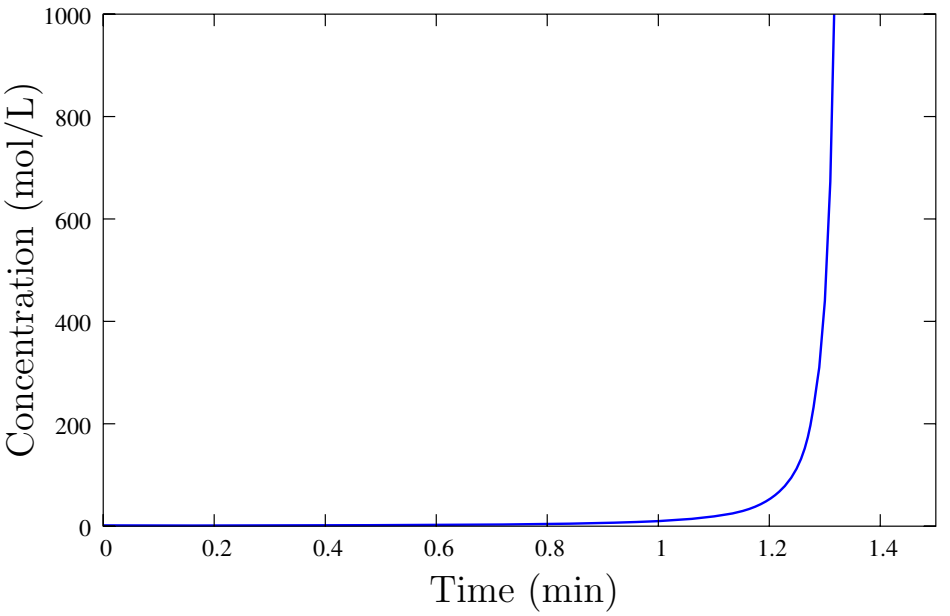


FIG. 4.1. *Upper bound for species B from Example* 4.1 *without utilizing a priori information.*

$G(t) = [\mathbf{x}^L(t), \mathbf{x}^U(t)]$, the right-hand side can be represented as an optimization over the set

$$\overline{\mathcal{X}}((t)) \cap G(t) = [\max\{x_A^L(t), \overline{x}_A^L(t)\}, \min\{x_A^U(t), \overline{x}_A^U(t)\}] \times \cdots \times [\max\{x_C^L(t), \overline{x}_C^L(t)\}, \min\{x_C^U(t), \overline{x}_C^U(t)\}]. \tag{4.2}$$

Therefore, much tighter bounds are achieved by simply assigning the values

$$\begin{aligned} x_A^L &:= \max\{0, x_A^L\}, & x_A^U &:= \min\{x_{A0} + x_{C0}, x_A^U\}, \\ x_B^L &:= \max\{x_{B0} - x_{A0}, x_B^L\}, & x_B^U &:= \min\{x_{B0} + x_{C0}, x_B^U\}, \\ x_C^L &:= \max\{0, x_C^L\}, & x_C^U &:= \min\{x_{A0} + x_{C0}, x_C^U\} \end{aligned}$$

into the right-hand sides of (4.1). The resulting bounds for species B together with the natural bounds are shown in Figure 4.2. There are several important features to note in these new bounds. First and foremost, the bounds no longer explode, because when the right-hand sides of the differential equations exceed a limit (dictated by the interval arithmetic), the right-hand side of the differential equation becomes a constant, thus bounding the slope of the state. Second, the bounds are smooth. This phenomenon occurs because the right-hand sides of the differential equations are continuous, assuring the everywhere existence of the derivatives of the states. Finally, while the upper bound is an improvement over the natural bound, we note that the lower bound is still not at least as good as the natural bounds on the system.

