A Variation of the Schwarz Alternating Method: The Domain Decomposition Reduction Method

Craig C. Douglas*

Abstract. Domain decomposition methods are highly parallel methods for solving elliptic partial differential equations. In many domain decomposition variants, the domain is partitioned into a number of (possibly overlapping) subdomains before computation begins. In contrast, the domain reduction method folds the domain into a number of smaller domains covering only a small portion of the entire domain. The solution over the entire domain is recovered by unfolding the solutions on the subdomains and summing them. The cost of the folding and unfolding is negligible. All steps of the algorithm are embarrassingly parallel.

1. Introduction. In this paper, the solution to an elliptic boundary value problem is approximated using a combination of domain decomposition, multigrid, and projection method techniques (see [1], [2], [4], [7], [13], [14], [16], and [17]).

Consider problems of the following form.

$$
(1.1) \qquad
\begin{cases}
\mathcal{L}u \; = f \quad \text{in } \Omega \subset \mathbb{R}^d, \;\; d \; > \; 0, \\[2mm]
u \; = \; 0 \quad \text{on } \partial\Omega.
\end{cases}
$$

More complicated boundary conditions are discussed in [3] and [8]. That (1.1) is well posed and will be discretized by a finite element, volume, or difference scheme is assumed throughout this paper. Typically,

$$
\mathcal{L}u \; = \; - \sum_{i,j=1}^{d} \frac{\partial}{\partial x_i} \left(a_{ij}(x) \frac{\partial u}{\partial x_j} \right) + a_0(x)u.
$$

The domain reduction method uses properties of a partial differential equation to split the problem into several subproblems. Each subproblem should be solved in parallel using the fastest known solution method appropriate.

* Mathematical Sciences Department, IBM Research Division, Thomas J. Watson Research Center, P. O. Box 218, Yorktown Heights, NY 10598.

The basic method was first derived in [9] as a parallel multilevel method for linear systems of equations where smoothing was unnecessary. The applicability to shared or distributed memory, coarse or fine grained parallel computers was discussed informally in [10]. The elliptic partial differential equation case (with an emphasis on one and two dimensions) was analyzed in [11], and a technique for doubling the basic parallelism was introduced in [3]. This was extended to three dimensional problems in [8]. In the general case, or when the subproblems are solved only approximately, the method becomes an iterative method or can be used as a preconditioner. Bounds on the resulting convergence factors and condition numbers are given in [8].

In §2, the method is defined as it pertains to this paper in a very abstract manner. In §3, a simple 2^d way domain reduction is defined when $\Omega = (-1, 1)^d$. In §4, more complicated domain reductions are defined. In §5, the algorithm is advocated as the inner part of a standard domain decomposition algorithm.

2. The Method. In this section, the domain reduction method is defined in three algorithms. The first two are special cases of the quite general third. Some theorems that pertain to the convergence of these algorithms are stated with references to where the proofs can be found.

A variational formulation of (1.1) is

(2.1) $$\text{find } u \in V \text{ such that } a(u,v) = f(v), \; \forall v \in V.$$

Typically, $V = V(\Omega)$ is the appropriate Sobolev space, and

$$a(u,v) = \int_\Omega a \nabla u \nabla v + a_0 uv dx \quad \text{and} \quad f(v) = \int_\Omega fv dx.$$

The method may be formulated quite abstractly.

ALGORITHM 1. *To solve (2.1), subspaces $V_1, V_2, \ldots, V_n$ are found such that*

$$V = \bigoplus_{i=1}^{n} V_i \quad and \quad V_i \perp V_j, \; i \neq j,$$

in the $a(\cdot, \cdot)$ inner product. Then the solution is

$$u = \sum_{i=1}^{n} u_i,$$

where u_i is the solution of the subproblem

(2.2) $$find \; u_i \in V_i \; such \; that \; a(u_i, v) = f(v), \; \forall v \in V_i.$$

In reality, the cost of solving (2.2) by direct methods is often unreasonable. A more realistic definition of the method can be formulated.

