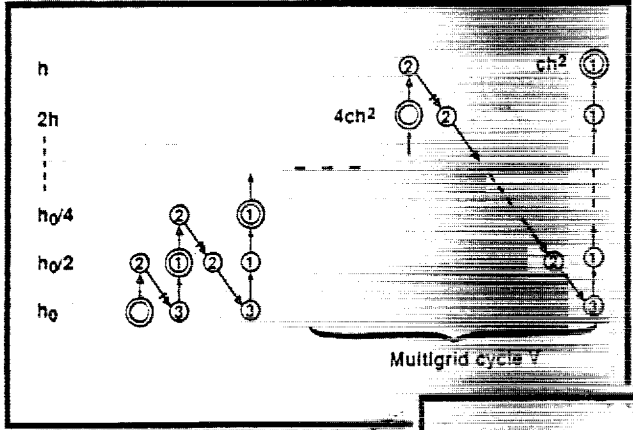
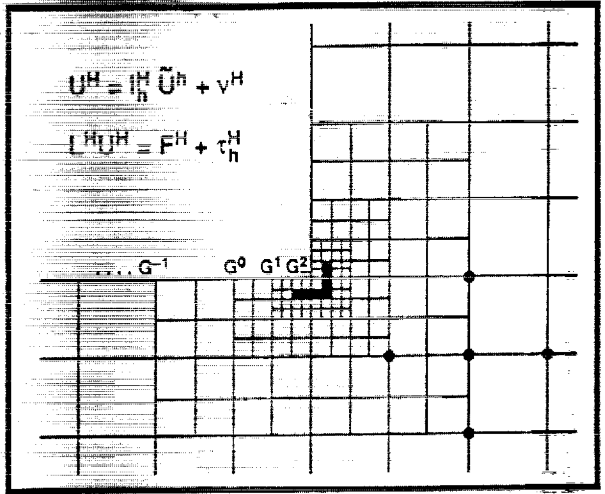
NASA Conference Publication 3224
Part 1

Sixth Copper Mountain Conference on Multigrid Methods





Proceedings of a workshop held at
Copper Mountain, Colorado
April 4-9, 1993

NASA

NASA Conference Publication 3224
Part 1

Sixth Copper Mountain Conference on Multigrid Methods

Edited by
N. Duane Melson
NASA Langley Research Center
Hampton, Virginia

T. A. Manteuffel and S. F. McCormick
University of Colorado
Denver, Colorado

Proceedings of a workshop cosponsored by the National Aeronautics and Space Administration, Washington, D.C., the Air Force Office of Scientific Research, Bolling Air Force Base, Washington, D.C., the Department of Energy, Washington, D.C., and the National Science Foundation, Washington, D.C., and held at Copper Mountain, Colorado
April 4–9, 1993

NASA
National Aeronautics and Space Administration
Office of Management
Scientific and Technical Information Program
1993

PREFACE

The Sixth Copper Mountain Conference on Multigrid Methods was held on April 4--9, 1993 at Copper Mountain Colorado and was cosponsored by NASA, the Air Force Office of Scientific Research, the Department of Energy, and the National Science Foundation. The University of Colorado at Denver, Front Range Scientific Computations, Inc., and the Society for Industrial and Applied Mathematics provided organizational support for the conference.

This document is a collection of many of the papers that were presented at the conference and thus represents the conference proceedings. NASA Langley graciously provided printing of this book so that all of the papers could be presented in a single forum. Each paper was reviewed by a member of the conference organizing committee under the coordination of the editors.

The multigrid discipline continues to expand and mature, as is evident from these proceedings. The vibrancy in this field is amply expressed in these important papers, and the collection clearly shows its rapid trend to further diversity and depth.

N. Duane Melson
NASA Langley Research Center

Steve F. McCormick and
Tom A. Manteuffel
University of Colorado at Denver

ORGANIZING COMMITTEE

CONTENTS

*Part 2 is presented under separate cover.

S1-64

197133

Part 1

N94-23674

A MULTIGRID SOLVER FOR THE SEMICONDUCTOR EQUATIONS

Bernhard Bachmann
Institut für Angewandte Mathematik der Universität Zürich,
Rämistr.74, 8001 Zürich, Switzerland
and
Asea Brown Boveri, Corporate Research,
5405 Baden-Dättwil, Switzerland.

SUMMARY

We present a multigrid solver for the exponential fitting method, applied to the current continuity equations of semiconductor device simulation in two dimensions. The exponential fitting method is based on a mixed finite element discretization using the lowest-order Raviart-Thomas triangular element. This discretization method yields a good approximation of front layers and guarantees current conservation. The corresponding stiffness matrix is an M-matrix. "Standard" multigrid solvers, however, cannot be applied to the resulting system, as this is dominated by an unsymmetric part, which is due to the presence of strong convection in part of the domain. To overcome this difficulty, we explore the connection between Raviart-Thomas mixed methods and the nonconforming Crouzeix-Raviart finite element discretization. In this way we can construct nonstandard prolongation and restriction operators using easily computable weighted L^2-projections based on suitable quadrature rules and the upwind effects of the discretization. The resulting multigrid algorithm shows very good results, even for real-world problems and for locally refined grids.

1. INTRODUCTION

The exponential fitting method applied to the current continuity equations is based on a mixed finite element discretization using the lowest-order Raviart-Thomas triangular element [1]. This discretization yields a good approximation of front layers and guarantees current conservation. The corresponding scheme results in a large sparse system of equations, which is dominated by an unsymmetric part. When applying multigrid algorithms to the resulting system (7), the most difficult part is the construction of suitable prolongation and restriction operators. Using the connection between Raviart-Thomas mixed methods and the nonconforming Crouzeix-Raviart finite element discretization, we overcome this difficulty.

In § 2 we give some results from [2] concerning the mixed finite element discretization. We determine the resulting system and show the interrelation with a nonconforming finite element method. § 3 deals with the solution of the system of linear equations by our multigrid solver. First we construct easily computable L^2−projections, based on suitable quadrature rules and the upwind effect of the discretization. Due to the presence of strong convection in part of the domain it is also necessary to consider special smoothers for the multigrid algorithm. We use a minimal residual method with ILU preconditioning. The results of the numerical tests are given in § 4.

2. THE EXPONENTIAL FITTING METHOD FOR CURRENT CONTINUITY EQUATIONS

2.1. Mixed Finite Element Method

Let $\Omega \subset \mathbb{R}^2$ be a connected, bounded and polygonal domain. $H^m(\Omega)$, for $m \in \mathbb{N}$, and $L^2(\Omega) := H^0(\Omega)$ denote the usual Sobolev and Lebesque spaces equipped with the norm

$$\|u\|_m := \{ \sum_{|\alpha| \le m} \int_\Omega |D^\alpha u|^2 \}^{\frac{1}{2}}.$$

For $f \in L^2(\Omega)$ and $g \in L^2(\Gamma_0)$, $\Gamma_0 \subset \partial\Omega$ closed with positive length, we consider the current continuity equation, as given in [3]:

Find $u \in H^1(\Omega)$ such that

$$\begin{cases} \text{div (grad } u + u \text{ grad } \psi) = f & \text{in } \Omega \subset \mathbb{R}^2, \\ u = g & \text{on } \Gamma_0 \subset \partial\Omega, \\ \dfrac{\partial u}{\partial n} + u \dfrac{\partial \psi}{\partial n} = 0 & \text{on } \Gamma_1 = \partial\Omega \setminus \Gamma_0. \end{cases} \tag{1}$$

The current is defined by $J = \text{grad } u + u \text{ grad } \psi$. Here, $\psi \in H^1(\Omega)$ is a given bounded function. To discretize problem (1) we introduce the classical method of changing variables from u to the socalled Slotboom variable ρ [3]

$$\rho = e^\psi \, u.$$

This results in the following symmetric form of problem (1):

Find $\rho \in H^1(\Omega)$ such that

$$\begin{cases} \text{div } (e^{-\psi} \text{grad } \rho) = f & \text{in } \Omega \subset \mathbb{R}^2, \\ \rho = \chi := e^\psi g & \text{on } \Gamma_0 \subset \partial\Omega, \\ \dfrac{\partial \rho}{\partial n} = 0 & \text{on } \Gamma_1 = \partial\Omega \setminus \Gamma_0. \end{cases} \tag{2}$$

Let $\{\mathcal{T}_k\}_{k \ge 0}$ be a regular sequence of decompositions of Ω into triangles. Denote by h_k the longest side of all triangles $T \in \mathcal{T}_k$. The set of edges of $\mathcal{T}_k$ is denoted by $\mathcal{E}_k$, where $\mathcal{E}_k^\partial$ are the boundary edges and $\mathcal{E}_k^0 = \mathcal{E}_k \setminus \mathcal{E}_k^\partial$ are all interelement boundaries. Denote by m_e the midpoint of an edge e of $\mathcal{E}_k$. Moreover, let P_m, $m \ge 0$, be the space of all polynomials of degree not greater than m. Following [1], we use the lowest order Raviart-Thomas mixed finite element to discretize (2). Therefore we define the following set of polynomial vectors

$$RT(T) := \{\tau = (\tau_1, \tau_2) : \tau_1 = \alpha + \beta x, \tau_2 = \gamma + \beta y, \ \alpha, \ \beta, \ \gamma \in \mathbb{R}\}, \quad \forall T \in \mathcal{T}_k,$$

and set

$$V_k := \{\tau \in (L^2(\Omega))^2 : \text{div } \tau \in L^2(\Omega), \ \tau \mathbf{n} = 0 \text{ on } \Gamma_1, \ \tau|_T \in RT(T) \ \forall T \in \mathcal{T}_k\},$$
$$W_k := \{\varphi \in L^2(\Omega) : \varphi|_T \in P_0(T) \ \forall T \in \mathcal{T}_k\}.$$

Then the mixed finite element discretization of (2) is defined as:

Find $(\mathbf{J_k}, \rho_k) \in V_k \times W_k$ *such that for all* $(\tau_\mathbf{k}, \varphi_k) \in V_k \times W_k$

$$\begin{cases} \int_\Omega e^\psi \mathbf{J_k} \tau_\mathbf{k} dx + \int_\Omega \rho_k \mathrm{div}\ \tau_\mathbf{k} dx = \int_{\Gamma_0} \chi \tau_\mathbf{k} \mathbf{n} ds, \\ \int_\Omega \varphi_k \mathrm{div}\ \mathbf{J_k} dx \qquad\qquad = \int_\Omega f \varphi_k dx. \end{cases} \tag{3}$$

The matrix associated with (3) is not coercive. To avoid this inconvenience we introduce a Lagrange multiplier. We define

$$\tilde{V}_k := \{\tau \in (L^2(\Omega))^2 : \tau|_T \in RT(T)\ \forall T \in \mathcal{T}_k\},$$

and for $\xi \in L^2(\Gamma_0)$

$$\Lambda_{k,\xi} := \{\mu : \mu \in L^2(\mathcal{E}_k),\ \mu|_e \in P_0(e)\ \forall e \in \mathcal{E}_k,\ \int_e (\mu - \xi) ds = 0\ \forall e \subset \Gamma_0\}.$$

Instead of (3) we now consider the mixed equilibrium discretization,

Find $(\tilde{\mathbf{J}}_\mathbf{k}, \tilde{\rho}_k, \lambda_k) \in \tilde{V}_k \times W_k \times \Lambda_{k,\chi}$ *such that for all* $(\tau_\mathbf{k}, \varphi_k, \mu_k) \in \tilde{V}_k \times W_k \times \Lambda_{k,0}$

$$\begin{cases} \int_\Omega e^\psi \tilde{\mathbf{J}}_\mathbf{k} \tau_\mathbf{k} dx + \sum_{T \in \mathcal{T}_k} \int_T \tilde{\rho}_k \mathrm{div}\ \tau_\mathbf{k} dx - \sum_{T \in \mathcal{T}_k} \int_{\partial T} \lambda_k \tau_\mathbf{k} \mathbf{n} ds = 0, \\ \sum_{T \in \mathcal{T}_k} \int_T \varphi_k \mathrm{div}\ \tilde{\mathbf{J}}_\mathbf{k} dx \qquad\qquad = \int_\Omega f \varphi_k dx, \\ \sum_{T \in \mathcal{T}_k} \int_{\partial T} \mu_k \tilde{\mathbf{J}}_\mathbf{k} \mathbf{n} ds \qquad\qquad = 0. \end{cases} \tag{4}$$

As shown in [3], problem (4) has a unique solution and $\mathbf{J_k} \equiv \tilde{\mathbf{J}}_\mathbf{k},\ \rho_k \equiv \tilde{\rho}_k$ holds. Moreover, λ_k is a good approximation of the solution of (2) at the interelement boundaries [2]. It is possible to eliminate the unknowns, corresponding to $\tilde{\mathbf{J}}_\mathbf{k}$ and $\tilde{\rho}_k$ in the resulting system, by static condensation [3]. This yields a matrix (acting only on the interelement multiplier λ_k), which is a symmetric positive definite matrix and which is an M-matrix if the triangulation is of the weakly acute type (i.e. no angle $> \frac{\pi}{2}$).

2.2. The Nonconforming Finite Element Formulation

To introduce the nonconforming finite element formulation we need the following definitions: Let Π_k^0 be the L^2–projection from $L^2(\mathcal{E}_k)$ onto

$$\Lambda_k := \{\mu_k \in L^2(\mathcal{E}_k) : \mu_k|_e \in P_0(e)\ \forall e \in \mathcal{E}_k\}$$

and P_k^0 be the L^2–projection from $L^2(\Omega)$ onto

$$\tilde{S}_k := \{v_k \in L^2(\Omega) : v_k|_T \in P_0(T)\ \forall T \in \mathcal{T}_k\},$$

$$\text{i.e.} \quad \Pi_k^0(\xi)|_e = \frac{1}{|e|}\int_e \xi ds, \quad \forall e \in \mathcal{E}_k \quad \text{and} \quad P_k^0(u)|_T = \frac{1}{|T|}\int_T u dx, \quad \forall T \in \mathcal{T}_k.$$

The Crouzeix-Raviart finite element space [4] is defined by

$$S_k := \{v_k \in L^2(\Omega) : v_k|_T \in P_1(T)\ \forall T \in \mathcal{T}_k,\ v_k \text{ is continuous at midpoints of edges}\}.$$

For $\xi \in L^2(\Gamma_0)$ we define

$$S_{k,\xi} := \{v_k \in S_k : v_k(m_e) = \Pi_k^0(\xi)|_e,\ e \subset \Gamma_0\}.$$

Notice that the standard basis functions of S_k are equal to one at the midpoint of exactly one edge and vanish at the midpoints of all other edges. Using the arguments concerning static condensation in [5], it is straightforward to prove the following lemma.

Lemma 2.1.

The solution λ_k of (4) can be written as $\lambda_k = \Pi_k^0(w_k)$, where w_k is the solution of the following nonconforming weak problem

Find $w_k \in S_{k,\chi}$ such that for all $v_k \in S_{k,0}$

$$\sum_{T\in\mathcal{T}_k} (P_k^0(e^{\psi}))^{-1} \int_T \text{grad } w_k \text{ grad } v_k dx = \sum_{T\in\mathcal{T}_k} P_k^0(f) \int_T (\frac{3}{2} - \frac{1}{2}\frac{e^{\psi}}{P_k^0(e^{\psi})}) v_k dx. \tag{5}$$

◇

Remark 2.2.

For w_k as in Lemma 2.1. and the solution ρ of (2) the following error estimate [2] holds:

$$\|\rho - w_k\|_0 \leq \gamma|h_k|^2(\|\rho\|_3 + \|\mathbf{J}\|_2)$$

with $\gamma = \gamma(e^{\psi})$ independent of ρ and h_k.

◇

The Lagrange multiplier λ_k is an approximation of $\rho = e^\psi\ u$. In semiconductor simulations the range of ψ is very large, so that λ_k is not suited for actual computations. Moreover we are interested in approximating the solution u of (1). Hence we introduce the following change of variable

$$\mu_k = (\Pi_k^0(e^\psi))^{-1}\ \lambda_k\ \in \Lambda_{k,(\Pi_k^0(e^\psi))^{-1}\chi}. \tag{6}$$

Denote the standard basis of S_k by $\varphi_e,\ e \in \mathcal{E}_k$. We define the linear operator $E_k : S_k \to S_k$ by

$$E_k(\varphi_e) = \Pi_k^0(e^\psi)|_e\varphi_e \quad \forall e \in \mathcal{E}_k.$$

For $f \in L^2(\Omega),\ G_k(f) \in L^2(\Omega)$ is defined by

$$G_k(f) = P_k^0(f)(\frac{3}{2} - \frac{1}{2}\frac{e^\psi}{P_k^0(e^\psi)}).$$

Finally we arrive at the following statement:

Lemma 2.3.

Let $\zeta = (\Pi_k^0(e^\psi))^{-1}\chi \in L^2(\Gamma_0)$. Then μ_k of (6) can be written as $\mu_k = \Pi_k^0(u_k)$, where u_k is the solution of the nonconforming weak problem:

Find $u_k \in S_{k,\zeta}$ such that for all $v_k \in S_{k,0}$

$$\sum_{T\in\mathcal{T}_k} (P_k^0(e^\psi))^{-1} \int_T \text{grad}\ E_k(u_k)\ \text{grad}\ v_k dx = \int_\Omega G_k(f)\ v_k dx. \tag{7}$$

◇

Remark 2.4.

Note that problem (7) is the usual nonconforming Crouzeix-Raviart discretization of the Laplace equation, if ψ and f are constant on Ω.

◇

We can use the error estimate of Remark 2.2. to obtain an estimate for the approximation error between the solution u_k from (7) and the solution u of (1), though the result is rather unsatisfying. To arrive at an improved error bound, one could use the fact that two Babuška-Brezzi conditions hold [6] for the corresponding bilinear form. The stability and the unique solvability of the discrete problem (7) also follow. In the following we construct a multigrid algorithm for problem (7). Therefore we define the bilinear form a_k on S_k by

$$a_k(u_k, v_k) := \sum_{T\in\mathcal{T}_k} (P_k^0(e^\psi))^{-1} \int_T \text{grad}\ E_k(u_k)\ \text{grad}\ v_k\ dx.$$

3. MULTIGRID METHOD

3.1. Adaptive Mesh-Refinement Techniques

In order to formulate the multigrid algorithm, we need a regular sequence of triangulations $\{\mathcal{T}_k\}_{k\geq 0}$. In our refinement process, two objectives are pursued. First, in order to improve approximation, we should refine the grid locally, where the solution behaves very badly. Second, we have to construct weakly acute triangulations to guarantee that the corresponding discretization matrix is an M-matrix. Therefore we define the strategy and rules below. Given a triangulation we refine its triangles as follows:

(1) The refinement process is started by a suitable error estimator, e.g. based on residuals, which marks some of the triangles as red.

(2) If a triangle is marked

  (i) *red*, it will be cut into four new ones by joining the midpoints of its edges,

  (ii) *green*, it will be cut into two new triangles by joining the midpoint of the longest edge to the vertex opposite to this edge, and

  (iii) *blue*, it will be cut into three new triangles by joining the midpoint of its longest edge to the vertex opposite to this edge and to the midpoint of one of the remaining edges (see Fig.1)

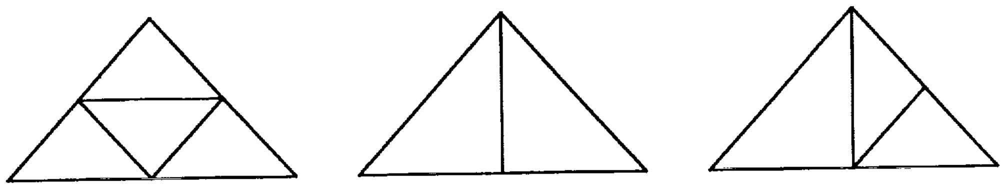


Figure 1. Red, green and blue refinement of a triangle.

(3) Hanging nodes are avoided using the following rules:

  (i) a triangle with three hanging nodes is marked *red*

  (ii) a triangle with two hanging nodes is marked *blue*, if one of the nodes lies on the longest edge of the triangle; otherwise it is marked *red*

  (iii) a triangle with one hanging node is marked *green*, if the node lies on the longest edge of the triangle; otherwise it is marked *blue*

Note that rules (ii) and (iii) may introduce new hanging nodes. However, one can prove that the refinement process obeying the above rules is finite. Moreover, assuming that $\mathcal{T}_0$ has only isosceles right-angled triangles, then it is guaranteed that all triangulations $\mathcal{T}_k$ are weakly acute.

3.2. The Prolongation

In order to solve problem (1), we have to find the solution u_k of the discrete problem (7). Since the Crouzeix-Raviart element is nonconforming and $S_{k-1} \not\subset S_k$, we must construct a suitable transfer operator between S_{k-1} and S_k. In addition, the discretization shows upwind effects due to the existence of strong convection in part of the domain. This also must be taken into account.

In [7, 8] a hierarchical basis multigrid method was used to solve a linear system arising from the convection diffusion equation by an upwind discretization. It was shown that the convergence of the hierarchical basis multigrid method depends on the strength of the convection term. When solving the discrete problem (7) with the multigrid algorithm [9], a similar effect can be seen in the numerical experiments. On the other hand, considering the one dimensional problem, one sees that a good interpolation has to regard the upwind effect. Therefore we introduce the following weighted L^2–projection. Define

$$(u,v)_k := \sum_{\tilde{T}\in\mathcal{T}_k} (P_k^0(e^{\psi})|_{\tilde{T}})^{-1} \sum_{\substack{T\in\mathcal{T}_{k+1}\\ T\subset\tilde{T}}} \int_T E_k(u)\, v dx \quad \forall\, u\in S_k,\ v\in S_k\cup S_{k+1}. \tag{8}$$

For all $u \in P_2(T)$, $T \in \mathcal{T}_k$, the quadrature rule

$$\int_T u dx = \frac{|T|}{3} \sum_{e\subset\partial T} u(m_e)$$

is exact, so that (8) can be written as

$$(u,v)_k = \sum_{\tilde{T}\in\mathcal{T}_k} (P_k^0(e^{\psi})|_{\tilde{T}})^{-1} \sum_{\substack{T\in\mathcal{T}_{k+1}\\ T\subset\tilde{T}}} \frac{|T|}{3} \sum_{e\subset\partial T} E_k(u)(m_e)\ v(m_e) \tag{9}$$

for all $u \in S_k$ and $v \in S_k \cup S_{k+1}$.

Remark 3.1.

Note that if $v \in S_k$ holds, (9) reduces to the equation

$$(u,v)_k = \sum_{\tilde{T}\in\mathcal{T}_k} (P_k^0(e^{\psi})|_{\tilde{T}})^{-1} \frac{|\tilde{T}|}{3} \sum_{e\subset\partial\tilde{T}} \Pi_k^0(e^{\psi})|_e\ u(m_e)\ v(m_e).$$

Moreover, if ψ is constant, we have

$$(u,v)_k = (u,v) \quad \forall\, u \in S_k,\ v \in S_k \cup S_{k+1},$$

where $(u,v) := \int_\Omega uv\ dx$ denotes the usual L^2–inner product.

◇

From (9) it follows that the standard basis functions of S_k are mutually orthogonal with respect to the inner product $(.,.)_k$. Therefore we can obtain an easily computable prolongation operator $P^k_{k-1} : S_{k-1} \to S_k$ by

$$(P^k_{k-1}u_{k-1}, v_k)_k = (u_{k-1}, v_k)_{k-1} \quad \forall\, u_{k-1} \in S_{k-1},\ v_k \in S_k.$$

It is straightforward to prove the following lemma:

Lemma 3.2.

Let $u_{k-1} \in S_{k-1}$, then:

If $e \in \mathcal{E}^{\partial}_k$ then $(P^k_{k-1}u_{k-1})(m_e) = \left(\frac{\Pi^0_k(e^\psi)|_e}{P^0_k(e^\psi)|_T}\right)^{-1} \frac{(E_{k-1}u_{k-1})(m_e)|_{\tilde{T}}}{P^0_{k-1}(e^\psi)|_{\tilde{T}}}$,
where T (resp. $\tilde{T}$) is the triangle in $\mathcal{T}_k$ (resp. $\mathcal{T}_{k-1}$) with $e \subset \partial T$ (resp. $e \subset \partial\tilde{T}$).

If $e \in \mathcal{E}^0_k$ then

$$(P^k_{k-1}u_{k-1})(m_e) = \left(|T^L|\frac{\Pi^0_k(e^\psi)|_e}{P^0_k(e^\psi)|_{T^L}} + |T^R|\frac{\Pi^0_k(e^\psi)|_e}{P^0_k(e^\psi)|_{T^R}}\right)^{-1}$$
$$\left(|T^L|\frac{(E_{k-1}u_{k-1})(m_e)|_{\tilde{T}^L}}{P^0_{k-1}(e^\psi)|_{\tilde{T}^L}} + |T^R|\frac{(E_{k-1}u_{k-1})(m_e)|_{\tilde{T}^R}}{P^0_{k-1}(e^\psi)|_{\tilde{T}^R}}\right),$$

where T^L, T^R (resp. $\tilde{T}^L, \tilde{T}^R$) are the two triangles in $\mathcal{T}_k$ (resp. $\mathcal{T}_{k-1}$) with $T^L \cap T^R = e$ and $T^L \subset \tilde{T}^L$, $T^R \subset \tilde{T}^R$. (see Fig.2)

$\diamond$

Remark 3.3.

If ψ of Lemma 3.2 is constant, we have the usual L^2−projection I^k_{k-1} as given in [9]. The coefficients $\frac{\Pi^0_k(e^\psi)|_e}{P^0_k(e^\psi)|_T}$, $k \geq 0$, are also computed during the construction of the stiffness matrix, hence the interpolation is not very expensive. On the other hand, as shown in [5] the coefficients $\frac{\Pi^0_k(e^\psi)|_e}{P^0_k(e^\psi)|_T}$, $k \geq 0$, introduce an upwind effect; i.e. the coefficient corresponding to the downwind node is equal to zero.

$\diamond$

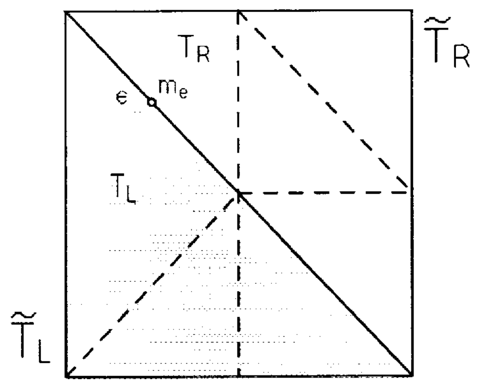


Figure 2. Interpolation.

3.3. The Smoother

A suitable smoother for the system (7) is given in [10] by a Gauss-Seidel-iteration with decoupling. This smoother is confined to special triangulations and does not allow adaptive grid refinements. Another candidate for problems with strong convection terms is the ILU-iteration. Here we restrict ourselves to a variant of the ILU-iteration. The ILU-decomposition of the linear system A_k, related to problem (7) and the standard basis of the Crouzeix-Raviart finite element space $S_{k,\zeta}$, can be written as

$$A_k = L_k U_k - D_k,$$

where L_k, U_k and D_k are given by the sparsity pattern of A_k. Denote by $\alpha_k = (\alpha_e)_{e\in\mathcal{E}_k}$ the coefficient vector of $u_k = \sum_{e\in\mathcal{E}_k} \alpha_e \varphi_e \ \in S_{k,\zeta}$ and by b_k the right hand side. Then the ILU-iteration is given by:

$$\begin{aligned} &\alpha_k^0 \text{ an arbitrary starting vector, } w \in (0,1], \\ &\alpha_k^i = \alpha_k^{i-1} + \omega(L_k U_k)^{-1}(b_k - A_k \alpha_k^{i-1}), \quad \forall i = 1, \cdots. \end{aligned}$$

In order to get a good smoothing rate, we must optimize the factor

$$\frac{\|A_k(\alpha_k^i - \alpha_k^*)\|_2}{\|A_k(\alpha_k^0 - \alpha_k^*)\|_2},$$

as mentioned in [11]. Here α_k^* is the solution of $A_k \alpha_k = b_k$. Therefore, by computing the optimal damping parameter ω in every step, our final smoothing algorithm is

Algorithm 3.4.

```
$\alpha_k^0$ an arbitrary starting vector, $r_k^0 = b_k - A_k\alpha_k^0$,
for $i = 1, \cdots$, compute:
    $d_k^{i-1} = (L_kU_k)^{-1}r_k^{i-1}$,
    $v_k^{i-1} = A_kd_k^{i-1}$,
    $\omega^{i-1} = \frac{v_k^{i-1^T} r_k^{i-1}}{v_k^{i-1^T} v_k^{i-1}}$,
    $\alpha_k^i = \alpha_k^{i-1} + \omega^{i-1}d_k^{i-1}$,
    $r_k^i = r_k^{i-1} - \omega^{i-1}v_k^{i-1}$,
end.
```

◇

Remark 3.5.

Algorithm 3.4. can be interpreted as a minimal residual method with ILU-preconditioning.

◇

3.4. Multigrid Algorithm

Now we are in the position to formulate our multigrid algorithm.

Algorithm 3.6. (One MG-iteration at level k)

(1) *Pre-smoothing:* Given $u_k^0 = \sum_{e\in\mathcal{E}_k} \alpha_e^0\varphi_e \in S_{k,\zeta}$. For $i = 1, \cdots, \nu_1$ compute u_k^i, using Algorithm 3.4.

(2) *Coarse-Grid Correction:* Denote by $u_{k-1}^* \in S_{k-1,0}$ the solution of the coarse grid problem

$$a_{k-1}(u_{k-1}, v_{k-1}) = (G_k(f), I_{k-1}^k v_{k-1}) - a_k(u_k^{\nu_1}, I_{k-1}^k v_{k-1}) \quad \forall v_{k-1} \in S_{k-1,0}. \qquad (*)$$

If $k = 1$, set $\tilde{u}_{k-1} = u_{k-1}^*$. If $k > 1$, compute an approximation $\tilde{u}_{k-1}$ to u_{k-1}^* by applying $\mu = 1$ or $\mu = 2$ iterations of the algorithm at level $k-1$ to problem $(*)$ and starting value 0. Set

$$u_k^{\nu_1+1} := u_k^{\nu_1} + P_{k-1}^k\tilde{u}_{k-1}.$$

(3) *Post-smoothing:* Apply ν_2 iterations of Algorithm 3.4. to $u_k^{\nu_1+1}$.

◇

Remark 3.7.

So far, there exists no convergence proof for Algorithm 3.6. The standard convergence analysis, as in [9, 11], cannot be used here, because the bilinear form $a_k(.,.)$ is unsymmetric.

◇

4. NUMERICAL RESULTS

In this section we present three numerical examples which demonstrate the behaviour of the proposed multigrid method. In all experiments we measure the performance of a method by the arithmetic mean of the convergence rates

$$\rho_i = \sqrt[i]{\frac{r_k^{i^T} r_k^i}{r_k^{0^T} r_k^0}},$$

where r_k^i is the defect of the i-th iteration.

The first model problem is taken from the papers of Brezzi, Marini and Pietra [3, 5]. We consider the domain $\Omega := (0,1) \times (0,1)$ with Neumann boundary

$$\Gamma_1 := \{(x,y) : \big((x=1) \wedge (y<0.75)\big) \vee \big((y=1) \wedge (x<0.75)\big)\}$$

and Dirichlet boundary $\Gamma_0 := \partial\Omega \setminus \Gamma_1$, right hand side $f \equiv 0$ and potential ψ defined as $\psi(x,y) := \dfrac{\psi_0(x,y)}{l}$ with

$$\psi_0(x,y) := \begin{cases} 0.0 & \text{if} \quad 0.0 \le r \le 0.8 \\ r - 0.8 & \text{if} \quad 0.8 \le r \le 0.9 \\ 0.1 & \text{if} \quad 0.9 \le r \end{cases} \quad \text{with } r := \sqrt{(x-1)^2 + (y-1)^2}.$$

On Γ_0 we have $g(x,y) = 0$ if $x = 0$ or $y = 0$ and $g(x,y) = 1$ otherwise. We use the initial triangulation $\mathcal{T}_0$ as given in Fig.3. and refine every triangulation by marking all triangles as red (uniform refinement). The numerical solution for $l = 10^6$ and a locally refined grid is shown in Fig.4.

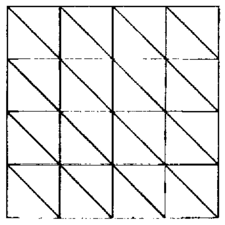


Figure 3. Initial triangulation 1.

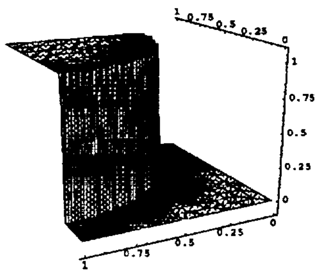


Figure 4. Numerical Solution.

We test our multigrid algorithm 3.6. with two pre- and two post-smoothing steps ($\nu_1 = \nu_2 = 2$) and with different values of μ (only smoothing : $\mu = 0$; V-cycle : $\mu = 1$; W-cycle : $\mu = 2$) for problems with varying $k_{\max}$ ($k_{\max} = 1, \cdots, 5$). The corresponding convergence rates for $l = 10$ and $l = 10^6$ are given in Tab.1 and Tab.2 respectively. In all experiments we used the same arbitrary starting vector.

$k_{\max}$	1	2	3	4	5
$\mu = 0$	.672	.854	.886	.904	.910
$\mu = 1$	.032	.103	.159	.208	.253
$\mu = 2$	.032	.074	.059	.059	.055

Table 1. Convergence rates ($l = 10$)

$k_{\max}$	1	2	3	4	5
$\mu = 0$	.736	.879	.890	.906	.910
$\mu = 1$	.096	.245	.358	.427	.482
$\mu = 2$	.096	.221	.266	.235	.201

Table 2. Convergence rates ($l = 10^6$)

In Tab.3 we show the results for $k_{\max} = 5$ and with varying μ and $l = 10^m$.

m	0	1	2	3	4	5	6
$\mu = 0$	.907	.910	.908	.906	.908	.910	.908
$\mu = 1$	.227	.253	.444	.473	.483	.482	.482
$\mu = 2$	.053	.055	.075	.174	.198	.201	.201

Table 3. Convergence rates ($k_{\max} = 5$)

In the second experiment we take

$$f = -2\ 10^5 \frac{e^{-\psi(x,y)}}{(\cosh(100\ (r - 0.65)))^2},$$

with $\psi(x,y) = 10^3\ (1 + \tanh(100\ (r - 0.65)))$ and $r = \sqrt{(x-1)^2 + (y-1)^2}$. Again we chose $\Omega = (0,1) \times (0,1)$. The Dirichlet boundary $\Gamma_0 = \partial\Omega$ and $g(x,y) = (x+y)\ e^{-\psi(x,y)}$. The exact solution is given by

$$u(x,y) = (x+y)\ e^{-\psi(x,y)}.$$

The numerical solution is shown in Fig.7. We used three different coarse grids, as given in Fig.3, Fig.5 and Fig.6, to show that the Algorithm 3.6. does not depend on the orientation of the grid. For uniform refinement and $k_{\max} = 5$ Tab.4 shows the results with varying μ ($\mu = 0, 1, 2$). In Tab.4 we also show the results for $k_{\max} = 6$ and adaptive refinement of the grid (see Fig.8).

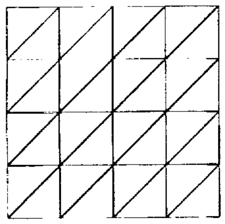


Figure 5. Initial triangulation 2.

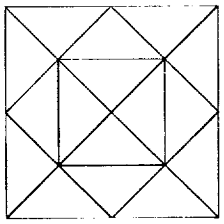


Figure 6. Initial triangulation 3.

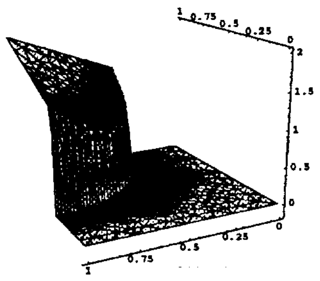


Figure 7. Numerical solution.

