

# USING SYMMETRIES AND ANTISYMMETRIES TO ANALYZE A PARALLEL MULTIGRID ALGORITHM: THE ELLIPTIC BOUNDARY VALUE PROBLEM CASE\*

CRAIG C. DOUGLAS<sup>†</sup> AND BARRY F. SMITH<sup>‡</sup>

*This paper is dedicated to Jim Douglas, Jr. on the occasion of his 60th birthday.*

**Abstract.** Symmetry and antisymmetry properties of a class of elliptic partial differential equations are exploited to prove when a particular parallel multilevel algorithm is a direct method rather than the usual iterative method. No smoothing is required for this result. Examples are presented, including variable coefficient ones. A connection between the algorithm in this article and domain decomposition is established, even though this algorithm is more general and different. The parallel algorithm is also analyzed when it is iterative and it is shown how to increase processor utilization. Hackbusch's robust multigrid algorithm is analyzed for some model problems and it is shown that the parallel algorithm in this article uses much less computer time with at most the same amount of storage.

**Key words.** partial differential equations, parallel multigrid, robust multigrid solver, domain decomposition, direct method, iterative method, convergence rates

**AMS(MOS) subject classifications.** 65W05, 65N15, 65N10

**1. The problem.** In this paper, we approximate the solution to the elliptic boundary value problem

$$(1.1) \quad \begin{cases} \mathcal{L}u = f & \text{in } \Omega, \\ u = \sum_{j=0}^k g_j D^j u & \text{on } \partial\Omega \end{cases}$$

using a parallel multigrid algorithm. We assume that (1.1) is well posed and has a unique solution  $u \in \mathcal{H}$ . Typically,  $\mathcal{H}$  is a Sobolev space. Our approximate solution lies in a finite-dimensional space  $\mathcal{M}$ , which is the natural space after (1.1) has been discretized using a finite difference, element, or volume method to form the discrete problem

$$(1.2) \quad \mathcal{A}U = F,$$

where  $\mathcal{A} \in \mathbb{R}^{N \times N}$  and  $U, F \in \mathbb{R}^N$ . We use a set of auxiliary nonnested finite-dimensional subspaces  $\{\mathcal{M}_j\}_{j=1}^d$  to compute the approximate solution.

Standard multigrid algorithms consist of two phases: smoothing on all the error components of a space and solving a smaller (coarse grid) correction problem on a subspace of the error components. Typically, the solution space is decomposed into two parts: low frequency components and everything else. A subspace is constructed from the low frequency components of the current space. This is done recursively, resulting in a (usually nested) sequence of subspaces. Doing this allows the dimension

---

\* Received by the editors November 30, 1987; accepted for publication (in revised form) March 10, 1988.

<sup>†</sup> Mathematical Sciences Department, IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, New York 10598. E-mail: *bells@ibm.com*.

<sup>‡</sup> Department of Mathematics, Courant Institute of Mathematical Sciences, New York University, 251 Mercer Street, New York, New York 10012. This research was done while visiting IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, New York.

of the subspaces to decrease geometrically, resulting in a fast algorithm. However, the algorithms are not useful for certain types of problems that cannot be represented adequately on the smaller spaces.

Our parallel multilevel algorithm decomposes the solution space completely. Every error component is represented in at least one of the subspaces. Computation on the subspaces is performed in parallel, so that all of the components are corrected to some degree during the correction phase. As a result, our algorithm can solve certain types of problems very well for which standard multigrid algorithms are not useful.

This paper is a continuation of a series of papers [10], [12], and [13]. In [12], we defined parallel algorithms based on serial multigrid and aggregation/disaggregation techniques. We analyzed them in a very abstract manner. In [10], we analyzed algorithms based on the nested iteration variant of multigrid and aggregation/disaggregation. In [13], we explored how these algorithms would perform on different classes of parallel processors: coarse grained machines with either shared or local memory, as well as fine grained machines with local memories. In these papers, no attempt was made to use properties of the underlying problems; only abstract knowledge was used.

Our parallel multilevel algorithms are not the only such algorithms. Brandt and Ta'asan [4], Frederickson and McBryan [15], and Hackbusch [16] have recently proposed special cases of our algorithms. The difference is that we consider the construction of the subspaces as a variable part of the parallel algorithm, whereas they provide specific, but different rules, for the construction.

In §2, we define the parallel multigrid algorithm and its auxiliary components. We also describe some properties that the subspaces should have in order for our algorithm to work well.

In §3, we analyze the algorithm when it is a direct method. We exploit symmetry and antisymmetry properties of a class of elliptic partial differential equations to prove when a particular parallel multilevel algorithm is a direct method rather than the usual iterative method. No smoothing is required for this result. Examples are presented, including variable coefficient ones.

In §4, we analyze the algorithm when it is an iterative method. We analyze Hackbusch's robust multigrid algorithm for some model problems and show that our parallel algorithm uses much less computer time and at most the same amount of storage. We also exploit the composition of two classes of restriction operators to produce many smaller problems to solve. This is similar to the total reduction method variant of standard multigrid [14]. By using more, smaller problems, we speed up our parallel algorithm noticeably.

In §5, we attempt to compare the running time of various flavors of the parallel multigrid algorithm. We do this using operation counts. We use two measures: *total operation counts* and *wall clock time*. The latter is the important measure since it measures the running time per iteration of our algorithm.

In §6, we interpret our algorithm as a domain decomposition method. We compare its interpretation with standard block domain decomposition methods. We show that our algorithm can be thought of as a pointwise domain decomposition method with global communication instead of the usual block domain decomposition method with local communication.

**2. The parallel multigrid algorithm.** In this section, we define our parallel multigrid algorithm and its auxiliary components. Assume we are given the following:

$$(2.1) \quad \{\mathcal{L}, \mathcal{A}, u_0, f, \mathcal{M}, \{\mathcal{P}_j, \mathcal{R}_j\}_{j=1}^d\}.$$

Here,  $u_0$  is the initial guess and  $f$  is the right-hand side. The subspaces  $\{\mathcal{M}_j\}_{j=1}^d$  of  $\mathcal{M}$  are defined through the restriction operators  $\{\mathcal{R}_j\}_{j=1}^d$ :

$$\mathcal{R}_j : \mathcal{M} \rightarrow \mathcal{M}_j.$$

The prolongation operators  $\{\mathcal{P}_j\}_{j=1}^d$  map the other way:

$$\mathcal{P}_j : \mathcal{M}_j \rightarrow \mathcal{M}.$$

Frequently, each prolongation operator is the adjoint of one of the restriction operators. We define operators  $\mathcal{L}_j$  and  $\mathcal{A}_j$  on the subspaces by

$$\mathcal{L}_j = \mathcal{R}_j(\mathcal{L}(\mathcal{P}_j))$$

and

$$\mathcal{A}_j = \mathcal{R}_j \mathcal{A} \mathcal{P}_j.$$

Each operator  $\mathcal{L}_j$  and  $\mathcal{A}_j$  must have an inverse. To guarantee that each  $\mathcal{L}_j$  exists requires  $\mathcal{M} \subset \mathcal{H}$ . This is not always practical, nor necessary. If we are only interested in solving the discrete problem (1.2), we can ignore this requirement.