ALGORITHM 2. *For an approximate solution u^m of (2.1), form the subproblems*

$$(2.3) \qquad find \ w_i \in V_i \ : \ a(w_i, v_i) \ = \ f(v_i) \ - \ a(u^m, v_i), \ \forall v_i \in V_i,$$

and find approximate solutions $\bar{w}_i \in V_i$ such that

$$(2.4) \qquad\qquad\qquad \|\bar{w}_i - w_i\| \ \leq \ \varepsilon \|w_i\|,$$

where $\| \cdot \|$ is the energy norm and $\varepsilon < 1$ is a constant. Then set

$$u^{m+1} \ = \ u^m \ + \ \sum_{i=1}^{n} \bar{w}_i.$$

Note that step (2.4) can be realized by solving (2.3) by an iterative method with convergence factor at worst ε in the energy norm, starting from an initial guess of zero. If more than one iterative method is used, then ε is the maximum of the convergence factors. If the subproblems are solved exactly, the method is related to the well-known Schwarz alternating method [21].

In implementation, full-rank restriction and prolongation operators are used to transfer information between the subspaces and V:

$$\mathcal{R}_i : V \to V_i \quad \text{and} \quad \mathcal{P}_i : V_i \to V.$$

Frequently, each prolongation operator is the adjoint of one of the restriction operators. Writing the problem (2.2) as $\mathcal{L}u = f$, the corresponding operators for the subproblems are then defined in the Galerkin sense:

$$\mathcal{L}_i \ = \ \mathcal{R}_i \mathcal{L} \mathcal{P}_i.$$

The operator $\mathcal{L}$ is discretized into a matrix $\mathcal{A}$. The subproblems, $\mathcal{A}_i$, are defined in the Galerkin sense, as well. Examples of several choices of restriction and prolongation operators, and the resulting $\mathcal{L}_i$ and $\mathcal{A}_i$, are given in [11].

The complete discrete algorithm is defined in a two level (multigrid-like) formulation.

ALGORITHM 3. *The domain reduction method is*
  Smooth s times on u to get u^0
  Do $j = 1, \cdots, m$ {
    Compute residual $r^j = f - \mathcal{A}u^{j-1}$
    Solve in parallel, $i = 1, \cdots, n$:
      $\mathcal{A}_i c^i = \mathcal{R}_i r^j$
    Set $c = u^{j-1} + \sum_{i=1}^{n} \mathcal{P}_i c^i$
    Smooth s times on c to get u^j
    }
  Set $u = u^n$
  }

The parallel solve step in Algorithm 3 can be another execution of Algorithm 3, producing a tree struction of subproblems. More typically, it is a direct solver (cf.,

Algorithm 1) or an iterative solver (cf., Algorithm 2). Algorithm 3 is related to algorithms analyzed in [5], [6], [12], [15], [18], [19], and [20].

The convergence of Algorithms 1 and 3 is guaranteed by the following theorem.

THEOREM 1. *Let $s = 0$ in Algorithm 3, $\Pi_i = \mathcal{P}_i \mathcal{R}_i$, and*

$$\sum_{i=1}^{n} \Pi_i \; = \; I.$$

Then Algorithms 1 and 3 converge to the exact solution if $[\Pi_i, A] = 0$ for all i, $1 \leq i \leq n$.

An immediate corollary is the following.

COROLLARY 1. *Theorem 1 remains true for Algorithm 1 if the condition $[\Pi_i, A] = 0$ is replaced by $\Pi_i A(\Pi_i - I) = 0$ for all i, $1 \leq i \leq n$.*

The proofs to Theorem 1 and Corollary 1 can be found in [9].

Suppose there is a set of internal interfaces

$$\{\Gamma_i\}_{i=1}^{\sigma},$$

where each Γ_j divides Ω into two equal sized subdomains, and a set of $n = 2^\sigma$ restriction operators

$$(2.5) \qquad \qquad \{\mathcal{R}_{\{\tau_i\}}\}, \quad i = 1, \cdots, \sigma,$$

where

$$\tau_i \; = \; \begin{cases} 1, & \text{when even functions are annihilated about } \Gamma_i, \\ 0, & \text{when odd functions are annihilated about } \Gamma_i. \end{cases}$$

An operator $\mathcal{L}$ *preserves even/odd functions about* Γ_k if $\mathcal{L}$ applied to any even (odd) function about Γ_k is a even (odd) function about Γ_k. For example, the Δ operator preserves even and odd functions. Similarly, an operator $\mathcal{L}$ *reverses even/odd functions about* Γ_k if $\mathcal{L}$ applied to any even (odd) function about Γ_k is an odd (even) function about Γ_k. Each prolongation operator is the adjoint of the correct restriction operator

$$(2.6) \qquad \qquad \mathcal{P}_{\{\tau_i\}} \; = \; \mathcal{R}_{\{\mu_i\}},$$

where

$$\mu_i \; = \; \begin{cases} \tau_i, & \text{when } \mathcal{L} \text{ preserves even/odd functions about } \Gamma_i, \\ 1 - \tau_i, & \text{when } \mathcal{L} \text{ reverses even/odd functions about } \Gamma_i. \end{cases}$$

If we renumber the restriction and prolongation operators as

$$\{\mathcal{P}_i, \mathcal{R}_i\}_{i=1}^{n},$$

then the following convergence result holds.