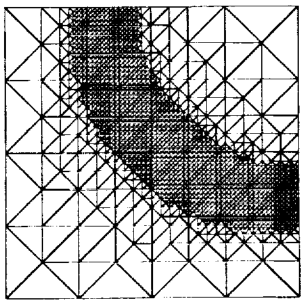


Figure 8. Adaptive refined grid ($k = 4$).

grid	init. triang. 1	init. triang. 2	init. triang. 3	loc. ref.
$\mu = 0$	.900	.903	.891	.905
$\mu = 1$	.311	.225	.216	.409
$\mu = 2$	.157	.087	.128	.355

Table 4. Convergence rates

Finally we consider an experiment with a real-world problem. Fig.9 shows the schematic structure of the doping of a thyristor. With an existing simulation program (ABBPISCES) we computed the solution u of (1) and the potential ψ of the coupled stationary semiconductor equations for a blocking-state (see Fig.11 resp. Fig.12) and an on-state of the thyristor (see Fig.13 resp. Fig.14). The so computed potential ψ was substituted into equation (1) and the resulting system was solved with our multigrid algorithm. Fig.10 shows the grid for an adaptive refinement ($k = 5$). Finally Tab.5 shows the convergence rates for Algorithm 3.6. with a suitable number of pre- and post-smoothing steps, with varying μ ($\mu = 0, 1, 2$) and $k_{\max} = 7$.

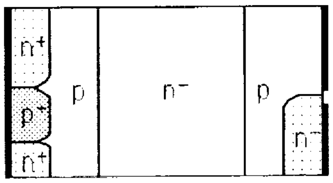


Figure 9. Schematic structure of the doping.

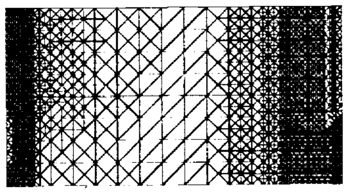


Figure 10. Adaptive grid ($k = 5$).

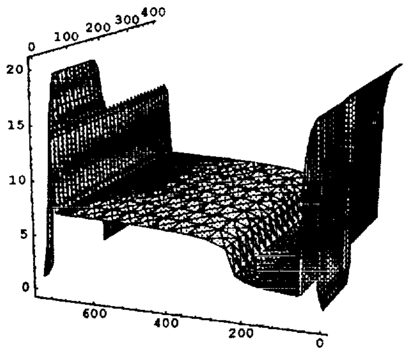


Figure 11. Solution (log).

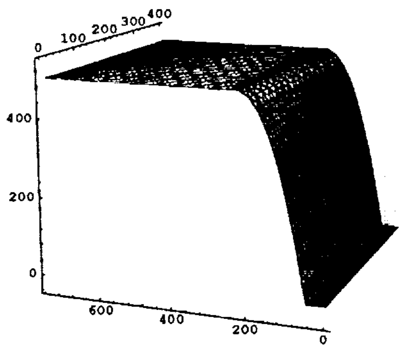


Figure 12. Potential.

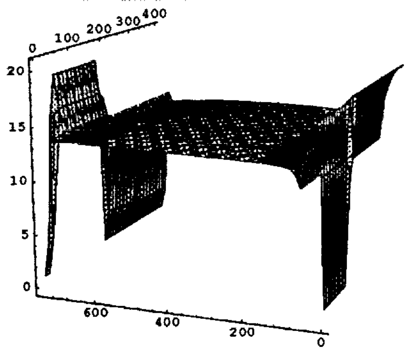


Figure 13. Solution (log).

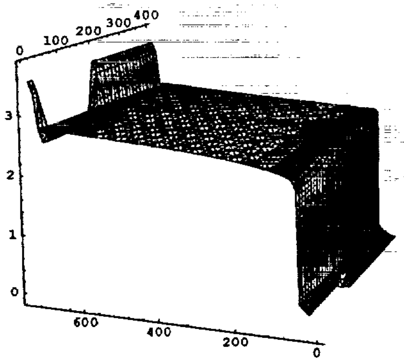


Figure 14. Potential.

state	blocking	on
$\mu = 0$	.843	.828
$\mu = 1$ $(\nu_1 = \nu_2 = 22)$	.249	.112
$\mu = 2$ $(\nu_1 = \nu_2 = 9)$	.108	.121

Table 5. Convergence rates

5. REFERENCES

1. Raviart, P.-A.; and Thomas, J.M.: A mixed finite element method for second order elliptic problems, *Mathematical Aspects of the Finite Element Method,* Lecture Notes in Mathematics, 606, Springer, Berlin, 1977, pp. 292-315.

2. Arnold, D.N.; and Brezzi, F.: Mixed and nonconforming finite element methods, postprocessing and error estimates, *RAIRO* M^2AN, vol.19, 1985, pp. 7-32.

3. Brezzi, F.; Marini, L.D.; and Pietra, P.: Two-dimensional exponential fitting and application to drift-diffusion models, *SIAM J. Numer. Anal.,* vol.26, 1989, pp. 1342-1355.

4. Crouzeix, M.; and Raviart, P.A.: Conforming and non-conforming finite element methods for solving the stationary Stokes equation, *RAIRO Anal. Numér.,* vol.7, 1973, pp. 33-76.

5. Brezzi, F.; Marini, L.D.; and Pietra, P.: Numerical simulation of semiconductor devices, *Comp. Meths. Appl. Mech. Engr.,* vol.75, 1989, pp. 493-513.

6. Bachmann, B.: *Adaptive Mehrgitterverfahren zur Lösung der stationären Halbleitergleichungen,* Dissertation, Universität Zürich, 1993.

7. Bank, R.E.; and Benbourenane, M.: A Fourier analysis of the two-level hierarchical basis multigrid method for convection-diffusion equations, *Proceedings of the Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations,* SIAM, Philadelphia, 1991, pp. 178-184.

8. Bank, R.E.; and Benbourenane, M.: The hierarchical basis multigrid method for convection-diffusion equations, *Numer. Math.,* vol.61, 1992, pp. 7-37.

9. Braess, D.; and Verfürth, R.: Multi-grid methods for non-conforming finite element methods, *SIAM J. Numer. Anal.,* vol.27, 1990, pp. 979-986.

10. Reusken, A.: Multigrid applied to mixed finite element schemes for current continuity equations, University Eindhoven, RANA 90-13, Nov. 1990.

11. Hackbusch, W.: *Multi-Grid Methods and Applications,* Springer, Berlin-Heidelberg, 1985.

S2-34
197134
P-13

N94-23675

FAS Multigrid Calculations of Three Dimensional Flow Using Non-staggered Grids

D. Matović[1], A. Pollard,
H. A. Becker and E. W. Grandmaison

*Centre for Advanced Gas Combustion Technology,
Departments of Mechanical and Chemical Engineering,
Queen's University, Kingston, Ontario K7L 3N6, Canada*

Abstract

Grid staggering is a well known remedy for the problem of velocity/pressure coupling in incompressible flow calculations. Numerous inconveniences occur, however, when staggered grids are implemented, particularly when a general-purpose code, capable of handling irregular three-dimensional domains, is sought. In several non-staggered grid numerical procedures proposed in the literature, the velocity/pressure coupling is achieved by either pressure or velocity (momentum) averaging. This approach is not convenient for simultaneous (block) solvers that are preferred when using multigrid methods. A new method is introduced in this paper that is based upon non-staggered grid formulation with a set of virtual cell face velocities used for pressure/velocity coupling. Instead of pressure or velocity averaging, a momentum balance at the cell face is used as a link between the momentum and mass balance constraints. The numerical stencil is limited to 9 nodes (in 2D) or 27 nodes (in 3D) both during the smoothing and inter-grid transfer, which is a convenient feature when a block point solver is applied. The results for a lid-driven cavity and a cube in a lid-driven cavity are presented and compared to staggered grid calculations using the same multigrid algorithm. The method is shown to be stable and produce a smooth (wiggle-free) pressure field.

[1]Ph.D. Student

1 Introduction

Multigrid methods are used in a number of applications in fluid dynamics, usually by applying the Full Approximation Scheme [1]. Incompressible flow calculations usually employ a staggered grid because of its strong coupling between the pressure and the velocity field (e.g. [2]. For complex geometries, however, as well as for calculations in non-orthogonal coordinates, the use of a staggered grid is a serious obstacle to efficient and well structured computer coding [3]. Additional complexities arise when a block-solver is used; for example, variables cannot be easily grouped into cell-bound blocks due to different node count. Some authors resort to asymmetric nodal clusters [5] while others update a symmetric block of variables around the cell centre node thereby updating face velocities twice in each relaxation sweep [5, 6]. Various levels of decoupled relaxation are also common. These include distributive relaxation, where all momentum equations are solved together and the pressure field is solved separately [1, 7], and sequential schemes that update variables throughout the flow field one by one [8, 9]. Some comparative studies of block *versus* sequential relaxation give no clear preference [10, 11]. There is a greater consensus that grid staggering is a necessary burden, particularly in the context of multigrid methods ([12, 13, 14, 15, 16] and even [1]). Comparison studies of staggered and non-staggered methods are sometimes conflicting in their assessment of the accuracy and stability of any given method. While some authors demonstrate that non-staggered methods match the staggered ones using both criteria ([13, 16, 17]), others question it ([18]). Despite this, the majority of finite volume incompressible calculations use staggered grids. The main reason may be that existing non-staggered grids *increase* rather than lessen the complexity of the staggered grid calculations. For example, the method of Rhie and Chow [19] (adopted by [13, 14, 15]) requires that both the nodal and cell face velocities are stored. Moreover, in a multigrid context, both the nodal and the face velocities need to be restricted [8], requiring even more computational work. Also, the computational cluster extends beyond either 9 or 27 point stencil in two- or three-dimensional formulations respectively for the first order discretisation and even more if the higher order methods are used.

The considerations mentioned above motivated the present contribution for a method suitable for block solvers on an irregular three-dimensional domain using a non-staggered grid.

In this paper a brief description of the multigrid procedure based on a new non-staggered method is given in section 2 and the multigrid implementation in section 3; the test cases and results are presented in section 4, followed by the discussion section where relative merits of the method are assessed.

2 The new non-staggered method

A transport equation for a general set of transported variables $\mathbf{u}$ in the volume Ω bounded by a boundary S can be expressed in an integral form suitable for finite volume formulation

$$\frac{\partial}{\partial t}\rho \mathbf{u} \mathrm{d}\Omega + \oint_S \left[\rho \mathbf{u} u_i - \Gamma_{\mathbf{u}} \left(\frac{\partial \mathbf{u}}{\partial x_i}\right)\right] n_i \mathrm{d}S = \mathbf{F}_{\mathbf{u}}, \tag{1}$$

where ρ is the density, u_i is the velocity component in the x_i direction and n_i is the component of unit normal to the boundary S. When (1) is applied to the momentum balance of a viscous incompressible fluid, the set $\mathbf{u}$ is a velocity vector $\mathbf{u} = \{u_j, j = 1, ..., d\}$ (d being the problem dimension), with the corresponding diffusion coefficient $\Gamma = \mu$ and the source terms

$$F_{u_j} = -\int_\Omega \left(\frac{\partial p}{\partial x_j}\right) \mathrm{d}\Omega + \oint_S \left[\mu \left(\frac{\partial u_i}{\partial x_j}\right) - \frac{2}{3}\delta_{ij}\mu \left(\frac{\partial u_k}{\partial x_k}\right)\right] n_i \mathrm{d}S \tag{2}$$

in the absence of external volume forces. The extension to other transportable properties (such as enthalpy, mass fraction, etc.) is straightforward by augmenting the vector $\mathbf{u}$ to include new variables and defining appropriate source terms and the diffusion coefficients. In the following presentation a three-dimensional implementation will be used.

The momentum equations are discretised using the hybrid (central/upwind) difference scheme [20] although higher order schemes can be employed[1]. The pressure field is resolved by means of mass conservation for the control volume around the node in a symmetric block manner as used by Vanka [5] for the staggered grid, although the extension to the line block around the node in a symmetric block manner as used by Vanka [5] for the staggered grid, although the extension to the line block formulation is

[1]For a multigrid implementation of a second order upwind scheme on a staggered grid see e.g. [21].

straightforward. The estimation of the face velocities that are substituted into the mass conservation equation is obtained by discretizing the momentum equation over a half-length volume around each cell face, directly involving the nodal pressure and velocities, while the lateral velocities are obtained by averaging values from the nearby nodes (see Fig. 1). More details on the coefficient generation are given in [22].

The principle of discretizing the face velocity using a half-size cell is applied also by Schneider and Raw [3], although in their method the coefficients of the face velocity are treated implicitly by incorporating them into the nodal velocity coefficient matrix. To ensure positive definiteness of the nodal velocities coefficient matrix, Schneider and Raw had to truncate the momentum equation applied to the face velocities [3, 16]. In the present method, the face velocities are explicitly expressed in terms of the surrounding nodal values and used in the continuity equation for the pressure correction calculation. The implication of this step is that the family of face velocities satisfies both the momentum and the mass conservation exactly at the positions where the convection velocities in a general transport equation are required. On the other hand, as an average of the (tentative) nodal velocities they are readily available without requirement for a permanent storage.

The boundaries of the flow field are coincident with the cell faces, enabling the definition of a set of boundary nodes there. This practice ensures consistency between the global mass balance of the whole calculation domain and the local mass balance of each cell, but calls for special treatment if Neumann boundary conditions are to be used. If the zero-gradient condition for the normal velocity is discretised in a usual way

$$\frac{\partial(u_i n_i)}{\partial(x_i n_i)} = \frac{(u_i n_i)_b - (u_i n_i)_{inn}}{(x_i n_i)_b - (x_i n_i)_{inn}}, \tag{3}$$

where subscripts b and inn denote boundary and the first inner node, respectively, the flow rate through the boundary will be linked to the velocity that does not belong to the mass preserving field, resulting in poor overall mass conservation. The correct way to implement Neumann boundary condition in this case is to use the face velocity. This way, the local and global mass balance become fully compatible. There is no need for any special treatment of the Dirichlet boundary conditions where the face velocities coincide with the boundary and are assumed known.

3 The multi-grid implementation

In the multigrid context, the nonlinear equation (1) can be expressed as

$$\mathcal{L}(\mathbf{u}) = \mathbf{F} \tag{4}$$

by grouping all terms that will result in a coefficient matrix (within a Newton iteration cycle) into the operator $\mathcal{L}$ and the remainder into the source term $\mathbf{F}$ as in [13, 23]. A more common practice of expressing Eqn. (4) as homogeneous (by absorbing $\mathbf{F}$ into $\mathcal{L}(\mathbf{u})$) [1, 24] is found by the present authors to be somewhat confusing, especially when defining residual transfer to the coarse grid.

The discretised (sparse, positive definite) Eqn. 4 for the grid l is linearised by a Newton iteration [24]

$$L^l \mathbf{u}^l = \mathbf{F}^l \tag{5}$$

and relaxed by a block Gauss-Seidel method.

The updates of the variable set

$$\mathbf{u}' = \alpha(\mathrm{diag}(L))^{-1}\mathbf{R}^l \tag{6}$$

are expressed in terms of the residual $\mathbf{R}^l = \mathbf{F}^l - L^l\mathbf{u}^l$, the inverse of the coefficient matrix diagonal $(\mathrm{diag}(L))^{-1}$ and the underrelaxation coefficient α. Variables at the node i, j, k are then updated by $\mathbf{u}_{i,j,k} = \mathbf{u}_{i,j,k} + \mathbf{u}'_{i,j,k}$.

Restriction is accomplished by grouping a cluster of eight cells into one. This leads to the following operator

$$\begin{aligned}\phi_{I,J,K} = \frac{1}{8}(&\phi_{i,j,k} + \phi_{i+1,j,k} + \phi_{i,j+1,k} + \phi_{i,j,k+1} + \\ &\phi_{i+1,j+1,k} + \phi_{i,j+1,k+1} + \phi_{i+1,j,k+1} + \phi_{i+1,j+1,k+1}),\end{aligned} \tag{7}$$

where $I = 2i - 2, \ldots$ The same operator is applied both to variable and residual restriction. After both the variables and residuals are transferred to the next coarser grid $(l-1)$, Eqn. (5) is approximated as

$$L^{l-1}(\mathbf{u}^{l-1}) = \overline{\mathbf{F}}^{l-1}, \tag{8}$$

where

$$\overline{\mathbf{F}}^{l-1} = \mathbf{F}^{l-1} - (\mathbf{F}_0^{l-1} - L_0^{l-1}\mathbf{u}_0^{l-1}) + \mathcal{R}_l^{l-1}\mathbf{R}^l \tag{9}$$

is the equivalent source term on the coarse grid. The restriction at Neumann boundaries is carried out using a divided form of the boundary conditions [1]. For the velocity component perpendicular to the boundary, an additional correction is made to preserve the mass flow rate through the boundary.

Prolongation is carried out by tri-linear interpolation using a seven point stencil, shown here for one cell and with injection only:

$$\phi_{i,j,k} = \frac{1}{6}(3\phi_{I,J,K} + \phi_{I-1,J,K} + \phi_{I,J-1,K} + \phi_{I,J,K-1}) \qquad (10)$$

with $i = (I+2)/2, \ldots$ The injection upon the first visit to the fine grid (FMG cycling is assumed) and the fine grid correction are done as

$$\mathbf{u}^l_{\text{first}} = \mathcal{P}^l_{l-1}\mathbf{u}_{l-1} \quad \text{or} \quad \mathbf{u}^l_{\text{new}} = \mathbf{u}^l_{\text{old}} + \mathcal{P}^l_{l-1}(\mathbf{u}_{l-1} - \mathbf{u}_{0,l-1}). \qquad (11)$$

4 Test cases

The non-staggered method presented in this paper is compared with the staggered three-dimensional calculations employing the block symmetric Gauss-Seidel algorithm of Vanka [5]. For both methods the coding and data structures are of the same style.

The flow in a three-dimensional cavity with a moving top is used as a first test case. The residual norm history is shown in Fig. 2. The rate of convergence obtained when calculating on a staggered grid is comparable with the results of Vanka [10] where 12 work units (w.u.) were necessary for a two orders of magnitude residual reduction. In our calculations 14 w.u. was necessary for the staggered grid calculation and 18 w.u. for the non-staggered calculations.[1] However, the change in slope of the non-staggered residual may indicate that the full potential of multigrid acceleration has yet to be achieved.

In a second test case, a cube is inserted in a cavity (Fig. 3), forcing the flow to negotiate this asymmetric three-dimensional obstacle, partly by the velocity magnitude change, partly by flow separation. It is believed that this flow geometry serves as a good test of the pressure/velocity coupling

[1]Or 23 w.u. for the same residual decrease; however, this is more arbitrary, because of the much lower initial residual at the finest grid.

because the major force behind the flow adjustment is the pressure field. The residual history, (Fig. 4) indicates very similar convergence rates for the staggered and non-staggered calculations. The resulting flow field in a symmetry plane (Fig. 5) indicates well resolved separation bubbles around the cube corners.

5 Discussion and conclusions

The new method of incompressible flow calculation using non-staggered grids and its multigrid implementation are examined for suitability in a complex flow field geometry. The presence of two sets of velocity values, both of which satisfy the (discrete form of the) governing equations increases the overall level of accuracy for a given grid size, although this remains to be quantified.

In the numerical experiments performed so far the method proved to be stable, without any tendency to produce an oscillating pressure field, which is a common feature of some non-staggered methods [18]. The method permits discretization on a trivially coarse grid (with a single node in the interior), which is very convenient in a FAS multigrid implementation, because it allows the coarse grid to contain the lowest number of nodes. Thus significantly coarser grids can be used in complex geometries. For example, in the case of a cube in a cavity (see the previous section) the coarsest grid (6x6x6 nodes) has only one control volume located between the cube and the cavity wall at one side. If the calculation method required two nodes at minimum, the overall node count at the coarse grid would increase eight times, thereby substantially increasing the work needed to obtain exact solution at the coarsest grid.

Various tests performed so far always produced smooth solutions both in velocity and pressure, which indicate a high ellipticity measure of the proposed method. The analytical evaluation of the ellipticity measure remains to be carried out (following e.g. [25, 16]).

The amount of computational work of the proposed method is slightly larger that of the Rhie and Chow [19] method. It is comparable to the work in the SCGS method of Vanka, requiring the same amount of work to calculate face velocities and pressure coefficients and, in addition, the calculation of the nodal velocity coefficients, i.e. approximately 25% increase in two-dimensional and 14% in three-dimensional calculations. This overhead exists only for the simplest flow problem because any additional variable that is solved permits nodal velocity coefficients to be reused (with proper scaling of the diffusion part).

References

[1] A. Brandt. Guide to multigrid development. In *Multigrid Methods: Proceedings,Köln-Porz, 1981*, pages 220–312. Springer-Verlag, Berlin, 1982.

[2 B. Favini and G. Guj. MG techniques for staggered differences. In Holstein D.J., Paddon H., editor, *Multigrid Methods for Integral and Differential Equations*, pages 253–262, Oxford, 1985. Clarendon Press.

[3] G.E. Schneider. A novel colloccated finite difference procedure for the numerical computation of fluid flow. In *Proceedings of 4th AIAA/ASME Joint Thermophysics and Heat Transfer Conference.* AIAA-86-1330, 1986.

[4] B. R. Hutchinson, P. F. Galpin, and G. D. Raithby. Application of additive correction multigrid to the coupled fluid flow equations. *Numerical Heat Transfer*, 13:133–147, 1988.

[5] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65:138–158, 1986.

[6] M. C. Thompson and J. H. Ferziger. An adaptive multigrid solution technique for the incompressible navier-stokes equations. *Journal of Computational Physics*, 83:94–121, 1989.

[7] L. Fuchs and H.-S. Zhao. Solution of three-dimensional viscous incompressible flows by a multi-grid method. *International Journal for Numerical Methods in Fluids*, 4:539–555, 1984.

[8] M. Hortmann, M. Peric, and G. Scheuerer. Finite volume multigrid prediction of laminar natural convection: Bench-mark solutions. *Journal for Numerical Methods in Fluids*, 11:189–207, 1990.

[9] P. A. Rubini, H. A. Becker, E. W. Grandmaison, A. Pollard, A. Sobiesiak, and C. Thurgood. Multi-grid acceleration of three dimensional turbulent variable density flows. *Numerical Heat Transfer, B: Fundamentals*, pages 163–177, 1992.

[10] S. P. Vanka. Fast numerical computation of viscous flow in a cube. *Numerical Heat Transfer, Part B: Fundamentals*, 20:255–261, 1991.

[11] P. H. Gaskell, A. K. C. Lau, and N. G. Wright. Comparison of two solution strategies for use with higher-order discretization schemes in fluid flow simulation. *International Journal for Numerical Methods in Fluids*, 8:1203–1215, 1988.

[12] C. Hirsch. *Numerical Computation of Internal and Eternal Flows*, volume 1: Fundamentals of Numerical Discretization. John Wiley & Sons, New York, 1988.

[13] M. Perić, R. Kessler, and G. Scheuerer. Comparison of finite-volume numerical methods with staggered and collocated grids. *Computers and Fluids*, 16:389–403, 1988.

[14] S. Majumdar. Role of underrelaxation in momentum interpolation for calculation of flow with nonstaggered grids. *Numerical Heat Transfer*, 13:125–132, 1988.

[15] P. Coelho, J. C. F. Pereira, and M. G. Carvalho. Calculation of laminar recirculating flows using a local non-staggered grid refinement system. *International Journal for Numerical Methods in Fluids*, 12:535–557, 1991.

[16] W. W. Armfield. Finite difference solutions of the navier-stokes equations on staggered and non-staggered grids. *Computers and Fluids*, 20(1):1–17, 1991.

[17] T. M. Shih and A. L. Ren. Primitive-variable formulations using nonstaggered grids. *Numerical Heat Transfer*, 7:413–428, 1984.

[18] T. M. Shih, C. H. Tan, and B. C. Hwang. Effects of grid staggering on numerical schemes. *International Journal for Numerical Methods in Fluids*, 9:413–428, 1989.

[19] C. M. Rhie and W. L. Chow. A numerical study of the turbulent flow past an isolated airfoil with trailing edge separation. AIAA-82-0998, 1982.

[20] D. B. Spalding. A novel finite-difference formulation for differential expressions involving both first and second derivatives. *International Journal of Numerical Methods in Engineering*, 4:551–559, 1972.

[21] D. Xiao, H. A. Becker, E. W. Grandmaison, and A. Pollard. The calculation of a jet in a crossflow using embedded/overlapping grid techniques, multi-grid acceleration and a modified QUICK differencing scheme. *Submitted to: International Journal of Computational Fluid Dynamics*, 1993.

[22] D. Matovic and A. Pollard. Evaluation of a new block non-staggered calculation method for an incompressible flow and its multigrid acceleration. Submitted for publication.

[23] W. Shyy and C-S. Sun. Development of a pressure-correction/staggered-grid based multigrid solver for incompressible recirculating flows. *Computers and Fluids*, 22:51–76, 1993.

[24] W. Hackbush. *Multi-Grid Methods and Applications.* Springer-Verlag, Berlin, 1985.

[25] A. Brandt and N. Dinar. Multigrid solutions to elliptic flow problems. In *Numerical Methods for Partial Differential Equations*, pages 53–147. Academic Press, New York, 1979.

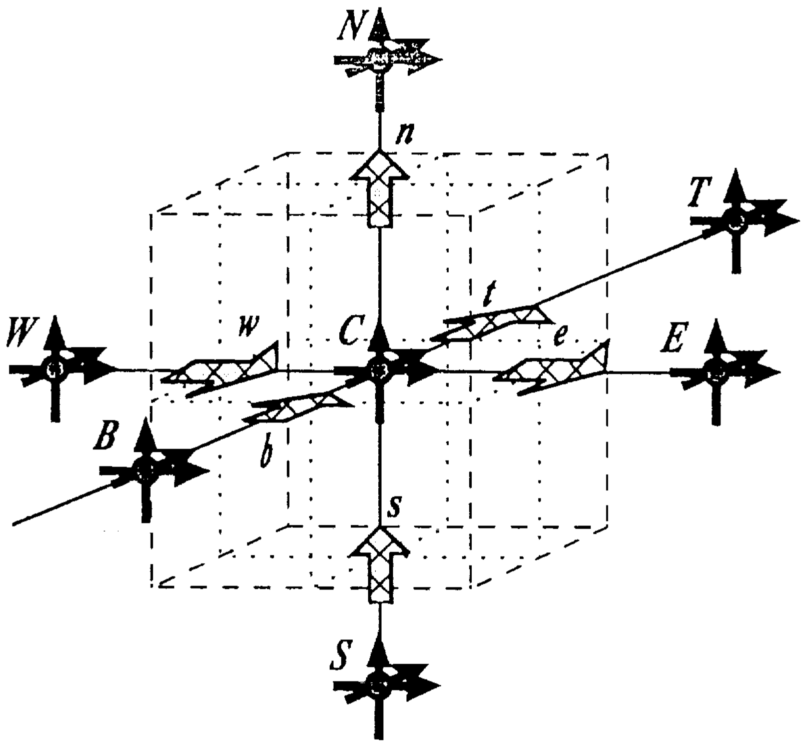


Figure 1: The layout of a non-staggered grid. Only the nodal variables require storage.

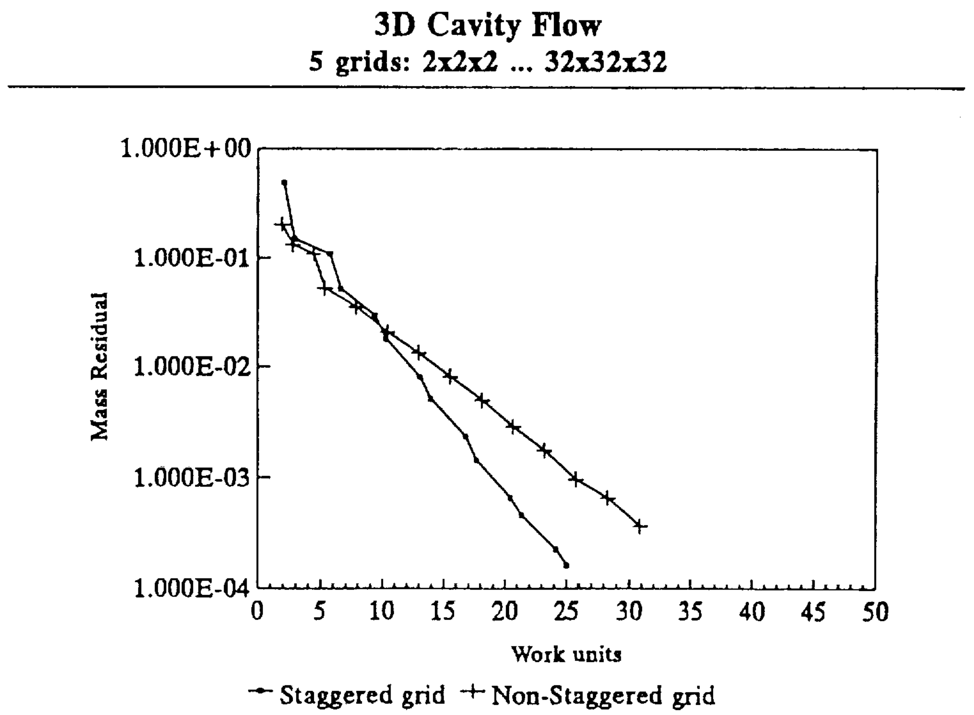


Figure 2: Mass residual history for a lid-driven cavity flow. Re = 400.

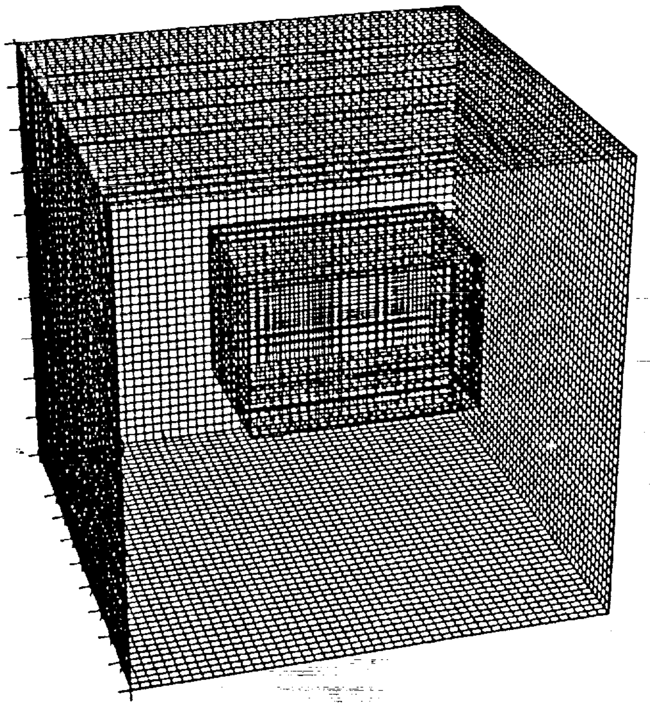


Figure 3: The grid for a flow around the cube in a lid-driven cavity.

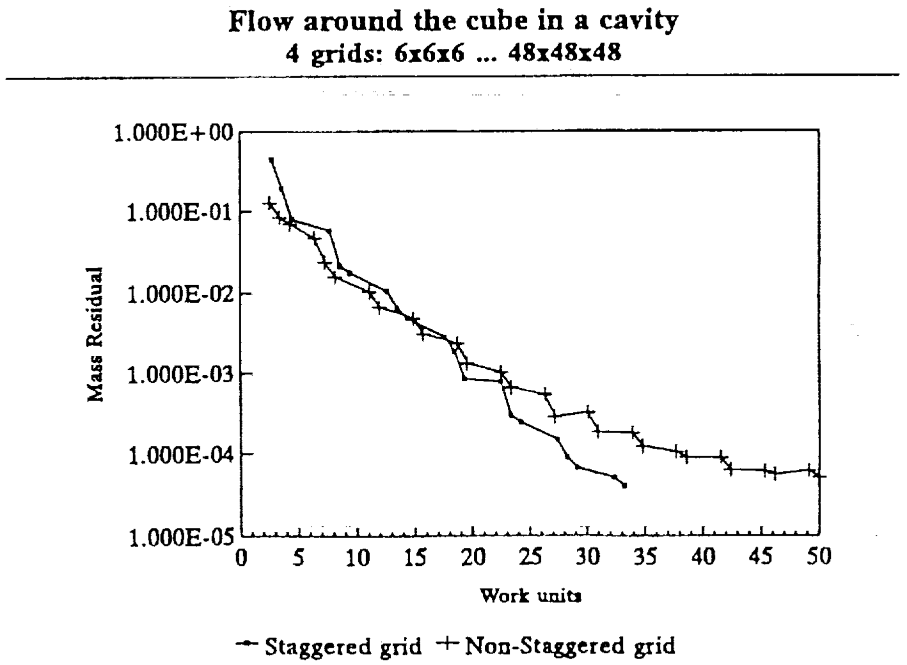


Figure 4: Mass residual history for the flow around the cube in a lid-driven cavity. Cavity Re=400.

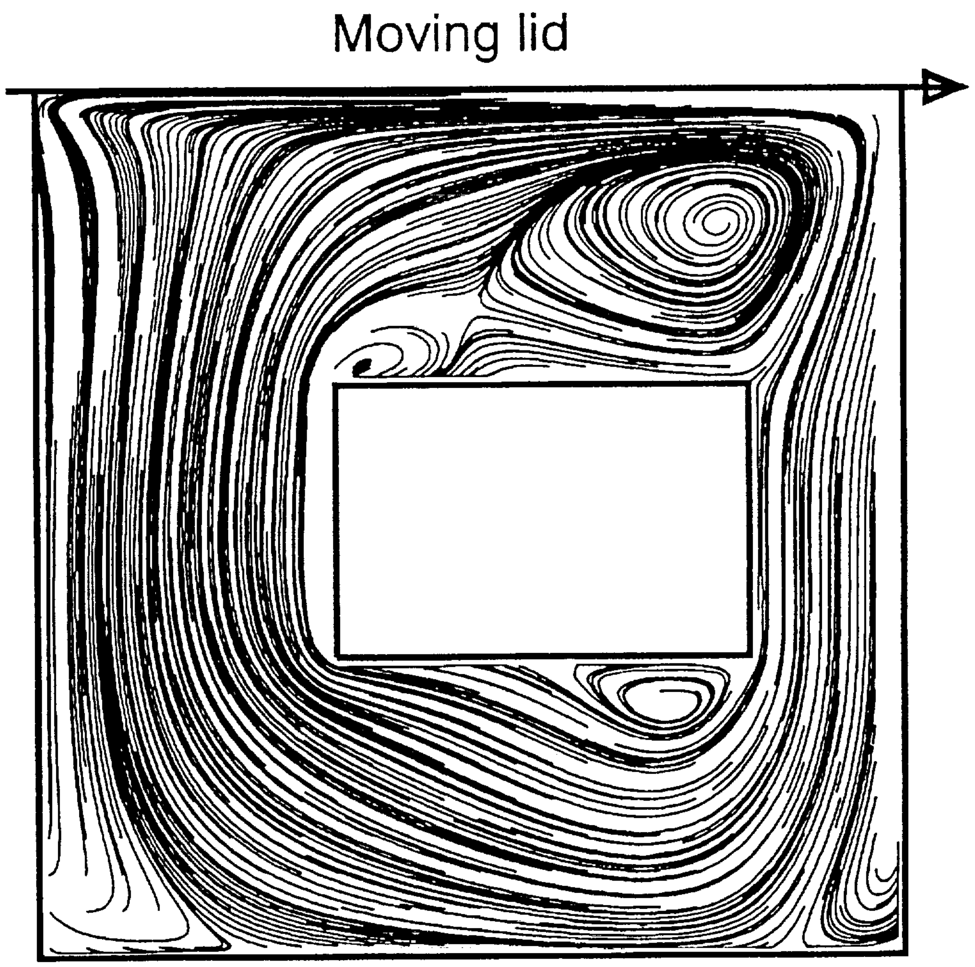


Figure 5: Flow around a cube in a lid-driven cavity: particle traces in a symmetry plane.

[illegible]

53-32
197135
p. 11

MULTIGRID AND CYCLIC REDUCTION APPLIED TO THE HELMHOLTZ EQUATION

N94-23676

Kenneth Brackenridge
Oxford University Computing Laboratory
Oxford OX1 3QD, U.K.

ABSTRACT

We consider the Helmholtz equation with a discontinuous complex parameter and inhomogeneous Dirichlet boundary conditions in a rectangular domain. A variant of the direct method of cyclic reduction is employed to facilitate the design of improved multigrid components, resulting in the method of CR-MG. We demonstrate the improved convergence properties of this method.

1 INTRODUCTION

Microwave heating of foods has revolutionised the food processing industry. Effective and efficient microwave heating depends very much on a detailed knowledge and understanding of the dielectric properties of the food to be processed. This need has given rise to extensive research into the dielectric properties of materials; see, for example, Tinga and Nelson [1].

Microwave heating can be compared to heating by alternating current. The electric field of alternating current changes direction approximately 100 times each second, whereas the microwave field changes direction approximately 5 billion times each second. The heating effect is accomplished by energy transfer to dipoles, most commonly water. Most foods contain between 50 and 90 percent water. By attempting to follow the very rapidly changing microwave electric field, the molecular vibrations of the dipoles are strengthened, thus increasing the temperature of the water and hence the food.