The second application of Corollary 2.6 alleviates the problem that the integrated state bounds exceed the a priori bounds at the expense of smoothness of the resulting bounds and ease of numerical integration. In the discussion following Corollary 2.6, the case where the optimization problem becomes infeasible was discussed. In addition

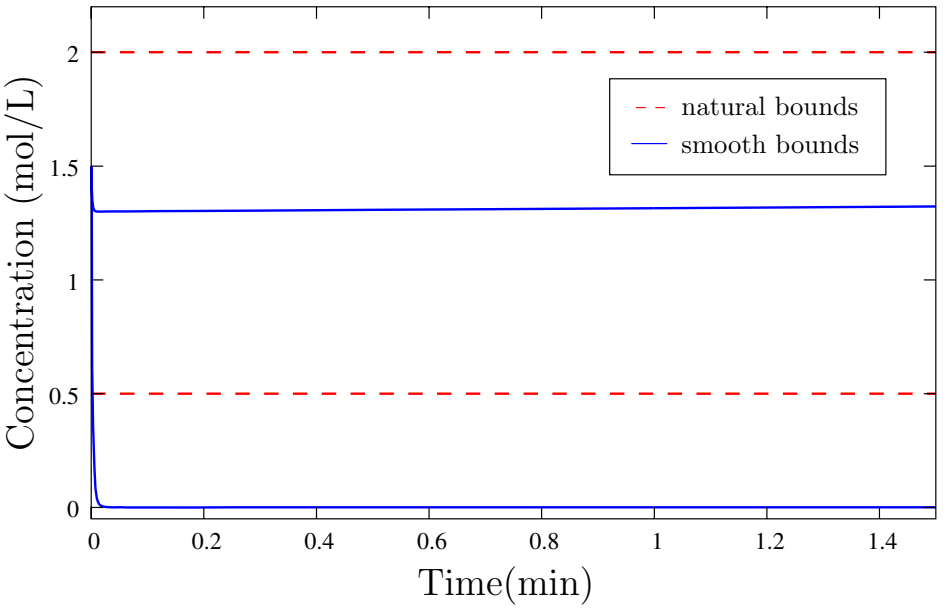


FIG. 4.2. *Bounds for species B from Example* 4.1 *utilizing a priori information.*

to optimizing over the constraint set $\overline{\mathcal{X}}(t) \cap G(t)$, we must also remember that we are subject to the equality constraint $z_i = v_i(t)$ for the lower bounding problems and $z_i = w_i(t)$ for the upper bounding problems. However, because of the constraint set $\overline{\mathcal{X}}(t)$, the equality $z_i = v_i(t)$ is impossible to satisfy when $x_i^L(t) < \overline{x}_i^L(t)$, and the equality $z_i = w_i(t)$ is impossible to satisfy when $x_i^U(t) > \overline{x}_i^U(t)$. In this situation, we can choose any finite number for the right-hand side of the differential equation. The obvious choice for this number is zero, for this forces the integrated bound to equal the natural bound. We note that for the upper bound, choosing a positive number yields a bound looser than the natural bound, and choosing a negative number causes chattering in the numerical integrator. Therefore, for $i = 1, \ldots, n_x$, very tight bounds can be obtained via Corollary 2.6 by defining the right-hand sides of the ODEs to equal the differential equations derived via interval analysis if $x_i^L, x_i^U \in \overline{\mathcal{X}}(t)$ and letting $\dot{x}_i^L = \dot{x}_i^U = 0$ otherwise. For Example 4.1, the lower bound for species B from this analysis and the lower bound as previously computed via the interval arithmetic approach to Corollary 2.6 are shown for comparison in Figure 4.3.

In the third method of bounding the differential equations, we incorporate an invariant in the differential equation defining the system. From an extent of reaction analysis, the following equality may be derived for this system:

$$x_A + x_B + 2x_C = x_{A0} + x_{B0} + 2x_{C0}. \tag{4.3}$$

Corollary 2.6 is now applied where the natural bounding set $\overline{\mathcal{X}}(t)$ is defined by the intersection of the plane in (4.3), and $\overline{\mathcal{X}}(t)$ is defined by (4.2). Effectively, for each bounding differential equation, (4.3) is utilized to eliminate one variable from the optimization problem defining the right-hand side of the bounding differential equation. The selection of which variables are eliminated in each optimization problem is not unique. Theoretically, the addition of this invariant creates more constrained optimization problems, and the infimum (supremum) of these new optimization problems