THEOREM 2. *Suppose $\mathcal{L}$ either preserves or reverses even and odd functions about each of the internal interfaces Γ_i, $i = 1, \cdots, \sigma$. Suppose the n restriction and prolongation operators are defined as in (2.5) and (2.6) with*

$$\sum_{i=1}^{d} \mathcal{R}_i^* \mathcal{R}_i = I$$

satisfied. Then Algorithm PMG converges to the exact solution in one iteration without smoothing if the subspace problems are solved exactly.

The proof to Theorem 2 can be found in [11].

The convergence of Algorithm 2 is guaranteed by the following theorem.

THEOREM 3. *Let P_{V_i} be the orthogonal projection onto V_i, and λ_{max} and λ_{min} be the maximal and the minimal eigenvalues, respectively, of $\sum_{i=1}^{n} P_{V_i}$,*

$$\lambda_{max} = \max \sigma \left(\sum_{i=1}^{n} P_{V_i} \right), \qquad \lambda_{min} = \max \sigma \left(\sum_{i=1}^{n} P_{V_i} \right).$$

Then the iterates produced by Algorithm 2 satisfy the error bound

$$(2.7) \qquad \|u^{m+1} - u\| \leq \left(\max\{\lambda_{max} - 1, 1 - \lambda_{min}\} + \varepsilon\lambda_{max} \right) \|u^m - u\|.$$

An immediate corollary is the following.

COROLLARY 2. *Using the same assumptions as in Theorem 3, if, in addition, $V_i \perp V_j$, for all i, j, then $\lambda_{max} = \lambda_{min} = 1$ and (2.7) reduces to*

$$\|u^{m+1} - u\| \leq \varepsilon\|u^m - u\|.$$

The proofs to Theorem 3 and Corollary 2 can be found in [8].

3. An Example. In this section, a simple example is described where the domain reduction method uses symmetries in (1.1) to split the problem into several subproblems. For $x = (x_1, \ldots, x_d) \in \Omega = (-1, 1)^d$, define

$$\mathcal{R}_{i_1, \ldots, i_d} x = ((-1)^{i_1} x_1, \ldots, (-1)^{i_d} x_d), \quad (i_1, \ldots, i_d) \in \{0, 1\}^d.$$

The subspaces $V_{i_1, \ldots, i_d} \subset V$ are defined by

$$V_{i_1, \ldots, i_d} = \{ u \in V : u \circ \mathcal{R}_{i_1, \ldots, i_d} = u \}, \quad (i_1, \ldots, i_d) \in \{0, 1\}^d.$$

It is immediate that the subspaces are mutually orthogonal in the $a(\cdot, \cdot)$ inner product as well as cover the entire solution space V. Therefore, (2.1) may be split into 2^d subproblems:

$$(3.1) \qquad \text{find } u_{i_1, \ldots, i_d} \in V_{i_1, \ldots, i_d} \text{ such that } a(u_{i_1, \ldots, i_d}, v) = f(v), \ \forall v \in V_{i_1, \ldots, i_d},$$

and recover the solution of (2.1) as

$$u = \sum_{i_1, \ldots, i_d = 0}^{1} u_{i_1, \ldots, i_d}.$$

Let $\tilde{\Omega} = (0,1)^d$. For a function $\tilde{v}$ on $\tilde{\Omega}$, define prolongation operators $\mathcal{P}_{i_1,\ldots,i_d}, (i_1, \ldots, i_d) \in \{0,1\}^d$, by

$$(\mathcal{P}_{i_1,\ldots,i_d}\tilde{v})(x) = (-1)^{i_1 i'_1 + \cdots + i_d i'_d}\tilde{v})(\tilde{x}) \quad \text{for } x = \mathcal{R}_{i'_1,\ldots,i'_d}\tilde{x}.$$

Then, (3.1) can be formulated as a problem on the domain $\tilde{\Omega}$ of the form