Let  $\mathcal{I}$  be the identity operator on  $\mathcal{M}$ . We require that

$$(2.2) \quad \sum_{j=1}^d \mathcal{R}_j^* \mathcal{R}_j = \mathcal{I},$$

where  $\mathcal{R}_j^*$  is the adjoint of  $\mathcal{R}_j$ . This is convenient in the proofs. It is also practical since natural restriction operators result in the right-hand side of (2.2) being  $C \cdot \mathcal{I}$ , where  $C \in \mathbb{R}$ . Simply scaling the restriction operators satisfies (2.2). Simple examples are contained in §3.

We also require the subspaces defined by the restriction operators to satisfy

$$(2.3) \quad \mathcal{M} = \bigoplus_{j=1}^d \mathcal{P}_j \mathcal{M}_j.$$

This requirement guarantees that every error component in  $\mathcal{M}$  is represented in at least one subspace.

A useful property for the subspaces to have is that

$$(2.4) \quad \dim(\mathcal{M}) = \sum_{j=1}^d \dim(\mathcal{M}_j).$$

This is a nontelegraphing property. One of the drawbacks of standard multigrid on parallel processors is that we have idle processors while computing on coarse levels. Typically, we will not have idle processors if this nontelegraphing property is satisfied (see [13] for an analysis of this property).

We use the following parallel multilevel algorithm.

ALGORITHM PMG  $(d, m, n, f, u, \mathcal{L}, \{\mathcal{L}_j, \mathcal{P}_j, \mathcal{R}_j\}_{j=1}^d)$  {

$d$	=	Number of subspaces ( $d > 0$ )
$m$	=	number of smoothing iterations ( $m \geq 0$ )
$n$	=	number of iterations ( $n \geq 0$ )
$f$	=	right-hand side
$u$	=	initial guess/approximate solution
$\mathcal{L}$	=	partial differential equation operator on solution space
$\{\mathcal{L}_j\}$	=	partial differential equation operators on subspaces
$\{\mathcal{P}_j\}$	=	prolongation operators
$\{\mathcal{R}_j\}$	=	restriction operators

Smooth  $m$  times on  $u$  to get  $u_0$

Do  $i = 1, \dots, n$  {  
  Compute residual  $r_i = f - \mathcal{L}u_{i-1}$   
  Solve in parallel,  $j = 1, \dots, d$ :  
   $\mathcal{L}_j c_j = \mathcal{R}_j r_i$   
  Set  $c = u_{i-1} + \sum_{j=1}^d \mathcal{P}_j c_j$   
  Smooth  $m$  times on  $c$  to get  $u_i$   
  }  
Set  $u = u_n$   
}

This generates an approximation to the solution  $u$  of (1.1). The discrete form of Algorithm PMG substitutes the symbol  $\mathcal{A}$  for each occurrence of the symbol  $\mathcal{L}$  (with or without a subscript) in the definition of Algorithm PMG.

The solve step can be done in one of two forms: either solve the problem quite well or recursively call Algorithm PMG to get an approximate solution. The latter is a true multilevel algorithm, while the former method is a two-level algorithm. Usual solution methods on the coarsest level are Gaussian elimination or some iterative method (e.g., preconditioned conjugate gradients or Orthomin).

We note that while only one step is labeled as *do in parallel*, the residual computation, smoothing, restriction, and prolongation steps may also be done in parallel. The determining factors are how many processors are available, the time required to transmit data between processors (on either shared or local memory systems), and if there are appropriate software tools available for this to make sense. In fact, each of the steps may be viewed as a candidate for parallel processing on many processors. See [10] for more details.

We allow general smoothers  $\mathcal{B} : \mathcal{M} \rightarrow \mathcal{M}$ . Let  $\hat{w} \in \mathcal{M}$  be defined by

$$(2.5) \quad \mathcal{B}(\hat{w} - w) = f - \mathcal{L}w.$$

The operator  $\mathcal{B}$  must be simple in comparison to  $\mathcal{L}$  to make (2.5) computationally attractive. The error propagation operator  $\mathcal{S}$  for one iteration of the the smoothing process is

$$\mathcal{S} = \mathcal{I} - \mathcal{B}^{-1}\mathcal{L}$$



FIG. 1. *Motivation in one dimension*

(see [3]). The error propagation operator  $\mathcal{C}$  for the correction process is simply

$$\mathcal{C} = \mathcal{I} - \sum_{j=1}^d \mathcal{P}_j \mathcal{L}_j^{-1} \mathcal{R}_j \mathcal{C}$$

(see [12]). When  $\mathcal{A}$  and  $\mathcal{B}$  are symmetric, positive definite, the energy norms,  $||| \cdot |||$ , of  $\mathcal{S}$  and  $\mathcal{C}$  are

$$||| \mathcal{S} ||| = \|\mathcal{I} - \mathcal{A}^{1/2} \mathcal{B}^{-1} \mathcal{A}^{1/2}\|$$

and

$$||| \mathcal{C} ||| = \|\mathcal{I} - \mathcal{A}^{1/2} (\sum_{j=1}^d \mathcal{P}_j \mathcal{A}_j^{-1} \mathcal{R}_j) \mathcal{A}^{1/2}\|,$$

where  $\|\cdot\|$  is the  $\mathcal{L}^2$  norm. Finally, the error propagation operator  $\mathcal{E}$  is defined as

$$(2.6) \quad \mathcal{E} = \mathcal{S}^m (\mathcal{C} \mathcal{S}^m)^n = \mathcal{S}^{m/2} (\mathcal{S}^{m/2} \mathcal{C} \mathcal{S}^{m/2})^n \mathcal{S}^{m/2}$$

There are many forms for  $\mathcal{E}$ . The appropriate choice depends on the properties of  $\mathcal{S}^m$ ,  $\mathcal{C}$ , and  $\mathcal{A}$  (see [3]).

**3. A direct method.** In this section, we exploit symmetry and antisymmetry properties of a class of elliptic partial differential equations to prove that a particular parallel multilevel algorithm is actually a direct method. No smoothing is required for this result. We motivate this material in §3.1 with two examples, prove the convergence result in §3.2, and provide some examples in §3.3.

**3.1. Motivation.** We motivate the main convergence theorem in this section with simple examples in one and two dimensions. In both cases, we take advantage of symmetries in the domains.