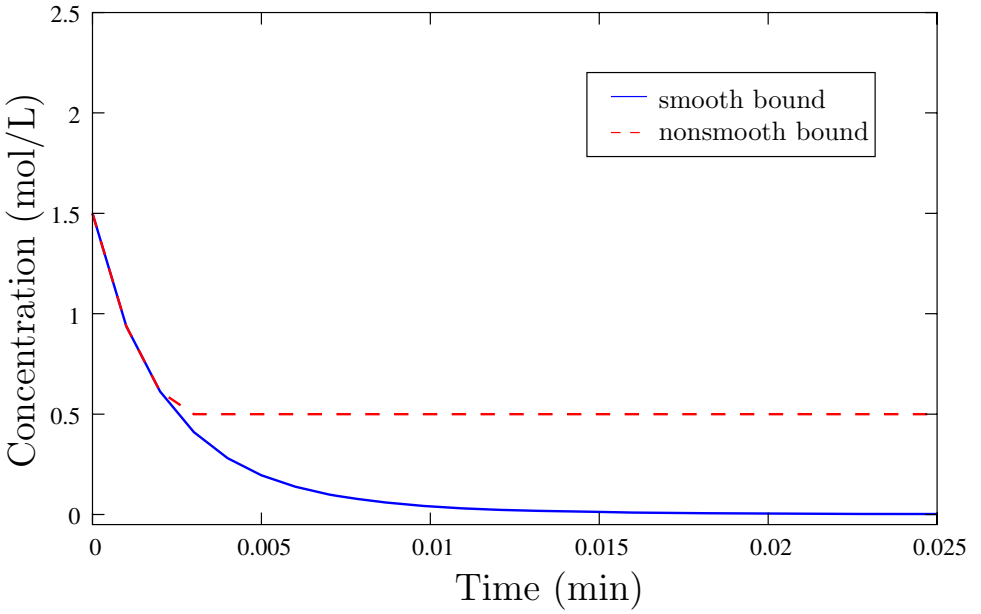


FIG. 4.3. *Bounds for species B from Example* 4.1 *utilizing a priori information and bounds truncation.*

is at least no worse than the infimum (supremum) of the optimization problems derived without the invariant. In practice, however, because the optimization problems are estimated using interval arithmetic, the quality of the state bound is very dependent on the quality of the interval arithmetic methods employed for bounding the right-hand sides of the differential equations. Both elimination of variables and distinct factorization strategies can lead to vastly different outcomes, and these different outcomes sometimes lead to worse state bounds than not including the invariant at all.

For this example, the invariant was used to eliminate x_C from the optimization problems defining $\dot{x}_A^L$, $\dot{x}_A^U$, $\dot{x}_B^L$, $\dot{x}_B^U$, and x_B from the equations defining $\dot{x}_C^L$, $\dot{x}_C^U$. The resulting differential equations for $\dot{x}_A^L$ and $\dot{x}_A^U$ are given by the following (see [6] for the complete system of ODEs):