The scalar potential ϕ associated with the microwave field satisfies the wave equation

$$\nabla^2\phi - \epsilon\mu\frac{\partial^2\phi}{\partial t^2} = 0, \qquad (1)$$

which is derived from Maxwell's equations of electromagnetics. The parameter ϵ describes the permittivity of the medium and the parameter μ the permeability. However, the radiation field in a microwave oven varies harmonically in time, and so we look for a solution of equation (1) in the form

$$\phi(\mathrm{x}\,,\,t) = e^{i\omega t}\,u(\mathrm{x}),$$

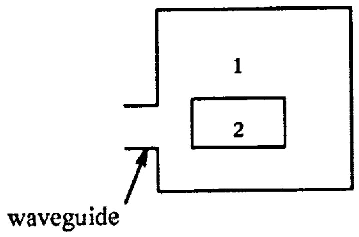


Figure 1: A two-dimensional model of a microwave oven.

where u is a time-independent scalar potential function and ω is the frequency of the microwave radiation. By substituting this expression into equation (1), we see that u satisfies the Helmholtz equation

$$\nabla^2 u + \delta u = 0,$$

where $\delta := \epsilon\mu\omega^2$. In general, ϵ and μ are complex numbers, with real parts related to a material's ability to store electrical and magnetic energy respectively, and imaginary parts related to a material's ability to dissipate electrical and magnetic energy respectively. However, the permeability of biological materials is close to that of free space, i.e. $\mu \approx \mu_0 = 4\pi \times 10^{-7}$ Hm^{-1}. Hence, since most domestic microwave ovens operate at a frequency of 2450 MHz, we can calculate δ for any given permittivity ϵ.

The oven is represented schematically (in two dimensions) by the rectangular domain depicted in Fig. 1. Region 1 corresponds to free space and so $\epsilon = \epsilon_0 = \frac{1}{36\pi} \times 10^{-9}$ Fm^{-1} and δ is a real constant in this region. Region 2 corresponds to the heated material and so δ is a complex constant in this region. Energy is fed into the system by a magnetron via the waveguide. Hence, in this paper, we consider the solution of the Helmholtz equation with a discontinuous complex parameter and inhomogeneous Dirichlet boundary conditions in a rectangular domain.

We close this section with a plan of the paper. In section 2 we describe the mathematical problem and discuss the smoothing abilities of two multigrid smoothers. In section 3 we describe the technique of approximate cyclic reduction and show how this can be used to design improved multigrid components. Numerical results are presented in section 4 and some concluding remarks are made in section 5.

2 MATHEMATICAL PROBLEM

Consider the complex two-dimensional Dirichlet boundary value problem

$$\begin{aligned} \nabla^2 u + \delta u = 0 \quad &\text{in} \quad \Omega = \Omega_1 \cup \Omega_2 \\ \text{s.t.} \quad u = g \quad &\text{on} \quad \partial\Omega, \end{aligned} \tag{2a}$$

with data

$$\delta = \begin{cases} \delta_1 & \text{in subdomain } \Omega_1 \\ \delta_2 & \text{in subdomain } \Omega_2 \end{cases}, \tag{2b}$$

where Ω_1 and Ω_2 are rectangular subdomains of Ω (as in Fig. 1).

Operator Definitions

Before attempting to solve this problem by the multigrid method, we need to carefully consider the definitions of the discretisation, restriction and prolongation operators. In [2], De Zeeuw considers the solution of general linear second order elliptic partial differential equations over similar domains. He notes that the rate of convergence of standard multigrid methods often deteriorates when the coefficients in the differential equation are discontinuous; he proposes matrix-dependent grid transfer operators to overcome these difficulties. However, in our case, the discontinuity occurs only in the coefficient of u (viz. δ), and not of ∇u. Hence we proceed in the following way to define operator $\mathcal{P} = \mathcal{P}(\delta)$ in the domain Ω, where $\mathcal{P}$ can be taken to represent the discretisation, restriction or prolongation operator. Firstly, if δ takes value δ_i in subdomain Ω_i $(i = 1, 2)$, then we set the value of δ on the interior boundary between Ω_1 and Ω_2 to $\delta_3 := \frac{1}{2}(\delta_1 + \delta_2)$. Secondly, $\mathcal{P}$ is defined piecewise by

$$\mathcal{P} = \begin{cases} \mathcal{P}(\delta_1) & \text{in } \Omega_1 \\ \mathcal{P}(\delta_2) & \text{in } \Omega_2 \\ \mathcal{P}(\delta_3) & \text{on } \partial\Omega_2 \end{cases} . \tag{3}$$

In practice, this definition of $\mathcal{P}$, for discontinuous δ, does not seem to impair the convergence of the multigrid algorithm for relevant values of δ.

Equivalent System of Real Equations

Consider the discrete analogue of problem (2). Suppose $\delta = \alpha + i\beta \in \mathbb{C}$ and $g \in \mathbb{R}$. Using a central difference discretisation on a mesh of $n \times n$ intervals, the matrix of the discrete system $A\mathbf{u} = \mathbf{f}$ is represented in stencil notation by

$$A \sim \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & p & 1 \\ & 1 & \end{bmatrix}, \tag{4}$$

where $A \in \mathbb{C}^{(n-1)^2} \times \mathbb{C}^{(n-1)^2}$, $h := \frac{1}{n}$ and $p := \delta h^2 - 4 = (\alpha h^2 - 4) + i\beta h^2$. Hence, while most linear systems which arise in practice have real coefficient matrices, the discretisation of this problem yields a complex linear system. Further applications which give rise to complex linear systems include discretisations of the time-dependent Schrödinger equation, inverse scattering problems and underwater acoustics.

A popular approach for solving complex linear systems is to solve the equivalent real linear systems for the real and imaginary parts of u. However, the following remarks, due to Freund [3], cast doubt on this approach. Suppose that A is a general complex $m \times m$ matrix. By taking real and imaginary parts, we can rewrite the complex system as the real linear $2m \times 2m$ system

$$B\bar{\mathbf{u}} = \begin{pmatrix} \mathcal{R}e\, A & \mathcal{I}m\, A \\ \mathcal{I}m\, A & -\mathcal{R}e\, A \end{pmatrix} \begin{pmatrix} \mathcal{R}e\, \mathbf{u} \\ -\mathcal{I}m\, \mathbf{u} \end{pmatrix} = \begin{pmatrix} \mathcal{R}e\, \mathbf{f} \\ \mathcal{I}m\, \mathbf{f} \end{pmatrix}.$$

It can then be shown that B has eigenspectrum

$$\sigma(B) = \{\lambda \in \mathbb{C} | \, \lambda^2 \in \sigma(\bar{A}A)\},$$

which means that $\sigma(B)$ is symmetric with respect to the real and imaginary axes and hence the eigenvalues *always* embrace the origin. Now if A is complex symmetric (as is the case with (4)), then B is a real symmetric matrix with real eigenvalues symmetrically distributed about the origin, i.e. B is symmetric indefinite. Therefore the equivalent real system is often harder to solve than the original complex one.

Smoothing Analysis

Multigrid smoothing methods are usually basic iterative methods, the properties of which are well understood. As the name suggests, the function of a multigrid smoothing method is to reduce the rough (high frequency) components of the error as efficiently as possible. This is basically a local task and so the smoothing efficiency of a method can be analysed by local Fourier mode analysis [4], neglecting interactions with boundaries. The smooth (low frequency) error components are reduced on the coarser grids. There is a natural distinction between high and low frequencies depending on the type of grid coarsening chosen. Essentially, the low frequencies are those which are visible on the coarser grids. In principle, smoothing methods need not be convergent (see [5], chpt 7), although in practice most are.

Consider the discrete analogue of problem (1), $A\mathbf{u} = \mathbf{f}$, defined on a mesh of $n \times n$ intervals. Basic iterative methods are based on a matrix splitting $A = M - N$ and are defined by

$$M\mathbf{u}^{(m+1)} = N\mathbf{u}^{(m)} + \mathbf{f}.$$

The algebraic error arising from the iterative solution of this system of equations is defined by $\mathbf{e}^{(m)} := \mathbf{u}^{(m)} - \mathbf{u}$ and satisfies the equation $M\mathbf{e}^{(m+1)} = N\mathbf{e}^{(m)}$. Denoting the stencils of M and N by $[M]$ and $[N]$ respectively, this equation can be rewritten in stencil form as $[M]\, e_{j\,k}^{(m+1)} = [N]\, e_{j\,k}^{(m)}$. Now if we define $\mathbf{e}^{(m+1)} := \lambda\, \mathbf{e}^{(m)}$ and note that the algebraic error can be represented as a combination of local Fourier modes

$$e_{j\,k}^{(m)} = \lambda^m \, e^{i(j\theta + k\phi)}, \; (\theta \,, \phi) \in \Theta := \{(\tfrac{2\pi p}{n} \,, \tfrac{2\pi q}{n}) : -\tfrac{n}{2} + 1 \leq p \,, q \leq \tfrac{n}{2}\},$$

then by substituting this into the stencil representation of the error recurrence we define the *error amplification factor*

$$\lambda(\theta \,, \phi) \;:=\; \frac{[N]\, e^{i(j\theta + k\phi)}}{[M]\, e^{i(j\theta + k\phi)}}.$$

The error amplification factor is the factor by which the amplitude of the $(\theta \,, \phi)$ Fourier mode is multiplied as a result of a single smoothing iteration. Now in the case of standard grid coarsening, the sets of smooth and rough frequencies are defined by

$$\Theta_s := \Theta \cap (-\tfrac{\pi}{2} \,, \tfrac{\pi}{2})^2,$$

$$\Theta_r := \Theta \backslash \Theta_s.$$

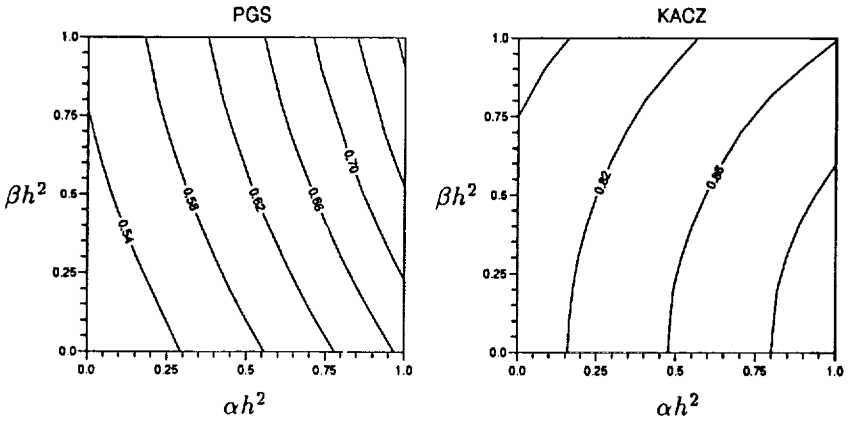


Figure 2: Fourier smoothing factors ρ^D for PGS and KACZ.

Hence the *Fourier smoothing factor*, which is the worst factor by which all high frequency error components are reduced per iteration, is defined by

$$\rho := \max_{(\theta,\phi)\in\Theta_r} |\lambda(\theta,\phi)|.$$

Note however that this definition of the smoothing factor is only valid for boundary conditions of harmonic type. The influence of Dirichlet boundary conditions can be taken into account heuristically (see [6] and [7], for example) in the following way. The error at the boundary is always zero and so we define a new set of rough frequencies as

$$\Theta_r^D := \Theta_r \cap \{(\theta,\phi)\in\Theta : \theta\neq 0 \text{ and/or } \phi\neq 0\}.$$

The corresponding Fourier smoothing factor is defined by

$$\rho^D := \max_{(\theta,\phi)\in\Theta_r^D} |\lambda(\theta,\phi)|.$$

This is a mesh-dependent definition. A mesh-independent definition, introduced by Brandt [4], is obtained by replacing the discrete set Θ with a continuous analogue, but this is more difficult to compute numerically and gives less realistic results in cases where the type of boundary condition has much influence.

There are many possibilities for the choice of smoothing method (see [7], for example), but for brevity we consider only two, point Gauß-Seidel iteration (PGS) and Kaczmarz iteration (KACZ). The latter of these two methods, dating back to 1937 [8], is considered here because, when applied to the complex linear system $A\mathbf{u} = \mathbf{f}$, the method converges for all distributions $\sigma(A)$ of eigenvalues of A. The reason for this is that solving the system $A\mathbf{u} = \mathbf{f}$ using KACZ is equivalent to solving the system $AA^H\mathbf{v} = \mathbf{f}$ with $\mathbf{u} = A^H\mathbf{v}$ (i.e. *postconditioning*) using PGS, and the matrix AA^H is Hermitian positive definite, thus guaranteeing convergence. Applying the smoothing analysis to stencil (4), the error amplification factors for PGS and KACZ are

$$\lambda_{\text{PGS}} = -\frac{e^{i\theta}+e^{i\phi}}{p+e^{-i\theta}+e^{-i\phi}},$$

$$\lambda_{\text{KACZ}} = -\frac{(e^{i\theta}+e^{i\phi})(e^{i\theta}+e^{i\phi}+p+\overline{p})+2e^{i(-\theta+\phi)}}{4+p\overline{p}+(e^{-i\theta}+e^{-i\phi})(e^{-i\theta}+e^{-i\phi}+p+\overline{p})+2e^{i(\theta-\phi)}},$$

for some $p = (\alpha h^2 - 4) + i\beta h^2$ and $(\theta\,,\,\phi) \in \Theta$. Fig. 2 displays contour plots of ρ^D_{PGS} and ρ^D_{KACZ} plotted as functions of αh^2 and βh^2. For fixed values of h and $\alpha = \mathcal{R}e\,\delta$, as $\beta = \mathcal{I}m\,\delta$ increases, ρ^D_{PGS} increases and ρ^D_{KACZ} decreases. Hence we might expect the multigrid convergence rate to improve slowly with a KACZ smoother and deteriorate more rapidly with a PGS smoother as β increases. This is borne out in practice. Finally, as a rule of thumb, a *good* smoothing method has a smoothing factor $\rho^D < \frac{1}{2}$. In this sense, neither of the two methods considered here is a good smoothing method for problem (2).

3 CYCLIC REDUCTION AND MULTIGRID

Cyclic reduction (CR) is a *direct* method of solution for tridiagonal and block-tridiagonal systems of linear algebraic equations [9], [10]. For tridiagonal systems which represent approximations to 1-D second order ordinary differential equations, CR is as efficient as multigrid (MG). For problems in higher dimensions CR becomes too computationally expensive due to fill-in within the blocks. However, the *design* of MG methods in higher dimensions can be facilitated by drawing comparisons between MG and CR (see Shaw [11]).

Approximate Cyclic Reduction

Consider the system of equations $L\mathbf{u} = \mathbf{f}$. If $\mathbf{v}$ is an approximation to the true solution $\mathbf{u}$, then we define the error vector as $\mathbf{e} := \mathbf{u} - \mathbf{v}$ and the residual vector as $\mathbf{r} := \mathbf{f} - L\mathbf{v} = L\mathbf{e}$. Then assuming that the error vector $\mathbf{e}$ is sufficiently smooth (a condition normally guaranteed by a few applications of a smoother in a MG algorithm), the fill-in can be minimised by making accurate Taylor expansion approximations of the outlying errors. This method is known as *approximate cyclic reduction* (ACR) [12].

Now consider a two-grid method applied to a two-dimensional Toeplitz system. Suppose the two grids have mesh sizes h and $H = 2h$ and the fine grid matrix has stencil

$$L_h \sim \begin{bmatrix} & b & \\ b & a & b \\ & b & \end{bmatrix},$$

where a and b are scalars. Given an initial approximation $\mathbf{v}$ to $\mathbf{u}$, we want to solve the equation $L_h\mathbf{e} = \mathbf{r}$ for $\mathbf{e}$ to obtain an improved approximation $\mathbf{v} + \mathbf{e}$. The method of ACR approaches this problem as follows.

Eliminate the outlying errors in the stencil using neighbouring equations to give

$$\tilde{L}_h \sim \begin{bmatrix} & & b^2 & & \\ & 2b^2 & 0 & 2b^2 & \\ b^2 & 0 & 4b^2 - a^2 & 0 & b^2 \\ & 2b^2 & 0 & 2b^2 & \\ & & b^2 & & \end{bmatrix}.$$

This first step of CR has destroyed the band structure of the original five-point operator. Further steps of CR would introduce more fill-in, resulting in a relatively inefficient process. Instead, assuming the errors are sufficiently smooth, approximate the errors at the NW, NE, SW and SE positions (in compass point notation) using accurate Taylor series expansions. This defines the ACR-modified coarse grid matrix, which has stencil

$$L_H \sim \sigma \begin{bmatrix} & & 2b^2 & & \\ & & 0 & & \\ 2b^2 & 0 & 8b^2 - a^2 & 0 & 2b^2 \\ & & 0 & & \\ & & 2b^2 & & \end{bmatrix},$$

where σ is an arbitrary scaling parameter. From the above information, the definition of restriction from the fine grid to the coarse grid can also be gleaned. The ACR-modified restriction operator has stencil

$$R_h^H \sim \sigma \begin{bmatrix} & b & \\ b & -a & b \\ & b & \end{bmatrix}. \tag{5}$$

For theoretical considerations it is very convenient to choose restriction and prolongation operators which satisfy the relation $P_H^h = R_h^{H*}$, where R_h^{H*} is the adjoint operator of R_h^H with respect to a suitably defined scalar product. However, the adjoint of the five-point restriction operator (5) is not a reasonable prolongation (see [13], p. 78). Alternative definitions of the prolongation operator are discussed in the following subsection.

ACR and the Helmholtz Equation

Consider a two-grid method, with mesh sizes h and $H = 2h$, applied to the fine grid Helmholtz differential operator $\mathcal{L}_h u := \nabla^2 u + \delta u$. Using a central difference discretisation on a mesh of $n \times n$ intervals, the fine grid matrix has stencil

$$L_h \sim \frac{1}{h^2} \begin{bmatrix} & 1 & \\ 1 & p & 1 \\ & 1 & \end{bmatrix},$$

where $h := \frac{1}{n}$ and $p := \delta h^2 - 4$. Hence $a = \frac{p}{h^2}$ and $b = \frac{1}{h^2}$. Now if we choose $\sigma = \frac{h^2}{8}$, then the ACR-modified coarse grid matrix and restriction operator have stencils

$$L_H \sim \frac{1}{(2h)^2} \begin{bmatrix} & & 1 & & \\ & & 0 & & \\ 1 & 0 & 4 - \frac{1}{2}p^2 & 0 & 1 \\ & & 0 & & \\ & & 1 & & \end{bmatrix},$$

$$R_h^H \sim \frac{1}{8} \begin{bmatrix} & 1 & \\ 1 & -p & 1 \\ & 1 & \end{bmatrix}.$$

Therefore the analogous coarse grid Helmholtz differential operator is defined as $\mathcal{L}_H u := \nabla^2 u + \delta(1 - \frac{\delta H^2}{32})u$, i.e. ACR suggests solving the Helmholtz equation with a different value of δ on the coarse grid in order to stabilise the MG process. For positive real values of δ for which L_h is indefinite, this corresponds to solving the Helmholtz equation with a *smaller* value of δ on the coarse grid, thus reducing the indefiniteness of L_H. There are various ways to define the prolongation operator. Possibilities include seven-point and nine-point prolongation [14]. However, a more effective definition of the prolongation operator for this interface problem is

$$P_H^h \sim \frac{1}{2p^2}\begin{bmatrix} 4 & -4p & 4 \\ -4p & 3p^2 & -4p \\ 4 & -4p & 4 \end{bmatrix},$$

which is derived from the tensor product of the one-dimensional ACR-modified prolongation operator. To extend these ideas to an m-grid process, where h_i is the mesh size of grid Ω_i and $h_{i+1} = 2h_i$, we proceed as follows.

Define $\delta_1 := \delta$ and $\delta_k := \delta_{k-1}(1 - \frac{\delta_{k-1}h_k^2}{32}) := \delta_{k-1}\, c_k$ $(2 \leq k \leq m)$ and $p_k := \delta_k h_k^2 - 4$. Then the differential operator on grid Ω_k is defined as

$$\mathcal{L}_k u := \nabla^2 u + \delta_k u,$$

for $1 \leq k \leq m$, provided $\sigma = \frac{h_k^2}{8}$. Therefore, the ACR-modified definitions of the matrix of the discrete system on grid Ω_k and the restriction and prolongation operators have stencils

$$L_k \sim \frac{1}{h_k^2}\begin{bmatrix} & 1 & \\ 1 & p_k & 1 \\ & 1 & \end{bmatrix},$$

$$R_k^{k+1} \sim \frac{1}{8}\begin{bmatrix} & 1 & \\ 1 & -p_k & 1 \\ & 1 & \end{bmatrix},$$

$$P_{k+1}^k \sim \frac{1}{2p_k^2}\begin{bmatrix} 4 & -4p_k & 4 \\ -4p_k & 3p_k^2 & -4p_k \\ 4 & -4p_k & 4 \end{bmatrix}$$

respectively. We call this ACR-modified multigrid process CR-MG. Note that the CR-MG restriction operator is similar to the operator naturally suggested by the principle of *total reduction* (see [15] and [16], for example). Further, for Laplace's equation (i.e. $\delta = 0$), $p_k = -4$ and the CR-MG restriction operator corresponds to half weighting.

4 NUMERICAL RESULTS

Consider the complex two-dimensional Dirichlet boundary value problem

$$\nabla^2 u + \delta u = 0 \quad \text{in} \quad \Omega = \Omega_1 \cup \Omega_2 \; : \text{unit square}$$

$$\text{s.t.} \quad u = g \quad \text{on} \quad \partial\Omega,$$

with data

$$\delta = \begin{cases} 30 + 10i & \text{in } \Omega_2 : \frac{3}{8} < x, y < \frac{5}{8} \\ 1 & \text{in } \Omega_1 : \Omega \backslash \overline{\Omega}_2 \end{cases} ,$$

$$g = \begin{cases} \sin(4y - \frac{3}{2})\pi & \text{on } x = 0 \, , \frac{3}{8} < y < \frac{5}{8} \\ 0 & \text{elsewhere on } \partial\Omega \end{cases} .$$

For convenience, we consider a domain Ω consisting of two concentric squares. The value of δ in Ω_2 is a typical value calculated from the data in [1]. In the following experiment we assess the efficiency of the CR-MG algorithm, as described in the preceding section, and compare it with standard MG using full weighting restriction and nine-point prolongation.

The problem is discretised piecewise according to (3) and (4), using central differences on a 65×65 grid. A four-grid method is employed, with standard grid coarsening. This ensures good resolution of the inner subdomain Ω_2 on the coarsest grid. The multigrid schedule used is the V-cycle with two pre-smoothing and two post-smoothing iterations, and LU decomposition with partial pivoting is used to solve the defect equation exactly on the coarsest grid. The initial estimate is taken to be the zero vector and convergence is measured by $\log_{10} ||r||_2$, where r is the residual vector and $||\cdot||_2$ is the usual Euclidean norm.

With convergence set to a tolerance of

$$\log_{10} ||r||_2 < -9,$$

the convergence times of MG and CR-MG with PGS and KACZ smoothers were measured and the results are displayed in Table 1. All convergence times were measured in seconds on a Sun SPARC-

Table 1: CPU Convergence Times

time (s)	PGS	KACZ
MG	22.8	191.6
CR-MG	18.5	155.9

workstation. We immediately notice that both MG and CR-MG converge much more rapidly with a PGS smoother than with a KACZ smoother. This is not unexpected, considering the smoothing properties of these two iterative methods. Further, KACZ is a more computationally intensive smoother than PGS, having a 13-point stencil as compared to the 5-point stencil of PGS.

However, most importantly, we find that with both smoothers the rate of convergence of CR-MG is significantly faster than that of MG. In fact, with both smoothers CR-MG provides a 19 percent saving in CPU time over MG. This is a significant saving, especially for larger problems. The rates of convergence of MG and CR-MG with a PGS smoother are compared graphically in Fig. 3. Both plots are approximately straight lines, a consequence of the grid-independent convergence of the multigrid method.

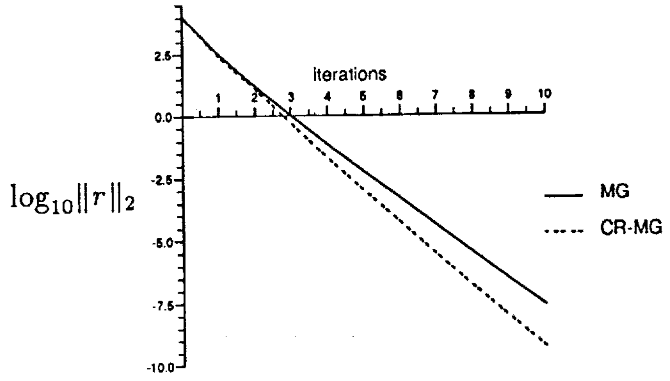


Figure 3: Convergence of MG and CR-MG with a PGS smoother.

5 CONCLUDING REMARKS

In this paper, attention has been focussed on improving the design of the standard multigrid method with respect to a particular problem, namely the complex-valued microwave oven problem. By drawing a comparison with the direct method of cyclic reduction, improved discretisation, restriction and prolongation operators have been designed, resulting in savings of up to 19 percent in CPU time used.

Only two smoothing methods have been considered here, point Gauß-Seidel and Kaczmarz. However, there are many more robust smoothers, such as alternating damped Jacobi, alternating symmetric line Gauß-Seidel and incomplete LU decomposition. These methods, and many more, have been summarised and analysed in detail in [7]. Improvements in the convergence properties of the modified multigrid method (CR-MG) will almost certainly be realised by using such smoothers.

Finally, attention in this paper has been restricted to the microwave oven problem, although the ideas presented here can be extended to other problems. For example, in [11], these ideas were applied to the convection-diffusion equation and it was shown that approximate cyclic reduction can be used to define the ideal quantity of coarse grid artificial viscosity and the direction in which it lies.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr D.F. Mayers, for his help and invaluable advice and Dr A.K. Parrott for reading through a draft of this paper. This work was supported by an SERC Research Studentship, for which I am grateful. I would also like to thank British Nuclear Fuels plc for their Bursary Award.

REFERENCES

[1] Tinga, W.R. and Nelson, S.O.: Dielectric Properties of Materials for Microwave Processing – Tabulated. *J. Micro. Power*, 8(1) : 23-65 (1973)

[2] De Zeeuw, P.M.: Matrix-Dependent Prolongations and Restrictions in a Blackbox Multigrid Solver. *J. Comp. Appl. Math.*, 33 : 1-27 (1990)

[3] Freund, R.W.: Conjugate Gradient Type Methods for Linear Systems with Complex Symmetric Coefficient Matrices. *SIAM J. Sci. Stat. Comput.*, 13 : 425-448 (1992)

[4] Brandt, A.: Multi-Level Adaptive Solutions to Boundary Value Problems. *Math. Comp.*, 31 : 333-390 (1977)

[5] Wesseling, P.: *An Introduction to Multigrid Methods.* John Wiley and Sons, Chichester (1992)

[6] Chan, T.F. and Elman, H.C.: Fourier Analysis of Iterative Methods for Elliptic Boundary Value Problems. *SIAM Rev.*, 31 : 20-49 (1989)

[7] Wesseling, P.: A Survey of Fourier Smoothing Analysis Results. *Int. Series of Num. Maths*, 98 : 105-127 (1991)

[8] Kaczmarz, S.: Angenäherte Auflösung von Systemen Linearer Gleichungen. *Bulletin de l'Academie Polonaise des Sciences et Lettres*, A35 : 355-357 (1937)

[9] Buzbee, B.L., Golub, G.H. and Nielson, C.W.: On Direct Methods for Solving Poisson's Equations. *SIAM J. Numer. Anal.*, 7 : 627-656 (1970)

[10] Sweet, R.A.: A Generalized Cyclic Reduction Algorithm. *SIAM J. Numer. Anal.*, 11 : 506-520 (1974)

[11] Shaw, G.J.: Cyclic Reduction and Multigrid. Mayers, D.F., Rollet J.S. and Shaw, G.J.: *Fast Iterative Solvers*, OUCL Lecture Notes, Oxford University, Oxford (1991)

[12] Swarztrauber, P.N.: Approximate Cyclic Reduction for Solving Poisson's Equation. *SIAM J. Sci. Stat. Comput.*, 8 : 199-209 (1987)

[13] Hackbusch, W.: *Multigrid Methods and Applications.* Computational Mathematics (4), Springer-Verlag, Berlin (1985)

[14] Stüben, K. and Trottenberg, U.: Multigrid Methods : Fundamental Algorithms, Model Problem Analysis and Applications. Hackbusch, W. and Trottenberg, U., eds.: *Multigrid Methods*, Lecture Notes in Mathematics (960), Springer-Verlag, Berlin (1982), pp.1-176

[15] Schröder, J., Trottenberg, U. and Witsch, K.: On Fast Poisson Solvers and Applications. Bulirsch, R., Griegorieff, R.D. and Schröder, J., eds.: *Numerical Treatment of Differential Equations*, Lecture Notes in Mathematics (631), Springer, Berlin (1978), pp.153-187

[16] Ries, M., Trottenberg, U. and Winter, G.: A Note on MGR Methods. *Lin. Alg. Appl.*, 49 : 1-26 (1983)

Uniform Convergence of Multigrid V-Cycle Iterations for Indefinite and Nonsymmetric Problems*

James H. Bramble
Cornell University
Ithaca, NY 14853-7901

Do Y. Kwak
Korea Advanced Institute of Science and Technology
Taejon, Korea 305-701

Joseph E. Pasciak
Brookhaven National Laboratory
Upton, NY 11973

To appear: SIAM Journal of Numerical Analysis
Dedicated to Professor Seymour Parter on the occasion of the sixty-fifth anniversary of his birthday.

ABSTRACT

In this paper, we present an analysis of a multigrid method for nonsymmetric and/or indefinite elliptic problems. In this multigrid method various types of smoothers may be used. One type of smoother which we consider is defined in terms of an associated symmetric problem and includes point and line, Jacobi and Gauss-Seidel iterations. We also study smoothers based entirely on the original operator. One is based on the normal form, that is, the product of the operator and its transpose. Other smoothers studied include point and line, Jacobi and Gauss-Seidel. We show that the uniform estimates of (ref. 6) for symmetric positive definite problems carry over to these algorithms. More precisely, the multigrid iteration for the nonsymmetric and/or indefinite problem is shown to converge at a uniform rate provided that the coarsest grid in the multilevel iteration is sufficiently fine (but not depending on the number of multigrid levels).

1. INTRODUCTION

The purpose of this paper is to study certain multigrid methods for second order elliptic boundary value problems including problems which may be nonsymmetric and/or indefinite. Multigrid methods are among the most efficient methods available for solving the discrete equations associated with approximate solutions of elliptic partial differential equations. Since their introduction by Fedorenko (ref. 15), there has been intensive research toward the mathematical understanding of such methods. The reader is referred to (ref. 19), (ref. 17) and (ref. 3) and the bibliographies therein. Most of these works concern symmetric, positive definite elliptic problems although a few consider nonsymmetric and/or indefinite problems. In particular, (ref. 1),(ref. 18), (ref. 10) and (ref. 24) deal with such multigrid algorithms and are most closely related to the subject of this paper. All of these papers share the requirement that the coarse grid be sufficiently fine. We shall briefly describe their contents.

*This manuscript has been authored under contract number DE-AC02-76CH00016 with the U.S. Department of Energy. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. This work was also supported in part under the National Science Foundation Grant No. DMS-9007185 and by the U.S. Army Research Office through the Mathematical Sciences Institute, Cornell University. The second author was also partially supported by the Korea Science and Engineering Foundation.

The paper by Bank (ref. 1) derives uniform convergence estimates for the W–cycle multigrid iteration with both a standard Jacobi smoother and a smoother which uses the operator times its adjoint. In each case, a sufficient number of smoothings are required and a sufficiently fine coarse grid, depending on the number of smoothings, is needed. Some regularity for the elliptic partial differential equation was also required.

Mandel studied the V–cycle iteration and showed that it was effective with only one smoothing and a sufficiently fine coarse grid. His result requires that the underlying partial differential equation satisfies the "full elliptic regularity" hypothesis and generalizes the results of Braess and Hackbusch (ref. 2) for the symmetric positive definite problem.

Bramble, Pasciak and Xu (ref. 10) studied the symmetric smoother introduced by Bank and showed that the W–cycle and variable V–cycle worked without making the undesirable requirement of "sufficiently many smoothings". Somewhat more than minimal regularity was needed.

In (ref. 24), Wang showed that, for the standard V–cycle with one smoothing, the "reduction factor" for the iteration error was bounded by $1 - C/J + C_1 h_1$ where J is the number of levels, h_1 is the size of the coarsest grid and C and C_1 are constants. This estimate deteriorates with the number of levels and will be less than one only if the coarse grid is subsequently finer as the number of levels increases. Minimal elliptic regularity was assumed.

In this paper uniform iterative convergence estimates for V-cycle multigrid methods applied to nonsymmetric and/or indefinite problems are proved under rather weak assumptions (e.g., the domain need not be convex). Uniform estimates were shown to hold in (ref. 6) and (ref. 8) for the V–cycle with one smoothing step in the symmetric positive definite case under such hypotheses. We show that these results carry over to the nonsymmetric and/or indefinite case for a variety of smoothers. The coarse grid must be fine enough but need not depend on the number of levels J. Such a condition seems unavoidable since, in many cases, it is needed even for the approximate problem to make sense.

In recent years, some other techniques have been proposed to handle the nonsymmetric indefinite case. One approach in (ref. 14), (ref. 4) and (ref. 7) is to precondition with a symmetric operator and then solve certain normal equations by the conjugate gradient method. One possible advantage of such a method is that some nonsymmetric problems which are not "compact perturbations" of symmetric ones may be treated. Of course, the usual normal equations may be formed and then preconditioned (cf. (ref. 7) and (ref. 20)); this approach seems to be rather restrictive in that good preconditioners may be difficult to construct. Other recent approaches have included Schwarz type methods (ref. 12) and two–level methods in which a "coarse space" is introduced to reduce the problem to one with a positive definite symmetric part (cf. (ref. 4), (ref. 13) and (ref. 25)).

The remainder of the paper is organized as follows: In Section 2, we describe a model problem and introduce the multigrid method. In Section 3, smoothers based on the symmetric problem (and used in our nonsymmetric and/or indefinite applications) are defined and the relevant properties which they satisfy are stated. Section 4 develops smoothers based on the original problem. The main results of the paper, which provide iterative convergence rates for the multigrid algorithms with the smoothers of Sections 3 and 4, are given in Section 5.

2. THE PROBLEM AND MULTIGRID ALGORITHM.

We set up the model nonsymmetric problem and the simplest multigrid algorithm in this section. We consider, for simplicity, the Dirichlet problem in two spatial dimensions approximated by piecewise linear finite elements on a quasi-uniform mesh. The multigrid convergence results hold for many extensions and generalizations as discussed at the end of Section 5.

We consider as our model problem the following second order elliptic equation with homogeneous boundary conditions.

$$-\sum_{i,j=1}^{2} \frac{\partial}{\partial x_j}(a_{ij}\frac{\partial u}{\partial x_i}) + \sum_{i=1}^{2} b_i \frac{\partial u}{\partial x_i} + au = f \quad \text{in} \quad \Omega, \qquad u = 0 \quad \text{on} \quad \partial\Omega, \tag{2.1}$$

where Ω is a polygonal domain (possibly nonconvex) in R^2 and $\{a_{ij}(x)\}$ is bounded symmetric, and uniformly positive definite for $x \in \Omega$. We assume that a_{ij} is in the Sobolev space $W_p^\gamma(\Omega)$ for $p > 2/\gamma$ (see, (ref. 16) for the definition of $W_p^\gamma(\Omega)$). Further, we assume that b_i is continuously differentiable on $\bar{\Omega}$ and that $|a|$ is bounded. Finally, we assume that the solution of (2.1) exists.

Let $H^1(\Omega)$ denote the Sobolev space of order one on Ω (cf., (ref. 16)) and let $H_0^1(\Omega)$ denote those functions in $H^1(\Omega)$ whose trace vanish on $\partial\Omega$. For $v, w \in H_0^1(\Omega)$, define

$$A(v,w) = \sum_{i,j=1}^{2} \int_\Omega a_{ij} \frac{\partial v}{\partial x_i}\frac{\partial w}{\partial x_j}\, dx + \sum_{i=1}^{2} \int_\Omega b_i \frac{\partial v}{\partial x_i} w\, dx + \int_\Omega avw\, dx. \tag{2.2}$$

The solution u of (2.1) satisfies

$$A(u,v) = (f,v) \qquad \text{for all } v \in H_0^1(\Omega), \tag{2.3}$$

where $(\cdot,\cdot)$ denotes the inner product in $L^2(\Omega)$.