**3.1.1. One dimension.** The first example is defined on the domain  $\Omega = (0, 1)$ . We divide  $\Omega$  into two equal parts about the midpoint  $m = 1/2$  and designate the left half as  $\Omega_1$ . We define

$$Qq = 1 - q \quad \forall q \in \Omega.$$

Figure 1 summarizes these definitions.

We say that a function  $U(x)$  is *symmetric* (even) *about*  $m$  in  $\Omega$  if

$$U(x) = U(Qx) \quad \forall x \in \Omega.$$

We say that a function  $U(x)$  is *antisymmetric* (odd) *about*  $m$  in  $\Omega$  if

$$U(x) = -U(Qx) \quad \forall x \in \Omega.$$

We say an operator  $\mathcal{L}$  *preserves symmetry and antisymmetry about  $m$*  if  $\mathcal{L}$  applied to any symmetric (antisymmetric) function about  $m$  is a symmetric (antisymmetric) function about  $m$ . For example, a second derivative operator preserves symmetric and antisymmetric functions. We say an operator  $\mathcal{L}$  *reverses symmetry and antisymmetry about  $m$*  if  $\mathcal{L}$  applied to any symmetric (antisymmetric) function about  $m$  is an antisymmetric (symmetric) function about  $m$ . For example, a first derivative operator reverses symmetric and antisymmetric functions.

REMARK 3.1. *These definitions determine how an operator  $\mathcal{L}$  maps symmetric and antisymmetric functions, not whether  $\mathcal{L}$  is a symmetric or nonsymmetric operator.*

Define for all  $x \in \Omega$ ,

$$(3.1) \quad \mathcal{R}_0 U(x) = 2^{-1/2} [ U(x) + U(Qx) ]$$

and

$$(3.2) \quad \mathcal{R}_1 U(x) = 2^{-1/2} [ U(x) - U(Qx) ].$$

Further,

$$\mathcal{R}_0^* U(x) = 2^{-1/2} \begin{cases} U(x), & \text{if } x \in \Omega_1, \\ U(Qx), & \text{otherwise.} \end{cases}$$

and

$$\mathcal{R}_1^* U(x) = 2^{-1/2} \begin{cases} U(x), & \text{if } x \in \Omega_1, \\ -U(Qx), & \text{otherwise.} \end{cases}$$

Note that

$$\mathcal{R}_0^* \mathcal{R}_0 + \mathcal{R}_1^* \mathcal{R}_1 = \mathcal{I}.$$

Suppose that  $U_0$  is a symmetric function about  $m$  and that  $U_1$  is an antisymmetric function about  $m$ . Then

$$(3.3) \quad \mathcal{R}_0^* \mathcal{R}_0 U_0(x) = U_0(x) \quad \text{and} \quad \mathcal{R}_0^* \mathcal{R}_0 U_1(x) = 0 \quad \forall x \in \Omega,$$

and

$$(3.4) \quad \mathcal{R}_1^* \mathcal{R}_1 U_1(x) = U_1(x) \quad \text{and} \quad \mathcal{R}_1^* \mathcal{R}_1 U_0(x) = 0 \quad \forall x \in \Omega.$$

Suppose that  $\mathcal{L}$  preserves symmetry and antisymmetry about  $m$ . If we take

$$\mathcal{P}_0 = \mathcal{R}_0^* \quad \text{and} \quad \mathcal{P}_1 = \mathcal{R}_1^*,$$

then Algorithm PMG converges in one iteration with no smoothing (see §3.2). On the other hand, suppose that  $\mathcal{L}$  reverses symmetry and antisymmetry about  $m$ . If we take

$$\mathcal{P}_0 = \mathcal{R}_1^* \quad \text{and} \quad \mathcal{P}_1 = \mathcal{R}_0^*,$$

then Algorithm PMG converges in one iteration with no smoothing.

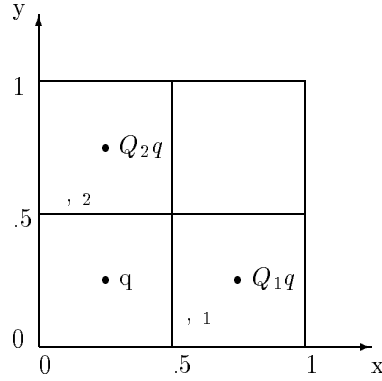


FIG. 2. Motivation in two dimensions

**3.1.2. Two dimensions.** The second example is defined on the domain  $\Omega = (0, 1) \otimes (0, 1)$ . We divide  $\Omega$  into four equal parts about the midpoint  $m = (.5, .5)$ . We define  $\Omega_1 = (0, .5) \otimes (0, .5)$ . Let  $q = (q_1, q_2)$  and define

$$Q_1q = (1 - q_1, q_2) \quad \text{and} \quad Q_2q = (q_1, 1 - q_2) \quad \forall q \in \Omega.$$

Figure 2 summarizes these definitions.

We say that a function  $U(x)$  is *symmetric* (even) about the line  $l_k$ ,  $k \in \{1, 2\}$ , in  $\Omega$  if

$$U(x) = U(Q_k x) \quad \forall x \in \Omega_1.$$

We say that a function  $U(x)$  is *antisymmetric* (odd) about the line  $l_k$  in  $\Omega$  if

$$U(x) = -U(Q_k x) \quad \forall x \in \Omega_1.$$

We say an operator  $\mathcal{L}$  *preserves symmetry and antisymmetry about  $l_k$*  if  $\mathcal{L}$  applied to any symmetric (antisymmetric) function about  $l_k$  is a symmetric (antisymmetric) function about  $l_k$ . For example, the  $\Delta$  operator preserves symmetric and antisymmetric functions. We say an operator  $\mathcal{L}$  *reverses symmetry and antisymmetry about  $l_k$*  if  $\mathcal{L}$  applied to any symmetric (antisymmetric) function about  $l_k$  is an antisymmetric (symmetric) function about  $l_k$ . For example, the  $\nabla$  operator reverses symmetric and antisymmetric functions.

For all  $x \in \Omega$ , define

$$(3.5) \quad \mathcal{R}_{00}U(x) = \frac{1}{2} [ U(x) + U(Q_1x) + U(Q_2x) + U(Q_2Q_1x) ],$$

$$(3.6) \quad \mathcal{R}_{11}U(x) = \frac{1}{2} [ U(x) - U(Q_1x) - U(Q_2x) + U(Q_2Q_1x) ],$$

$$(3.7) \quad \mathcal{R}_{10}U(x) = \frac{1}{2} [ U(x) + U(Q_1x) - U(Q_2x) - U(Q_2Q_1x) ],$$

and

$$(3.8) \quad \mathcal{R}_{01}U(x) = \frac{1}{2} [ U(x) - U(Q_1x) + U(Q_2x) - U(Q_2Q_1x) ].$$

Note that

$$\sum_{j,k=0}^1 \mathcal{R}_{jk}^* \mathcal{R}_{jk} = \mathcal{I}.$$

Properties for the  $\mathcal{R}_{jk}$  similar to (3.3) and (3.4) hold.