$$\begin{aligned}
\dot{x}_A^L = &-\max\{\min\{k_f^L x_A^L,\, k_f^U x_A^L\}\max\{x_B^L,\, 0.5\},\, \min\{k_f^L x_A^L,\, k_f^U x_A^L\}\min\{x_B^U,\, 2\},\\
&\qquad \max\{k_f^L x_A^L,\, k_f^U x_A^L\}\max\{x_B^L,\, 0.5\},\, \max\{k_f^L x_A^L,\, k_f^U x_A^L\}\min\{x_B^U,\, 2\}\}\\
&+\min\{k_r^L(1.75 - 0.5x_A^L - \max\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^L(1.75 - 0.5x_A^L - \min\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^U(1.75 - 0.5x_A^L - \max\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^U(1.75 - 0.5x_A^L - \min\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\})\},\\
\dot{x}_A^U = &-\min\{\min\{k_f^L x_A^U,\, k_f^U x_A^U\}\max\{x_B^L,\, 0.5\},\, \min\{k_f^L x_A^U,\, k_f^U x_A^U\}\min\{x_B^U,\, 2\},\\
&\qquad \max\{k_f^L x_A^U,\, k_f^U x_A^U\}\max\{x_B^L,\, 0.5\},\, \max\{k_f^L x_A^U,\, k_f^U x_A^U\}\min\{x_B^U,\, 2\}\}\\
&+\max\{k_r^L(1.75 - 0.5x_A^U - \max\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^L(1.75 - 0.5x_A^U - \min\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^U(1.75 - 0.5x_A^U - \max\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\}),\\
&\qquad k_r^U(1.75 - 0.5x_A^U - \min\{0.5\max\{x_B^L,\, 0.5\},\, 0.5\min\{x_B^U,\, 2\}\})\}.
\end{aligned}$$

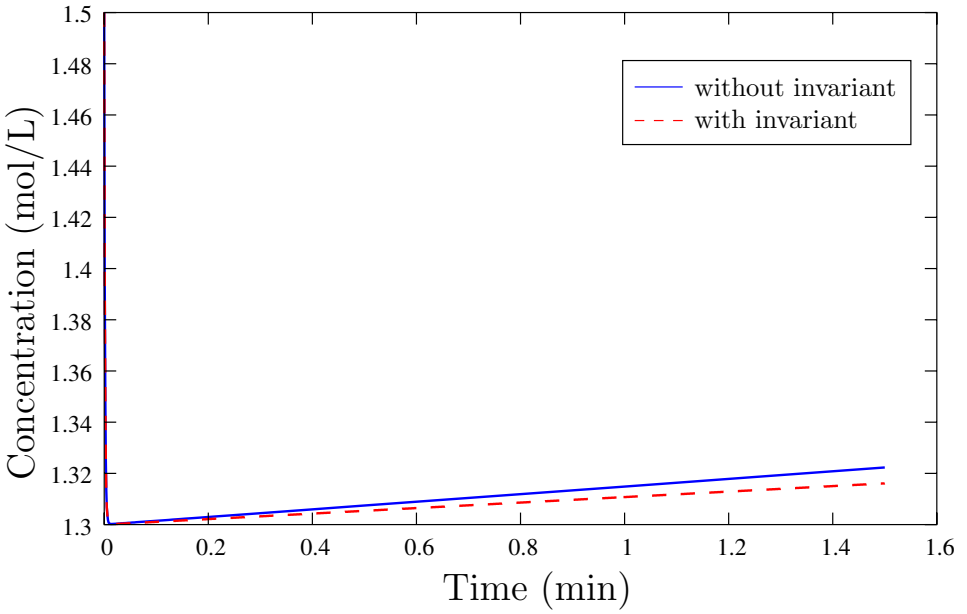


FIG. 4.4. *Bounds for species B from Example* 4.1 *utilizing the reaction invariant.*

The marginal improvement of utilizing the reaction invariant is illustrated for the upper bound of species B in Figure 4.4. We note that even for this simple example problem, different factorizations of the objective function prior to applying interval arithmetic can lead to bounds worse than those computed without the invariant.

5. Automatic generation of relaxations. From the previous simple example, one quickly notes that application of the theory presented in this paper is both tedious and error-prone, even for trivial problems. As previously mentioned, the techniques developed in this paper were originally motivated by a desire to solve globally optimization problems containing differential equations. In order for this objective to be accomplished feasibly, a suite of programs, named GDOC [9], was developed to solve dynamic optimization problems automatically, utilizing the relaxation strategy developed in this paper. GDOC has proven its utility for globally solving dynamic optimization problems in several case studies [6, 7, 10].