For the analysis, we introduce a symmetric positive definite form $\hat{A}(\cdot,\cdot)$ which has the same second order part as $A(\cdot,\cdot)$. We define $\hat{A}(\cdot,\cdot)$ by

$$\hat{A}(u,v) = \sum_{i,j=1}^{2} \int_\Omega a_{ij} \frac{\partial u}{\partial x_i}\frac{\partial v}{\partial x_j}\, dx + \int_\Omega uv\, dx.$$

The difference is denoted by

$$D(u,v) = A(u,v) - \hat{A}(u,v).$$

The form $D(\cdot,\cdot)$ satisfies the inequalities

$$|D(u,v)| \le C\|u\|_1 \|v\| \text{ and } |D(u,v)| \le C\|u\|\,\|v\|_1. \tag{2.4}$$

Here $\|\cdot\|_1$ and $\|\cdot\|$ denote the norms in $H^1(\Omega)$ and $L^2(\Omega)$ respectively. The second inequality above follows from integration by parts. Here and throughout the paper, c or C, with or without subscript, will denote a generic positive constant. These constants can take on different values in different occurrences but will always be independent of the mesh size and the number of levels in multigrid algorithms.

By the assumptions on the coefficients appearing in the definition of $\hat{A}(\cdot,\cdot)$, it follows that the norm $\hat{A}(v,v)^{1/2}$ for $v \in H^1(\Omega)$ is equivalent to the norm on $H^1(\Omega)$. Thus, we take

$$\|v\|_1 = \hat{A}(v,v)^{1/2}.$$

We develop a sequence of nested triangulations of Ω in the usual way. We assume that a coarse triangulation $\{\tau_1^i\}$ of Ω is given. Successively finer triangulations $\{\tau_m^i\}$ for $m > 1$ are defined by subdividing each triangle (in a coarser triangulation) into four by connecting the midpoints of the edges. The mesh size of $\{\tau_1^i\}$ will be denoted to be d_1 and can be taken to be the diameter of the largest triangle. By similarity, the mesh size of $\{\tau_m^i\}$ is $2^{1-m}d_1$.

For theoretical and practical purposes, the coarsest grid in the multilevel algorithms must be sufficiently fine. In practice, however, the coarse grid is still considerably coarser than the solution grid. Let L and J be greater than or equal to one and set M_k, for $k = 1, \ldots, J$, to be the functions which are piecewise linear with respect to the triangulation $\{\tau^i_{k+L}\}$, continuous on Ω and vanish on $\partial\Omega$. Since the triangulations are nested, it follows that

$$M_1 \subset M_2 \subset \ldots \subset M_J.$$

The space M_k has a mesh size of $h_k = 2^{1-L-k} d_1 = 2^{1-k} h_1$.

Fix k in $\{1, 2, \ldots\}$. Let us temporarily assume that for every $u \in M_k$,

$$A(u, v) = 0 \quad \text{for all } v \in M_k \quad \text{implies} \quad u = 0. \tag{2.5}$$

This assumption immediately implies the existence and uniqueness of solutions to problems of the form: Given a linear functional $F(\cdot)$ defined on M_k, find $u \in M_k$ satisfying

$$A(u, \phi) = F(\phi) \qquad \text{for all } \phi \in M_k.$$

In particular, the projection operator $P_k : H^1(\Omega) \mapsto M_k$ satisfying

$$A(P_k u, v) = A(u, v) \quad \text{for all } v \in M_k,$$

is well defined.

Clearly, if (2.2) has a positive definite symmetric part then (2.5) holds. More generally, if solutions of (2.1) satisfy regularity estimates of the form

$$\|u\|_{1+\alpha} \leq C\|f\|_{-1+\alpha}, \tag{2.6}$$

then, it is well known (cf., (ref. 22)) that there exists a constant h_0 such that for $h_k \leq h_0$, (2.5) holds and furthermore

$$\|(I - P_k)u\| \leq c h_k^\alpha \|(I - P_k)u\|_1. \tag{2.7}$$

and finally,

$$\|P_k u\|_1 \leq C\|u\|_1. \tag{2.8}$$

Even if regularity estimates of the form of (2.6) are not known to hold, then (2.5) is known from a recent result by Schatz and Wang (ref. 23).

Lemma 2.1 (ref. 23). *There exists an h_0 such that (2.5) holds for $h_k \leq h_0$. Moreover, given $\epsilon > 0$, there exists an $h_0(\epsilon) > 0$ such that for all $h_k \in (0, h_0]$, (2.8) holds and*

$$\|(I - P_k)u\| \leq \epsilon\|(I - P_k)u\|_1. \tag{2.9}$$

Remark 2.1. The above ϵ will appear in our subsequent analysis. We note that ϵ can be taken arbitrarily small. However, L will be taken large enough so that (2.5), (2.8) and (2.9) hold. Thus, the coarse grid size (i.e., L) for any estimate in which ϵ appears will depend on ϵ.

In our analysis, we shall use the orthogonal projectors $\hat{P}_k : H^1_0(\Omega) \mapsto M_k$ and $Q_k : L^2(\Omega) \mapsto M_k$ which, respectively, denote the elliptic projection corresponding to $\hat{A}(\cdot,\cdot)$ and the $L^2(\Omega)$ projection. These are defined by

$$\hat{A}(\hat{P}_k u, v) = \hat{A}(u, v) \quad \text{for all } v \in M_k,$$

and

$$(Q_k u, v) = (u, v) \quad \text{for all } v \in M_k.$$

The multigrid algorithms will be defined in terms of an additional inner product $(\cdot,\cdot)_k$ on $M_k M_k$. Examples of this inner product in our applications will be given in the next section. Additional operators are defined in terms of this inner product as follows: For each k, define $A_k : M_k \rightarrow M_k$ and $\hat{A}_k : M_k \rightarrow M_k$ by

$$(A_k u, v)_k = A(u, v) \quad \text{for all } v \in M_k,$$

and

$$(\hat{A}_k u, v)_k = \hat{A}(u, v) \quad \text{for all } v \in M_k.$$

Finally, the restriction operator $P^0_{k-1} : M_k \mapsto M_{k-1}$ is defined by

$$(P^0_{k-1} u, v)_{k-1} = (u, v)_k \qquad \text{for all } v \in M_{k-1}.$$

We seek the solution of

$$A(u, v) = (f, v), \quad \text{for all } v \in M_J. \tag{2.10}$$

This can be rewritten in the above notation as

$$A_J u = Q_J f. \tag{2.11}$$

We describe the simplest V–cycle multigrid algorithm for iteratively computing the solution u of (2.3). Given an initial iterate $u_0 \in M_J$, we define a sequence approximating u by

$$u_{i+1} = \mathrm{Mg}_J(u_i, Q_J f). \tag{2.12}$$

Here $\mathrm{Mg}_J(\cdot,\cdot)$ is a map of $M_J M_J$ into M_J and is defined as follows.

Definition MG. *Set $Mg_1(v, w) = A_1^{-1} w$. Let $k > 1$ and v, w be in M_k. Assuming that $Mg_{k-1}(\cdot,\cdot)$ has been defined, we define $Mg_k(v, w)$ by:*

(1) $x_k = v + R_k(w - A_k v)$.

(2) $Mg_k(v, w) = x_k + q$, *where q is defined by*

$$q = Mg_{k-1}(0, P^0_{k-1}(w - A_k x_k)).$$

Here $R_k : M_k \mapsto M_k$ is a linear smoothing operator. Note that in this V–cycle, we smooth only as we proceed to coarser grids.

In Section 3, we define R_k in terms of smoothing operators defined for the form $\hat{A}(\cdot,\cdot)$. Specifically, the smoothing procedure for the symmetric problem will be denoted $\hat{R}_k : M_k \mapsto M_k$ and we set $R_k = \hat{R}_k$. In Section 4, we consider smoothers which are directly defined in terms of the original operator A_k.

A straightforward mathematical induction argument shows that $\mathrm{Mg}_J(\cdot,\cdot)$ is a linear map from $M_J M_J$ into M_J. Moreover, the scheme is consistent in the sense that $v = \mathrm{Mg}_J(v, A_J v)$ for all $v \in M_J$. It easily follows that the linear operator $E = \mathrm{Mg}_J(\cdot, 0)$ is the error reduction operator for (2.12), that is

$$u - u_{i+1} = E(u - u_i).$$

Let $T_k = R_k A_k P_k$ for $k > 1$ and set $T_1 = P_1$. Using the facts that $P_{k-1}^0 A_k = A_{k-1}P_{k-1}$ and $P_{k-1}P_k = P_{k-1}$ and Definition MG, a straightforward manipulation gives that for $k > 1$ and any $u \in M_J$,

$$u - \mathrm{Mg}_k(0, A_k P_k u) = (I - T_k)u - \mathrm{Mg}_{k-1}(0, A_{k-1}P_{k-1}(I - T_k)u).$$

Let $E_k u = u - \mathrm{Mg}_k(0, A_k P_k u)$. In terms of E_k, the above identity is the same as

$$E_k = E_{k-1}(I - T_k).$$

Moreover, by consistency, $E = E_J$ and hence

$$E = (I - T_1)(I - T_2)\cdots(I - T_J). \tag{2.13}$$

The product representation of the error operator given above will be a fundamental ingredient in the convergence analysis presented in Section 4. Similar representations in the case of multigrid algorithms for symmetric problems were given in (ref. 9).

The above algorithm is a special case of more general multigrid algorithms in that we only use pre-smoothing. Alternatively, we could define an algorithm with just post-smoothing or both pre- and post-smoothing. The analysis of these algorithms is similar to that above and will not be presented.

Often algorithms with more than one smoothing are considered (ref. 3), (ref. 17), (ref. 19). This is not advised in the above algorithm since the smoothing iteration is generally unstable.

3. SMOOTHERS BASED ON THE SYMMETRIC PROBLEM.

In this section, we consider smoothers which are based on the symmetric problem. The symmetric smoother will be denoted by $\hat{R}_k$. We state a number of abstract conditions concerning these smoothing operators. We then give three examples of smoothing procedures which satisfy these assumptions. In Section 5, we provide convergence estimates for multigrid algorithms with $R_k = \hat{R}_k$ in Definition MG.

The first two conditions are standard assumptions used in earlier multigrid analyses. For $k > 1$, let $\hat{K}_k = I - \hat{R}_k \hat{A}_k$ (defined on M_k) and $\hat{T}_k = \hat{R}_k \hat{A}_k P_k$ (defined on M_J). We assume that:

(1) There is a constant C_R such that

$$\frac{(u,u)_k}{\lambda_k} \le C_R(\bar{R}_k u, u)_k, \quad \text{for all } u \in M_k, \tag{C.1}$$

where $\bar{R}_k = (I - \hat{K}_k^* \hat{K}_k)\hat{A}_k^{-1}$ and λ_k is the largest eigenvalue of $\hat{A}_k$. Here and in the remainder of this paper, $*$ denotes the adjoint with respect to the inner product $\hat{A}(\cdot,\cdot)$.

(2) There is a constant $\theta < 2$ not depending on k satisfying

$$\hat{A}(\hat{T}_k v, \hat{T}_k v) \le \theta \hat{A}(\hat{T}_k v, v) \qquad \text{for all } v \in M_k. \tag{C.2}$$

Provided that (C.2) holds, (C.1) is equivalent to

$$\frac{(u,u)_k}{\lambda_k} \le C(\hat{R}_k u, u)_k, \quad \text{for all } u \in M_k. \tag{3.1}$$

When $\hat{R}_k$ is symmetric with respect to $(\cdot,\cdot)_k$, (C.2) states that the norm of $\hat{T}_k$ is less than or equal to θ. Even in the case of non-symmetric $\hat{R}_k$, (C.2) implies stability of $(I - \hat{T}_k)$. In fact, for any $w \in M_J$, (C.2) implies that

$$\begin{aligned}\hat{A}((I - \hat{T}_k)w, (I - \hat{T}_k)w) &= \hat{A}(w,w) - 2\hat{A}(\hat{T}_k w, w) + \hat{A}(\hat{T}_k w, \hat{T}_k w)\\ &\le \hat{A}(w,w) - (2-\theta)\hat{A}(\hat{T}_k w, w) \le \hat{A}(w,w).\end{aligned} \tag{3.2}$$

The final condition is that for $k > 1$, there exists a constant C satisfying

$$(C.3) \qquad (\hat{T}_k u, \hat{T}_k u)_k \le C\lambda_k^{-1}\hat{A}(\hat{T}_k u, u) \qquad \text{for all } u \in M_k.$$

A simple change of variable shows that (C.3) is the same as

$$(\hat{R}_k v, \hat{R}_k v)_k \le C\lambda_k^{-1}(\hat{R}_k v, v)_k \qquad \text{for all } v \in M_k.$$

In the case when $\hat{R}_k$ is symmetric, this is equivalent to

$$(3.3) \qquad (\hat{R}_k v, v)_k \le C\lambda_k^{-1}(v, v)_k \qquad \text{for all } v \in M_k$$

and is the opposite inequality of (3.1). Note that both (C.2) and (C.3) hold on M_J.

Remark 3.1. If Conditions (C.1)–(C.3) hold for a smoother R_k then they hold for its adjoint R_k^t with respect to the inner product $(\cdot,\cdot)_k$. This means that (C.1) holds for $\bar{R}_k = (I - \hat{K}_k\hat{K}_k^*)\hat{A}_k^{-1}$ and that (C.2) and (C.3) hold with $\hat{T}_k^*$ replacing $\hat{T}_k$. In the case of (C.2) and (C.3), the corresponding inequalities hold with the same constants as those appearing in the original inequalities.

Example 1. The first example of a smoother is the operator

$$\hat{R}_k = \bar{\lambda}_k^{-1} I$$

where I denotes the identity operator on M_k and $\lambda_k \le \bar{\lambda}_k \le C\lambda_k$. In this case, (3.1) holds with $C = \bar{\lambda}_k/\lambda_k$, (C.2) holds with $\theta = 1$ and (3.3) holds with $C = \lambda_k/\bar{\lambda}_k$. To avoid the inversion of L^2 Gram matrices in the multigrid algorithm, we use the inner product

$$(3.4) \qquad (u, v)_k = h_k^2 \sum_i u(x_i)v(x_i).$$

Here the sum is taken over all nodes x_i of the subspace M_k. Note that $(\cdot,\cdot)_k$ is uniformly (independent of k) equivalent to $(\cdot,\cdot)$ on M_k.

The remaining smoothers correspond to Jacobi and Gauss-Seidel, point and line iteration methods. We shall present these smoothers in terms of subspace decompositions. Specifically, we write

$$(3.5) \qquad M_k = \sum_{i=1}^{l} M_k^i$$

where M_k^i is the one dimensional subspace spanned by the nodal basis function ϕ_k^i or the subspace spanned by the nodal basis functions along a line. The number of such spaces $l = l(k)$ will often depend on k. These spaces satisfy the following inequality.

$$(3.6) \qquad \|v\| \le Ch_k \|v\|_1 \qquad \text{for all } v \in M_k^i.$$

Example 2. For the second example, we consider the additive smoother defined by

$$(3.7) \qquad \hat{R}_k = \gamma \sum_{i=1}^{l} \hat{A}_{k,i}^{-1} Q_{k,i}.$$

Here $\hat{A}_{k,i} : M_k^i \to M_k^i$ is the defined by

$$(\hat{A}_{k,i} v, \chi)_k = \hat{A}(v, \chi) \quad \text{for all } \chi \in M_k^i$$

and $Q_{k,i} : M_k \to M_k^i$ is the projection onto M_k^i with respect to the inner product $(\cdot,\cdot)_k$. The constant γ is a scaling factor which is chosen to ensure that (C.2) is satisfied (see, e.g., (ref. 11),(ref. 5)). Note that $\hat{R}_k$ is symmetric with respect to the inner product $(\cdot,\cdot)_k$. In addition, (3.1) and (3.3) are shown to hold in (ref. 11) with point Jacobi. When the subspaces M_k^i are defined in terms of lines, (3.1) was proved in (ref. 5). The estimate (3.3) easily follows in the line case using the support properties of the basis functions and (3.6). For this example, we take $(\cdot,\cdot)_k = (\cdot,\cdot)$ for all k.

Example 3. We next consider the multiplicative smoother. Given $f \in M_k$, we define $\hat{R}_k$ by

(1) Set $v_0 = 0 \in M_k$.
(2) Define v_i, for $i = 1,\ldots,l$, by
$$v_i = v_{i-1} + \hat{A}_{k,i}^{-1} Q_{k,i}(f - \hat{A}_k v_{i-1}).$$
(3) Set $\hat{R}_k f = v_l$.

Conditions (C.1) and (C.2) are known for this operator (see, e.g., (ref. 5)). The next lemma shows that (C.3) holds for this choice of $\hat{R}_k$. For this case, we also take $(\cdot,\cdot)_k = (\cdot,\cdot)$ for all k.

Lemma 3.1. *(C.3) holds when $\hat{R}_k$ is defined to be the multiplicative smoother of Example 3.*

Proof. The proof uses the techniques for analyzing smoothers presented in (ref. 5). Fix $k > 1$ and let

$$\hat{\mathcal{E}}_i = (I - \hat{P}_k^i)(I - \hat{P}_k^{i-1}) \cdots (I - \hat{P}_k^1) \tag{3.8}$$

where $\hat{P}_k^i$ denotes the $\hat{A}(\cdot,\cdot)$ projection onto M_k^i and $\hat{\mathcal{E}}_0 = I$. Note that $(I - \hat{T}_k) = \hat{\mathcal{E}}_l$ and $\hat{\mathcal{E}}_{i-1} = \hat{\mathcal{E}}_i + \hat{P}_k^i \hat{\mathcal{E}}_{i-1}$. Hence

$$\hat{T}_k = I - \hat{\mathcal{E}}_l = \sum_{i=1}^{l} \hat{P}_k^i \hat{\mathcal{E}}_{i-1}$$

and for every $u \in M_k$, (cf., (ref. 5))

$$\begin{aligned}\hat{A}((2I - \hat{T}_k)u, \hat{T}_k u) &= \hat{A}(u,u) - A(\hat{\mathcal{E}}_l u, \hat{\mathcal{E}}_l u) \\ &= \sum_{i=1}^{l} \hat{A}(\hat{P}_k^i \hat{\mathcal{E}}_{i-1} u, \hat{\mathcal{E}}_{i-1} u).\end{aligned}$$

Since $h_k^2 \le c\lambda_k^{-1}$, the proof of the lemma will be complete if we can show that

$$(\hat{T}_k u, \hat{T}_k u) \le c h_k^2 \sum_{i=1}^{l} \hat{A}(\hat{P}_k^i \hat{\mathcal{E}}_{i-1} u, \hat{\mathcal{E}}_{i-1} u). \tag{3.9}$$

Expanding the left hand side of (3.9) gives

$$(\hat{T}_k u, \hat{T}_k u) = \sum_{i=1}^{l} \sum_{j=1}^{l} (\hat{P}_k^i \hat{\mathcal{E}}_{i-1} u, \hat{P}_k^j \hat{\mathcal{E}}_{j-1} u). \tag{3.10}$$

Because of the support properties of $\{\phi_k^i\}$, the subspaces $\{M_k^i\}$ satisfy a limited interaction property in that for every i, the number of subspaces j for which $(v^i, v^j) \ne 0$, with $v^i \in M_k^i$ and $v^j \in M_k^j$ is bounded by a fixed constant n_0 not depending on k or l. Lemma 3.1 of (ref. 5) implies that the double sum of (3.10) can be bounded by n_0 times its diagonal, i.e.

$$(\hat{T}_k u, \hat{T}_k u) \le n_0 \sum_{i=1}^{l} (\hat{P}_k^i \hat{\mathcal{E}}_{i-1} u, \hat{P}_k^i \hat{\mathcal{E}}_{i-1} u). \tag{3.11}$$

Applying (3.6) gives

$$(\hat{P}_k^i\hat{\mathcal{E}}_{i-1}u, \hat{P}_k^i\hat{\mathcal{E}}_{i-1}u) \leq Ch_k^2\hat{A}(\hat{P}_k^i\hat{\mathcal{E}}_{i-1}u, \hat{\mathcal{E}}_{i-1}u). \tag{3.12}$$

Combining (3.11) and (3.12) proves (3.9). This completes the proof of the lemma.

Remark 3.2. The same analysis could be used for successive overrelaxation type iteration. In that case,

$$\hat{\mathcal{E}}_l = (I - \beta\hat{P}_k^l)(I - \beta\hat{P}_k^{l-1})\cdots(I - \beta\hat{P}_k^1)$$

where $\beta \in (0, 2)$ is the relaxation parameter.

4. SMOOTHERS BASED ON A_k.

In this section, we consider smoothing operators R_k which are defined directly in terms of the nonsymmetric and/or indefinite operator A_k. The first smoother is one that was originally analyzed in (ref. 1) and subsequently studied in (ref. 10).

Example 4. For our first example of a smoother based on A_k, we consider R_k defined by

$$R_k = \bar{\lambda}_k^{-2}A_k^t.$$

Here, A_k^t is the adjoint of A_k with respect to the inner product $(\cdot,\cdot)_k$ and $\bar{\lambda}_k$ is as in Example 1. A possible motivation for such a choice is that, on M_k, the iteration

$$v^i = v^{i-1} + \bar{\lambda}_k^{-2}A_k^t(f - A_kv^{i-1})$$

is stable in the norm $(\cdot,\cdot)_k^{1/2}$ provided that $\bar{\lambda}_k^2$ is greater than or equal to half the largest eigenvalue of $A_k^tA_k$.

Example 5. This example is closely related to the second example of the previous section. As in that example, we define the line or point subspaces $\{M_k^i\}$ for $i = 1, \ldots, l$. Note that the form $A(\cdot,\cdot)$ satisfies a Gårding inequality

$$c_1\hat{A}(u,u) - c\,\|u\|^2 \leq A(u,u) \qquad \text{for all } u \in H_0^1(\Omega).$$

Consequently, by (3.6),

$$(c_1 - Ch_k^2)\hat{A}(u,u) \leq A(u,u) \qquad \text{for all } u \in M_k^i.$$

We will assume that h_2 is sufficiently small so that

$$Ch_2^2 \leq c_1/2. \tag{4.1}$$

This means that $A(\cdot,\cdot)$ restricted to M_k^i has a positive definite symmetric part. Hence, the projector $P_k^i : M_k \mapsto M_k^i$ satisfying

$$A(P_k^iv, w) = A(v, w) \qquad \text{for all } w \in M_k^i$$

is well defined and satisfies

$$\left\|P_k^iu\right\|_1 \leq C\,\|u\|_{1,\Omega_k^i}\,. \tag{4.2}$$

The second norm is taken only over the subdomain Ω_k^i which is the set of points of Ω where the functions in M_k^i are nonzero. In addition, the operator $A_{k,i} : M_k^i \mapsto M_k^i$ defined by

$$(A_{k,i}v, w)_k = A(v, w) \qquad \text{for all } v, w \in M_k^i,$$

is invertible. We set R_k by

$$R_k = \gamma \sum_{i=1}^{l} A_{k,i}^{-1} Q_{k,i}.$$

We choose γ as in Example 2 so that the symmetric smoother defined by (3.7) satisfies (C.2).

Example 6. Our final example is that of Gauss-Seidel directly applied to the nonsymmetric/indefinite equations. We assume that the subspaces $\{M_k^i\}$ satisfy the conditions of the previous example. The block Gauss-Seidel algorithm (based on A_k) is given as follows:

(1) Set $v_0 = 0 \in M_k$.
(2) Define v_i, for $i = 1, \ldots, l$, by
$$v_i = v_{i-1} + A_{k,i}^{-1} Q_{k,i}(f - A_k v_{i-1}).$$
(3) Set $R_k f = v_l$.

5. ANALYSIS OF THE MULTIGRID ITERATION (2.12).

We provide an analysis of the multigrid iteration (2.12) in this section. This analysis is based on the product representation of the error operator (2.13). All of the analysis of this section is based on perturbation from the uniform convergence estimates for multigrid applied to symmetric problems.

We start by stating a result from (ref. 6) estimating the rate of convergence for the multigrid algorithm applied to the symmetric problem. Specifically, we replace A_k by $\hat{A}_k$ and R_k by $\hat{R}_k$ in Definition MG. Set $\hat{T}_1 = \hat{P}_1$. From the earlier discussion, the error operator associated with this iteration applied to finding a solution for the symmetric problem

$$\hat{A}_J u = Q_J f$$

is given by $\hat{E} = \hat{E}_J$ where

$$\hat{E}_k = (I - \hat{T}_1)(I - \hat{T}_2) \cdots (I - \hat{T}_k). \tag{5.1}$$

We then have the following theorem.

Theorem 5.1 (ref. 6). *For $k > 1$, let $\hat{R}_k$ satisfy (C.1) and (C.2). Under the assumptions on the domain Ω and the coefficients of (2.1) given in Section 2, there exists a positive constant $\delta < 1$ not depending on J such that*

$$\hat{A}(\hat{E}_J u, \hat{E}_J u) \le \delta^2 A(u,u) \qquad \text{for all } u \in M_J.$$

To analyze the multigrid algorithms using the smoothers of Section 3, we use the perturbation operator

$$Z_k = T_k - \hat{T}_k.$$

We note that for any $u, v \in M_J$, for $k > 1$,

$$\hat{A}(Z_k u, v) = D(u, \hat{T}_k^* v). \tag{5.2}$$

Indeed, by definition,

$$\begin{aligned}\hat{A}(T_k u, v) &= (T_k u, \hat{A}_k \hat{P}_k v)_k = (A_k P_k u, \hat{R}_k^t \hat{A}_k \hat{P}_k v)_k \\ &= (A_k P_k u, \hat{T}_k^* v)_k = A(P_k u, \hat{T}_k^* v) \\ &= A(u, \hat{T}_k^* v) = \hat{A}(u, \hat{T}_k^* v) + D(u, \hat{T}_k^* v).\end{aligned}$$

The equality (5.2) immediately follows.

To handle the case of $k = 1$, we have

$$\hat{A}(Z_1 u, v) = D((I - P_1)u, \hat{P}_1 v). \tag{5.3}$$

In fact, by definition,

$$\begin{aligned}\hat{A}(P_1 u, v) &= \hat{A}(P_1 u, \hat{P}_1 v)\\ &= A(u, \hat{P}_1 v) - D(P_1 u, \hat{P}_1 v)\\ &= \hat{A}(\hat{P}_1 u, v) + D((I - P_1)u, \hat{P}_1 v).\end{aligned}$$

The following theorem provides an estimate for the multigrid algorithm when the smoothers of Section 3 are used.

Theorem 5.2. *Let $R_k = \hat{R}_k$ and assume that (C.1)–(C.3) hold. Given $\epsilon > 0$, there exists an $h_0 > 0$ such that for $h_1 \le h_0$,*

$$\hat{A}(Eu, Eu) \le \delta^2 \hat{A}(u, u) \qquad \text{for all } u \in M_J,$$

for $\delta = \hat{\delta} + c(h_1 + \epsilon)$. Here $\hat{\delta}$ is less than one (independently of J) and is given by Theorem 5.1.

Proof. For an arbitrary operator $\mathcal{O} : M_J \mapsto M_J$, let $||\mathcal{O}||_{\hat{A}}$ denote its operator norm, i.e.,

$$||\mathcal{O}||_{\hat{A}} = \sup_{u,v \in M_J} \frac{\hat{A}(\mathcal{O}u, v)}{\hat{A}(u, u)^{1/2} \hat{A}(v, v)^{1/2}}.$$

Applying (2.4), (2.9) and (2.8) to (5.3) gives

$$|\hat{A}(Z_1 u, v)| \le C\epsilon \, ||(I - P_1)u||_1 \, ||v||_1 \le C\epsilon \, ||u||_1 \, ||v||_1 \, .$$

This means that the operator norm of Z_1 is bounded by $C\epsilon$. Since the operator norm of $(I - \hat{P}_1)$ is less than or equal to one, the triangle inequality implies that the operator norm of $(I - P_1) = (I - \hat{P}_1 - Z_1)$ is bounded by $1 + C\epsilon$.

For $k > 1$, applying (2.4), (C.3), Remark 3.1, and (3.2) to (5.2) gives

$$\begin{aligned}|\hat{A}(Z_k u, v)| &\le c h_k \, ||u||_1 \, \hat{A}(\hat{T}_k v, v)^{1/2}\\ &\le c h_k \, ||u||_1 \, ||v||_1 \, ,\end{aligned}$$

i.e., the operator norm of Z_k is bounded by $c h_k$. Since, by (3.2), the operator norm of $(I - \hat{T}_k)$ is less than or equal to one, the triangle inequality implies that the operator norm of $(I - T_k) = (I - \hat{T}_k - Z_k)$ is less than or equal to $1 + c h_k$. Hence, it follows that

$$||E_k||_{\hat{A}} \le (1 + C\epsilon) \prod_{i=2}^{k} (1 + c h_i) \le C.$$

It is immediate from the definitions that

$$E_k - \hat{E}_k = (E_{k-1} - \hat{E}_{k-1})(I - \hat{T}_k) - E_{k-1} Z_k. \tag{5.4}$$

By (3.2) and the above estimates, for $k > 1$,

$$\begin{aligned}||E_k - \hat{E}_k||_{\hat{A}} &\le ||E_{k-1} - \hat{E}_{k-1}||_{\hat{A}} ||I - \hat{T}_k||_{\hat{A}} + ||E_{k-1}||_{\hat{A}} ||Z_k||_{\hat{A}}\\ &\le ||E_{k-1} - \hat{E}_{k-1}||_{\hat{A}} + C h_k.\end{aligned} \tag{5.5}$$

Repetitively applying (5.5) and using

$$||E_1 - \hat{E}_1||_{\hat{A}} = ||Z_1||_{\hat{A}} \leq C\epsilon$$

gives that

$$||E_J - \hat{E}_J||_{\hat{A}} \leq C\epsilon + C\sum_{k=2}^{J} h_k \leq c(h_1 + \epsilon).$$

The theorem follows from the triangle inequality and Theorem 5.1.

Remark 5.1. Note that ϵ can be made arbitrarily small by taking h_1 small enough. Consequently, Theorem 5.2 shows that the multigrid iteration converges with a rate which is independent of J provided that the coarse grid is fine enough. The coarse grid mesh size can also be taken to be independent of J.

We next consider the case of Example 4. For this example, we consider first the multigrid algorithm for the symmetric problem which uses

(5.6) $$\hat{R}_k = \bar{\lambda}_k^{-2}\hat{A}_k$$

as a smoother. From the discussion in Section 2, the iteration (2.12) with $\hat{R}_k$ (given by (5.6)) and $\hat{A}_k$ replacing, respectively, R_k and A_k in Definition MG, gives rise to the error operator given by (5.1) where, as above, for $k > 1$, $\hat{T}_k = \hat{R}_k\hat{A}_k\hat{P}_k$. The smoother (5.6) does not satisfy (C.1) and so the first step in the analysis of the nonsymmetric and/or indefinite example is to provide a uniform estimate for $\hat{E}_J$ given by (5.1). Such an estimate is provided in the following theorem. Its proof is given in the appendix.

Theorem 5.3. *Let $\hat{E}_J$ be given by (5.1) where $\hat{T}_k = \hat{R}_k\hat{A}_k\hat{P}_k$ and $\hat{R}_k$ is defined by (5.6). Then,*

$$\hat{A}(\hat{E}_J u, \hat{E}_J u) \leq \hat{\delta}^2 A(u,u) \qquad \textit{for all } u \in M_J.$$

Here $\hat{\delta}$ is less that one and independent of J.

We can now prove the convergence estimate for multigrid applied to (2.1) using the smoother of Example 4.

Theorem 5.4. *Let R_k be defined by Example 4. Given $\epsilon > 0$, there exists an $h_0 > 0$ such that for $h_1 \leq h_0$,*

$$\hat{A}(Eu, Eu) \leq \delta^2 \hat{A}(u,u) \qquad \textit{for all } u \in M_J,$$

for $\delta = \hat{\delta} + c(h_1 + \epsilon)$. Here $\hat{\delta}$ is less than one (independently of J) and is given by Theorem 5.3.

Proof. For $k > 1$, we consider the perturbation operator

$$Z_k = T_k - \hat{T}_k = \bar{\lambda}_k^{-2}(A_k^t A_k P_k - \hat{A}_k^2\hat{P}_k).$$

Clearly,

(5.7) $$Z_k = \bar{\lambda}_k^{-2}[A_k^t(A_kP_k - \hat{A}_k\hat{P}_k) + (A_k^t - \hat{A}_k)\hat{A}_k\hat{P}_k].$$

As in (5.2),

$$\bar{\lambda}_k^{-1}\hat{A}((A_kP_k - \hat{A}_k\hat{P}_k)u, v) = \bar{\lambda}_k^{-1}D(u, \hat{A}_k\hat{P}_k v)$$

from which it follows using (2.4) that

$$||\bar{\lambda}_k^{-1}(A_kP_k - \hat{A}_k\hat{P}_k)||_{\hat{A}} \leq ch_k.$$

A similar argument shows that

$$||\bar{\lambda}_k^{-1}(A_k^t - \hat{A}_k)\hat{P}_k||_{\hat{A}} \leq ch_k.$$

It is not difficult to show that

$$\|A_k^t\|_{\hat{A}} \le C\bar{\lambda}_k.$$

Combining the above estimates with (5.7) gives

$$\begin{aligned}\|Z_k\|_{\hat{A}} \le \|\bar{\lambda}_k^{-1}A_k^t\|_{\hat{A}}\|\bar{\lambda}_k^{-1}(A_kP_k - \hat{A}_k\hat{P}_k)\|_{\hat{A}} \\ + \|\bar{\lambda}_k^{-1}(A_k^t - \hat{A}_k)\hat{P}_k\|_{\hat{A}}\|\bar{\lambda}_k^{-1}\hat{A}_k\hat{P}_k\|_{\hat{A}} \le ch_k.\end{aligned}$$

The remainder of the proof is exactly the same as that of Theorem 5.2. This completes the proof of the theorem.

We next consider the case of Example 5. We use perturbation from the multigrid algorithm for $\hat{A}$ which uses the smoother $\hat{R}_k$ defined by Example 2. Theorem 5.1 provides a uniform estimate for the operator norm of $\hat{E}_J$.

Theorem 5.5. *Let R_k be defined by Example 5. Given $\epsilon > 0$, there exists an $h_0 > 0$ such that for $h_1 \le h_0$,*

$$\hat{A}(Eu, Eu) \le \delta^2 \hat{A}(u,u) \qquad \textit{for all } u \in M_J,$$

for $\delta = \hat{\delta} + c(h_1 + \epsilon)$. Here $\hat{\delta}$ is less than one (independently of J) and is given by Theorem 5.1 applied to $\hat{R}_k$ defined in Example 2.

Proof. For this case, the perturbation operator Z_k is given by

$$Z_k = \gamma \sum_{i=1}^{l} (P_k^i - \hat{P}_k^i).$$

As in (5.3),

$$\hat{A}((P_k^i - \hat{P}_k^i)u, v) = D((I - P_k^i)u, \hat{P}_k^i v).$$

Applying (2.4), (3.6) and (4.2) gives

(5.8) $$\hat{A}((P_k^i - \hat{P}_k^i)u, v) \le ch_k \|u\|_{1,\Omega_k^i} \|v\|_{1,\Omega_k^i}$$

and hence

$$\hat{A}(Z_k u, v) \le ch_k \sum_{i=1}^{l} \|u\|_{1,\Omega_k^i} \|v\|_{1,\Omega_k^i}.$$

Using the limited overlap properties of the domains, Ω_k^i gives

$$\|Z_k\|_{\hat{A}} \le ch_k.$$

The remainder of the proof of the theorem is exactly the same as that given in the proof of Theorem 5.2.

We finally consider the case of Example 6. We use perturbation from the multigrid algorithm for $\hat{A}$ which uses the smoother $\hat{R}_k$ defined by Example 3. Theorem 5.1 provides a uniform estimate for the operator norm of $\hat{E}_J$.

Theorem 5.6. *Let R_k be defined by Example 6. Given $\epsilon > 0$, there exists an $h_0 > 0$ such that for $h_1 \le h_0$,*

$$\hat{A}(Eu, Eu) \le \delta^2 \hat{A}(u,u) \qquad \textit{for all } u \in M_J,$$

for $\delta = \hat{\delta} + c(h_1 + \epsilon)$. Here $\hat{\delta}$ is less than one (independently of J) and is given by Theorem 5.1 applied with $\hat{R}_k$ defined as in Example 3.