Suppose that  $\mathcal{L}$  preserves symmetry and antisymmetry about both  $\tau_1$  and  $\tau_2$ . If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{jk}^*, \quad j, k = 0, 1,$$

then Algorithm PMG converges in one iteration with no smoothing (see §3.2). Suppose that  $\mathcal{L}$  reverses symmetry and antisymmetry about both  $\tau_1$  and  $\tau_2$ . If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{1-j,1-k}^*, \quad j, k = 0, 1,$$

then Algorithm PMG converges in one iteration with no smoothing. Suppose that  $\mathcal{L}$  reverses symmetry and antisymmetry about  $\tau_1$ , but preserves symmetry and antisymmetry about  $\tau_2$ . If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{1-j,k}^*, \quad j, k = 0, 1,$$

then Algorithm PMG converges in one iteration with no smoothing. Finally, suppose that  $\mathcal{L}$  reverses symmetry and antisymmetry about  $\tau_2$ , but preserves symmetry and antisymmetry about  $\tau_1$ . If we take

$$\mathcal{P}_{jk} = \mathcal{R}_{j,1-k}^*, \quad j, k = 0, 1,$$

then Algorithm PMG converges in one iteration with no smoothing.

**3.2. Convergence result.** In this section, we show that if  $\mathcal{L}$  preserves or reverses symmetric or antisymmetric functions about internal interfaces, then Algorithm PMG can be made to converge in one iteration without smoothing to the exact solution if the subspace problems are solved exactly.

Suppose we have a set of internal interfaces

$$\{\tau_j\}_{j=1}^\sigma,$$

where each  $\tau_j$  divides  $\Omega$  into two equal sized subdomains, and a set of  $d = 2^\sigma$  restriction operators

$$(3.9) \quad \{\mathcal{R}_{\{\tau_j\}}\}, \quad j = 1, \dots, \sigma,$$

where

$$\tau_j = \begin{cases} 1, & \text{when symmetric functions are annihilated about } \tau_j, \\ 0, & \text{when antisymmetric functions are annihilated about } \tau_j. \end{cases}$$

Each prolongation operator is the adjoint of the correct restriction operator

$$(3.10) \quad \mathcal{P}_{\{\tau_j\}} = \mathcal{R}_{\{\mu_j\}},$$

where

$$\mu_j = \begin{cases} \tau_j, & \text{when } \mathcal{L} \text{ preserves symmetric/antisymmetric functions about } \gamma_j, \\ 1 - \tau_j, & \text{when } \mathcal{L} \text{ reverses symmetric/antisymmetric functions about } \gamma_j. \end{cases}$$

If we renumber the restriction and prolongation operators as

$$\{\mathcal{P}_j, \mathcal{R}_j\}_{j=1}^d,$$

then we can prove the following convergence result.

**THEOREM 3.1.** *Suppose  $\mathcal{L}$  either preserves or reverses symmetric and antisymmetric functions about each of the internal interfaces  $\gamma_j$ ,  $j = 1, \dots, \sigma$ . Suppose the  $d$  restriction and prolongation operators are defined as in (3.9) and (3.10) with (2.2) satisfied. Then Algorithm PMG converges to the exact solution in one iteration without smoothing if the subspace problems are solved exactly.*

*Proof.* Without loss of generality, assume a zero initial guess. Then we are solving, in parallel,  $d$  equations of the form

$$\mathcal{R}_j \mathcal{L} \mathcal{P}_j c_j = \mathcal{R}_j f, \quad j = 1, \dots, d.$$

Apply  $\mathcal{R}_j^*$  to each equation:

$$\mathcal{R}_j^* \mathcal{R}_j \mathcal{L} \mathcal{P}_j c_j = \mathcal{R}_j^* \mathcal{R}_j f, \quad j = 1, \dots, d.$$

Using (2.2),

$$\left[ \mathcal{I} - \sum_{k=1, k \neq j}^d \mathcal{R}_k^* \mathcal{R}_k \right] \mathcal{L} \mathcal{P}_j c_j = \mathcal{R}_j^* \mathcal{R}_j f, \quad j = 1, \dots, d.$$

Noting that  $\mathcal{R}_k \mathcal{L} \mathcal{P}_j c_j = 0$  for all  $j \neq k$  (see (3.9) and (3.10)), we get

$$\mathcal{L} \mathcal{P}_j c_j = \mathcal{R}_j^* \mathcal{R}_j f, \quad j = 1, \dots, d.$$

Summing all  $d$  equations gives us

$$\mathcal{L} \left( \sum_{j=1}^d \mathcal{P}_j c_j \right) = \left( \sum_{j=1}^d \mathcal{R}_j^* \mathcal{R}_j \right) f = f,$$

or

$$u = \sum_{j=1}^d \mathcal{P}_j c_j. \quad \square$$

**3.3. Examples.** In this section, we provide several examples. The first is a constant coefficient problem in  $\sigma$  dimensions,  $\sigma \geq 1$ . Depending on the choice of the coefficients, standard multigrid methods are not useful. However, Algorithm PMG solves these problems in one iteration. The second example is a variable coefficient problem in two dimensions.











We are now ready to analyze the convergence behavior of Algorithm PMG using either  $\{\mathcal{T}_0, \mathcal{T}_1\}$  or  $\{\mathcal{T}_2, \mathcal{T}_3\}$  as the restriction operators.

**THEOREM 4.1.** *Using  $\{\mathcal{R}_j\}_{j=1}^2 = \{\mathcal{T}_k\}_{k=0}^1$ ,  $\mathcal{P}_j = \mathcal{R}_j^T$ , and no smoothing ( $m = 0$ ), each iteration of the main loop of Algorithm PMG reduces the error in the energy norm by a factor bounded above by  $1/3$ , independent of  $N$ , for the problem in (4.1).*

*Proof.* We only provide the details when  $N$  is even. A similar argument works when  $N$  is odd. Note that

$$K_0 = \bar{\mathcal{T}}_0(\bar{\mathcal{T}}_0\mathcal{A}_1\bar{\mathcal{T}}_0^T) + \bar{\mathcal{T}}_0^T = QQ\bar{\mathcal{T}}_0Q(Q\bar{\mathcal{T}}_0Q\Lambda Q\bar{\mathcal{T}}_0^TQ) + Q\bar{\mathcal{T}}_0^TQQ.$$