(3.2)   find $\tilde{u}_{i_1,\ldots,i_d} \in \tilde{V}_{i_1,\ldots,i_d}$ such that $\tilde{a}(\tilde{u}_{i_1,\ldots,i_d}, \tilde{v}) = f(\tilde{v}), \ \forall \tilde{v} \in \tilde{V}_{i_1,\ldots,i_d}$,

where

$$\tilde{V}_{i_1,\ldots,i_d} = V_{i_1,\ldots,i_d} \cap V(\tilde{\Omega}),$$

$$V_{i_1,\ldots,i_d} = \mathcal{P}_{i_1,\ldots,i_d}\tilde{V}_{i_1,\ldots,i_d},$$

$$\tilde{f}(\tilde{v}) = f(\mathcal{P}_{i_1,\ldots,i_d}\tilde{v}),$$

$$\tilde{a}(\tilde{u}, \tilde{v}) = a(\mathcal{P}_{i_1,\ldots,i_d}\tilde{u}, \mathcal{P}_{i_1,\ldots,i_d}\tilde{v}),$$

and

$$u_{i_1,\ldots,i_d} = \mathcal{P}_{i_1,\ldots,i_d}\tilde{u}_{i_1,\ldots,i_d}.$$

Actually, (3.2) is discretized on a uniform mesh, V is the space of finite element functions associated with the mesh (or the vector space associated with a finite difference discretization), and $V_{i_1,\ldots,i_d}$ are then subspaces of it. The problems (3.2) can be then written as discretized boundary value problems on the reduced domain $\tilde{\Omega}$.

To visualize the domain reduction process, consider the following example, which illustrates that the method is applicable to a wider range of problems than just elliptic problems with constant coefficients. Let

$$(3.3) \quad \begin{cases} \mathcal{L}u = -\sum_{i=1}^{d} (a_2(x)u_{x_i})_{x_i} - (a_1(x)u)_{x_i} + a_0(x)u = f \quad \text{in } \Omega = (-1,1)^d, \\ \\ u = 0 \quad \text{on } \partial\Omega, \end{cases}$$

where a_2 and a_0 are even functions about the axes, and a_1 is an odd function about the axes. For example,

$$a_2(x) = \prod_{j=1}^{d} \sin(\pi x_j), \qquad a_1(x) = \prod_{j=1}^{d} \cos(\pi x_j), \qquad \text{and} \qquad a_0(x) = \beta \in \mathbb{R}$$

are suitable choices.

By definition of $u_{i_1,\ldots,i_d}$, (3.2) can be written as

$$\mathcal{L}\tilde{u}_{i_1,\ldots,i_d} = \tilde{f}_{i_1,\ldots,i_d} \qquad in \ \tilde{\Omega}$$

with boundary conditions

$$(3.4) \quad \begin{cases} \tilde{u}_{i_1,\ldots,i_d} = 0 \quad \text{on the } d \text{ sides } x_1 = 1, \ldots, \text{ and } x_d = 1, \\[2mm] \tilde{u}_{i_1,\ldots,i_d} = 0 \quad \text{on side } x_{i_k} = 0 \text{ if } i_k = 1, \; k = 1,\ldots,d, \; . \\[2mm] \frac{\partial}{\partial n}\tilde{u}_{i_1,\ldots,i_d} = 0 \quad \text{on side } x_{i_k} = 0 \text{ if } i_k = 0, \; k = 1,\ldots,d. \end{cases}$$

Note that when (3.3) is discretized on a uniform mesh with a central difference scheme, 2^d systems of equations must be solved. By suitable choices of a_2, a_1, and a_0, the resulting systems can be nonsymmetric, indefinite.

Since the subspaces are orthogonal, Corollary 2 applies. Hence, the correct way to monitor the error over the entire domain is to monitor the error of each of the subproblems: the largest subproblem error is the largest error on the entire domain. Running times on both shared and distributed memory parallel computers (Encore and Intel) are contained in [8].

4. Many Way Reduction. In this section, a more subtle approach to domain reduction is described. A careful examination of (3.4) shows that a purely Dirichlet problem has been converted into a collection of similar subproblems with Dirichlet and Neumann boundaries.

$$(4.1) \quad \begin{cases} \mathcal{L}u = f \quad \text{in } \Omega \subset \mathbb{R}^d, \; d > 0, \\[2mm] u = 0 \quad \text{on } \partial\Omega_D, \quad \text{where } \partial\Omega = \partial\Omega_D \cup \partial\Omega_N, \\ \qquad\qquad\qquad\qquad\quad \text{and } \partial\Omega_D \cap \partial\Omega_N = \phi, \\[2mm] \frac{\partial}{\partial n}u = 0 \quad \text{on } \partial\Omega_N. \end{cases}$$

By folding the subproblem domains so that boundary conditions match, a higher degree of parallelism can be attained.