Proof. The perturbation operator for this example is

$$Z_k = T_k - \hat{T}_k = \hat{\mathcal{E}}_l - \mathcal{E}_l$$

where $\hat{\mathcal{E}}_l$ is given by (3.8) and

$$\mathcal{E}_i = (I - P_k^i)(I - P_k^{i-1}) \cdots (I - P_k^1)$$

with $\mathcal{E}_0 = I$. As in (5.4),

$$\hat{\mathcal{E}}_i - \mathcal{E}_i = (I - \hat{P}_k^i)(\hat{\mathcal{E}}_{i-1} - \mathcal{E}_{i-1}) - (\hat{P}_k^i - P_k^i)\mathcal{E}_{i-1}.$$

Since the last two terms are orthogonal with respect to $\hat{A}(\cdot,\cdot)$ we have that

$$\|(\hat{\mathcal{E}}_i - \mathcal{E}_i)u\|_{\hat{A}}^2 = \|(I - \hat{P}_k^i)(\hat{\mathcal{E}}_{i-1} - \mathcal{E}_{i-1})u\|_{\hat{A}}^2 + \|(\hat{P}_k^i - P_k^i)\mathcal{E}_{i-1}u\|_{\hat{A}}^2.$$

Because of (5.8) and the fact that the operator norm of $(I - \hat{P}_k^i)$ is bounded by one, it follows that

$$\|(\hat{\mathcal{E}}_i - \mathcal{E}_i)u\|_{\hat{A}}^2 \le \|(\hat{\mathcal{E}}_{i-1} - \mathcal{E}_{i-1})u\|_{\hat{A}}^2 + Ch_k^2 \|\mathcal{E}_{i-1}u\|_{1,\Omega_k^i}^2 .$$

Summing over i, since $\hat{\mathcal{E}}_0 = \mathcal{E}_0 = I$, we obtain

$$\|(\hat{\mathcal{E}}_l - \mathcal{E}_l)u\|_{\hat{A}}^2 \le Ch_k^2 \sum_{i=1}^{l} \|\mathcal{E}_{i-1}u\|_{1,\Omega_k^i}^2 . \tag{5.9}$$

We shall show that

$$\sum_{i=1}^{l} \|\mathcal{E}_{i-1}u\|_{1,\Omega_k^i}^2 \le C\|u\|_{\hat{A}}^2. \tag{5.10}$$

By the arithmetic–geometric mean inequality, the definition $\mathcal{E}_i$ and the limited interaction property (see (3.10) and above) it follows that

$$\begin{aligned}
\sum_{i=1}^{l} \|\mathcal{E}_{i-1}u\|_{1,\Omega_k^i}^2 &\le 2\sum_{i=1}^{l} \|u\|_{1,\Omega_k^i}^2 + 2\sum_{i=1}^{l} \|u - \mathcal{E}_{i-1}u\|_{1,\Omega_k^i}^2 \\
&\le C\|u\|_{\hat{A}}^2 + 2\sum_{i=1}^{l} \left\| \sum_{m=1}^{i-1} P_k^m \mathcal{E}_{m-1}u \right\|_{1,\Omega_k^i}^2 \\
&\le C(\|u\|_{\hat{A}}^2 + \sum_{m=1}^{l}\sum_{i=1}^{l} \|P_k^m \mathcal{E}_{m-1}u\|_{1,\Omega_k^i}^2) \\
&\le C(\|u\|_{\hat{A}}^2 + \sum_{m=1}^{l} \|P_k^m \mathcal{E}_{m-1}u\|_{\hat{A}}^2).
\end{aligned} \tag{5.11}$$

In order to estimate the last term on the right of (5.11) we write

$$\begin{aligned}
\|P_k^m \mathcal{E}_{m-1}u\|_{\hat{A}}^2 &= \hat{A}(P_k^m \mathcal{E}_{m-1}u, P_k^m \mathcal{E}_{m-1}u) \\
&= \hat{A}((\mathcal{E}_{m-1} - \mathcal{E}_m)u, (\mathcal{E}_{m-1} - \mathcal{E}_m)u) \\
&= \hat{A}((\mathcal{E}_{m-1} - \mathcal{E}_m)u, (\mathcal{E}_{m-1} + \mathcal{E}_m)u) - 2\hat{A}(P_k^m \mathcal{E}_{m-1}u, \mathcal{E}_m u) \\
&= \hat{A}(\mathcal{E}_{m-1}u, \mathcal{E}_{m-1}u) - \hat{A}(\mathcal{E}_m u, \mathcal{E}_m u) \\
&\quad - 2\hat{A}(P_k^m \mathcal{E}_{m-1}u, (I - P_k^m)\mathcal{E}_{m-1})u).
\end{aligned} \tag{5.12}$$

Now by (5.8)

$$\begin{aligned}
\hat{A}(P_k^m \mathcal{E}_{m-1}u, (I - P_k^m)\mathcal{E}_{m-1}u) &= \hat{A}(P_k^m \mathcal{E}_{m-1}u, (\hat{P}_k^m - P_k^m)\mathcal{E}_{m-1})u) \\
&\le Ch_k \|P_k^m \mathcal{E}_{m-1}u\|_{\hat{A}} \|\mathcal{E}_{m-1}u\|_{1,\Omega_k^m} .
\end{aligned} \tag{5.13}$$

Hence, combining (5.12) and (5.13), we have

$$||P_k^m \mathcal{E}_{m-1}u||_{\hat{A}}^2 \le C[\hat{A}(\mathcal{E}_{m-1}u, \mathcal{E}_{m-1}u) - \hat{A}(\mathcal{E}_m u, \mathcal{E}_m u)] + Ch_k^2 ||\mathcal{E}_{m-1}u||_{1,\Omega_k^m}^2 .$$

Summing over m we conclude that

$$\sum_{m=1}^{\ell} ||P_k^m \mathcal{E}_{m-1}u||_{\hat{A}}^2 \le C||u||_{\hat{A}}^2 + Ch_k^2 \sum_{m=1}^{\ell} ||\mathcal{E}_{m-1}u||_{1,\Omega_k^m}^2 .$$

This together with (5.11) yields (5.10) when h_k is small enough. Finally, we obtain from (5.10) and (5.9) that for $k > 1$,

$$||Z_k||_{\hat{A}} \le Ch_k.$$

The remainder of the proof of this theorem is the same as that of Theorem 5.2.

Remark 5.2. The same analysis could be used for successive overrelaxation type iteration. In that case,

$$\hat{\mathcal{E}}_l = (I - \beta P_k^l)(I - \beta P_k^{l-1}) \cdots (I - \beta P_k^1)$$

where $\beta \in (0, 2)$ is the relaxation parameter.

Remark 5.3. Many extensions and generalizations of the techniques given above are possible. These techniques lead to uniform estimates for multigrid iteration methods for solving nonsymmetric and/or indefinite problems for the following applications.

(1) Approximations using higher order nodal finite element spaces.
(2) Three dimensional problems.
(3) Problems with discontinuous coefficients as discussed in (ref. 6).
(4) More general boundary conditions.
(5) Problems with local mesh refinement as described in (ref. 11).
(6) Finite element approximation of problems on domains with nonpolygonal boundaries as discussed in (ref. 6).

In addition, the perturbation analysis given above can be combined with results for additive multilevel algorithms, for example, Theorem 3.1 of (ref. 6). This leads to new estimates for additive multilevel preconditioning iterations applied to indefinite and nonsymmetric problems. Provided that the coarse grid is sufficiently fine, the operator

$$P = \sum_{k=1}^{J} T_k$$

has a uniformly (independent of J) positive definite symmetric part with respect to the inner product $\hat{A}(\cdot,\cdot)$ and has a uniformly bounded operator norm. These results extend to all of the applications discussed in Remark 5.3.

6. APPENDIX

We provide a proof of Theorem 5.3 in this appendix. We will apply the analysis given in the proof of Theorem 3.2 of (ref. 6). Note that we cannot directly apply Theorem 3.2 of (ref. 6) since the smoother $\hat{R}_k = \hat{\lambda}_k^{-2}\hat{A}_k$ does not satisfy (C.1). We note, however, that Theorem 5.3 will follow from the proof of Theorem 3.2 of (ref. 6) if we show that (C.2) holds as well as (3.5) and (3.6) of (ref. 6) with $\tilde{T}_k$ replaced by $\hat{T}_k$ defined above. Clearly, (C.2) holds with $\theta = 1$. The remaining two inequalities corresponding to (3.5) and (3.6) of (ref. 6) are

(6.1) $$\hat{A}(\hat{T}_k v, v) \le (\tilde{C}\eta^{k-l})^2 \hat{A}(v, v) \qquad \text{for all } v \in M_l,\ l < k$$

and

$$\hat{A}(v,v) \le C\sum_{k=1}^{J} \hat{A}(\hat{T}_k v, v) \qquad \text{for all } v \in M_J. \tag{6.2}$$

Here η is less than one and independent of k and l.

From the definition of $\bar{\lambda}_k$, we obviously have

$$\hat{A}(\hat{T}_k v, v) \le \bar{\lambda}_k^{-1}\hat{A}(\hat{A}_k v, v) = \hat{A}(\tilde{T}_k v, v).$$

As in (ref. ?), we have set $\tilde{T}_k = \bar{\lambda}_k^{-1}\hat{A}_k$. Inequality (6.1) follows from Lemma 4.2 of (ref. ?).

Inequality (6.2) can be rewritten,

$$\hat{A}(u,u) \le C\Big(\hat{A}(\hat{P}_1 u, u) + \sum_{k=2}^{J} \bar{\lambda}_k^{-2}\left\|\hat{A}_k\hat{P}_k u\right\|_1^2\Big). \tag{6.3}$$

To prove this we proceed as follows. Let $u \in M_J$ and $Q_0 = 0$. Then

$$\begin{aligned}\hat{A}(u,u) &= \sum_{k=1}^{J}\hat{A}(u,(Q_k - Q_{k-1})u)\\ &\le \Big(\hat{A}(\hat{P}_1 u,u) + \sum_{k=2}^{J}\bar{\lambda}_k^{-2}\left\|\hat{A}_k\hat{P}_k u\right\|_1^2\Big)^{1/2}\Big(\hat{A}(Q_1u,Q_1u)\\ &\quad + \sum_{k=2}^{J}\bar{\lambda}_k^{2}(\hat{A}_k^{-1}(Q_k-Q_{k-1})u,(Q_k-Q_{k-1})u)_k\Big)^{1/2}\end{aligned} \tag{6.4}$$

Now, for $k > 1$,

$$\begin{aligned}(\hat{A}_k^{-1}&(Q_k - Q_{k-1})u,(Q_k-Q_{k-1})u)_k\\ &= \sup_{\phi\in M_k}\frac{(\hat{A}_k^{-1/2}(Q_k-Q_{k-1})u,\phi)_k^2}{(\phi,\phi)_k}\\ &= \sup_{\psi\in M_k}\frac{((Q_k-Q_{k-1})u,(Q_k-Q_{k-1})\psi)_k^2}{\|\psi\|_1^2}.\end{aligned}$$

By well known approximation properties,

$$((Q_k - Q_{k-1})\psi,(Q_k-Q_{k-1})\psi)_k^{1/2} \le C\,\|(Q_k-Q_{k-1})\psi\| \le Ch_k\,\|\psi\|_1\,.$$

Combining the above estimates gives

$$\begin{aligned}\hat{A}(Q_1u,Q_1u) &+ \sum_{k=2}^{J}\bar{\lambda}_k^{2}(\hat{A}_k^{-1}(Q_k-Q_{k-1})u,(Q_k-Q_{k-1})u)_k\\ &\le C\Big(\hat{A}(Q_1u,Q_1u)+\sum_{k=2}^{J}\bar{\lambda}_k\,\|(Q_k-Q_{k-1})u\|^2\Big)\\ &\le C\hat{A}(u,u).\end{aligned} \tag{6.5}$$

The last inequality of (6.5) is (4.5) of (ref. ?) and also can be found in (ref. ?). Combining (6.4) and (6.5) proves (6.3) and hence completes the proof of the theorem.

References

1. R. Bank, *A comparison of two multilevel iterative methods for nonsymmetric and indefinite elliptic finite element equations*, SIAM J. Numer. Anal. **18** (1981), 724–743.

2. Braess, D. and Hackbusch, W., *A new convergence proof for the multigrid method including the V-cycle*, SIAM J. Numer. Anal. **20** (1983), 967–975.

3. J.H. Bramble, *Multigrid Methods*, Cornell Mathematics Department Lecture Notes, 1992.

4. J.H. Bramble, Z. Leyk, and J.E. Pasciak, *Iterative schemes for non-symmetric and indefinite elliptic boundary value problems*. Math. Comp. (to appear).

5. J.H. Bramble and J.E. Pasciak, *The analysis of smoothers for multigrid algorithms*, Math. Comp. **58** (1992), 467–488.

6. J.H. Bramble and J.E. Pasciak, *New estimates for multigrid algorithms including the V-cycle*, Math. Comp. (to appear).

7. J.H. Bramble and J.E. Pasciak, *Preconditioned iterative methods for nonselfadjoint or indefinite elliptic boundary value problems*, Unification of finite element methods, (Ed. H. Kardestuncer), Elsevier Science Publ. (North-Holland), New York, 1984, pp. 167 – 184.

8. J.H. Bramble and J.E. Pasciak, *Uniform convergence estimates for multigrid V-cycle algorithms with less than full elliptic regularity* (1992), Brookhaven Nat. Lab. #BNL-47892.

9. J.H. Bramble, J.E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp. **57** (1991), 23–45.

10. J.H. Bramble, J.E. Pasciak, and J. Xu, *The analysis of multigrid algorithms for nonsymmetric and indefinite elliptic problems*, Math. Comp. **51** (1988), 389–414.

11. J.H. Bramble, J.E. Pasciak, and J. Xu, *Parallel multilevel preconditioners*, Math. Comp. **55** (1990), 1–22.

12. X.-C. Cai and O. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Stat. Comp. (to appear).

13. X.-C. Cai and J. Xu, *A preconditioned GMRES method for nonsymmetric and indefinite problems*, (Preprint).

14. H.C. Elman, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Yale Univ. Dept. of Comp. Sci. Rep. 229, (1982).

15. Fedorenko, R.P., *The speed of convergence of one iterative process*, USSR Comput. Math. and Math. Phys. (1961), 1092-1096.

16. P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Pitman, Boston, 1985.

17. Hackbusch, W., *Multi-Grid Methods and Applications*, Springer-Verlag. New York, 1985.

18. Mandel, J., *Multigrid convergence for nonsymmetric, indefinite variational problems and one smoothing step*, Proc. Copper Mtn. Conf. Multigrid Methods, vol. 19, Applied Math. Comput., 1986, pp. 201-216.

19. J. Mandel, S. McCormick, and R. Bank. *Variational multigrid theory*, Multigrid Methods, Ed. S. McCormick, SIAM, Philadelphia, Penn., 1987, pp. 131–178.

20. T.A. Manteuffel and S.V. Parter, *Preconditioning and boundary conditions*, SIAM J. Numer. Anal. **27** (1990), 656–694.

21. P. Oswald, *On discrete norm estimates related to multilevel preconditioners in the finite element method*, (Preprint).

22. A.H. Schatz, *An observation concerning Ritz-Galerkin methods with indefinite bilinear forms*, Math. Comp. **28** (1974), 959–962.

23. A.H. Schatz and J. Wang, *New error estimates in finite element methods*, (Preprint).

24. J. Wang, *Convergence analysis of multigrid algorithms for non-selfadjoint and indefinite elliptic problems* (1991), Proceedings of the 5th Copper Mountain Conference on Multigrid Methods.

25. J. Xu, *A new class of iterative methods for nonsymmetric boundary value problems*, (Preprint).

55-64

N94-23678

197137

p. 14

A MULTILEVEL COST-SPACE APPROACH TO SOLVING THE BALANCED LONG TRANSPORTATION PROBLEM*

Kevin J. Cavanaugh
U.S. Coast Guard
Research and Development Center
Groton, CT

Van Emden Henson
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943

SUMMARY

We develop a multilevel scheme for solving the balanced long transportation problem, that is, given a set $\{c_{kj}\}$ of shipping costs from a set of M supply nodes S_k to a set of N demand nodes D_j, we seek to find a set of flows, $\{x_{kj}\}$, that minimizes the total cost $\sum_{k=1}^{M}\sum_{j=1}^{N} x_{kj}c_{kj}$. We require that the problem be balanced, that is, the total demand must equal the total supply. Solution techniques for this problem are well known from optimization and linear programming. We examine this problem, however, in order to develop principles that can then be applied to more intractible problems of optimization.

We develop a multigrid scheme for solving the problem, defining the grids, relaxation, and intergrid operators. Numerical experimentation shows that this line of research may prove fruitful. Further research directions are suggested.

INTRODUCTION

The transportation problem is the simplest of network flow problems. It is posed on a bipartite graph, consisting of a set of M supply nodes, a set of N demand nodes, and a set of arcs connecting them. Each supply node S_i has a fixed amount s_i of a commodity which it can provide. Each demand node D_j has a fixed requirement d_j for that commodity, and for each arc (i, j) connecting supply node S_i to demand node D_j there is an associated cost per unit flow c_{ij}. When the total supply equals the total demand the problem is *balanced.* When $M << N$, the problem is referred to as a *long* transportation problem. Denoting the flow on arc (i, j) by x_{ij}, the transportation problem

*This work was supported in part by Naval Postgraduate School Research Council, Grant No. MA000-MA999/4476-4479

can be expressed

$$\text{Minimize } \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} \quad \text{subject to: } \sum_{j=1}^{N} x_{ij} = s_i, \quad \sum_{i=1}^{M} x_{ij} = d_j, \quad x_{ij} \geq 0.$$

Let b denote an $(M+N)$-vector whose first M entries are the available supplies s_i at nodes S_1 through S_M, and whose last N entries are (negatives of) the required demands d_j at demand nodes D_1 through D_N. Let K be the number of arcs in the problem. Throughout this work we shall assume that every supply node is connected to every demand node, so that $K = MN$. Let the K-vector x be composed of the flow on the arcs from the M supply-nodes to the N demand nodes in some order, and the K-vector c be the cost of shipping on those arcs in the same order. We denote by A the incidence matrix of the graph, so that A has as many rows as there are nodes in the problem, $M+N$, and as many columns as there are arcs (MN). Each column of A is associated with one arc of the problem, and they are arranged in an order that matches the order of the vectors c and x. Each column has exactly two non-zero entries: a $+1$ in the row corresponding to the tail (supply) node S_i of the arc, and a -1 in the row corresponding to the head (demand) node D_j. Each row of A is associated with one of the constraints of the problem [1]. Then the problem may be written in matrix notation as

$$\begin{aligned} \text{Minimize:} \quad & c^T x \\ \text{Subject to:} \quad & Ax = b, \\ & x \geq 0. \end{aligned}$$

A simple example is presented in Figure 1. In the example, there are four supply nodes, having 12, 15, 10, and 7 units of the commodity to deliver. There are three demand nodes, requiring 13, 20, and 11 units of the commodity. We seek to find a flow vector

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{21} & x_{22} & x_{23} & x_{31} & x_{32} & x_{33} & x_{41} & x_{42} & x_{43} \end{pmatrix}^T$$

given that the vector of costs, written in corresponding order, is

$$\begin{pmatrix} 2 & 1 & 5 & 6 & 4 & 3 & 1 & 7 & 4 & 2 & 3 & 4 \end{pmatrix}^T.$$

The algebraic description of this problem is to find x such that $c^T x$ is minimized, subject to the system of constraints

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{21} \\ x_{22} \\ x_{23} \\ x_{31} \\ x_{32} \\ x_{33} \\ x_{41} \\ x_{42} \\ x_{43} \end{pmatrix} = \begin{pmatrix} 12 \\ 15 \\ 10 \\ 7 \\ -13 \\ -20 \\ -11 \end{pmatrix}.$$

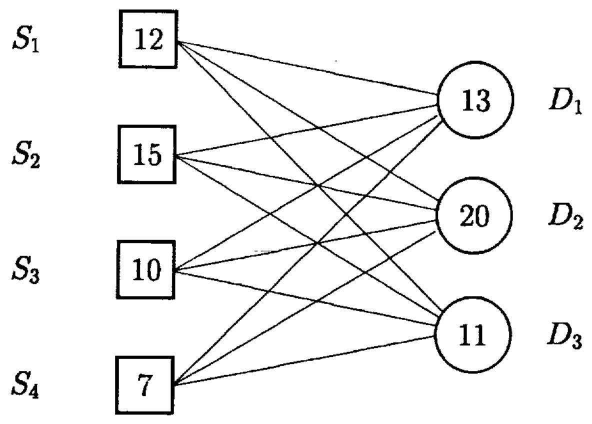


Figure 1: *A simple example of a transportation problem*

Very little work has been done on multigrid methods for discrete optimization problems. Significant studies to date are [2], [3], [4], and [5]. The traditional optimization algorithm which most closely resembles a multilevel algorithm is aggregation/disaggregation [6], [7], and [8], in which nodes are aggregated in a logical way in order to reduce the size of the problem, and the solution to the smaller problem is disaggregated to provide an initial estimate for the solution to the original problem. The most successful work to date, and the work that inspired this study, is that of Kaminsky [4].

COST-SPACE

In [4] it is required that the demand nodes occupy a physical location in space, and that a relationship exist between transportation costs and distances. This is done so that the coarsening step may be performed by aggregating together demand nodes that are physically near one another. For this to make sense, it is necessary that shipment to each of the aggregated demand nodes involve a similar cost, which naturally occurs if the shipping cost is a function of distance. For many applications this makes perfect sense; the cost of shipping a commodity is often directly linked to the distance the commodity must be shipped. This restriction is overly limiting for other types of problems, however. For example, the manpower assignment problem, in which a specified number of jobs must be assigned a given set of workers, can be formulated as a transportation problem. There is no distance involved in such a problem, and cost of assignment is related to other factors, such as the cost of training an individual for a specific task.

In order to address problems that have no geometrical dependence of cost on distance, we employ a change of coordinate systems from physical space to a space we describe as cost space. For the $M \times N$ problem, cost space is the M-dimensional space in which each of the coordinate axes is the cost of shipping from one of M supply nodes. Each of the N demand nodes is placed in cost space at the point whose coordinates are the unit costs of shipping from the supply nodes to it. For example, the three demand nodes in Figure 1 would be placed in a four-dimensional cost space, and would have the coordinates $D_1 = (2, 6, 1, 2)$, $D_2 = (1, 4, 7, 3)$, and $D_3 = (5, 3, 4, 4)$. This change of coordinate systems means shipping cost becomes the metric of the problem, so that two demand nodes are "near" each other only if the shipping costs are similar, and the aggregation of neighboring demand nodes automatically ensures the similarity of their costs.

Posed in cost space, the dimensionality of the problem equals the number of supply nodes. In traditional multigrid methods, one typically uses grids that are tensor products of one-dimensional grids, each having a cardinality of gridpoints that is a power of two. In the cost space approach this would lead to a very rapid growth in the size of the problem; for this reason the cost space approach can be applied only to problems with a relatively small number of supply nodes. This is one reason for restricting our attention to the long transportation problem.

Reduced dimension cost space

If at least one supply node is connected to all demand nodes (and in our work we assume this to be true of all supply nodes) then we can transform the $M \times N$ transportation to an equivalent $(M-1) \times N$ problem, which we call the *reduced dimension* problem. Since we are dealing with the long problem, the transformed problem is somewhat simpler and less expensive to solve. The transformation is accomplished as follows. Suppose that supply node S_l is connected to all demand nodes. Then for each demand node D_j, we subtract c_{lj}, the cost of shipping from supply node S_l to D_j, from all of the shipping costs into demand node D_j. That is, we form an auxilliary cost vector $\tilde{c}_{ij} = c_{ij} - c_{lj}$. The result is that for supply node S_l, all the demand nodes map to the origin in cost space. Effectively S_l has been removed from the problem, leaving an $(M-1) \times N$ problem to be solved. For example, if we use the cost of shipping from S_2 on the example in Figure 1, the transformed cost vector becomes

$$\tilde{c} = \begin{pmatrix} -4 & -3 & 2 & 0 & 0 & 0 & -5 & 3 & 1 & -4 & -1 & 3 \end{pmatrix}^T .$$

We can show that while the objective function value is different for the new problem, a solution for one is equivalent to a solution for the other.

Theorem 1 *Let the $M \times N$ balanced long transportation problem be represented by a bipartite graph G, and suppose that supply node S_l is connected to all demand nodes. Let b be the $(M+N)$ length column vector whose first M entries are the supplies at the supply nodes and whose remaining N entries are the negatives of the demands at the demand nodes. Let A be the adjacency matrix of the graph G; that is, for each arc (i,j) we have $A(i,(i-1)N+j) = 1$ and $A(M+j,(i-1)N+j) = -1$. Let c be the $(M+N)$ length vector whose $k = (i-1)N+j$ element is the cost c_{ij} of shipping from node S_i to node D_j along arc (i,j). Define $\tilde{c}$ to be the vector whose k^{th} entry is $\tilde{c}_k = c_{ij} - c_{lj}$. Then x^* is a solution to the problem*

$$\begin{array}{rl} \textit{Minimize:} & c^T x \\ \textit{Subject to:} & Ax = b, \\ & x \geq 0 , \end{array}$$

if and only if it is a solution to the problem:

$$\begin{array}{rl} \textit{Minimize:} & \tilde{c}^T x \\ \textit{Subject to:} & Ax = b, \\ & x \geq 0 . \end{array}$$

Proof:

$$\begin{aligned}
\tilde{c}^T x &= \sum_{k=1}^{M}\sum_{j=1}^{N} \tilde{c}_{kj}x_{kj} = \sum_{k=1}^{M}\sum_{j=1}^{N}(c_{kj}-c_{lj})x_{kj} \\
&= \sum_{k=1}^{M}\sum_{j=1}^{N}(c_{kj}x_{kj}-c_{lj}x_{kj}) \\
&= \sum_{k=1}^{M}\sum_{j=1}^{N} c_{kj}x_{kj} - \sum_{k=1}^{M}\sum_{j=1}^{N} c_{lj}x_{kj} \\
&= c^T x - \sum_{j=1}^{N} c_{lj}\sum_{k=1}^{M} x_{kj} = c^T x - \sum_{j=1}^{N} c_{lj}d_j,
\end{aligned}$$

since $\sum_{k=1}^{M} x_{kj} = d_j$ in the balanced problem. But $\sum_{j=1}^{N} c_{lj}d_j$ does not depend on x, and therefore $\tilde{c}^T x$ achieves its extreme values precisely when $c^T x$ does. ∎

This transformation of the costs to reduced-dimension space maps all costs of shipping from S_l to the origin in cost space. As will be shown in the next section, our algorithm requires that the demand nodes be sorted once according to the cost of shipping. Since sorting is a fairly expensive operation, the savings generated by reducing the dimension of the problem are tangible. Once the transformation to reduced-dimension cost-space has been performed, the resulting problem may be solved with no further consideration of the transformation. Therefore, in the remainder of this work it is assumed that when an $M \times N$ problem is to be solved, it may be the reduced dimension version of a problem that was originally $(M+1) \times N$.

A MULTIGRID APPROACH TO THE TRANSPORTATION PROBLEM

Following traditional multigrid design approaches, we develop the necessary tools to devise a multigrid V-cycle, which we will combine with a nested iteration to create an *FMG* algorithm. In particular, it is necessary to devise restriction and prolongation methods, some form of local relaxation, and to weave them into an algorithm.

Restriction

To devise a restriction algorithm, it is first necessary to define a coarse grid. We use an approach in which each gridpoint on the coarse grid is a demand node for the coarse grid problem, and represents a pair of demand nodes on the fine grid. This is accomplished as follows. The demand nodes are first sorted by increasing cost of shipping from S_1, and divided into two groups about the median of the sorted cost. This procedure results in two groups of demand nodes, one with a lower cost of shipping from S_1, and one for which shipping from S_1 is more expensive. Each of these groups are then sorted according to increasing cost of shipping from S_2 and divided into two groups about the median cost. This results in four groups, one for which shipping is expensive from both supply nodes, one group for which shipping is inexpensive from both supply nodes, one group for which shipping is expensive from S_2 and inexpensive from S_1, and one group where shipping is expensive from S_1 and inexpensive from S_2.

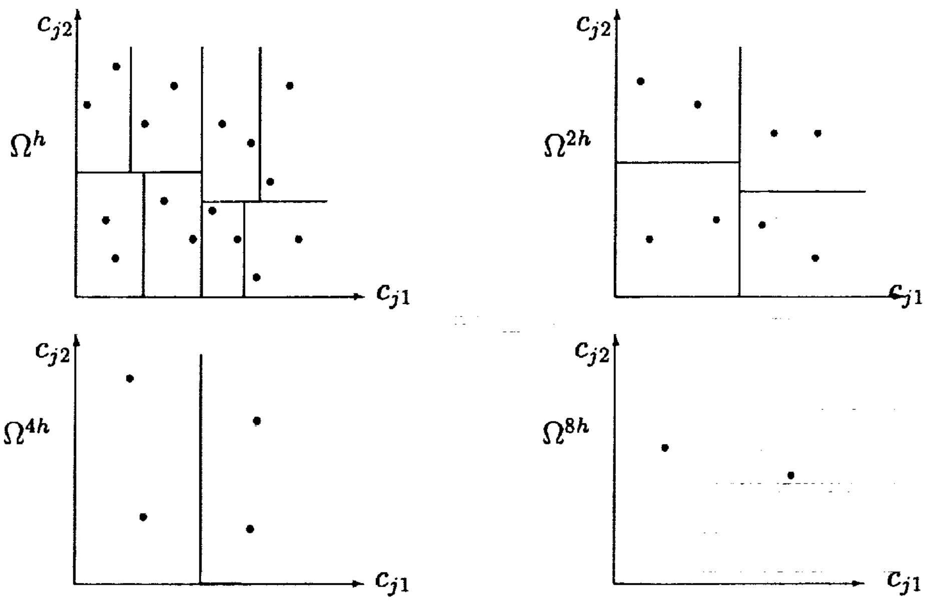


Figure 2: *A simple example of a coarsening process for the transportation problem*

If there are more than two supply nodes in the problem, this process is continued. The four groups are each sorted by cost of shipping from supply node S_3, then divided into smaller groups if necessary and sorted again, according to cost from S_4, etc. If the groups contain more than two nodes after the nodes have been sorted according to cost from all supply nodes, the sorting process begins again with cost from S_1 on each of the groups. Eventually, the nodes will be sorted into pairs that have similar shipping costs from all supply nodes. Each of these pairs of demand nodes is then replaced with a single coarse grid demand node, the collection of which constitutes the first coarse grid.

Further coarsening is accomplished by repeating the procedure described above on the coarse grids to produce still coarser grids. Figure 2 shows a simple example of the coarsening process. If the number of points on the original grid is a power of two, then in the limit a coarsest grid would consist of a single demand node. As in traditional multigrid methods, once the hierarchy of grids is established it is stored, so that the sorting process need never be repeated.

Three quantities must be restricted when aggregating a pair of fine grid demand nodes into a coarse grid demand node: the demands, flows, and costs. Let D_m^{2h} be the coarse grid node representing the fine grid nodes D_j^h and D_l^h. It seems natural that the demands can be restricted simply by summing the demands of the two fine grid nodes to produce the demand at the coarse grid node, $d_m^{2h} = I_h^{2h}[d_j^h, d_l^h] = d_j^h + d_l^h$. Similarly, the flow x_{km}^{2h} from any supply node S_k into the coarse demand node should be the sum of the flows from S_k to each of the fine grid demand nodes that make up the coarse grid node, $x_{km}^{2h} = I_h^{2h}[x_{kj}^h, x_{kl}^h] = x_{kj}^h + x_{kl}^h$.

Restricting the cost of shipment is more complicated, and no obvious "best" approach is apparent. However several methods can be considered. The simplest of these is to define the coarse cost c_{km}^{2h} to be the minimum of the fine costs, i.e., $c_{km}^{2h} = I_h^{2h}[c_{kj}^h, c_{kl}^h] = \min(c_{kj}^h, c_{kl}^h)$. Other simple schemes are readily devised, such as using the maximum of the fine costs, or a weighted average of the fine grid costs. We use a weighted average of the fine grid costs. Again, there are several possible weightings, each having valid arguments for and against it. Three schemes were tested in some depth, equal weighting, flow weighting, and demand weighting:

$$
\begin{array}{llcll}
\text{Equal weighting:} & c^{2h}_{km} & = & I^{2h}_h[c^h_{kj}, c^h_{kl}] & = \dfrac{c^h_{kj} + c^h_{kl}}{2}, \\
\text{Demand weighting:} & c^{2h}_{km} & = & I^{2h}_h[c^h_{kj}, c^h_{kl}] & = \dfrac{d^h_j c^h_{kj} + d^h_l c^h_{kl}}{d^h_j + d^h_l}, \\
\text{Flow weighting:} & c^{2h}_{km} & = & I^{2h}_h[c^h_{kj}, c^h_{kl}] & = \dfrac{x^h_{kj} c^h_{kj} + x^h_{kl} c^h_{kl}}{x^h_{kj} + x^h_{kl}}.
\end{array}
$$

With flow weighting, provision must be made for the case where there is zero flow on both arcs. In such a case flow weighting can be replaced with either demand weighting or equal weighting. In general, we found that demand weighting most consistently gave the best results, and adopted it for our algorithm.

Prolongation, or Interpolation

Suppose that the problem has been solved on the coarse grid Ω^{2h}. We seek a method of prolongation, that is, a way in which the coarse grid solution can be interpolated onto the fine grid. In the coarse grid solution there is some quantity of flow x^{2h}_{km} giving the flow from each supply node S_k to each coarse grid demand node d^{2h}_m. Each such demand node on the coarse grid, however, represents the aggregation of two demand nodes on the fine grid, d^h_j and d^h_l. An interpolation of the coarse grid solution, therefore, can be constructed by treating the M flows $x^{2h}_{1m}, x^{2h}_{2m}, \ldots, x^{2h}_{Mm}$, into the coarse grid demand node d^{2h}_m, as supplies. Interpolation, then, consists of solving for each coarse grid demand node, the $M \times 2$ transportation problem with those supply values, the two demand nodes d^h_j and d^h_l, and the shipping costs c^{2h}_{km}, $k = 1, 2, \ldots, M$. (Figure 4 shows schematically how the interpolation process appears.)

Having defined the interpolation process as finding the solutions to many small transportation problems, we turn our attention to the mechanism for finding these solutions. A method for solving such $M \times 2$ problems is described below. The method is a special case of Vogel's approximation method.

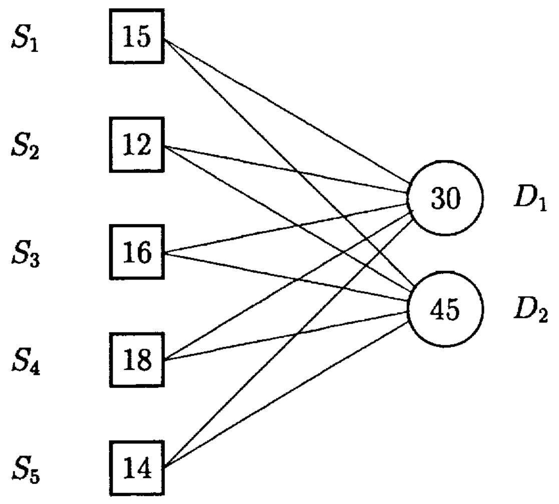
Algorithm 1 *Solving the $M \times 2$ Balanced Transportation Problem*

1. *For each supply node S_k, find the difference in cost of shipping $\delta_k = |c^h_{kj} - c^h_{kl}|$ to the two fine grid demand nodes d^h_j and d^h_l.*

2. *Rank the M supply nodes in decreasing order of these cost differentials, so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_M$.*

3. *Repeat until all supply nodes are removed from the problem:*

   (a) *Denote the supply node at the top of the ordered list as the "current" supply node, and allocate flow to the demand node with the lower cost of shipment, that is, along the least expensive arc, thus determining a "current" demand node. (In the event that more than one node has the largest differential cost, select from among them the node with the*



$$c_{11} = 5 \quad c_{12} = 1 \quad c_{21} = -2 \quad c_{22} = 6 \quad c_{31} = 6 \quad c_{32} = 2 \quad c_{41} = 8 \quad c_{42} = 2 \quad c_{51} = 4 \quad c_{52} = 3$$

$$\delta_1 = 4 \quad \delta_2 = 8 \quad \delta_3 = 4 \quad \delta_4 = 6 \quad \delta_5 = 1$$

Figure 3: *Example problem illustrating the solution method for an $M \times 2$ problem.*

smallest cost along one of its two arcs). Allocate flow along this arc until either the demand at the current demand node is satisfied or until the supply at the current supply node is exhausted.