A similar expression exists for  $K_1 = \bar{\mathcal{T}}_1(\bar{\mathcal{T}}_1\mathcal{A}_1\bar{\mathcal{T}}_1^T) + \bar{\mathcal{T}}_1^T$ . Both  $K_0$  and  $K_1$  decouple into  $N/2$   $2 \times 2$  independent block matrices. Let  $[QK_jQ]_i$  be the  $i$ th decoupled block of  $[QK_jQ]$ . Then

$$[QK_0Q]_i = (x_i y_i)^{-1} \begin{bmatrix} y_i^2 & -x_i y_i \\ -x_i y_i & x_i^2 \end{bmatrix}$$

and

$$[QK_1Q]_i = (x_i^3 + y_i^3)^{-1} \begin{bmatrix} x_i^2 & x_i y_i \\ x_i y_i & y_i^2 \end{bmatrix}.$$

Combining the last two equations gives us

$$[Q(K_0 + K_1)Q]_i = (x_i y_i (x_i^3 + y_i^3))^{-1} \begin{bmatrix} y_i^2(x_i^3 + y_i^3) + x_i^3 y_i & x_i y_i (x_i y_i - x_i^3 - y_i^3) \\ x_i y_i (x_i y_i - x_i^3 - y_i^3) & x_i^2(x_i^3 + y_i^3) + y_i^3 x_i \end{bmatrix}.$$

The  $i$ th decoupled block of the error  $\mathcal{E}$  iteration matrix (see (2.6)) is

$$[Q\mathcal{E}Q]_i = [Q(I - \Lambda^{1/2}(K_0 + K_1)\Lambda^{1/2})Q]_i.$$

Let  $\theta_i = i\pi h$ . Then

$$x_i - y_i = -\cos \theta_i, \quad x_i y_i = \frac{1}{4} \sin^2 \theta_i, \quad \text{and}$$

$$x_i^3 + y_i^3 = \frac{1}{4}(1 + \cos^2 \theta_i).$$

Hence,

$$[Q\mathcal{E}Q]_i = (1 + 3 \cos^2 \theta_i)^{-1} \begin{bmatrix} \sin^2 \theta_i \cos \theta_i & 2 \sin \theta_i \cos^2 \theta_i \\ 2 \sin \theta_i \cos^2 \theta_i & -\sin^2 \theta_i \cos \theta_i \end{bmatrix},$$

which has eigenvalues

$$\lambda^2 = (\sin^2 \theta_i \cos^2 \theta_i)(1 + 3 \cos^2 \theta_i)^{-1}.$$

The maximum value of  $\lambda$  occurs when  $\cos \theta = 1/\sqrt{3}$  with maximum value

$$\lambda_{\max} = \frac{1}{3}.$$

TABLE 1  
Contraction factors for one-dimensional model problem

$N = 150$ ,  $m =$  number of smoothing iterations

$m$	$\{\mathcal{T}_0, \mathcal{T}_1\}$	$\{\mathcal{V}_0, \mathcal{V}_1\}$	$\{\mathcal{S}_{00}, \mathcal{S}_{11}, \mathcal{S}_{10}, \mathcal{S}_{01}\}$
$0 (N \rightarrow \infty)$	1/3	0	1/3
0	.3333	0	.3004
1	.1822	0	.2289
2	.1934	0	.1922

Finally,

$$\rho(\mathcal{E}) = \max \rho([Q\mathcal{E}Q]_i) \leq \frac{1}{3}. \quad \square$$

We state the following theorem without proof, which is analogous to that of Theorem 4.1.

**THEOREM 4.2.** *Using  $\{\mathcal{R}_j\}_{j=1}^2 = \{\mathcal{T}_k\}_{k=2}^3$ ,  $\mathcal{P}_j = \mathcal{R}_j^T$ , and no smoothing ( $m = 0$ ), each iteration of the main loop of Algorithm PMG reduces the error in the energy norm by a factor bounded above by  $1/3$ , independent of  $N$ .*

The third set of restriction operators are compositions of the first two sets, using intermediate grids. This is similar to the total reduction method variant of multigrid. Define

$$\tilde{N} = \begin{cases} N/2, & N \text{ even,} \\ (N+1)/2, & N \text{ odd.} \end{cases}$$

Define the restriction operators  $\{\mathcal{S}_{jk}\}_{j,k=0}^1$  by

$$(4.3) \quad \mathcal{S}_{jk} = \mathcal{T}_{\alpha_{jk}} \mathcal{V}_k,$$

where  $\mathcal{T}_{\alpha_{jk}}$  and  $\mathcal{V}_k$  are the appropriate sizes, and

$$(4.4) \quad \alpha_{jk} = \begin{cases} 2k + j, & \tilde{N} \text{ even,} \\ 2(1-k) + j, & \tilde{N} \text{ odd.} \end{cases}$$

We are generating four problems, each with approximately  $N/4$  unknowns. We have verified numerically the following conjecture.

**CONJECTURE 4.1.** *Using  $\{\mathcal{R}_i\}_{i=1}^4 = \{\mathcal{S}_{jk}\}_{j,k=0}^1$  and  $\mathcal{P}_i = \mathcal{R}_i^T$ , and no smoothing ( $m = 0$ ), each iteration of the main loop of Algorithm PMG reduces the error in the energy norm by a factor bounded above by  $1/3$ , independent of  $N$ , for the problem in (4.1).*

Table 1 contains contraction factors for one iteration of the main loop of Algorithm PMG for the problem in (4.1).

**4.2. Two dimensional problems.** We re-investigate the model domain of §3.1.2 applied to the problem

$$(4.5) \quad \begin{cases} -au_{xx} - bu_{yy} + cu = f & \text{in } \Omega = (0, 1) \otimes (0, 1), \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

Using an  $N = n \times n$  uniform mesh for the unknowns and central differences leads to a block tridiagonal coefficient matrix  $\mathcal{A}_2$  of the form

$$(4.6) \quad \mathcal{A}_2 = [ -a, \dots, -b, 2(a+b) + ch^2, -b, \dots, -a ],$$

where  $h = 1/(n+1)$ . Let  $I_n$  be the  $n \times n$  identity matrix. If  $a = b = 1$  and  $c = 0$  in (4.5), then

$$\mathcal{A}_2 = \mathcal{A}_1 \otimes I_n + I_n \otimes \mathcal{A}_1,$$

where  $\mathcal{A}_1$  is defined in (4.2).