One component of GDOC, the gdoc-compiler, was designed specifically to accept a text input file written in the GDOC language, apply the theorems developed in this paper, and generate a Fortran 77 residual file for input to a numerical ODE solver. The compiler consists of three separate components: the compiler front end, the intermediate code representation and symbolic analysis tool, and the compiler back end. The compiler front end is responsible for scanning, parsing, and generating the intermediate code representation. The compiler back end is a custom written output routine that accepts output from the symbolic analyzer, automatically formats the results in Fortran 77, and writes an ODE residual file. The intermediate code and symbolic analysis tool is responsible for applying the theory developed in this paper; this step is described in more detail below.

The scanning and parsing steps in the gdoc-compiler accept equations written as text and generate object-oriented parse trees to be operated on by the symbolic

analyzer. The symbolic analyzer first transforms the parse tree representation of the input equations into a code tape representation. For the purposes of this discussion, a code tape is a representation of a system of equations as a decomposition into a sequence of elementary operations containing only unary or binary operations of the input variables or generated intermediate variables [1]; transformation to a code tape eliminates the computation of common subexpressions. Second, the symbolic analyzer creates a second code tape that applies Corollary 2.6 to each equation in the input code tape in order to generate state bounds; Corollary 2.6 is implemented via interval arithmetic. Third, the symbolic analyzer creates a third code tape that computes the convex and concave relaxations of the right-hand side of the input ODE. This code tape is generated by applying McCormick's relaxation technique [4] to the code tape of the original ODE using the bounds generated by the second code tape as necessary. Fourth, the symbolic analyzer creates a fourth code tape by linearizing the relaxation code tape along the user-specified reference trajectory. The linearizations are generated by applying the forward mode of algorithmic differentiation to the relaxation code tape. Steps three and four combined represent the application of Theorem 3.2. All four code tapes are then passed to the compiler back end, where the residual file is generated. The gdoc-compiler is described in much greater detail in [6].

6. Control problem. In this section, the theory developed in this paper is applied to a problem for which a closed form solution is impossible and manual application of the theory is impractical. The problem considered is an optimal control problem contrived to demonstrate the application of the theory to a nonlinear system of ODEs. The problem is stated as follows:

$$\min_{u(t)} x_4(t_f),$$

subject to

$$\begin{aligned} \dot{x}_1 &= \exp(-x_2) + x_1/x_4, \\ \dot{x}_2 &= x_3^2/(u+0.1), \\ \dot{x}_3 &= x_4 x_3 u, \\ \dot{x}_4 &= 3x_1/(t+1), \\ \mathbf{x}_0 &= (1,\ -1,\ 1,\ 0.5), \\ t \in (0,1], &\quad 0 \le u(t) \le 4. \end{aligned}$$

Because this problem does not derive from a physical system, no bounding set $\mathcal{X}$ exists. Consider a piecewise constant control profile on two equally spaced time intervals. The objective function is nonconvex, and hence, in order to solve this problem globally, convex and concave relaxations $c_{x_4}(t_f, \mathbf{u})$ and $C_{x_4}(t_f, \mathbf{u})$ are required at each node in the branch-and-bound tree. Utilizing the compiler component of GDOC, a plot of the objective function and its relaxations is depicted in Figure 6.1; the relaxations were generated for the reference trajectory $(\mathbf{x}^*, \mathbf{u}^*) = (\mathbf{x}^{mid}, \mathbf{u}^{mid})$. Because the GDOC compiler generates lengthy Fortran files, the generated source code for this problem is not included in this paper. However, both the input and output files for this example are available from the second author's website, http://yoric.mit.edu/gdoc/index.html. We note that in [7] we prove that $c_{x_4}(t_f, \mathbf{u})$ and $C_{x_4}(t_f, \mathbf{u})$ both converge to $x_4(t_f, \mathbf{u})$ as the interval $U = [0,4]^2$ shrinks to any degenerate point $\mathbf{u}^{\mathbf{d}} \in \mathbf{U}$. This convergence property implies the convergence of the branch-and-bound algorithm utilized to solve this problem globally.