(b) *If the supply at the current supply node is exhausted, remove that supply node from the problem.*

(c) *If the demand at the current demand node is satisfied, remove that demand node from the problem, allocate the remaining supply from the current supply node to the remaining demand node, and remove the current supply node from the problem.*

4. Stop.

As an example of this procedure, consider the five by two problem shown in Figure 3. The five supply nodes $S_1, S_2, \ldots, S_5$ have, respectively, 15, 12, 16, 18 and 14 units of the commodity to deliver. The demands of the two demand nodes D_1 and D_2 are 30 and 45. Let $\delta = (4\ 8\ 4\ 6\ 1)^T$ be the vector whose i^{th} entry is the difference δ_i between shipping cost from supply node S_i to the two demand nodes (the costs themselves are given for each arc in the figure). Sorting from largest to smallest value of δ_i, the supply nodes are ordered $(S_2, S_4, S_1, S_3, S_5)$. Note that, while the differences for nodes S_1 and S_3 are the same, the cost c_{12} along the arc from node S_1 to node D_2 is less expensive than either of the arcs incident from node S_3. Starting with node S_2, then, as much flow as possible is sent along the least expensive arc. In this case, that is the arc to demand node D_2. Since this demand exceeds the available supply from node S_2, all of the flow from node S_2 goes along this arc. Similarly, node S_4 and then node S_1 send all of their supply to node D_2. When node S_3 has sent 12 units of flow along its least expensive arc, the demand at node D_1 is completely met. Thus node S_3 sends its remaining units to node D_2, as does node S_5. Although the arc from node S_5 to D_1 is less expensive, the demand at D_1 has been met from supply nodes where the difference in arc costs is greater.

We can show now that because of the special structure of the $M \times 2$ problem, i.e., the fact that there are only two demand nodes, this procedure produces an optimal solution.

Theorem 2 *Let x be the vector of flows assigned for the $M \times 2$ problem using the algorithm given above. Then x is an optimal solution to the $M \times 2$ problem.*

Proof: Suppose that x is not an optimal solution. Then there exists a flow $x^* \neq x$ such that $z^* = c^T x^*$ is optimal. We will show that if x is determined by the algorithm given above and $z = c^T x$, then $z^* \geq z$, contradicting the assumption that x is not an optimal solution. Letting $\delta_k = |c_{k1} - c_{k2}|$ for each $k = 1, 2, m \ldots, M$, assume the supply nodes have been ordered in decreasing order of δ_k so that $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_M$. Let i be the first supply node for which x^* differs from x, and without loss of generality, assume that $c_{i1} \leq c_{i2}$. Let $\Delta = x_{i1} - x^*_{i1}$. We first observe five useful facts:

1. Since the problem is balanced, total flow out of S_k equals the supply, so that $s_k = x_{k1} + x_{k2} = x^*_{k1} + x^*_{k2}$ for every k, implying $x_{k1} - x^*_{k1} = x^*_{k2} - x_{k2}$.
2. In particular, since $\Delta = x_{i1} - x^*_{i1}$ then $-\Delta = x^*_{i2} - x_{i2}$.
3. Since the problem is balanced, total flow into D_1 equals demand, so that $d_1 = \sum_{j=1}^{M} x_{j1}$ and $d_1 = \sum_{j=1}^{M} x^*_{k1}$. Subtracting these two relations, and noting that x and x^* do not differ for $j = 1, 2, \ldots, i-1$, we find that $0 = x_{i1} - x^*_{i1} + \sum_{j=i+1}^{M}(x_{j1} - x^*_{j1})$, implying that $\Delta = \sum_{j=i+1}^{M}(x^*_{j1} - x_{j1})$.
4. Using similar reasoning, we obtain $-\Delta = \sum_{j=i+1}^{M}(x^*_{j2} - x_{j2})$.
5. By construction, since $c_{i1} \leq c_{i2}$, then x_{i1} is as large as it can possibly be, so that if x and x^* differ for node i then $x_{i1} > x^*_{i1}$, implying that $\Delta > 0$.

Next, we observe that

$$\begin{aligned} z^* &= z + z^* - z \\ &= z + c^T x^* - c^T x \\ &= z + \sum_{j=i}^{M} \left(x^*_{j1} c_{j1} + x^*_{j2} c_{j2} \right) - \sum_{j=i}^{M} \left(x_{j1} c_{j1} + x_{j2} c_{j2} \right), \end{aligned}$$

where we have used the fact that x^* and x do not differ for $j < i$. Separating the flows from supply node i, we can write

$$\begin{aligned} z^* &= z + (x^*_{i1} - x_{i1})\, c_{i1} + (x^*_{i2} - x_{i2})\, c_{i2} + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) c_{j1} + \sum_{j=i+1}^{M} \left(x^*_{j2} - x_{j2} \right) c_{j2} \\ &= z - \Delta\, c_{i1} + \Delta\, c_{j2} + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) (c_{j1} - c_{j2}) \end{aligned}$$

where we have used the fact that $x_{k1} - x^*_{k1} = x^*_{k2} - x_{k2}$ for all k. Recalling that $\delta_k = |c_{k1} - c_{k2}|$, and that since $c_{i1} \leq c_{i2}$ then $\delta_i = c_{i1} - c_{i2}$, we observe that

$$z^* \geq z + \Delta\, \delta_i + \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1} \right) (-\delta_j).$$

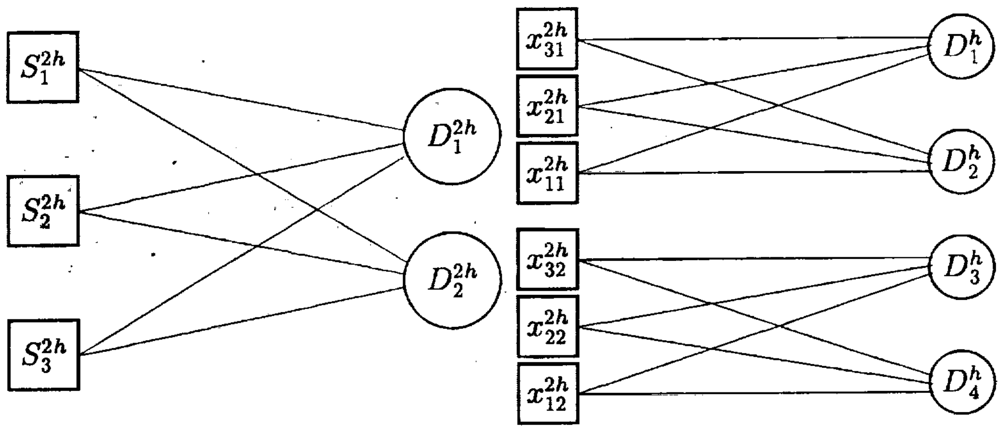


Figure 4:
An illustration of interpolation by local optimization. The flows in the 3×2 *coarse grid problem are the supplies for the two* 3×2 *local problems. The combination of the flows solving those two local problems makes up the interpolated solution to the* 3×4 *fine grid problem.*

Since the nodes are ordered in decreasing order of δ_k, we know that $-\delta_i \leq -\delta_j$ for all $j \geq i$, and therefore

$$z^* \geq z + \Delta\,\delta_i - \delta_i \sum_{j=i+1}^{M} \left(x^*_{j1} - x_{j1}\right).$$

Finally, recalling that $\Delta = \sum_{j=i+1}^{M}(x^*_{j1} - x_{j1})$, we obtain

$$z^* \geq z + \Delta\,\delta_i - \delta_i\,\Delta.$$

Therefore $z^* \geq z$, contradicting the assumption that x is not an optimal solution.

∎

Relaxation

Suppose that we have solved the coarse grid problem, and have interpolated that solution by solving an $M \times 2$ transportation problem for each coarse grid demand node in order to pass the local solution to the two fine grid demand nodes represented by each coarse grid node. The supplies for this local $M \times 2$ problem are the coarse grid flows. Figure 4 displays a schematic showing how the interpolation process appears graphically.

It is important to note that while each of the local $M \times 2$ problems has been solved optimally, there is no reason to expect that the total set of fine grid flows thus assigned will be optimal. For this reason, it is essential that we devise some kind of "relaxation" scheme, whose task is to smooth or correct errors left by the interpolation scheme.

When two $M \times 2$ subproblem solutions are viewed from a more global perspective, as a solution to an $M \times 4$ problem (or as a portion of a solution to a still larger problem), this combination of locally optimized solutions may be flawed, in that too many arcs may have flow on them. This is because the minimum value of the objective function for a balanced transportation problem can always be obtained with a flow regime having flow on at most $M + N - 1$ arcs. This simply reflects the fact from linear programming theory that an extreme point solution has flow on $M + N - 1$ arcs, if the solution is non-degenerate [9], and that an optimal solution can always be found at one of the extreme points (a degenerate solution is one in which distinct subsets of demand nodes are supplied by distinct subsets of supply nodes). If the solution is degenerate, there will be fewer arcs with flow on them. For example, if $N > M$, then in the extreme degenerate case each supply node provides flow to a disjoint subset of the demand nodes. This means that each demand node has exactly one arc with flow on to it, so that precisely N arcs have flow. If $N < M$, then the extreme degenerate case is when each supply node has exactly one arc with flow, giving M such arcs.

When interpolating from Ω^{2h} to Ω^h, each coarse demand node generates two fine grid demand nodes and the optimal solution to the $M \times 2$ subproblem has $M + 1$ arcs with flow, in the non-degenerate case. If the subproblem solution is degenerate, then M arcs have flow. If there are $N/2$ demand nodes on Ω^{2h}, then after the interpolation the collection of subproblem solutions (viewed as the initial feasible solution to the Ω^h problem) will have flow on at least $NM/2$ and at most $(NM + N)/2$ arcs, depending on how many subproblems are degenerate.

Thus, whenever $NM/2$ is greater than $M + N - 1$, (which is true for any long transportation problem where $M > 2$ and $N > 3$), the collection of local solutions has too many arcs with flow to be an extreme point solution to the fine grid problem, and is probably less than optimal. The local relaxation scheme developed here is designed to reduce the number of arcs with flow for the fine grid problem, which will generally have the effect of moving the global solution toward an optimal solution.

The mechanism by which we do this is cycle removal. Since there are $M + N - 1$ arcs in a spanning tree over $M + N$ nodes, and the addition of a single arc (or more) to a tree results in a graph with at least one cycle, then for most problems, the interpolation process will introduce cycles. We note that while this has been developed in the setting of the *entire* collection of local solutions, it is also true in a pairwise sense. That is, each of two $M \times 2$ local solutions will have either $M + 1$ or M arcs with flow. Viewing the pair as a solution to an $M \times 4$ problem, we observe that the combined solution will have at least $2M$ arcs with flow. If $M > 2$ this equals or exceeds the $M + 3$ arcs with flow that would be present in an extreme point solution.

To illustrate this, consider the possibilities when two 3×2 local solutions are combined into a 3×4 solution, as shown in Figure 5. In *a*), two non-degenerate solutions are combined. Numbering the demand nodes of the combined problem clockwise from the upper left and the supply nodes from top to bottom, we observe that there are three cycles in the combined solution $(S_1, D_3, S_2, D_1, S_1)$, $(S_1, D_3, S_3, D_4, S_2, D_1, S_1)$, and $(S_2, D_3, S_3, D_4, S_2)$. In *b*), a degenerate solution is combined with a non-degenerate solution, yielding a combined solution with one cycle. In *c*), two degenerate solutions are combined into a solution that has no cycles, while in *d*), two degenerate solutions are combined into a solution that has one cycle.

A reasonable candidate for a local relaxation process is to adjust the flow in the initial solution produced by the interpolation process, so that cycles are removed and the objective function is reduced. The effect of this procedure is to adjust the locally optimal flows which result from interpolation so that they are more nearly optimal in the global problem.

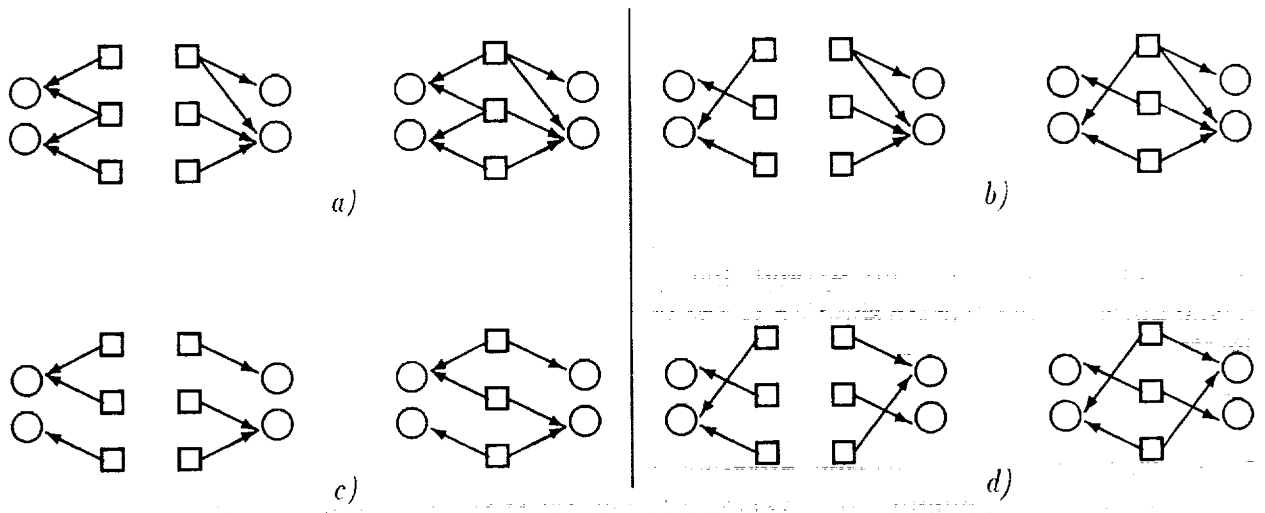


Figure 5: *Combining two* 3×2 *solutions into a* 3×4 *solution. Four typical cases are shown.*

Cycles are detected in the algorithm using a *depth first search (DFS)*. The DFS proceeds as follows:

Algorithm 2 *Depth First Search*

1. *Initialize all nodes with DFS number 0, to indicate they have not yet been visited.*
2. *Start at any node. Assign this node a DFS number of 1, and define node 0 to be the predecessor of this node.*
3. *If any node adjacent to the current node has been previously visited, and has a DFS number lower than the predecessor of the current node, then the path from that node through the current node and back is a cycle. Stop DFS and call the cycle removal routine.*
4. *If no adjacent nodes have lower DFS numbers, then look for any adjacent nodes which have not been visited. If there are any unvisited adjacent nodes, identify the current node as the predecessor of the unvisited node, make the unvisited node the current node, and assign the current node a DFS number equal to the DFS number of its predecessor plus 1.*
5. *If there are no unvisited nodes adjacent to the current node, make the predecessor of the current node the current node. If the current node is node 0, stop. Otherwise, return to step 3.*

Once a cycle is detected, a cycle removal algorithm is used to adjust the flows. The technique is illustrated in Figure 6. The effect of a unit increase in flow in the clockwise direction around the cycle is determined by adding together the costs of the arcs whose flow increases and subtracting the cost of the arcs whose flow decreases. The change in objective function value per unit change in flow in one direction will be the negative of the change in the opposite direction. An example is shown in Figure 6, with the initial flow regime on the left, and the flow after cycle removal on the right. The supplies and demands are shown in the boxes and circles, while the numbers in parentheses above each arc give the cost and flow for that arc. For example, the cost c_{34} is 5, while

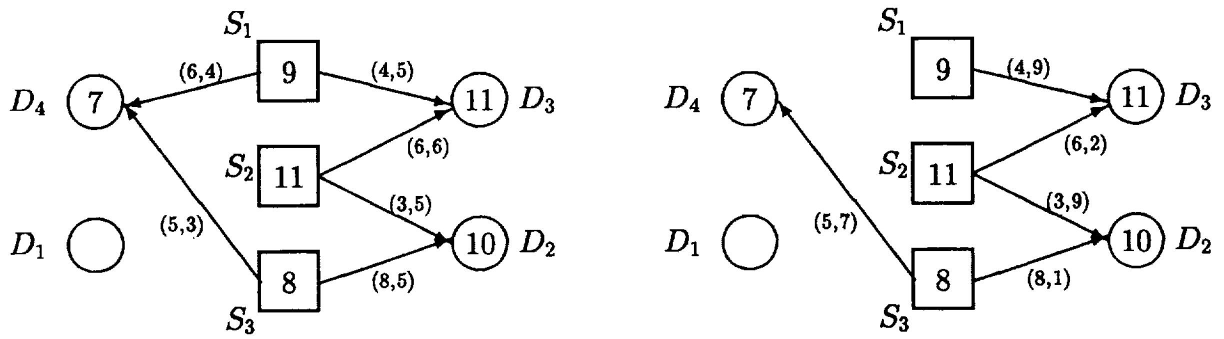


Figure 6: *An initial solution with a cycle (left), and the improved flow regime after cycle removal (right). The numbers in parentheses above each arc are* (c_{ij}, x_{ij}).

there is initially 3 units of flow on that arc. A unit increase in flow clockwise around the cycle (beginning at S_1) will cause a change in the objective function value of $4 - 6 + 3 - 8 + 5 - 6 = -4$, a net decrease. A unit increase in the counter-clockwise direction therefore yields 6, a net increase in the objective function. Clearly, increasing the clockwise flow is profitable, so flow is increased in this direction. Flow values will thus be increased for x_{13}, x_{22} and x_{34}, while flow is decreased for x_{23}, x_{32}, and x_{14}. That is, flow is increased on the arcs in the cycle which point in the profitable direction, and decreased on the the other arcs, until one of the decreasing arcs reaches zero flow. At this point, the cycle has been removed, and the value of the objective function has been decreased. In Figure 6, increasing the flow clockwise around the cycle by four units breaks the cycle by eliminating flow along x_{14}, and reduces the value of the objective function by 16 units. The improved flow regime is shown on the right side of the figure.

This technique is used as a local relaxation method by applying it to pairs of subproblems. Two subproblems which are adjacent in cost space are joined to form an $M \times 4$ problem, which is inspected for cycles. If any are found, they are removed and the problem is searched again.

Two different methods for applying this technique are investigated. The first, termed total relaxation, is to join adjacent pairs of $M \times 2$ problems, remove the cycles, then repeat the process by joining adjacent $M \times 4$ pairs, removing the cycles, then to join $M \times 8$ problems, and so on, until the global problem for the current level is inspected and certified to be cycle-free. This approach is, however, extremely expensive. The second approach only employs a local relaxation, and is therfore true to multigrid principles. In this case, only pairs of $M \times 2$ are checked for cycles. The gain in speed from using this second method is significant, while the decrease in accuracy is negligible (see Table 1 in the next section).

EXPERIMENTAL RESULTS AND CONCLUSIONS

The algorithm employed in this work is an *FMG* algorithm, using demand-weighting as the restriction method for computing costs, interpolation by local optimization, and local relaxation by cycle removal. The results are displayed in Table 1. While the multilevel algorithm performed well on problems with only two or three supply nodes, the results for the five supply node problem are unsatisfactory. The table clearly indicates that relaxation by cycle removal is as effective when applied over a local area as when applied globally, and the computational effort required for local relaxation is an order of magnitude smaller.

Problem Size	Relaxation Method	Run Time	% Above Optimality
2 × 1024	Total	1.210835	0.02 %
2 × 1024	Local	0.131437	0.02 %
3 × 1024	Total	1.15788	8.41 %
3 × 1024	Local	0.108765	8.41 %
5 × 1024	Total	1.18411	58.4 %
5 × 1024	Local	0.106392	59.7 %

We note also that this algorithm is not now competitive with the state of the art in network flow optimization methods. No numerical data are available as a careful comparison has not been made, however, some rough comparisons indicate that much remains to be done before a competitive algorithm could be obtained.

The most significant contribution of the current research is the removal of the requirement for a physical interpretation of the problem, and the dependence on a relationship between distance and shipping costs. By mapping the problem into cost-space, a multilevel approach can be applied to a much broader class of problems. Of course, there is a limit to the number of supply nodes which this approach can handle, due to the increasing dimensionality of the problem. However, for problems with few supply nodes, this approach can be helpful. We predict that further work will yield the result that problems which have either a very small number of supply nodes, or a geometrical interpretation, can be solved to within an acceptable degree of optimality using a multilevel approach. However, problems which do not meet either of these criteria probably cannot be solved with currently known multilevel methods.

Further Research

An algorithm analogous to the full approximation scheme (FAS) should be developed. In the current work, we were unable to find an effective method of extracting a *correction* from the solution on Ω^{2h} and applying it to the approximation on Ω^h, while still maintaining feasibility. Instead, we compute the solution on Ω^{2h} and use interpolation to replace the solution on Ω^h. Since a direct analog to the residual in a PDE is unknown for in an optimization problem, FAS is likely the method of choice, however, the difficulty mentioned above must be overcome.

Another possibility for improving this algorithm is to begin the procedure by overlaying the cost-space with a regular M-dimensional grid. The first step of the restriction process would then be to map the demand nodes from their natural irregularly spaced positions in cost-space to the regular grid points. Later, the final interpolation step would be to transfer from the regular grid back to the original demand points. This approach overcomes a shortcoming in the current algorithm, which aggregates demand nodes which are closest in relative distance in cost-space, regardless of the absolute distance between them. In using a regular grid, a demand node on Ω^{2h} would reflect only the demand at nodes a distance of $2h$ or less away from it. Another important potential advantage is that the work on each coarser level is reduced by 2^{-M}, instead of by one half as in the current research. If the regular grid approach proves worthwhile, then it could be extended to a fast adaptive composite (FAC) grid approach. In a network optimization setting, this might be done by overlaying a fine grid on those regions of cost-space where the density of demand nodes is high, and a coarser grid on the areas of low density. In this way, the flow to nodes which are most similar to their nearest neighbors in cost-space will receive the benefit of a finer grid spacing, while nodes which are naturally more distinct from their neighbors will only enter the problem on the coarser levels.

REFERENCES

[1] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network flows.* John Wiley and Sons, 1990.

[2] Dorit Ron. *Development of Fast Numerical Solvers for Problems in Optimization and Statistical Mechanics.* PhD thesis, Weizmann Institute of Science, 1987.

[3] Achi Brandt, Dorit Ron, and D. J. Amin. Multi-level approaches to discrete-state and stochastic problems. In W. Hackbusch and U. Trottenberg, editors, *Multigrid methods II (Proceedings, Cologne 1985)*, Lecture notes in mathematics 1228. Springer-Verlag, 1985.

[4] Ron Kaminsky. Multilevel solution of the long transportation problem. Master's thesis, Weizmann Institute of Science, 1986.

[5] B. Nilo. The transportation problem: a multi-level approach. Master's thesis, Weizmann Institute of Science, 1986.

[6] P. H. Zipkin. *Aggregation in linear programming.* PhD thesis, Yale University, 1977.

[7] P. H. Zipkin. Bounds on the effect of aggregating variables in linear programs. *Operations Research*, 28(4):903–916, 1980.

[8] E. Balas. Solution of large-scale transportation problems through aggregation. *ORSA*, 1965.

[9] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimization.* John Wiley and Sons, 1988.

N94-23679

VECTORIZATION AND PARALLELIZATION OF THE FINITE STRIP METHOD FOR DYNAMIC MINDLIN PLATE PROBLEMS *

56-64
197138
p. 15

Hsin-Chu Chen
Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA

Ai-Fang He
Department of Mathematics
Illinois State University
Bloomington, IL

SUMMARY

The finite strip method is a semi-analytical finite element process which allows for a discrete analysis of certain types of physical problems by discretizing the domain of the problem into finite strips. This method decomposes a single large problem into m smaller independent subproblems when m harmonic functions are employed, thus yielding natural parallelism at a very high level. In this paper we address vectorization and parallelization strategies for the dynamic analysis of simply-supported Mindlin plate bending problems and show how to prevent potential conflicts in memory access during the assemblage process. The vector and parallel implementations of this method and the performance results of a test problem under scalar, vector, and vector-concurrent execution modes on the Alliant FX/80 are also presented.

INTRODUCTION

More and more parallel computers have been developed and made available to the engineering and scientific computing community in recent years. To take advantage of current and future advanced multiprocessors, however, a great deal of efforts remain to be made in the search for efficient and parallel implementations. In this paper we address both the coarse-grain and fine-grain parallelism offered by the finite strip method (FSM) for the dynamic analysis of Mindlin plate bending problems and present our vector and parallel implementations on multiprocessors with vector processing capabilities. FSM, first developed in the context of thin plate bending analysis, is a semi-analytical finite element process [6, 22]. This method allows for a discrete analysis of

*This work was supported by the U.S. Department of Energy under Grant No. DOE DE-FG02-85ER25001 while the authors were with the Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign.

PAGE 76 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

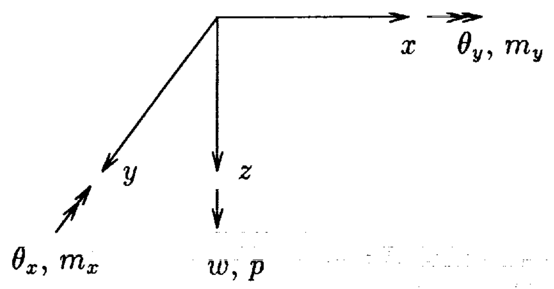


Figure 1: The coordinate system and sign convention.

certain types of physical problems by discretizing their domains into finite strips, involving an approximation of the true solution using a continuous harmonic series in one direction and piecewise interpolation polynomials in the others. Because of the orthogonality properties of the harmonic functions in the stiffness and mass matrix formulation, FSM decomposes a problem, when applicable, into many smaller and independent subproblems which yields coarse-grain parallelism in an extremely easy and natural way.

Although not as versatile as the finite element method, FSM has been applied to a wide range of plate, folded plate, shell, and bridge deck problems [4, 6, 7, 8, 10, 18] because of its efficiency and simplicity. The performance induced by the coarse-grain parallelism of this method in a multiprocessing environment has been shown in [9] for the static analysis of Mindlin plate problems and in [20] for groundwater modeling. In this paper, we report and compare the performance results of our implementation for the dynamic analysis of a simply-supported rectangular Mindlin plate using scalar, vector, and vector-concurrent execution modes on an Alliant FX/80.

THE PROBLEM

In this section we describe briefly the mathematical modeling of Mindlin plate problems [17]. Let Ω be the space domain in $\Re^2$, Γ the boundary, and T the time domain. Let also the stress resultants, generalized strains, displacements, dynamic surface loadings, and inertia forces be denoted respectively by $\mathbf{s}$, $\mathbf{r}$, $\mathbf{d}$, $\mathbf{p}$, and $\mathbf{q}$:

$$\mathbf{s} = \begin{bmatrix} M_x \\ M_y \\ M_{xy} \\ Q_x \\ Q_y \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \gamma_x \\ \gamma_y \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix}, \quad \mathbf{d} = \begin{bmatrix} w \\ \theta_x \\ \theta_y \end{bmatrix}, \quad \mathbf{p} = \begin{bmatrix} p \\ m_x \\ m_y \end{bmatrix}, \quad \text{and} \quad \mathbf{q} = \begin{bmatrix} -\rho h \ddot{w} \\ \frac{1}{12}\rho h^3 \ddot{\theta}_x \\ \frac{1}{12}\rho h^3 \ddot{\theta}_y \end{bmatrix}$$

where ρ stands for the mass density (per unit volume), h the thickness of the plate, and $\ddot{v}$ ($v = w$, θ_x, or θ_y) the second derivative of v with respect to time t: $\ddot{v} = \partial^2 v/\partial t^2$. The subscripts x, y, and z above represent the directions in the Cartesian coordinate system. The sign convention for the displacements and external loadings is shown in Figure 1. Neglecting the damping effect of the plate, the differential equations which govern the state of stress resultants, generalized strains, and displacements in an elastic plate can be expressed as

1. Equilibrium equations: $L_1^T\mathbf{s} + \mathbf{p} + \mathbf{q} = 0$ in $\Omega \otimes T$, subject to some appropriate boundary conditions on Γ,
2. Stress-strain equations: $\mathbf{s} = \mathbf{D}\mathbf{r}$, and
3. Strain-displacement equations: $\mathbf{r} = L_2\mathbf{d}$.

Here $\mathbf{D}$ is the material property matrix of an elastic plate. L_1 and L_2 are the differential operators:

$$L_1^T = \begin{bmatrix} 0 & 0 & 0 & \partial/\partial x & \partial/\partial y \\ \partial/\partial x & 0 & \partial/\partial y & -1 & 0 \\ 0 & \partial/\partial y & \partial/\partial x & 0 & -1 \end{bmatrix} \tag{1}$$

and

$$L_2^T = \begin{bmatrix} 0 & 0 & 0 & \partial/\partial x & \partial/\partial y \\ -\partial/\partial x & 0 & -\partial/\partial y & -1 & 0 \\ 0 & -\partial/\partial y & -\partial/\partial x & 0 & -1 \end{bmatrix} \tag{2}$$

where the superscript T denotes the transpose of a matrix.

For orthotropic material, the matrix $\mathbf{D}$ takes the form

$$\mathbf{D} = \begin{bmatrix} D_x & D_1 & & & \\ D_1 & D_y & & & \\ & & D_{xy} & & \\ & & & \alpha G_x & \\ & & & & \alpha G_y \end{bmatrix} \tag{3}$$

where D_x, D_1, ..., G_y are the standard flexural and shear rigidities of plates and α is a modification coefficient to account for the deviation of shear strain distribution from uniformity [4] ($\alpha = 5/6$ for rectangular cross section; see [21, p. 371]). The rest of the entries in $\mathbf{D}$ are zero. If the material is isotropic, then the nonzero entries take the following values:

$$D_x = D_y = \frac{Eh^3}{12(1-\nu^2)}, \quad D_1 = \nu D_x, \quad D_{xy} = \frac{1-\nu}{2}D_x, \quad \text{and} \quad G_x = G_y = \frac{Eh}{2(1+\nu)}$$

where E, h, and ν represent the material modulus, plate thickness, and Poisson's ratio, respectively. The total potential energy of the plate due to the dynamic surface loading $\mathbf{p}$ [17, 16, 14] can be written as

$$\Pi = \int_0^t \left(\frac{1}{2} \int_\Omega (L_2\mathbf{d})^T \mathbf{D} (L_2\mathbf{d})\, d\Omega - \int_\Omega \mathbf{p}^T\mathbf{d}\, d\Omega - \frac{1}{2}\int_\Omega \dot{\mathbf{d}}^T \Lambda \dot{\mathbf{d}}\, d\Omega \right) dt \tag{4}$$

where $\dot{\mathbf{d}} = \partial\mathbf{d}/\partial t$ and $\Lambda = \text{diag}\left[-\rho h,\ \frac{1}{12}\rho h^3,\ \frac{1}{12}\rho h^3\right]$, a diagonal matrix.

A STRIP ELEMENT FOR MINDLIN PLATES

We now outline the FSM formulation for the Mindlin plates using linear elements [4, 19]. We shall confine our discussions to rectangular Mindlin plate problems simply supported on two

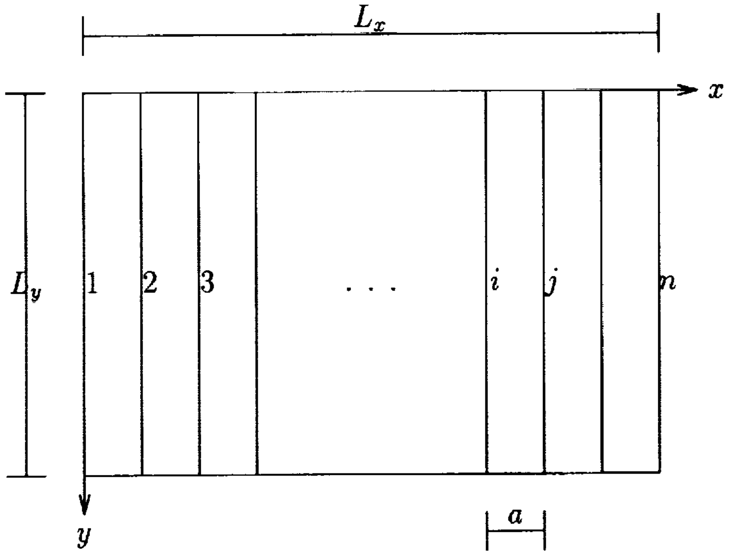


Figure 2: A discretized plate.

opposite sides. Figure 2 shows a rectangular plate discretized into $n-1$ finite strips. The plate is assumed to be simply supported on edges $y=0$ and $y=L_y$. Shown in Figure 3 is the mid-plane of a typical linear strip plate element of constant thickness h, whose local coordinate system is denoted by (x', y', z') where $x'=x-x_i$, $y'=y$, and $z'=z$. Let $\Omega_{(e)}$ be the domain of the e^{th} strip element and i and j be the two longitudinal edges (nodal lines) of the element, as shown in Figure 3. Let $\mathbf{d}_{(e)}(x,y,t)$ and $\mathbf{u}^l_{(e)}(t)$ be defined as

$$\mathbf{d}_{(e)}(x,y,t)=[w(x,y,t)\ \ \theta_x(x,y,t)\ \ \theta_y(x,y,t)]^T\,,\quad (x,y)\in\Omega_{(e)}$$

and

$$\mathbf{u}^l_{(e)}(t)=\begin{bmatrix}\mathbf{u}^l_i(t)\\ \mathbf{u}^l_j(t)\end{bmatrix}=\left[w^l_i(t)\ \ \theta^{\,l}_{xi}(t)\ \ \theta^{\,l}_{yi}(t)\mid w^l_j(t)\ \ \theta^{\,l}_{xj}(t)\ \ \theta^{\,l}_{yj}(t)\right]^T$$

where $w^l_i(t)$ denotes the l^{th} harmonic coefficient (amplitude) of $w_i(y,t)$ which is the displacement along edge i, etc. For a linear strip element with m harmonic terms specified, the approximation to $\mathbf{d}_{(e)}$ is given [4, 18] by

$$\mathbf{d}_{(e)}(x,y,t)\approx\sum_{l=1}^{m}\mathbf{F}^l(x,y)\mathbf{u}^l_{(e)}(t) \tag{5}$$

with

$$\mathbf{F}^l=\begin{bmatrix}N_iS_l & 0 & 0 & N_jS_l & 0 & 0\\ 0 & N_iS_l & 0 & 0 & N_jS_l & 0\\ 0 & 0 & N_iC_l & 0 & 0 & N_jC_l\end{bmatrix}$$

where S_l and C_l are the l^{th} harmonic functions of y, and N_i and N_j are the linear shape functions of x, defined by

$$S_l=sin\frac{l\pi y}{L_y},\quad C_l=cos\frac{l\pi y}{L_y},$$

$$N_i=\frac{1-r_{(e)}}{2},\quad\text{and}\quad N_j=\frac{1+r_{(e)}}{2}$$

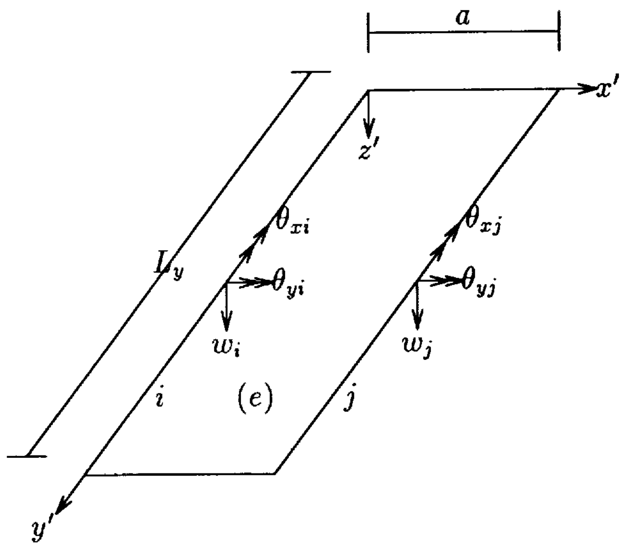


Figure 3: A typical plate strip element.

where $r_{(e)}$, ranging from -1 to 1, is the natural coordinate in x-direction of the e^{th} element. Note that $r_{(e)} = -1 + 2\frac{x-x_i}{x_j-x_i}$ for the element shown in Figure 3. It should be observed that the approximation to the displacement vector in (5) satisfies the simply supported boundary conditions on edges $y = 0$ and $y = L_y$; i.e., w, θ_x, $\partial w/\partial x$, $\partial \theta_x/\partial x$, and $\partial \theta_y/\partial y$ all vanish on these two edges. The dynamic surface loading on the e^{th} element, $\mathbf{p}_{(e)}(x, y, t)$, can often be approximated by the sum of a harmonic series in the longitudinal direction as shown below

$$\mathbf{p}_{(e)}(x, y, t) \approx \sum_{l=1}^{m} \mathbf{H}^l(y)\mathbf{p}^l_{(e)}(x, t) \tag{6}$$

where $\mathbf{H}^l = \text{diag}\,[S_l, \ \ S_l, \ \ C_l]$ and $\mathbf{p}^l_{(e)} = \begin{bmatrix} q^l & m^l_x & m^l_y \end{bmatrix}^T_{(e)}$. The subscript (e) outside the brackets indicates that every component of the vector is associated only with the e^{th} element.