We define restriction operators as the tensor products of the one-dimensional operators. Let

$$\mathcal{V}_{jk} = \mathcal{V}_j \otimes \mathcal{V}_k, \quad j, k = 0, 1,$$

and

$$\mathcal{T}_{jk} = \mathcal{T}_j \otimes \mathcal{T}_k, \quad j, k = 0, 1, \quad \text{or} \quad j, k = 2, 3.$$

**THEOREM 4.3.** *Using  $\{\mathcal{R}_j\}_{j=1}^4 = \{\mathcal{T}_{ik}\}_{i,k=0}^1$ ,  $\mathcal{P}_j = \mathcal{R}_j^T$ , and no smoothing ( $m = 0$ ), each iteration of the main loop of Algorithm PMG reduces the error in the energy norm by a factor bounded above by  $1/3$ , independent of  $N$ , for the problem in (4.5) when  $a = b = 1$  and  $c = 0$ .*

*Proof.* Define

$$\bar{\mathcal{T}}_{jk} = \bar{\mathcal{T}}_j \otimes \bar{\mathcal{T}}_k, \quad j, k = 0, 1,$$

and

$$K_{jk} = \bar{\mathcal{T}}_{jk} (\bar{\mathcal{T}}_{jk} \mathcal{A}_2 \bar{\mathcal{T}}_{jk}^T)^+ \bar{\mathcal{T}}_{jk}^T.$$

Using the rules of matrix multiplication of tensor products of square matrices, we can show that

$$\begin{aligned} K_{jk} = & (\bar{\mathcal{T}}_j \otimes \bar{\mathcal{T}}_k)^T \{ (\bar{\mathcal{T}}_j \mathcal{A}_1 \bar{\mathcal{T}}_j^T) \otimes (\bar{\mathcal{T}}_k \bar{\mathcal{T}}_k^T) + \\ & (\bar{\mathcal{T}}_j \bar{\mathcal{T}}_j^T) \otimes (\bar{\mathcal{T}}_k \mathcal{A}_1 \bar{\mathcal{T}}_k^T) \}^+ (\bar{\mathcal{T}}_j \otimes \bar{\mathcal{T}}_k). \end{aligned}$$

Each  $K_{jk}$  decouples into  $N/4$   $4 \times 4$  independent block matrices similar to the one-dimensional case. The remainder of the proof mimics the proof of Theorem 4.1. We used the Scratchpad II symbol manipulation system to determine the polynomial representations for the eigenvalues and then maximized those to complete the result.  $\square$

When  $N$  is divisible by four, the block structure of each of the four subproblems  $\mathcal{T}_{jk}^T \mathcal{A}_2 \mathcal{T}_{jk}$  is identical to the original problem on a grid with  $N/2$  points on a side. Hence, the amount of storage required by this form of Algorithm PMG is at least as great as that required when  $\mathcal{V}_{jk}^T \mathcal{A}_2 \mathcal{V}_{jk}$  are used.

We can construct 16 restriction operators  $\{\mathcal{S}_{ijkl}\}_{i,j,k,l=0}^1$  similar to the one-dimensional equivalents ( $\{\mathcal{S}_{jk}\}_{j,k=0}^1$ ) by using intermediate grids. For  $N$  even, define

$$\mathcal{S}_{ijkl} = (\mathcal{T}_{\alpha_{ik}} \mathcal{V}_k) \otimes (\mathcal{T}_{\alpha_{jl}} \mathcal{V}_l) = \mathcal{T}_{\alpha_{ik}, \alpha_{jl}} \mathcal{V}_{kl},$$

TABLE 2  
Contraction factors for two-dimensional model problems

$$a = b = c = 1$$

$m$	$\{\mathcal{T}_{jk}\}_{j,k=0}^1$	$\{\mathcal{V}_{jk}\}_{j,k=0}^1$	$\{\mathcal{S}_{ijkl}\}_{i,j,k,l=0}^1$
0 ( $N \rightarrow \infty$ )	1/3	0	1/3
0	.3323	0	.3263
1	.2653	0	.2581
2	.2175	0	.2120
3	.1795	0	.1756

$$a = 10^{-5}, \quad b = 10^6, \quad c = 1$$

$m$	$\{\mathcal{T}_{jk}\}_{j,k=0}^1$	$\{\mathcal{V}_{jk}\}_{j,k=0}^1$	$\{\mathcal{S}_{ijkl}\}_{i,j,k,l=0}^1$
0 ( $N \rightarrow \infty$ )	1/3	0	1/3
0	.3327	0	.3269
1	.1998	0	.1946
2	.1319	0	.1303
3	.0976	0	.1000

$N = 16 \times 16$ ,  $m =$  number of smoothing iterations

where  $\mathcal{T}_{\alpha_{ik}, \alpha_{jl}}$  and  $\mathcal{V}_{kl}$  are the appropriate sizes, and

$$\alpha_{ik} = \begin{cases} 2k + i & \text{if } 4 \mid n, \\ 2(1-k) + i & \text{if } 2 \mid n \text{ and } 4 \nmid n. \end{cases}$$

REMARK 4.1. *When  $N$  is odd, a more complicated definition of  $\mathcal{S}_{ijkl}$  exists.*

We are generating 16 problems, each with approximately  $N/16$  unknowns. We have verified numerically the following conjecture:

CONJECTURE 4.2. *Using  $\{\mathcal{R}_i\}_{i=1}^{16} = \{\mathcal{S}_{ijkl}\}_{i,j,k,l=0}^1$ ,  $\mathcal{P}_i = \mathcal{R}_i^T$ , and no smoothing ( $m = 0$ ), each iteration of the main loop of Algorithm PMG reduces the error in the energy norm by a factor bounded above by  $1/3$ , independent of  $N$ , for the problem in (4.5) when  $a = b = 1$  and  $c = 0$ .*

Table 2 contains contraction factors for one iteration of the main loop of Algorithm PMG for the problem in (4.1) with two sets of coefficients  $\{a, b, c\}$ :  $\{1, 1, 1\}$  and  $\{10^{-5}, 10^6, 1\}$ .

We conclude this section by noting that comparing convergence rates is interesting only if there are operation counts or timing information available, too. We determine this information in §5 and make concrete conclusions about which set of restriction operators to use.

**5. Timing comparisons.** In this section, we attempt to compare the running time of Algorithm PMG for three sets of restriction operators. We use two measures: *total operation counts* and *wall clock time*. By total operation counts, we include all of the arithmetic operations on every level. By wall clock time, we include the slowest

operations for a given level and part of Algorithm PMG. This is actually the figure that counts since it measures the running time per iteration of our algorithm.

Throughout this section, the number of unknowns on the finest level is  $N$  and the number of unknowns for any problem on a level  $j$  is  $N_j$ . We assume that

$$\varphi = N_j / N_{j-1} > 1, \quad j > 1,$$

and that (2.4) holds. Let  $x \in \mathbb{R}^{N_j}$ . The cost per major operation is given by

Operation	Cost
smoothing	$C_0 N_j$
residual	$C_1 N_j$
coarse level solves	$C_2 N_1^\Phi$
$\mathcal{T}x$	$C_3 N_j$
$\mathcal{V}x$	$C_4 N_j$
$\mathcal{S}x$	$C_5 N_j$

The value of  $C_1$  is dependent on the value of  $m$  (the number of smoothing iterations) and the smoother. Many smoothers compute the residual as part of their algorithm (see [2]), in which case,  $C_1 = 0$  whenever  $m > 0$ .