In the case of a square, an eight way domain reduction is possible. Starting from $\Omega = (-1,1)^2$, four subproblems are solved on subdomains $(0,1)^2$ with Dirichlet (D) and Neumann (N) boundary conditions given as follows.
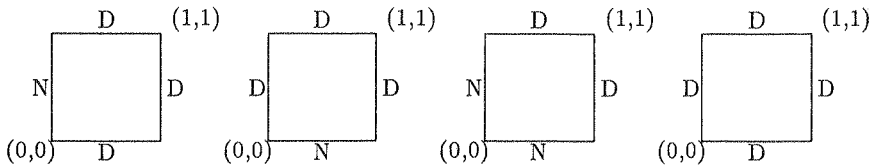
The subdomain with two Neumann boundary sides can be folded across the diagonal to produce two new subproblems on triangles.
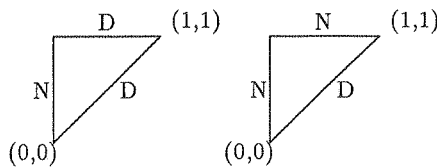
The subdomains with one Neumann boundary side can be folded across the middle of that boundary to produce two new subproblems on rectangles. For example,
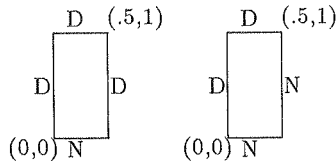
Finally, the subdomain which is isomorphic to the original domain can be reduced onto two triangles, two rectangles, or recursively onto an eight way set of subproblems. More details are contained in [3].

The three dimensional cube offers a real challenge. Starting from $\Omega = (-1,1)^3$, eight subproblems are solved on subdomains $(0,1)^3$ with Dirichlet (D) and Neumann (N) boundary conditions given as in (3.4).
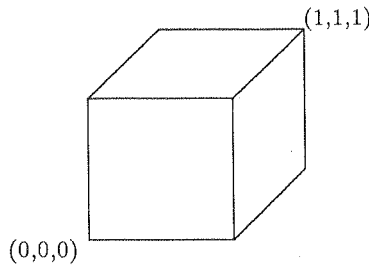
In this case, the subproblems have the following number of Neumann and Dirichlet Neumann boundary faces.

Number of subproblems	Number of faces Neumann	Dirichlet	Number of new subproblems
1	0	6	8
3	1	5	3×8
3	2	4	3×8
1	3	3	4 or 8
Total			60 or 64

The subdomain with no Neumann boundary face can be folded into the cube $(.5, 1)^3$ similarly to (3.4). The subdomains with exactly one Neumann boundary face can be folded into the following prism shaped domain.
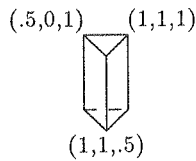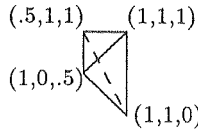

The subdomains with exactly one Neumann boundary face can be folded into the
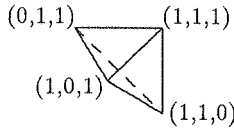
following wedge shaped domain.
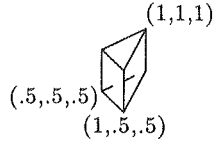
In each of these new subdomains, all of the angles are either 90° or 45°. The subdomain with exactly three Neumann boundary faces can be folded into the following tetrahedra.

This is not a conveniently shaped domain for finite difference discretizations. A better shaped one is as follows.


Unfortunately, this domain has twice the volume of the previous one.

Normally, three dimensional grid generated problems should never be solved directly since the cost is approximately $cN^{5/3}$, $c \in \mathbb{R}$. Using a 64 way domain reduction, the cost of solving each subproblem directly is at most

$$c(N/64)^{5/3} = \frac{c}{1024}N^{5/3}.$$

Using exactly 64 processors results in an elapsed time speedup of a factor of 1024. Switching to an iterative solver for each subproblem could be characterized as greedy.