Following the standard finite element procedure and taking advantage of the orthogonality properties of the harmonic functions, we obtain a linear algebraic differential system of block diagonal form [5] depicted by:

$$\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f} \tag{7}$$

where

$$\mathbf{M} = \mathbf{M}^{11} \oplus \mathbf{M}^{22} \oplus \cdots \oplus \mathbf{M}^{mm} \quad \text{and} \quad \mathbf{K} = \mathbf{K}^{11} \oplus \mathbf{K}^{22} \oplus \cdots \oplus \mathbf{K}^{mm}$$

are block diagonal matrices of the same block structure. The vectors $\mathbf{u}$ and $\mathbf{f}$ are accordingly partitioned,

$$\mathbf{u}^T = \begin{bmatrix} (\mathbf{u}^1)^T & (\mathbf{u}^2)^T & \cdots & (\mathbf{u}^m)^T \end{bmatrix} \quad \text{and} \quad \mathbf{f}^T = \begin{bmatrix} (\mathbf{f}^1)^T & (\mathbf{f}^2)^T & \cdots & (\mathbf{f}^m)^T \end{bmatrix}.$$

In (7), the symbol $\oplus$ stands for the direct sum of square matrices. $\mathbf{M}^{ll}$, $\mathbf{K}^{ll}$, $\mathbf{u}^l$, and $\mathbf{f}^l$ are the system mass matrix, system stiffness matrix, system displacement amplitude vector, and system load amplitude vector due to the l^{th} harmonic mode, respectively. In the rest of the paper, we shall drop the term *amplitude* and simply call $\mathbf{u}^l$ ($\mathbf{f}^l$) the l^{th} system displacement (load) vector for brevity. $\mathbf{M}^{ll}$ is assembled from the strip mass matrix $\mathbf{M}^{ll}_{(e)}$, $\mathbf{K}^{ll}$ from the strip stiffness matrix $\mathbf{K}^{ll}_{(e)}$, and $\mathbf{f}^l$ from the strip load vector $\mathbf{f}^l_{(e)}$ where

$$\mathbf{M}^{ll}_{(e)} = \int_{\Omega_{(e)}} (\mathbf{F}^l)^T \mathbf{\Lambda} \mathbf{F}^l d\Omega_{(e)}, \quad l = 1,\ m, \tag{8}$$

$$\mathbf{K}^{ll}_{(e)} = \int_{\Omega_{(e)}} (L_2\mathbf{F}^l)^T \mathbf{D} (L_2\mathbf{F}^l) d\Omega_{(e)}, \quad l = 1,\ m, \tag{9}$$

$$\mathbf{f}^l_{(e)} = \int_{\Omega_{(e)}} (\mathbf{F}^l)^T \mathbf{H}^l \mathbf{p}^l_{(e)} d\Omega_{(e)}, \quad l = 1,\ m. \tag{10}$$

For a plate discretized with n nodal lines, $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$ are square matrices of order $3n$ for each l. ($\mathbf{K}^{ll}_{(e)}$ and $\mathbf{M}^{ll}_{(e)}$ are of order 6.) Once the entire system stiffness matrix $\mathbf{K}$, system mass matrix $\mathbf{M}$, and system load vector $\mathbf{f}$ are assembled and the boundary conditions imposed, the remaining major work is to solve the linear algebraic differential system (7) for $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$.

PARALLEL AND VECTOR IMPLEMENTATIONS

Computational Procedure. Similar to the finite element method, FSM normally consists of the following three main computational components: (1) the generation of strip stiffness/mass matrices and strip load vectors for all strip elements, (2) the assemblage of the entire system stiffness/mass matrix and system load vector, and (3) the solution process of the resulting linear differential system $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}$. There are many step-by-step integration methods available for solving the 2nd-order linear differential equations. Among them are the central difference, Houbolt, Wilson θ, and Newmark β methods. The central difference method is an explicit scheme and the other three are implicit. Regardless of whether the method employed is implicit or explicit, the procedure basically involves an initial calculation of an effective coefficient matrix and then solves an effective linear system, after an effective load vector is formed, at each time step. In this paper, we employ the Newmark integration method whose procedure is shown below, where a_0, a_1, $\cdots$, a_7 are the Newmark integration constants [3, pp. 311]:

(1) initial calculation of the effective stiffness matrix $\hat{\mathbf{K}} = \mathbf{K} + a_0\mathbf{M}$, the factorization of $\hat{\mathbf{K}}$ into $\mathbf{LL^T}$ or $\mathbf{LDL}^T$ form, and then for each time step t_{k+1}, $k = 0, 1, \cdots$
(2) forming the effective load vector $\hat{\mathbf{f}}$ at time t_{k+1}: $\hat{\mathbf{f}}_{k+1} = \mathbf{f}_{k+1} + \mathbf{M}(a_0\mathbf{u}_k + a_2\dot{\mathbf{u}}_k + a_3\ddot{\mathbf{u}}_k)$,
(3) solving the effective linear system at time t_{k+1}: $\hat{\mathbf{K}}^{ll}\mathbf{u}_{k+1} = \hat{\mathbf{f}}_{k+1}$,
(4) calculating the acceleration and velocity vectors $\ddot{\mathbf{u}}_{k+1}$ and $\dot{\mathbf{u}}_{k+1}$:

$$\ddot{\mathbf{u}}_{k+1} = a_0(\mathbf{u}_{k+1} - \mathbf{u}_k) - a_2\dot{\mathbf{u}}_k - a_3\ddot{\mathbf{u}}_k, \quad \dot{\mathbf{u}}_{k+1} = \dot{\mathbf{u}}_k + a_6\ddot{\mathbf{u}}_k + a_7\ddot{\mathbf{u}}_{k+1}.$$

Note that the first step need be performed only once. The last three steps, however, must be performed at every time step and therefore constitute the most time-consuming part in the entire analysis.

To address the parallel implementation of FSM, we should first employ the decoupled structure of the system stiffness matrix depicted by (7), due to the orthogonality properties of harmonic functions. This decoupling leads to m independent sets of differential equations. Therefore, solving (7) is equivalent to solving

$$\mathbf{M}^{ll}\ddot{\mathbf{u}}^{l} + \mathbf{K}^{ll}\mathbf{u}^{l} = \mathbf{f}^{l}, \quad l = 1,\ m$$

where $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$, $l = 1, \cdots, m$, are block tridiagonal matrices with each block of order only 3×3 for the ordering shown in Figure 2. Furthermore, each $\mathbf{M}^{ll}$ consists of only three nonzero diagonals. Since there is no data dependency among these m subsystems, not only can the generation of $\mathbf{M}^{ll}_{(e)}$, $\mathbf{K}^{ll}_{(e)}$, and $\mathbf{f}^{l}_{(e)}$ and the assemblage of $\mathbf{M}^{ll}$, $\mathbf{K}^{ll}$, and $\mathbf{f}^{l}$ for each harmonic term be performed independently, but all the subsystems can be solved in parallel. In a parallel computing environment with parallelism of two levels (considering vectorization as the first level), this special feature leads FSM to a fully parallelizable approach when the number of harmonic terms matches the number of processors. The following pseudo-Fortran code outlines its computational procedure and indicates where parallelism can be exploited for vector/concurrent executions.

```
C — Initial calculations
    DO 200 $l = 1,\ m$              (concurrent, one CPU per iteration)
        DO 100 $e = 1,\ N_s$                       (to be discussed)
            Generate $\mathbf{K}^{ll}_{(e)}$, $\mathbf{M}^{ll}_{(e)}$, and $\mathbf{f}^{l}_{(e)}$
            Assemble $\mathbf{K}^{ll}$, $\mathbf{M}^{ll}$, and $\mathbf{f}^{l}$
        END 100
        Initialize $\mathbf{u}^{l}$, $\dot{\mathbf{u}}^{l}$, and $\ddot{\mathbf{u}}^{l}$               (vector)
        Form $\hat{\mathbf{K}}^{ll}$ from $\mathbf{K}^{ll}$ and $\mathbf{M}^{ll}$                (vector)
        Factorize $\hat{\mathbf{K}}^{ll}$ into $\mathbf{LL}^{\mathbf{T}}$ or $\mathbf{LDL}^{\mathbf{T}}$ form       (vector)
    END 200
C — Calculations for each time step
    DO until the last time step                    (sequential)
        DO 400 $l = 1,\ m$          (concurrent, one CPU per iteration)
            DO 300 $e = 1,\ N_s$                   (to be discussed)
                Generate $\mathbf{f}^{l}_{(e)}$ and assemble $\mathbf{f}^{l}$
            END 300
            Form effective load vector $\hat{\mathbf{f}}^{l}$          (vector)
            Solve $\hat{\mathbf{K}}^{ll}\mathbf{u}^{l} = \hat{\mathbf{f}}^{l}$ for $\mathbf{u}^{l}$                  (vector)
            Calculate $\ddot{\mathbf{u}}^{l}$ and $\dot{\mathbf{u}}^{l}$                    (vector)
        END 400
        DO 600 $l = 1,\ m$                         (sequential)
            Accumulate displacements $w$ for all strips  (vector-concurrent)
        END 600
    END DO
```

In the above pseudo-code, we neglect the step of imposing boundary conditions because they can be performed in the generation step. The word *concurrent* inside the parentheses after the DO statements is used to show that all iterations in this loop may be performed in parallel, on the basis of one processor per iteration ; and the word *vector* (or *vector-concurrent*) indicates computations involved in the statement should be performed in vector (or vector-concurrent) mode whenever possible and desirable. Whether a vector operation is desirable depends on the startup overhead and the vector length of the operation.

Data Structure and Parallelization. To allow current code restructurers to automatically vectorize or parallelize certain computations, the Fortran statements related to that part of computations are usually written in the form of DO loops or array constructs . Potential memory access conflict must also be resolved. Therefore, the data structure of the code plays an essential role. In our implementations, the system stiffness matrix $\mathbf{K}$ and system mass matrix $\mathbf{M}$ are represented by two 3D arrays SK(1:nbk,1:n,1:m) and SM(1:nbm,1:n,1:m), respectively, where *nbk* (*nbm*) is the semi-bandwidth of $\mathbf{K}$ ($\mathbf{M}$), n the number of equations in each harmonic term, and m the number of harmonic terms. It should be noted that in many situations, it is more beneficial to interchange the first two dimensions of both $\mathbf{K}$ and $\mathbf{M}$, or to concatenate the first two dimensions into a single dimension. The system load vector $\mathbf{f}$ is represented by a 2D array SF(1:n,1:m) and the vectors $\mathbf{u}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$ are similarly represented by 2D arrays SU, SV, and SA, respectively. This representation allows parallelization across harmonic terms to be performed in the outermost loop. It also makes the passing of references to subroutines an easy task.

To serve as an example, we consider the *DO 200* loop where the computations inside the loop are now translated into subroutines as shown below (the *DO 400* loop follows the same approach).

```
CVD$L CNCALL        ! an Alliant directive
    DO 200 L = 1, m        ! **concurrent**, one CPU per iteration
        CALL GenAss (SK(1,1,L), SM(1,1,L), SF(1,L), L, n, nbk, nbm, ns, ...)
        CALL Initialize (SU(1,L), SV(1,L), SA(1,L), ...)  ! Initialize $\mathbf{u}_0$, $\dot{\mathbf{u}}_0$, and $\ddot{\mathbf{u}}_0$.
        CALL Form (SK(1,1,L), SM(1,1,L), n, nbk, nbm, a0)  ! Form $\hat{\mathbf{K}}^{ll}$ and overwrite SK.
        CALL Factorize (SK(1,1,L), n, nbk)  ! Factorize $\hat{\mathbf{K}}^{ll}$ and overwrite SK.
    END 200
```

where GenAss is a subroutine performing the task of the *DO 100* loop in the previous pseudo code. The other three subroutines are self-explanatory. In the above code, the argument *ns* denotes the number of strips N_s and *a0* is the Newmark constant a_0. Using this approach, each processor will have an identical local copy, automatically generated by the compiler, of the subroutines inside the loop and its own reference space (via the index L) in locating $\mathbf{K}^{ll}$, $\mathbf{M}^{ll}$, and $\mathbf{f}^{l}$; yielding concurrent execution for all harmonic terms because distinct processors will hold different values of L. This not only prevents memory access conflicts in performing these tasks but also enables us to use a single set of subroutines for all harmonic terms. The same applies to the other three subroutines as well. Note that the index L is also passed to the subroutine GenAss as a local variable because it is required for evaluating $\mathbf{K}^{ll}_{(e)}$, $\mathbf{M}^{ll}_{(e)}$, and $\mathbf{f}^{l}_{(e)}$ whose dimensions should be declared inside GenAss and will become local variables.

Vectorization. To address vectorization, we now turn to the computations for a single harmonic term. First we note that the formation of the effective stiffness matrix $\hat{\mathbf{K}}^{ll}$ and effective load vector $\hat{\mathbf{f}}^{l}$, and the calculation of $\ddot{\mathbf{u}}^{l}$ and $\dot{\mathbf{u}}^{l}$ consist mainly of matrix-matrix (vector-vector) additions and matrix-vector multiplications and are thus highly vectorizable. The vectorization and parallelization of factorizing $\hat{\mathbf{K}}^{ll}$ and solving the linear system $\hat{\mathbf{K}}^{ll}\mathbf{u}^{l} = \hat{\mathbf{f}}^{l}$ have been under intensive studies; see [13, 15, 23] for example. In this paper, we shall only focus on approaches to vectorizing the generation of $\mathbf{K}^{ll}_{(e)}$ and the assemblage of $\mathbf{K}^{ll}$. The generation of $\mathbf{M}^{ll}_{(e)}$ ($\mathbf{f}^{l}_{(e)}$) and the assemblage of $\mathbf{M}^{ll}$ ($\mathbf{f}^{l}$) follow the same way and, thus, need not be discussed.

There are two approaches to vectorizing the generation of $\mathbf{K}^{ll}_{(e)}$. The first, referred to as Vectorization within a Single Strip (VSS), is to generate the entries of $\mathbf{K}^{ll}_{(e)}$ in vector mode. This approach requires a minimal storage because $\mathbf{K}^{ll}_{(e)}$ for all strips can share the same storage of a single strip stiffness matrix, which is usually the case for most traditional finite strip or finite element programs. The disadvantage is that the vector length available for vectorization is limited by the order of the strip stiffness matrix, 6 in our case, which is rather small. In addition, the generation step may not even involve any loop structure because most of the Fortran statements may simply be assignment statements when the entries of $\mathbf{K}^{ll}_{(e)}$ are explicitly integrated. Therefore, we resort to the second approach: Vectorization across Multiple Strips (VMS). This approach generates the matrix entries component-wise across many different strips by employing the fact that each strip matrix can be generated independently of the others. It, however, requires a manual change in the data structure of the strip matrix in the computer program because current code restructurers can hardly accomplish this task automatically. One way of achieving our goal is to add one more dimension (preferably the first dimension) to the array that stores a strip matrix so that the new array can store all strip stiffness matrices. For example, let EKL(1:6,1:6) be the array used in the VSS approach for storing a single strip stiffness matrix and be shared by all strips, one at a time. (For simplicity, we ignore the symmetry of the matrix.) When the VMS approach is employed, we can simply change EKL to a 3D array, say EKL(1:ns,1:6,1:6), so that the first dimension is associated with strip identifications, allowing vector execution to be performed across strips. Although the change in data structure may impose some programming difficulty in modifying an existing code, this approach indeed provides a very good way for both vectorization and parallelization.

So far as the assemblage of the l^{th} system stiffness matrix $\mathbf{K}^{ll}$ is concerned, both VSS and VMS are still applicable if potential data dependencies are avoided. Note that assemblying an entry of $\mathbf{K}^{ll}_{(e)}$ to $\mathbf{K}^{ll}$ has no conflict with assemblying the other entries of the same matrix to $\mathbf{K}^{ll}$. Vectorization obviously can be performed within any single strip matrix without any difficulty, subject to the same disadvantage of short vector length as the case in the generation step. The following Fortran code indicates where vectorization can be performed using VSS for assemblying the stiffness matrix, where the rows of SKL store the upper diagonals of the band symmetric matrix $\mathbf{K}^{ll}$ using the Linpack format [12] with the main diagonal of $\mathbf{K}^{ll}$ stored in the last row of SKL.

```
DO 100 I = 1, NBK                        ! NBK (=6): Semi-bandwidth of K^ll
    SKL(I, 1:N) = 0.0     (vector)       ! Initialization. N: No. of equations of K^ll
END 100
DO 300 K = 1, NS                         ! NS: No. of strips
    K1 = 3 * (K-1)
    DO 200 J = 1, 6
        J1 = K1 + J
        I1 = NBK - J + 1
        SKL(I1:NBK, J1) = SKL(I1:NBK, J1) + EKL(1:J, J)    (vector)
                                         ! Vector length too short.
    END 200
END 300
```

Care, however, must be taken when the VMS approach is employed for assembling $\mathbf{K}^{ll}$. This is because different strips may have some nodes in common, which amounts to saying that the entries of $\mathbf{K}^{ll}_{(e)}$ from different strips may contribute themselves to the same location in $\mathbf{K}^{ll}$. Therefore, in order to vectorize the assemblage of $\mathbf{K}^{ll}$ from $\mathbf{K}^{ll}_{(e)}$ across multiple strip elements, we must find a way to avoid potential simultaneous updates of a common matrix entry. A general approach to avoid this situation is to use graph coloring techniques to partition strips so that all strips in the same group do not contain any common nodes. For our plate problems under consideration, two colors are enough: one for odd strips and the other for even strips. When a natural ordering is imposed as shown in Figure 2, however, a better approach to enhancing vectorization can be employed by assemblying entries component-wise (or node-wise) across all strip elements as shown below, assuming the i^{th} strip starts from nodal line i to nodal line $i+1$ and all strip stiffness matrices are available.

```
DO 100 I = 1, NBK                        ! NBK (=6): Semi-bandwidth of K^ll
    SKL(I, 1:N) = 0.0     (vector)       ! N: No. of equations of K^ll
END 100
DO 300 J = 1, 6
    JS = 3 * (NS-1) + J                  ! NS: No. of strips
    DO 200 I = 1, J
        IJ = NBK - J + I
        SKL(IJ, J:JS:3) = SKL(IJ, J:JS:3) + EKL(1:NS, I, J)    (vector)
    END 200
END 300
```

Note that the array EKL now has one dimension more than the one used in the previous code. The storage can be reduced by about half if symmetry of the matrix is taken into account. Finally, we would like to mention that for a cluster-based multiprocessor with parallelism of three levels like the Cedar [11], FSM is a perfect candidate because the decoupling at the system level offers

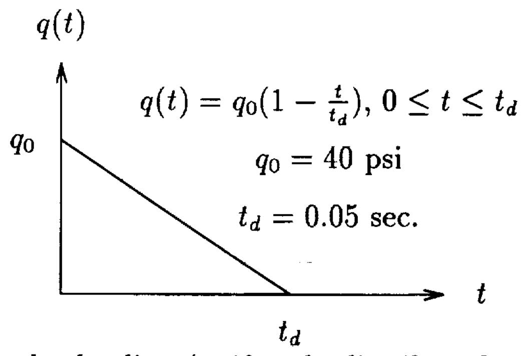


Figure 4: The triangular loading (uniformly distributed on the entire plate).

a great deal of freedom for the problem to be solved using all levels of parallelism. For example, we need exploit only the first two levels of parallelism in a linear system solver instead of three because the highest level of parallelism can be employed across multiple linear subsystems.

NUMERICAL EXPERIMENTS

To demonstrate the effectiveness and parallelizability of FSM, we consider the dynamic Mindlin analysis of a thin steel plate that is simply supported on all of its four edges and is subject to a uniformly distributed triangular loading $q(t)$ as shown in Figure 4. This plate, adapted from [2], is 60 inches (L_x) wide, 40 inches (L_y) long, and one inch thick throughout the entire plate. The material of the plate is assumed to be isotropic with Young's modulus $E = 30 \times 10^6$ ksi, Poisson ratio $\nu = 0.25$, and a mass density of $m = 0.00073$ lb-sec^2/in^4. The time step size Δt is set to 0.00001 sec. In evaluating the strip stiffness matrices, reduced integration with one Gaussian point is used to overcome the shear locking behavior [18]. The strip mass matrices are evaluated using the consistent mass approach. The linear algebraic differential equations are solved using the Newmark integration method with parameters $\alpha = 0.25$ and $\delta = 0.50$ [3, pp. 311]. A banded direct solver is used to solve the resulting linear subsystems in each time step.

In Figure 5, we compare the numerical solution of the displacement w at the center of the plate using 16 Mindlin strip elements with the exact solution (Fourier series) derived from the Kirchhoff thin plate theory. Eight harmonic terms are used in the finite strip approximation. From Figure 5, it is clear that the finite strip solution is in good agreement with the exact solution of the Kirchhoff theory. The performance of this method on an Alliant FX/80 is shown in Tables 2 and 3. In Table 2, we compare the CPU time (all in seconds) consumed in the entire analysis, including the generation, assemblage, and solution of the linear algebraic differential equations and finally the calculation of the displacements. Three different execution modes: scalar (S), vector (V), and vector-concurrent (VC) are considered. The compiler options [1] used for these modes are shown in Table 1.

Table 2 shows the vector speedup (the ratio of the 1-processor CPU time spent under the S mode to that under the V mode) for the entire process. As seen from this table, the vector

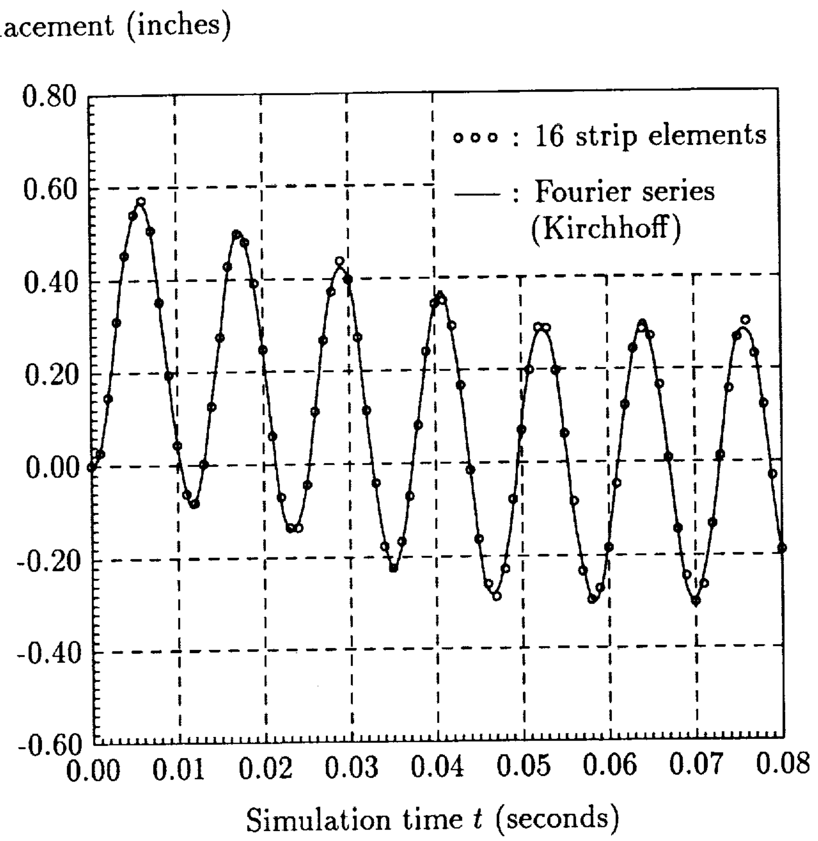


Figure 5: Displacement at the center of the plate.

Table 1: Compiler options

Execution mode	Compiler options	Subprograms compiled
Scalar (S)	-Og -AS -pg	the entire program
Vector (V)	-Ogv -AS -pg	the entire program
Vector-Concurrent (VC)	-Ogv -AS -Ogvc -AS	recursively-called subroutines others

Table 2: CPU time (in seconds) and vector speedup on the Alliant FX/80 using one processor.

Step	Scalar (S)	Vector (V)	S/V	Remark
Solve $\mathbf{LDL}^T\mathbf{u} = \hat{\mathbf{f}}$	177.1	137.1	1.29	semi-bandwidth too small
Compute $\hat{\mathbf{f}}$, $\dot{\mathbf{u}}$, $\ddot{\mathbf{u}}$ (Newmark)	91.0	25.3	3.60	mainly DAXPY operations.
Generate $\mathbf{f}^l_{(e)}$ and assemble $\mathbf{f}$	42.7	12.4	3.45	using the VMS approach
Initialization and I/O	1.72	1.70	1.01	no manual optimization
Total	312.4	176.4	1.77	

C-2.

Table 3: Parallel performance under the vector-concurrent mode.

No. of processors k	1	2	4	8
CPU time in seconds	165.7	84.14	45.01	25.08
Concurrency speedup S_k	1.00	1.97	3.68	6.61
Efficiency E_k (%)	100.0	98.5	92.0	82.6

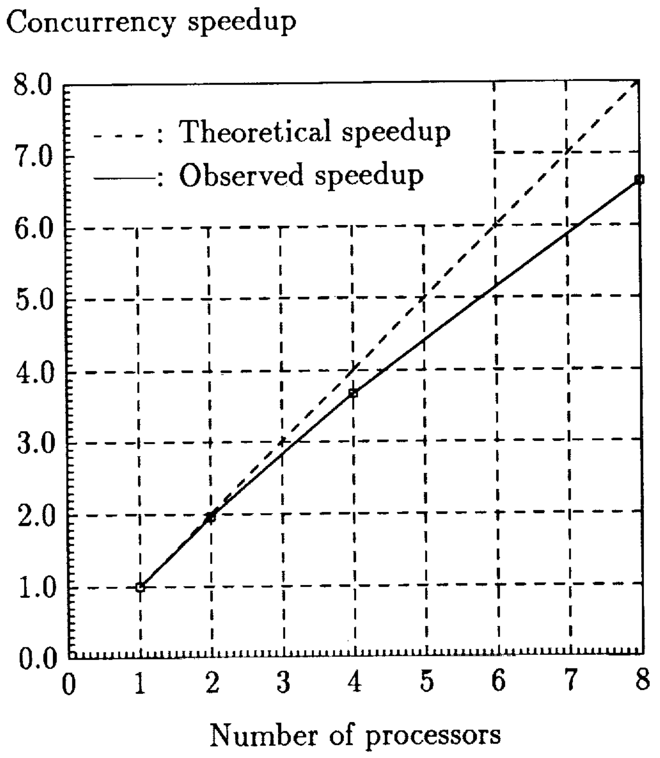


Figure 6: Concurrency speedup on the Alliant FX/80.

speedups for the three most time-consuming parts: (1) solving $\hat{\mathbf{K}}u = \hat{\mathbf{f}}$, (2) computing $\hat{\mathbf{f}}$, $\dot{\mathbf{u}}$, and $\ddot{\mathbf{u}}$, and (3) generating $\mathbf{f}^{l}_{(e)}$ and assemblying $\mathbf{f}$ are 1.29, 3.60, and 3.45, respectively. Note that the semi-bandwidth of the system stiffness matrix is only 6 in this example, which is obviously not long enough for a banded direct linear system solver to take advantage of vector instructions in solving the linear system. The vector speedups for the other two parts, however, are very satisfactory. It deserves mentioning that in generating $\mathbf{f}^{l}_{(e)}$ and assemblying $\mathbf{f}$, we employed the VMS approach which yields a much better vector performance than the VSS approach. Table 3 shows the concurrency speedup S_k, defined to be the ratio of the CPU time spent under the VC execution mode of the entire program using only one processor to that using k processors and the efficiency E_k ($= S_k/k$), the ratio of the concurrency speedup S_k to the number of processors k. Figure 6 plots the speedup against the number of processors used. As seen from Table 3, the concurrency speedups observed using 2, 4, and 8 processors are 1.97, 3.68, and 6.61, respectively. This impressive performance clearly indicates the parallelizability of FSM on multiprocessors when the number of harmonic terms used matches the number of processors available.

CONCLUSIONS

The effectiveness and parallelizability of the finite strip method (FSM) for the dynamic analysis of a class of Mindlin plates have been addressed and vector/parallel implementations presented. The performance of this method on the Alliant FX/80 has also been tested using a rectangular plate that is simply supported on all edges and is subject to a uniformly distributed triangular loading. From the experiments performed, we have obtained concurrency speedups of 1.97, 3.68, and 6.61 using 2, 4, and 8 processors, respectively. These speedups are satisfactory and very encouraging. It clearly demonstrates the superiority of FSM in a parallel processing environment. For vectorization, good performance has also been observed for the Newmark integration scheme and for the generation/assemblage process using the VMS (vectorization across multiple strips) approach. In summary, we conclude that, although vector performance during the solution stage may be hindered by the small semi-bandwidth of the subsystems if a direct solver is employed, FSM is highly parallelizable and, therefore, suitable for computation on multiprocessor or multicluster computers. This is especially true when the problem requires a large number of harmonic terms to yield accurate results.

References

[1] Alliant Computer Systems Corporation, *FX/FORTRAN Programmer's Handbook*, Alliant Computer Systems Corporation, Acton, Massachusetts, 1987.

[2] A. Assadi-Lamouki and T. Krauthammer, An explicit finite difference approach for the Mindlin plate analysis, *Computers & Structures*, Vol.31, No.4 (1989), pp. 487-494.

[3] K. J. Bathe and E. L. Wilson, *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976.

[4] P. R. Benson and E. Hinton, A thick finite strip solution for static, free vibration and stability problems, *Int. J. for Numer. Meth. in Eng.*, 10 (1976), pp. 665-678.

[5] J. M. Canet, B. Suárez, and E. Oñate, Dynamic analysis of structures using a Reissner-Mindlin finite strip formulation, *Computers & Structures*, Vol.31, No.6 (1989), pp. 967-975.

[6] Y-K. Cheung, The finite strip method in the analysis of elastic plates with two opposite simply supported ends, *Proc. Inst. Civ. Eng.*, 40(1968), pp. 1-7.

[7] Y-K. Cheung, Finite strip method analysis of elastic slabs, *ASCE J. of Mechanics Div.*, 94 (1968), pp. 1365-1378.

[8] Y-K. Cheung, *Finite Strip Method in Structural Analysis*, Pergamon Press, New York, 1976.

[9] H-C. Chen and A-F. He, Implementation of the finite strip method for structural analysis on a parallel computer, *Proc. 1990 Int'l. Conf. on Parallel Processing*, Vol. III: Algorithms and Applications (ed. P-C. Yew), August 1990, pp. 372-373.

[10] A. R. Cusens and Y. C. Loo, Applications of the finite strip method in the analysis of concrete box bridges, *Proc. Inst. Civ. Eng.*, 57-II (1974), pp. 251-273.

[11] E. Davidson, D. Kuck, D. Lawrie, and A. Sameh, Supercomputing tradeoffs and the Cedar system, *CSRD Tech. Rept. 577*, Center for Supercomputing Research and Development, University of Illinois at Urbana-Champaign, 1986.

[12] J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, *LINPACK User's Guide*, SIAM, 1979.

[13] C. Farhat and E. Wilson, A parallel active column equation solver, *Computers & Structures*, Vol.28, No.2 (1988), pp. 289-304.

[14] D. G. Fertis, *Dynamics and Vibration of Structures*, John Wiley & Sons, New York, 1973.

[15] D. Goehlich, L. Komzsik, R. E. Fulton, Application of a parallel equation solver to static FEM problems, *Computers & Structures*, Vol.31, No.2 (1989), pp. 121-129.

[16] K. H. Huebner, *The Finite Element Method for Engineers*, John Wiley & Sons, New York, 1975.

[17] R. D. Mindlin, Influence of rotatory inertia and shear on flexural motions of isotropic, elastic plates, *J. of Applied Mechanics*, 18 (1951), pp. 31-38.

[18] E. Oñate and B. Suarez, A unified approach for the analysis of bridges, plates and axisymmetric shells using the linear Mindlin strip element, *Computers & Structures*, 17 (1983), pp. 407-426.

[19] E. Oñate and B. Suarez, A comparison of the linear quadratic and cubic Mindlin strip elements for the analysis of thick and thin plates, *Computers & Structures*, 17 (1983), pp. 427-439.

[20] J. A. Puckett and R. J. Schmidt, Finite strip method for groundwater modeling in a parallel computing environment, *Eng. Comput.*, 7 (1990), pp. 167-172.

[21] S. P. Timoshenko and J. M. Gere, *Mechanics of Materials*, Van Nostrand Co., New York, 1972.

[22] O. C. Zienkiewicz, *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.

[23] D. Zois, Parallel processing techniques for FE analysis: system solution, *Computers & Structures*, Vol.28, No.2 (1988), pp. 261-274.

S7-64
197139

N94-23680

DOMAIN DECOMPOSITION METHODS FOR NONCONFORMING FINITE ELEMENT SPACES OF LAGRANGE-TYPE*

Lawrence C. Cowsar
Department of Computational and Applied Mathematics
Rice University
Houston, Texas

SUMMARY

In this article, we consider the application of three popular domain decomposition methods to Lagrange-type nonconforming finite element discretizations of scalar, self-adjoint, second order elliptic equations. The additive Schwarz method of Dryja and Widlund, the vertex space method of Smith, and the balancing method of Mandel applied to nonconforming elements are shown to converge at a rate no worse than their applications to the standard conforming piecewise linear Galerkin discretization. Essentially, the theory for the nonconforming elements is inherited from the existing theory for the conforming elements with only modest modification by constructing an isomorphism between the nonconforming finite element space and a space of continuous piecewise linear functions.

INTRODUCTION

We consider the convergence properties of domain decomposition methods applied to Lagrange-type nonconforming finite element discretizations of scalar, self-adjoint, second order elliptic problems. An isomorphism between the nonconforming finite element space with the natural norm induced by the elliptic problem and a conforming piecewise linear space with the H^1-seminorm is constructed. Using the isomorphism, we are able to apply the existing analysis of domain decomposition methods for conforming elements to nonconforming elements with only modest modifications. As examples of this technique, we show that the operators arising in three popular domain decomposition methods, specifically, the additive Schwarz method of Dryja and Widlund [1], the vertex space method of Smith [2], and balancing method of Mandel [3], applied to nonconforming finite elements have condition numbers that satisfy the same bounds as the ones given in [4] and [5] for conforming finite elements.

The same technique was used in [6] and [7] to analyze the rate of convergence of balancing domain decomposition and the standard additive Schwarz method for the dual-variable mixed finite element formulation. Moreover, as a corollary of the analysis of Smith's method for the nonconforming spaces presented in this paper, we have a new bound for Smith's method applied to mixed finite elements.

*This work was supported in part by the National Science Foundation under Grant No. DMS-9112847.

After the research for this paper was completed, the author was made aware of some related work done concurrently by Sarkis [8]. In particular, the isomorphism used herein was independently suggested by Sarkis for linear nonconforming elements. In [8], Sarkis constructs and analyzes special coarse spaces such that when the overlapping additive Schwarz method is applied, the condition number of the resulting operator is bounded by a constant times $(1 + \log(H/h))(1 + H/\delta)$ in both two and three dimensions. Here H and h are the characteristic sizes of the subdomains and mesh, respectively, and δ is a measure of the overlap of subdomains. The notable characteristic of Sarkis' bound is that the constant is independent of jumps in the coefficients across subdomain boundaries. If the techniques of this paper were used to derive bounds that were independent of the jumps in coefficients, the resulting bound would include one log factor in two dimensions using [1, 9], but two logs in three dimensions using [5, 10, 11].

The remainder of this paper is divided into six sections. In the next section, we set some notation, formulate the nonconforming problem, and construct an equivalent representation in terms of the nodal values. In Section 3, we construct an isomorphism between the nonconforming space and a continuous space of piecewise linear functions. The isomorphism is used in Section 4 to analyze the rate of convergence of the Dryja-Widlund additive Schwarz method. In the last three sections, we consider the substructuring methods of Smith and Mandel applied to the nonconforming problem.