The value of  $\Phi$  depends on the solution method on level 1. If a band solver on a single processor is used, then  $\Phi = 1$  for one-dimensional problems, and  $\Phi = 2$  for two-dimensional problems. If a conjugate gradient method on  $N_1$  processors is used, then  $\Phi = 1$ . These are the extreme cases. There are cases where the cost is actually  $N_1^\Phi \log N_1$ .

Consider the one-dimensional problem from §4.1. For a uniform mesh,

Constant	Multiplications	Additions
$C_3$	2	2
$C_4$	1	1
$C_5$	2	2

For two levels with the subspace problems solved on individual processors, the total cost per iteration of Algorithm PMG by set of restriction operators is given by

Restriction operators	Cost
$\{\mathcal{T}_0, \mathcal{T}_1\}$	$(C_1 + C_2 + 4C_3)N$
$\{\mathcal{V}_0, \mathcal{V}_1\}$	$(C_1 + C_2 + 4C_4)N$
$\{\mathcal{S}_{00}, \mathcal{S}_{11}, \mathcal{S}_{10}, \mathcal{S}_{01}\}$	$(C_1 + C_2 + 8C_5)N$

Using more than two levels actually increases the running time.

The cost and storage requirements of Algorithm PMG using either  $\{\mathcal{T}_0, \mathcal{T}_1\}$  or  $\{\mathcal{V}_0, \mathcal{V}_1\}$  are nearly identical. Hence, the only criterion to determine which set of restriction operators is the contraction factor for each set. Based on Theorem 3.1, Conjecture 4.1, and Table 1, it is clear that the correct choice is  $\{\mathcal{V}_0, \mathcal{V}_1\}$ . For the one-dimensional example,  $\{\mathcal{S}_{00}, \mathcal{S}_{11}, \mathcal{S}_{10}, \mathcal{S}_{01}\}$  is not competitive.

Consider the two-dimensional problem from §4.2. For a uniform mesh,

Constant	Multiplications	Additions
$C_3$	3	8
$C_4$	1	3
$C_5$	1.75	5

For two levels with the subspace problems solved on individual processors, the total cost per iteration of Algorithm PMG is given by

$$(mC_0 + C_1 + 2\varphi C_k)N + \varphi C_2 N_1^2, \quad k \in \{3, 4, 5\},$$

or

Restriction operators	Cost
$\{\mathcal{T}_{jk}\}$	$(mC_0 + C_1 + 8C_3)N + C_2 N^2/4$
$\{\mathcal{V}_{jk}\}$	$(mC_0 + C_1 + 8C_4)N + C_2 N^2/4$
$\{\mathcal{S}_{ijkl}\}$	$(mC_0 + C_1 + 32C_5)N + C_2 N^2/16$

The total cost and storage requirements of Algorithm PMG using either  $\{\mathcal{V}_{jk}\}$  are bounded above by those for  $\{\mathcal{T}_{jk}\}$ . Hence, the only criterion to determine which set of restriction operators is the contraction factor for each set. Based on Theorem 3.1, Conjecture 4.2, and Table 2, it is clear that the correct choice is  $\{\mathcal{V}_{jk}\}$ . The cost per iteration for  $\{\mathcal{S}_{ijkl}\}$  is significantly less than that for  $\{\mathcal{V}_{jk}\}$ . To determine which to use is a function of how accurate an approximation is required and whether or not space is a consideration.

When more than two levels are used, with the subspace problems on level 1 solved on individual processors, the total cost per iteration of Algorithm PMG is asymptotically (in the number of levels)

$$(mC_0 + C_1 + 2\varphi C_k)N \log_\varphi N, \quad k \in \{3, 4, 5\}.$$

This can be reduced significantly by using more than one processor during the smoothing, residual computation, and/or restriction/prolongation phases. The total cost and storage requirements of Algorithm PMG using either  $\{\mathcal{V}_{jk}\}$  are bounded above by those for  $\{\mathcal{T}_{jk}\}$ . As before, the correct choice is clearly  $\{\mathcal{V}_{jk}\}$ . The cost per iteration for  $\{\mathcal{S}_{ijkl}\}$  is not less than that for  $\{\mathcal{V}_{jk}\}$ . To determine which to use is a function of how accurate an approximation is required and whether or not space is a consideration.

The total number of operations is not always the best measure of a parallel algorithm. Another approach is to measure the cost of the slowest part of any parallel step in an algorithm. For two levels with the subspace problems solved on individual processors, the wall clock costs per iteration of Algorithm PMG are given by

$$(mC_0 + C_1 + 2C_k)N + C_2(N_1/\varphi)^2, \quad k \in \{3, 4, 5\},$$

or

Restriction operators	Cost
$\{\mathcal{T}_{jk}\}$	$(mC_0 + C_1 + 2C_3)N + C_2 N^2/16$
$\{\mathcal{V}_{jk}\}$	$(mC_0 + C_1 + 2C_4)N + C_2 N^2/16$
$\{\mathcal{S}_{ijkl}\}$	$(mC_0 + C_1 + 2C_5)N + C_2 N^2/256$

In this case, it is obvious that the cost per iteration for  $\{\mathcal{S}_{ijkl}\}$  is significantly less than that for either  $\{\mathcal{T}_{jk}\}$  or  $\{\mathcal{V}_{jk}\}$ .

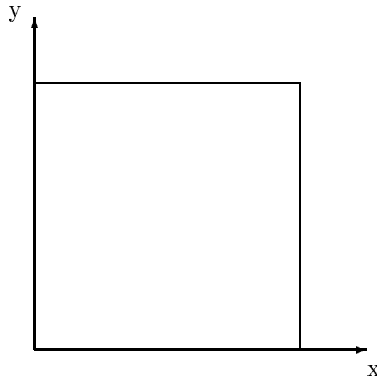
When more than two levels are used, with the subspace problems on level 1 solved on individual processors, the total cost per iteration of Algorithm PMG is asymptotically (in the number of levels)

$$\frac{\varphi}{\varphi - 1}(mC_0 + C_1 + 2C_k)N, \quad k \in \{3, 4, 5\}.$$

This can be reduced significantly by using more than one processor during the smoothing, residual computation, and/or restriction/prolongation phases. The wall clock time of Algorithm PMG using  $\{\mathcal{V}_{jk}\}$  is bounded above by that for  $\{\mathcal{T}_{jk}\}$ . As before, the correct choice is clearly  $\{\mathcal{V}_{jk}\}$ . The wall clock time per iteration for  $\{\mathcal{S}_{ijkl}\}$  is less than that for  $\{\mathcal{V}_{jk}\}$ . To determine which to use is a function of how accurate an approximation is required and whether or not space is a consideration.

**6. Domain decomposition interpretation.** In this section, we interpret Algorithm PMG as a domain decomposition method. We compare its interpretation with standard block domain decomposition methods. While the definition of Algorithm PMG is more general than standard domain decomposition methods, its use in this paper is not.