5. Domain reduction and domain decomposition methods. In this section, the domain reduction algorithm is advocated as the inner part of a standard domain decomposition algorithm. The techniques of §§2–4 would be difficult to apply easily to a problem on the following domain.
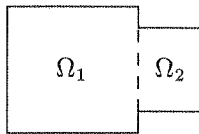
The obvious domain decomposition is the following.
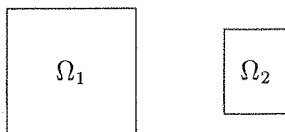
If possible, a domain reduction technique should be used to solve the problems that arise by using a standard domain decomposition method on each of Ω_1 and Ω_2 instead of a preconditioned conjugate direction method or multigrid. Note that the subproblems that naturally arise in the domain reduction method can be solved by parallel variants of preconditioned conjugate direction methods or multigrid.

6. Conclusions. The domain reduction method offers an interesting mathematical preconditioning technique for solving real problems. When it is directly applicable to a problem, it offers a way of producing a direct method which is embarrassingly parallel. When the cost of solving the subproblems by a direct method is prohibitive, iterative methods should be used, similar to domain decomposition methods. Finally, all steps of the algorithm are embarassingly parallel, making this ideal for implementation on existing parallel computers.

REFERENCES

[1] R. E. BANK AND C. C. DOUGLAS, *Sharp estimates for multigrid rates of convergence with general smoothing and acceleration*, SIAM J. Numer. Anal., 22 (1985), pp. 617–633.

[2] A. BRANDT, *Multi-level adaptive solution to boundary-value problems*, Math. Comp., 31 (1977), pp. 333–390.

[3] F. BREZZI, C. C. DOUGLAS, AND L. D. MARINI, *A parallel domain reduction method*, Numer. Meth. for PDE, 5 (1989), pp. 195–202.

[4] T. CHAN, R. GLOWINSKI, G. A. MEURANT, J. PÉRIAUX, AND O. WIDLUND, eds., *Domain Decomposition Methods for Partial Differential Equations II*, Philadelphia, 1989, Society for Industrial and Applied Mathematics.

[5] H. C. CHEN, *The SAS domain decomposition method for structural analysis*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1988.

[6] 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.

[7] C. C. DOUGLAS, *Multi-grid algorithms with applications to elliptic boundary-value problems*, SIAM J. Numer. Anal., 21 (1984), pp. 236–254.

[8] 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, S. F. McCormick, J. E. Dendy, C. Farhat, G. Lonsdale, S. V. Parter, J. W. Ruge, and K. Stüben, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1989, pp. 149–160.

[9] C. C. DOUGLAS AND W. L. MIRANKER, *Constructive interference in parallel algorithms*, SIAM J. Numer. Anal., 25 (1988), pp. 376–398.

[10] ———, *Some nontelescoping 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.

[11] C. C. DOUGLAS AND B. F. SMITH, *Using symmetries and antisymmetries to analyze a parallel multigrid algorithm*, SIAM J. Numer. Anal., 26 (1989), pp. 1439–1461.

[12] 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.

[13] R. GLOWINSKI, G. H. GOLUB, G. A. MEURANT, AND J. PÉRIAUX, eds., *On the Schwarz alternating method I*, Philadelphia, 1988, Society for Industrial and Applied Mathematics.

[14] W. HACKBUSCH, *Multigrid methods and applications*, Springer-Verlag, Berlin, 1985.

[15] ———, *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.

[16] P. L. LIONS, *On the Schwarz alternating method I*, in Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1988, pp. 1–42.

[17] ———, *On the Schwarz alternating method II*, in Domain Decomposition Methods for Partial Differential Equations II, T. Chan, R. Glowinski, G. A. Meurant, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1989, pp. 47–70.

[18] J. MANDEL AND S. F. McCORMICK, *Iterative solution of elliptic equations with refinement: the model multi-level case*, in Domain Decomposition Methods for Partial Differential Equations II, T. Chan, R. Glowinski, G. A. Meurant, J. Périaux, and O. Widlund, eds., Society for Industrial and Applied Mathematics, Philadelphia, 1989, pp. 93–102.

[19] S. F. McCORMICK, *Fast adaptive composite grid (FAC) methods*, in Defect correction methods, K. Böhmer and H. J. Stetter, eds., Springer-Verlag, Vienna, 1984, pp. 115–121.

[20] S. F. McCORMICK AND J. THOMAS, *The fast adaptive composite grid (FAC) method for elliptic equations*, Math. Comp., 46 (1986), pp. 439–456.

[21] H. A. SCHWARZ, *Über einige Abbildungsaufgaben*, Ges. Math. Abh., 11 (1869), pp. 65–83.