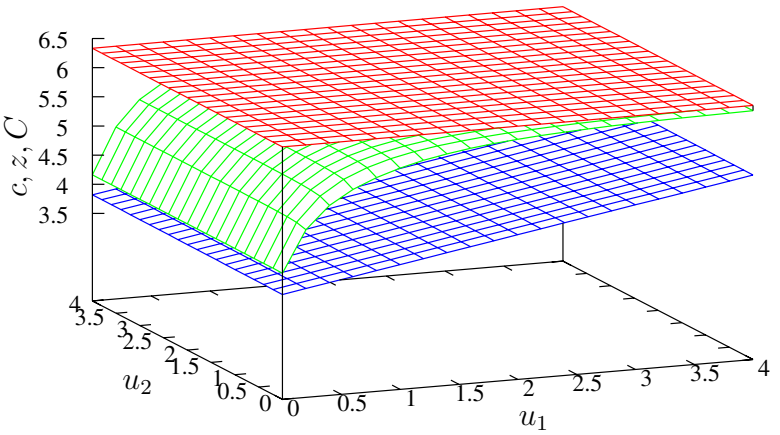


FIG. 6.1. *Objective function and relaxations for a control problem.*

7. Conclusions. In this paper, we have presented two novel results on bounding the solution of ODEs. In the first section, we demonstrated a method for extending differential inequalities to include natural bounds on the system. The method is useful for practical problems, for it provides a differential technique to generate reasonably tight bounds for systems that are not quasi-monotone. Utilizing these time varying state bounds, we presented a method for providing pointwise in time convex and concave relaxations for the image of the solution of a system of nonlinear parametric differential equations on a parameter space. The method is applied by constructing a convex and concave relaxation of the right-hand sides of the differential equations. These relaxations are then linearized at a reference trajectory. By construction, the linearizations are known to rigorously bound the original right-hand sides of the differential equations; therefore, integration provides rigorous bounds for the original states. Convexity and concavity results from the affine nature of the solution to a system of linear differential equations.

In the second half of the paper, two examples were presented demonstrating the application of the theory developed in the first half of the paper. The first example illustrated the utilization of Corollary 2.6 for an analytically tractable problem. The difficulty in manually applying the theory motivated the discussion of a program, GDOC, that utilizes algorithmic differentiation techniques to apply the theory automatically. A second example then demonstrated the application of Theorem 3.2 on a nonlinear problem utilizing GDOC.

REFERENCES

[1] A. GRIEWANK, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, Frontiers in Appl. Math. 19, SIAM, Philadelphia, 2000.

[2] G. HARRISON, *Dynamic models with uncertain parameters*, in Proceedings of the First International Conference on Mathematical Modeling, X. Avula, ed., vol. 1, 1977, pp. 295–304.

[3] C. K. LEE, A. B. SINGER, AND P. I. BARTON, *Global optimization of linear hybrid systems with explicit transitions*, Systems Control Lett., 51 (2004), pp. 363–375.

[4] G. P. MCCORMICK, *Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems*, Math. Programming, 10 (1976), pp. 147–175.

[5] R. E. MOORE, *Methods and Applications of Interval Analysis*, Studies in Appl. Math. 2, SIAM, Philadelphia, 1979.

[6] A. B. Singer, *Global Dynamic Optimization*, Ph.D. thesis, Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA, 2004.
[7] A. B. Singer and P. I. Barton, *Global optimization with nonlinear ordinary differential equations*, J. Global Optim., in press.
[8] A. B. Singer and P. I. Barton, *Global solution of linear dynamic embedded optimization problems*, J. Optim. Theory Appl., 121 (2004), pp. 613–646.
[9] A. B. Singer and B. Chachuat, *GDOC Version* 1.0 *Manual*, Technical report, Massachusetts Institute of Technology, Cambridge, MA, 2005.
[10] A. B. Singer, J. W. Taylor, W. H. Green, and P. I. Barton, *Global dynamic optimization for parameter estimation in chemical kinetics*, J. Phys. Chem. A, 110 (2006), pp. 971–976.
[11] W. Walter, *Differential and Integral Inequalities*, Springer-Verlag, Berlin, 1970.