Standard domain decomposition methods break the domain into a collection of blocks with local communication along (or near) the internal borders between the blocks. For example, suppose the domain is



We can decompose this into

3	3	3	3	4	4	4	4
3	3	3	3	4	4	4	4
3	3	3	3	4	4	4	4
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2
1	1	1	1	2	2	2	2

The numbers 1, 2, 3, and 4 correspond to the subdomain which contains the data point. We compute, by some technique, in each of the four quadrants (in parallel), pass boundary information between relevant processors, and continue processing. In essence, we have a block method with local communication between blocks of data.

Algorithm PMG with  $\{\mathcal{T}_{jk}\}_{j,k=0}^1$  or  $\{\mathcal{V}_{jk}\}_{j,k=0}^1$  decomposes the domain pointwise

as something similar to

```

3 4 3 4 3 4 3 4
1 2 1 2 1 2 1 2
3 4 3 4 3 4 3 4
1 2 1 2 1 2 1 2
3 4 3 4 3 4 3 4
1 2 1 2 1 2 1 2

```

Here, the numbers 1, 2, 3, and 4 correspond to the subspace into which the data point is principally mapped. Clearly, this can be expanded to correspond to the case of  $\{\mathcal{S}_{ijkl}\}_{i,j,k,l=0}^1$  or any similar abstraction. The correction produced by each subspace interferes with future computation in each of its neighboring points on the fine grid. Hence, we can think of this as global communication.

The principal difference between this narrow interpretation of Algorithm PMG and standard domain decomposition methods for partial differential equations is that Algorithm PMG can be a pointwise decomposition method with global communication and standard domain decomposition methods are block decomposition methods with local communication.

**7. Conclusions.** We have demonstrated that operator and domain properties can easily be employed to produce fast parallel direct methods based on multigrid algorithms. Due to the geometric nature of the idea in this paper, the technique can easily be applied to well-posed problems on strangely shaped domains. Further, constructing a more parallel method that is iterative is simple and computationally attractive (both in time and storage).

**Acknowledgments.** Since writing this paper, we have become aware that a different interpretation of the analysis of §3 was done independently by Chen, Kamath, and Sameh (see [6], [7], and [17]).

The reader should be aware that significant extensions to the theory of §3 have occurred since this paper was written. In [5], the two-dimensional example on a square (see Fig. 2) is solved using an eight-way technique. This is extended in [11] to a three-dimensional problem on a cube that is solved using 8, 60, and 64-way techniques. Finally, in [11], the analysis of this paper is extended to the case when the subproblems are solved inexactly (i.e., using iterative methods) and when the subspaces are not orthogonal.

Last, but not least, we would like to thank Franco Brezzi, Donatella Marini, Willard Miranker, and Steve McCormick for their extensive comments about this paper.

## REFERENCES

- [1] R. E. ALCOUFFE, A. BRANDT, J. J. E. DENDY, AND J. W. PAINTER, *The multi-grid methods for the diffusion equation with strongly discontinuous coefficients*, SIAM J. Sci. Stat. Comp., 2 (1981), pp. 430–454.
- [2] R. E. BANK AND C. C. DOUGLAS, *An efficient implementation of the SSOR and ILU preconditionings*, Appl. Numer. Math., 1 (1985), pp. 489–492.
- [3] ———, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22 (1985), pp. 617–633.
- [4] A. BRANDT AND S. TA'ASAN, *Multigrid method for nearly singular and slightly indefinite problems*, in Multigrid Methods II, W. Hackbusch and U. Trottenberg, eds., Springer-Verlag, New York, 1986, pp. 100–121.

- [5] F. BREZZI, C. C. DOUGLAS, AND L. D. MARINI, *A parallel domain reduction method*, Numer. Meth. for PDE, 5 (1989), pp. 195–202.
- [6] H. C. CHEN, *The SAS domain decomposition method for structural analysis*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1988.
- [7] H. C. CHEN AND A. H. SAMEH, *A matrix decomposition method for orthotropic elasticity problems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 39–64.
- [8] C. C. DOUGLAS, *Multi-grid algorithms for elliptic boundary-value problems*, PhD thesis, Yale University, May 1982. Also, Computer Science Department, Yale University, Technical Report 223.
- [9] ———, *Multi-grid algorithms with applications to elliptic boundary-value problems*, SIAM J. Numer. Anal., 21 (1984), pp. 236–254.
- [10] C. C. DOUGLAS, S. C. MA, AND W. L. MIRANKER, *Generating parallel algorithms through multi-grid and aggregation/disaggregation techniques*, in Computational Acoustics: Algorithms and Applications, D. Lee, R. L. Sternberg, and M. H. Schultz, eds., Elsevier, North-Holland, 1987, pp. 133–147.
- [11] C. C. DOUGLAS AND J. MANDEL, *The domain reduction method: high way reduction in three dimensions and convergence with inexact solvers*, in Fourth Copper Mountain conference on multigrid methods, J. Mandel and S. F. McCormick, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1989. To appear.
- [12] C. C. DOUGLAS AND W. L. MIRANKER, *Constructive interference in parallel algorithms*, SIAM J. Numer. Anal., 25 (1988), pp. 376–398.
- [13] ———, *Some nontelegraphing parallel algorithms based on serial multigrid/aggregation/disaggregation techniques*, in Multigrid Methods: Theory, Applications, and Supercomputing, S. F. McCormick, ed., Marcel Dekker, New York, 1988, pp. 167–176.
- [14] H. FOERSTER, K. STUBEN, AND U. TROTTEMBERG, *Non-standard multigrid techniques using checkerboard relaxation and intermediate grids*, in Elliptic Problem Solvers, M. H. Schultz, ed., Academic Press, New York, 1981, pp. 285–300.
- [15] P. FREDERICKSON AND O. MCBRYAN, *Parallel superconvergent multigrid*, in Multigrid Methods: Theory, Applications, and Supercomputing, S. F. McCormick, ed., Marcel Dekker, New York, 1988, pp. 195–210.
- [16] W. HACKBUSCH, *A new approach to robust multi-grid solvers*, in ICIAM'87: Proceedings of the First International Conference on Industrial and Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1988, pp. 114–126.
- [17] C. KAMATH AND A. H. SAMEH, *A projection method for solving nonsymmetric linear systems on multiprocessors*, Parallel Comp., 9 (1989), pp. 291–312.
- [18] J. W. RUGE AND K. STÜBEN, *Algebraic multigrid*, in SIAM Frontiers on Applied Mathematics, Volume 3, Multigrid Methods, S. F. McCormick, ed., SIAM, 1987, pp. 73–130.
- [19] R. S. VARGA, *Matrix Iterative Analysis*, Prentice-Hall, New York, 1962.