PRELIMINARIES

We consider the following self-adjoint, uniformly elliptic problem for p on the polygonal domain $\Omega \subset \mathbb{R}^n$, $n = 2, 3$, with boundary $\partial\Omega$:

$$-\nabla \cdot A \nabla p = f \quad \text{in } \Omega, \qquad p = 0 \quad \text{on } \partial\Omega, \tag{1}$$

where A is a uniformly positive definite, bounded, symmetric second order tensor, and $f \in L^2(\Omega)$. The uniform ellipticity of (1) implies the existence of positive constants c_*, c^* such that the following bound holds:

$$c_* \xi^T \xi \le \xi^T A(x) \xi \le c^* \xi^T \xi \quad \forall \xi \in \mathbb{R}^n, \forall x \in \Omega. \tag{2}$$

In order to set a length scale, we assume that the diameter of Ω is one. We introduce a two level quasi-regular triangulation of Ω: a division first into subdomains $\{\Omega_i\}_{i=1}^M$ with diameter $O(H)$, and a refinement of the first into elements with diameter $O(h)$. Following [12], define the scaled Sobolev norms

$$\|u\|_{1,\Omega_i}^2 = |u|_{1,\Omega_i}^2 + \frac{1}{H^2}\|u\|_{0,\Omega_i}^2, \qquad \|u\|_{1/2,\partial\Omega_i}^2 = |u|_{1/2,\partial\Omega_i}^2 + \frac{1}{H}\|u\|_{0,\partial\Omega_i}^2$$

where

$$\|u\|_{0,\Omega_i}^2 = \int_{\Omega_i} |u(x)|^2 \, dx, \qquad \|u\|_{0,\partial\Omega_i}^2 = \int_{\partial\Omega_i} |u(s)|^2 \, ds,$$

$$|u|_{1,\Omega_i}^2 = \int_{\Omega_i} | \nabla u(x)|^2 \, dx, \qquad |u|_{1/2,\partial\Omega_i}^2 = \int_{\partial\Omega_i}\int_{\partial\Omega_i} \frac{| u(t) - u(s) |^2}{| t - s |^n} \, dt \, ds.$$

Let $\mathcal{N}(\Omega)$ be a finite dimensional nonconforming finite element space of Lagrange-type defined subordinate to the triangulation $\mathcal{T}$ that vanishes at all degrees of freedom on $\partial\Omega$. Since $\mathcal{N}(\Omega)$ is of Lagrange-type, the elements in $\mathcal{N}(\Omega)$ may be expressed in terms of a nodal basis, and we may identify an element in $\mathcal{N}(\Omega)$ with the values it attains at the nodal points. For convenience, we assume that the subdomains and the elements are triangular in two dimensions or tetrahedral in three dimensions. Extensions to other shape regular decompositions are straightforward.

We consider the problem of finding $p_h \in \mathcal{N}(\Omega)$ such that

$$d(p_h, q_h) = \int_\Omega f q_h \, dx \quad \forall q_h \in \mathcal{N}(\Omega), \tag{3}$$

where d is the generalized Dirichlet form:

$$d(p_h, q_h) = d_\Omega(p_h, q_h), \qquad d_{\Omega'}(p_h, q_h) \equiv \sum_{\tau \in \mathcal{T},\, \tau \subset \Omega'} \int_\tau A \nabla p_h \cdot \nabla q_h \, dx.$$

We now introduce several conventions used in this paper. In this paper, we shall only be concerned with the solution of this finite dimensional problem, and will henceforth drop the "h" subscript.

Having defined a parent finite element space of functions $\mathcal{X}(\Omega)$ with a nodal basis and a set $\Omega' \subset \Omega$, we will simply write $\mathcal{X}(\Omega')$ for the restriction of $\mathcal{X}(\Omega)$ to Ω', i.e.

$$\mathcal{X}(\Omega') = \{\phi_{|\Omega'} \mid \phi \in \mathcal{X}(\Omega)\}.$$

By an abuse of notation, we consider an element $\phi \in \mathcal{X}(\Omega')$ also to be an element of $\mathcal{X}(\Omega)$ by setting ϕ to zero at all nodes outside of Ω'.

We will write $\mathcal{Q}_1 \simeq \mathcal{Q}_2$ if two quadratic forms $\mathcal{Q}_1$ and $\mathcal{Q}_2$ with the same domain $\mathcal{D}$ are *equivalent*, i.e. if there exists constants $c_1, c_2 > 0$ such that

$$c_1 \mathcal{Q}_1(\phi, \phi) \le \mathcal{Q}_2(\phi, \phi) \le c_2 \mathcal{Q}_1(\phi, \phi), \quad \forall \phi \in \mathcal{D}.$$

In what follows, C will be used to denote a generic constant that may not be the same from one line to the next. This constant, as well as the constants involved in the equivalence of quadratic forms, will always be independent of h and H, but can depend on the constants in (2), the shape regularity of the subdomains, the degree of the nonconforming finite elements, and the regularity of the triangulation.

To conclude this section, we prove a lemma that provides an equivalent quadratic form for $d(\cdot,\cdot)$ in terms of the nodal degrees of freedom. The proof of this lemma was suggested by Joseph Pasciak in the context of the mixed finite methods considered in [6, 7].

Lemma 1 *Let $\Omega' \subseteq \Omega$ be the union of elements of $\mathcal{T}$. And let $A(x) = \alpha(x)\widehat{A}(x)$, where α is a positive, piecewise constant function with value α_τ on $\tau \in \mathcal{T}$. Then for every $p \in \mathcal{N}(\Omega')$,*

$$d_{\Omega'}(p, p) \simeq \sum_{\substack{\tau \in \mathcal{T}, \\ \tau \subset \Omega'}} \alpha_\tau |\tau|^{1-2/n} \sum_{\substack{\text{nodes}: \\ n_i, n_j \in \tau}} (p(n_i) - p(n_j))^2. \tag{4}$$

The constants that appear in the definition of the equivalence do not depend on the constants in (2), but rather on constants that arise when A is replaced by $\widehat{A}$.

Proof. The local kernel of $d_\tau(\cdot,\cdot)$ in $\mathcal{N}(\tau)$ is exactly the constant functions on τ since for $p \in \mathcal{N}(\tau)$

$$d_\tau(p, q) = 0 \quad \forall q \in \mathcal{N}(\tau), \quad \Longleftrightarrow \quad \nabla p = 0.$$

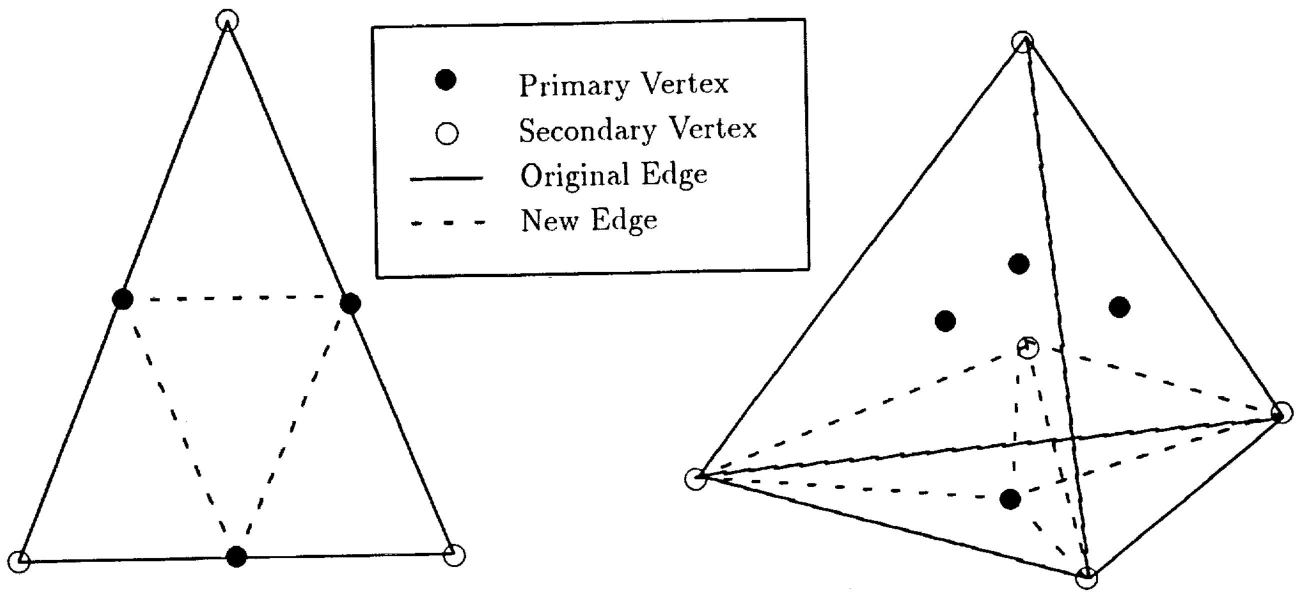


Figure 1: Refinement of the 2D P-1 element and a partial refinement of the 3D P-1 element.

Hence, $(d_\tau(\cdot,\cdot))^{1/2}$ is a norm on $\mathcal{N}(\tau)/\mathbb{R}$. Since all norms are equivalent on finite dimensional spaces, we see that

$$d_\tau(p,p) \simeq \alpha_\tau |\tau|^{1-2/n} \sum_{\substack{\text{nodes}:\\ n_i, n_j \in \tau}} (p(n_i) - p(n_j))^2,$$

by a simple scaling argument. The proof is completed by summing over the elements of $\mathcal{T}$ in Ω'. $\square$

A CONFORMING EQUIVALENCE

In this section, we construct a conforming space that is isomorphic to $\mathcal{N}(\Omega)$ using the techniques in [6, 7] and recall some basic properties about the isomorphism.

Given an element $\tau \in \mathcal{T}$, let $\hat{\mathcal{T}}_\tau$ be a subtriangulation of τ such that the vertices of the subtriangulation include the vertices of τ and the nodal points in τ pertaining to the degrees of freedom of $\mathcal{N}(\tau)$. Every element in the new triangulation should have at least one vertex that corresponds to a nodal point of $\mathcal{N}(\tau)$. Moreover, the subtriangulations should be constructed in such a way that the union of subtriangulations gives rise to a refined quasi-regular triangulation of Ω which we denote by

$$\hat{\mathcal{T}} \equiv \bigcup_{\tau\in\mathcal{T}} \hat{\mathcal{T}}_\tau.$$

A vertex of $\hat{\mathcal{T}}$ will be called *primary* if it was a nodal point corresponding to a degree of freedom of $\mathcal{N}(\Omega)$; otherwise, we call the vertex *secondary*. We say that two vertices of the triangulation $\hat{\mathcal{T}}$ are *adjacent* if there exists an edge of $\hat{\mathcal{T}}$ connecting the vertices. An example of the subtriangulation of the P-1 element that has nodal degrees of freedom at the center of its edges (faces) is given in Figure 1.

Let $U_h(\Omega)$ denote the space of continuous piecewise linear functions subordinate to the triangulation $\hat{\mathcal{T}}$ that vanish on $\partial\Omega$. For $\Omega' \subset \Omega$, a union of elements, define $U_h(\Omega')$ by restriction, i.e.

$$U_h(\Omega') = \{u_{|\Omega'} \mid u \in U_h(\Omega)\}.$$

Since the functions in $U_h(\Omega')$ are naturally parameterized by the values they attain at the vertices, we can define a pseudo-interpolation operator $\tilde{\mathcal{I}}^{\Omega'}$ into $U_h(\Omega')$ for any function ϕ defined at the primary vertices contained in Ω' by

$$\tilde{\mathcal{I}}^{\Omega'}\phi(x) = \begin{cases} 0, \text{ if } x \in \partial\Omega' \cap \partial\Omega; \\ \phi(x), \text{ if } x \text{ is a primary vertex not in } \partial\Omega' \cap \partial\Omega; \\ \text{The average of all adjacent primary vertices on the boundary} \\ \quad \text{of } \Omega', \text{ if } x \text{ is a secondary vertex in } \partial\Omega' \setminus \partial\Omega; \\ \text{The average of all adjacent primary vertices, if } x \text{ is a secondary} \\ \quad \text{vertex in the interior of } \Omega'; \\ \text{The continuous piecewise linear interpolant of the above vertex} \\ \quad \text{values, if } x \text{ is not a vertex of } \hat{\mathcal{T}}. \end{cases} \tag{5}$$

Since $\tilde{\mathcal{I}}^{\Omega'}$ is well defined for any function defined at the primary vertices, by an abuse of notation, we can understand $\tilde{\mathcal{I}}^{\Omega'}$ both as a map from $\mathcal{N}(\Omega')$ into $U_h(\Omega')$ and a map from $U_h(\Omega')$ into itself.

For any Ω' that is the union of elements in $\mathcal{T}$, let $\tilde{U}_h(\Omega') \subset U_h(\Omega')$ denote the range of $\tilde{\mathcal{I}}^{\Omega'}$; that is,

$$\tilde{U}_h(\Omega') = \{\psi = \tilde{\mathcal{I}}^{\Omega'}q, q \in \mathcal{N}(\Omega)\}.$$

We now prove that $\tilde{\mathcal{I}}^{\Omega'} : \mathcal{N}(\Omega') \to \tilde{U}_h(\Omega')$ preserves the norm induced by the bilinear form $d_{\Omega'}(\cdot,\cdot)$ on $\mathcal{N}(\Omega')$ and the H^1-seminorm on $\tilde{U}_h(\Omega')$. Since $\tilde{\mathcal{I}}^{\Omega'}$ is a bijection between $\mathcal{N}(\Omega')$ and $\tilde{U}_h(\Omega')$ by construction, this proves that $\mathcal{N}(\Omega')$ and $\tilde{U}_h(\Omega')$ are isomorphic.

Theorem 2 *Let $\Omega' \subset \Omega$ be the union of elements. Then for all $p \in \mathcal{N}(\Omega')$,*

$$d_{\Omega'}(p,p) \simeq |\tilde{\mathcal{I}}^{\Omega'}p|^2_{1,\Omega'}. \tag{6}$$

Proof. This proof is an expanded version of the proof given in [7]. Recall that for $\phi \in U_h(\Omega')$,

$$|\phi|^2_{1,\Omega'} \simeq \sum_{\substack{\tau \in \hat{\mathcal{T}}, \\ \tau \subset \Omega'}} |\tau|^{1-2/n} \sum_{\substack{\text{vertices}: \\ v_i, v_j \in \tau}} (\phi(v_i) - \phi(v_j))^2. \tag{7}$$

By virtue of Lemma 1 and Equation (7), it is enough to show that

$$\sum_{\substack{\tau \in \mathcal{T}, \\ \tau \subset \Omega'}} |\tau|^{1-2/n} \sum_{\substack{\text{nodes}: \\ n_i, n_j \in \tau}} (p(n_i) - p(n_j))^2 \simeq \sum_{\substack{\tau \in \hat{\mathcal{T}}, \\ \tau \subset \Omega'}} |\tau|^{1-2/n} \sum_{\substack{\text{vertices}: \\ v_i, v_j \in \tau}} ((\tilde{\mathcal{I}}^{\Omega'}p)(v_i) - (\tilde{\mathcal{I}}^{\Omega'}p)(v_j))^2. \tag{8}$$

Since vertices of $\hat{\mathcal{T}}_\tau$ contain the nodal points of τ and $p = \tilde{\mathcal{I}}^{\Omega'}p$ at these points, we have

$$\sum_{\substack{\text{nodes}: \\ n_i, n_j \in \tau}} (p(n_i) - p(n_j))^2 \leq C \sum_{\hat{\tau} \in \hat{\mathcal{T}}_\tau} \sum_{\substack{\text{vertices}: \\ v_i, v_j \in \hat{\tau}}} \left(\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_i) - \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_j)\right)^2,$$

where the constant is controlled by the regularity of the subtriangulation. Hence, by summing over the elements of $\mathcal{T}$ in Ω', we conclude that the right hand side of (8) dominates the left hand side.

To prove that the left hand side dominates the right, we note that the differences in the right hand side are of three types: the difference at two primary vertices, the difference at two secondary vertices, and the difference at a primary and a secondary vertex. Since p and $\tilde{\mathcal{I}}^{\Omega'}p$ agree at primary vertices of $\hat{\mathcal{T}}$, the difference at two primary vertices occurs as a term in the left hand side. For two secondary vertices v_1, v_2 in an element $\tau \in \hat{\mathcal{T}}$ containing a primary vertex v_p, we see that

$$\left(\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_1) - \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_2)\right)^2 \leq 2\left(\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_1) - \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_p)\right)^2 + 2\left(\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_2) - \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_p)\right)^2.$$

Hence, it is enough to bound the difference at a secondary and primary vertex by terms in the left hand side of (8).

Let v_{n+1} be a secondary vertex with adjacent primary vertices $v_1, \ldots, v_n$, and let $p_j = p(v_j)$. Noting that for $j = 1, \ldots, n$

$$\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_j) = p_j, \qquad \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_{n+1}) = \frac{1}{n}\sum_{j=1}^{n}\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_j) = \frac{1}{n}\sum_{j=1}^{n} p_j,$$

we see that

$$\left(\left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_{n+1}) - \left(\tilde{\mathcal{I}}^{\Omega'}p\right)(v_i)\right)^2 = \frac{1}{n^2}\left(\sum_{j=1}^{n}(p_j - p_i)\right)^2 \leq \frac{n}{n^2}\sum_{j=1}^{n}(p_j - p_i)^2,$$

by the Cauchy-Schwarz inequality. The proof is completed by summing over all triangles of $\hat{\mathcal{T}}$. The number of such terms, and hence the constant in the bound, is controlled since the regularity of the mesh implies that there is an a priori maximum number of adjacent elements that can share a secondary point. □

Using the techniques in the proof of Theorem 2, the following lemma is easy to prove.

Lemma 3 *There exists a constant C depending only on the regularity of the triangulation $\mathcal{T}$ and the degree of the nonconforming space such that for any $\Omega' \subset \Omega$, the union of elements of $\mathcal{T}$,*

$$|\tilde{\mathcal{I}}^{\Omega'}\phi|_{k,\Omega'} \leq C|\phi|_{k,\Omega'} \quad \forall\phi \in U_h(\Omega'),\ k = 0, 1. \tag{9}$$

THE DRYJA-WIDLUND ADDITIVE SCHWARZ METHOD

The presentation in this section and the next follows the treatment of Schwarz methods given by Dryja and Widlund in [4]. We concentrate only on the additive Schwarz methods with exact solves. The convergence rate of the multiplicative Schwarz method may be estimated in terms of the same quantities (see [13]) and is easily worked out. Extensions to inexact solves are likewise direct.

Recall that the additive Schwarz method with exact solves for (3) is completely determined by a decomposition of the finite element space $\mathcal{N}(\Omega) = \mathcal{N}_0 + \mathcal{N}_1 + \ldots + \mathcal{N}_M$. For each subspace $\mathcal{N}_i$, define an operator $P_i : \mathcal{N}(\Omega) \to \mathcal{N}_i$ by

$$d(P_i p, q) = d(p, q) \quad \forall q \in \mathcal{N}_i. \tag{10}$$

The *additive Schwarz algorithm* with exact solves for (3) involves the solution of

$$Pp = \hat{f}, \quad P \equiv \sum_{i=0}^{M} P_i, \quad \hat{f} \equiv \sum_{i=0}^{M} f_i, \tag{11}$$

where $f_i \in \mathcal{N}_i$ is defined by

$$d(f_i, q) = \int_\Omega fq\, dx \quad \forall q \in \mathcal{N}_i.$$

Abstract bounds on the condition number of P have been derived in terms of two quantities, C_0 and the spectral radius of $\mathcal{E}$, which we now define. Let C_0 be a constant such that for every $p \in \mathcal{N}$ there exists a representation $p = \sum_{i=0}^{M} p_i$ with $p_i \in \mathcal{N}_i$ satisfying

$$\sum_{i=0}^{M} d(p_i, p_i) \leq C_0 d(p, p). \tag{12}$$

Let $\rho(\mathcal{E})$ denote the spectral radius of $\mathcal{E} = \{\epsilon_{ij}\}$, the matrix of strengthened Cauchy-Schwarz constants; that is, ϵ_{ij} is the smallest constant for which

$$|d(p_i, p_j)| \leq \epsilon_{ij} d(p_i, p_i)^{\frac{1}{2}} d(p_j, p_j)^{\frac{1}{2}} \quad \forall p_i \in \mathcal{N}_i,\ \forall p_j \in \mathcal{N}_j,\ i, j \geq 1. \tag{13}$$

The next theorem, due to Dryja and Widlund [14], bounds the condition number of the additive Schwarz method in terms of C_0 and $\rho(\mathcal{E})$:

Theorem 4 *The eigenvalues and the condition number $\kappa(P)$ of P satisfy*

$$\lambda_{\min}(P) \geq C_0^{-1}, \quad \lambda_{\max}(P) \leq (\rho(\mathcal{E}) + 1), \quad \kappa(P) \leq C_0(\rho(\mathcal{E}) + 1). \tag{14}$$

To construct the decomposition of $\mathcal{N}(\Omega)$ to be used in our application of the additive Schwarz algorithm for nonconforming elements, we first create an overlapping decomposition of the domain Ω by extending each subdomain Ω_i to a larger region Ω_i' which is also the union of elements of $\mathcal{T}$. We characterize the extent of the overlap of the partition $\{\Omega_i'\}_{i=1}^{M}$ by δ, where

$$\delta = \min_{i=1,\ldots,M} \operatorname{dist}(\partial\Omega_i \setminus \partial\Omega, \partial\Omega_i' \setminus \partial\Omega).$$

The decomposition $\{\Omega_i'\}_{i=1}^{M}$ gives rise to a natural decomposition of $\mathcal{N}(\Omega)$ by letting $\mathcal{N}_i \subset \mathcal{N}(\Omega)$ denote the set of functions that vanish at all nodes in the closure of $(\Omega \setminus \Omega_i')$. In order to provide a mechanism for global exchange of information between subdomains so as to enhance the rate of convergence, we also use a low dimensional space defined by

$$\mathcal{N}_0 = \{p \in \mathcal{N}(\Omega) \mid p = \mathcal{I}^{\mathcal{N}}\phi, \phi \in U_H(\Omega)\},$$

where $\mathcal{I}^{\mathcal{N}}$ is nodal interpolation into $\mathcal{N}(\Omega)$, and $U_H(\Omega)$ is the space of continuous functions that are linear on each subdomain Ω_i. Note that the subspaces for the nonconforming space are exactly the nodal interpolants of the standard decomposition of the conforming space $U_h(\Omega)$, namely, $U_h(\Omega) \cap H_0^1(\Omega_i')$.

In the following lemma we recall the crux of the proof due to Dryja and Widlund (Theorem 3 of [4]) that the Schwarz method applied to the conforming Galerkin discretization has a condition number that is $O(1 + (H/\delta))$.

Lemma 5 *For every $\phi \in U_h(\Omega)$, there exists a decomposition $\phi = \sum_{i=0}^{M} \phi_i$ with $\phi_0 \in U_H(\Omega)$, $\phi_i \in U_h(\Omega) \cap H_0^1(\Omega_i')$, $1 \le i \le M$ and a constant C independent of h, H, and δ, such that*

$$\sum_{i=0}^{M} |\phi_i|_{1,\Omega}^2 \le C\left(1 + \frac{H}{\delta}\right) |\phi|_{1,\Omega}^2. \tag{15}$$

We now show that the application of the Schwarz method to the nonconforming space converges at the same rate.

Theorem 6 *The condition number $\kappa(P)$ of the additive Schwarz operator P defined by (11) induced by the decomposition $\mathcal{N}(\Omega) = \mathcal{N}_0 + \ldots + \mathcal{N}_M$ of the nonconforming finite element space satisfies*

$$\kappa(P) \le C\left(1 + \frac{H}{\delta}\right).$$

The constant C is independent of h, δ, and H.

Proof. The verification that the largest eigenvalue of P is bounded by a constant is standard. Since $d(p_i, p_j) \equiv 0$ for $p_i \in \mathcal{N}_i$, $p_j \in \mathcal{N}_j$ with $\Omega_i' \cap \Omega_j' = \emptyset$, P may be written as the sum of an a priori bounded number of disjoint projections. Since projections have unit norm, a constant bound on the largest eigenvalue of P is immediate. See, e.g., Lemma 3.1 of [2].

For $p \in \mathcal{N}(\Omega)$, let $(\tilde{\mathcal{I}}^{\Omega} p)_i$ denote the decomposition of $\tilde{\mathcal{I}}^{\Omega} p \in U_h(\Omega)$ arising in Lemma 5, and set $p_i = \mathcal{I}^{\mathcal{N}}((\tilde{\mathcal{I}}^{\Omega} p)_i)$. It is easy to check that $p_i \in \mathcal{N}_i$ and $p = \sum_{i=0}^{M} p_i$. Using Theorem 2 and Lemma 3, we see that for $i = 0, \ldots, M$,

$$d(p_i, p_i) \le C |\tilde{\mathcal{I}}^{\Omega}((\tilde{\mathcal{I}}^{\Omega} p)_i)|_{1,\Omega}^2 \le C |(\tilde{\mathcal{I}}^{\Omega} p)_i|_{1,\Omega}^2.$$

Summing and applying Lemma 5 and Theorem 2, we conclude that

$$\sum_{i=0}^{M} d(p_i, p_i) \le C \sum_{i=0}^{M} |(\tilde{\mathcal{I}}^{\Omega} p)_i|_{1,\Omega}^2 \le C\left(1 + \frac{H}{\delta}\right) |\tilde{\mathcal{I}}^{\Omega} p|_{1,\Omega}^2 \le C\left(1 + \frac{H}{\delta}\right) d(p, p).$$

Hence, C_0 in (12) is bounded by $C(1 + H/\delta)$. An application of Theorem 4 completes the proof. □

SUBSTRUCTURING DOMAIN DECOMPOSITION

The remaining two methods considered in this paper are domain decomposition methods applied to a reduced problem involving only the degrees of freedom on the internal interfaces of subdomains $\Gamma = \cup_{i=1}^{M} \partial\Omega_i \setminus \partial\Omega$. Following [4], we recall the construction of the reduced problem. Since $\mathcal{N}(\Omega)$ is of Lagrange-type, we may associate with functions $p, q \in \mathcal{N}(\Omega)$ the vectors of values they attain at the nodes. Let x and y denote the vectors of nodal values of p and q, respectively, and $x^{(i)}, y^{(i)}$ the subvectors of degrees of freedom in $\overline{\Omega}_i$. Let $D^{(i)}$ denote the local stiffness matrix arising from $d_{\Omega_i}(\cdot, \cdot)$, and let D denote the global stiffness matrix, i.e.

$$x^{(i)^T} D^{(i)} y^{(i)} = d_{\Omega_i}(p, q), \quad x^T D y = \sum_{i=1,\ldots,M} x^{(i)^T} D^{(i)} y^{(i)} = d(p, q).$$

For each subdomain, we can partition the degrees of freedom $x^{(i)}$ into two sets, the ones related to nodes on the boundary of Ω_i denoted $x_B^{(i)}$, and the ones corresponding to nodes in the interior of Ω_i denoted $x_I^{(i)}$. Such a partitioning induces a partitioning of $D^{(i)}$ given by

$$x^{(i)^T} D^{(i)} y^{(i)} = \begin{pmatrix} x_I^{(i)} \\ x_B^{(i)} \end{pmatrix}^T \begin{pmatrix} D_{II}^{(i)} & D_{IB}^{(i)} \\ D_{IB}^{(i)T} & D_{BB}^{(i)} \end{pmatrix} \begin{pmatrix} y_I^{(i)} \\ y_B^{(i)} \end{pmatrix}.$$

The interior unknowns of each subdomain may be eliminated in terms of the boundary unknowns. The resulting matrix, S, is the Schur complement with respect to the interface unknowns defined by

$$x_B^T S y_B = \sum_{i=1,\ldots,M} x_B^{(i)^T} S^{(i)} y_B^{(i)}, \quad \text{where} \quad S^{(i)} = D_{BB}^{(i)} - D_{IB}^{(i)T} (D_{II}^{(i)})^{-1} D_{IB}^{(i)}.$$

It will be convenient to work with the bilinear forms induced by S and $S^{(i)}$, and so we define

$$s(p,q) = x_B^T S y_B, \quad s_i(p,q) = x_B^{(i)^T} S^{(i)} y_B^{(i)}.$$

For a function $p \in \mathcal{N}(\Omega)$, we note that unlike conforming spaces, the restriction of p to the interfaces, $p_{|\Gamma}$, is not solely determined by the nodal values on Γ since $\mathcal{N}(\Omega)$ is nonconforming. Hence, we are careful to understand $\mathcal{N}(\Gamma)$ as a subset of $\mathcal{N}(\Omega)$ parameterized by the nodal values on Γ consisting of the discrete harmonic extension of the nodal values to the interior of the subdomains. Specifically, if $p \in \mathcal{N}(\Gamma)$ has the vector of nodal values $x_B^{(i)}$ on $\partial\Omega_i$, then $p_{|\Omega_i}$ is the function associated with the vector of nodal values $(x_I^{(i)}, x_B^{(i)})^T$ where $x_I^{(i)}$ satisfies

$$D_{II}^{(i)} x_I^{(i)} = -D_{IB}^{(i)} x_B^{(i)}.$$

A linear functional g is easily constructed such that finding $p \in \mathcal{N}(\Gamma)$ satisfying

$$s(p,q) = g(q) \quad \forall q \in \mathcal{N}(\Gamma) \tag{16}$$

is essentially equivalent to (3).

We now construct a conforming space of functions that is isomorphic to $\mathcal{N}(\Gamma)$ with the norm induced by the bilinear form $s(\cdot,\cdot)$. Let $U_h(\Gamma)$ denote the restriction of $U_h(\Omega)$ to $\cup_{i=1}^M \partial\Omega_i$. Since functions in $U_h(\Gamma)$ vanish on $\partial\Omega$ (because functions in $U_h(\Omega)$ do), functions in $U_h(\Gamma)$ can be parameterized in the natural nodal basis by the values they attain at the vertices of $\hat{\mathcal{T}}$ in Γ. Analogous to (5), for Γ' the union of edges (and faces in 3D) in the triangulation $\mathcal{T}$ and ϕ a function defined at the primary vertices in Γ', define a pseudo-interpolant $\tilde{\mathcal{I}}^{\Gamma'}\phi \in U_h(\Gamma')$ by

$$\tilde{\mathcal{I}}^{\Gamma'}\phi(x) = \begin{cases} 0, \text{ if } x \in \Gamma' \cap \partial\Omega; \\ \phi(x), \text{ if } x \text{ is a primary vertex not in } \Gamma' \cap \partial\Omega; \\ \text{The average of all adjacent primary vertices on } \Gamma' \text{ if } x \text{ is a} \\ \quad \text{secondary vertex on } \Gamma'; \\ \text{The continuous piecewise linear interpolant of the above vertex} \\ \quad \text{values, if } x \text{ is not a vertex of } \hat{\mathcal{T}}. \end{cases} \tag{17}$$

Note that if $\Gamma' = \partial\Omega'$, then $\tilde{\mathcal{I}}^{\Gamma'}\phi = (\tilde{\mathcal{I}}^{\Omega'}\tilde{\phi})_{|\partial\Omega'}$ for all $\tilde{\phi}$ in $\mathcal{N}(\Omega')$ that agree with ϕ at the nodal degrees of freedom of $\partial\Omega'$.

Since $\tilde{\mathcal{I}}^{\Gamma'}$ is well defined for any function defined at primary vertices, by an abuse of notation, we can understand $\tilde{\mathcal{I}}^{\Gamma'}$ both as a map from $\mathcal{N}(\Gamma')$ into $U_h(\Gamma')$ and a map from $U_h(\Gamma')$ into $U_h(\Gamma')$. We denote the range of $\tilde{\mathcal{I}}^{\Gamma'}$ by

$$\tilde{U}_h(\Gamma') = \{(\tilde{\mathcal{I}}^{\Gamma'}\psi)_{|\Gamma'} | \psi \in U_h(\Gamma')\}.$$

The equivalences in the following lemma are a combination of the standard trace theorem and an extension theorem for $\tilde{U}_h(\partial\Omega_i)$. In particular, the proof of this lemma given in [6] shows that the space $\tilde{U}_h(\Omega_i)$ is rich enough to inherit the Extension Theorem of Widlund [15] from $U_h(\Omega_i)$.

Lemma 7 *For $\hat{\phi} \in \tilde{U}_h(\partial\Omega_i)$,*

$$\|\hat{\phi}\|_{1/2,\partial\Omega_i} \simeq \inf_{\substack{\phi \in \tilde{U}_h(\Omega_i) \\ \phi_{|\partial\Omega_i} = \hat{\phi}}} \|\phi\|_{1,\Omega_i}, \qquad |\hat{\phi}|_{1/2,\partial\Omega_i} \simeq \inf_{\substack{\phi \in \tilde{U}_h(\Omega_i) \\ \phi_{|\partial\Omega_i} = \hat{\phi}}} |\phi|_{1,\Omega_i}. \tag{18}$$

Additionally, there exists a constant C independent of mesh parameters such that

$$|\tilde{\mathcal{I}}^{\partial\Omega_i}\hat{\phi}|_{k,\partial\Omega_i} \leq C|\hat{\phi}|_{k,\partial\Omega_i}, \quad \forall\hat{\phi} \in U_h(\partial\Omega_i),\ k = 0, 1/2. \tag{19}$$

The following theorem plays the role of Theorem 2 for the interface problem.

Theorem 8 *For all $p \in \mathcal{N}(\Gamma)$,*

$$s_i(p,p) \simeq |\tilde{\mathcal{I}}^{\partial\Omega_i}p|^2_{1/2,\partial\Omega_i}. \tag{20}$$

Proof. By a direct computation followed by an application of Theorem 2 and Lemma 7 noting that $\tilde{U}_h(\Omega_i) = \tilde{\mathcal{I}}^{\Omega_i}(\mathcal{N}(\Omega_i))$, we see that

$$s_i(p,p) = \inf_{\substack{\tilde{p} \in \mathcal{N}(\Omega_i) \\ \tilde{p}_{|\partial\Omega_i} = p_i}} d_{\Omega_i}(\tilde{p},\tilde{p}) \simeq \inf_{\substack{\tilde{p} \in \mathcal{N}(\Omega_i) \\ \tilde{p}_{|\partial\Omega_i} = p}} |\tilde{\mathcal{I}}^{\Omega_i}\tilde{p}|^2_{1,\Omega_i} \simeq |\tilde{\mathcal{I}}^{\partial\Omega_i}p|^2_{1/2,\partial\Omega}.$$

□

SMITH'S VERTEX SPACE METHOD

Smith's vertex space method [2] is an additive Schwarz method applied to the interface problem (16). The decomposition of $\mathcal{N}(\Gamma)$ is constructed slightly differently in two and three dimensions. In both cases, we first partition Γ into overlapping subsets based on its decomposition as the boundary of subdomains. In two dimensions, for each vertex V_j of Γ, let $\Gamma_\delta^{V_j}$ denote the set of points on Γ that are less than a distance δ from V_j. For each edge E_i of Γ, let $\Gamma_\delta^{E_i}$ denote the interior of the edge E_i. In three dimensions, for each vertex V_j, each edge E_i, and each face F_k of Γ, define $\Gamma_\delta^{V_j}$ as above, let $\Gamma_\delta^{F_k}$ denote the interior of the face F_k, and let $\Gamma_\delta^{E_i}$ denote the set of all points in strips of width δ on all faces which share the common edge E_i.

Understanding the set of faces to be empty in two dimensions, the decomposition of Γ into subsets induces a decomposition of $\mathcal{N}(\Gamma)$ by considering

$$\mathcal{N}(\Gamma) = \sum_{G \in \{H, E_i, V_j, F_k\}} \mathcal{N}(\Gamma_\delta^G),$$

where for $G \in \{E_i, V_j, F_k\}$, $\mathcal{N}(\Gamma_\delta^G) \subset \mathcal{N}(\Gamma)$ are those functions that vanish at all nodal points on Γ that are outside of the set Γ_δ^G, and $\mathcal{N}(\Gamma_\delta^H) \subset \mathcal{N}(\Gamma)$ are those functions that are the nodal interpolant of the restriction to Γ of continuous functions that are linear on each subdomain Ω_i and vanish on $\partial\Omega$.

The following lemma is the crux of the analysis of Smith's method by Dryja and Widlund [4] for conforming elements.

Lemma 9 *For every $\phi \in U_h(\Gamma)$, there exists a decomposition*

$$\phi = \sum_{G \in \{H, E_i, V_j, F_k\}} \phi_G$$

with $\phi_H \in U_H(\Gamma)$, $\phi_G \in U_h(\Gamma_\delta^G) = U_h(\Gamma) \cap H_0^1(\Gamma_\delta^G)$ for $G \in \{E_i, V_j, F_k\}$ such that

$$\sum_{G \in \{H, E_i, V_j, F_k\}} \sum_{i=1}^{M} |\phi_G|^2_{1/2, \partial\Omega_i} \leq C \left(1 + \log(H/\delta)\right)^2 \sum_{i=1}^{M} |\phi|^2_{1/2, \partial\Omega_i}. \tag{21}$$

The constant C is independent of the choice of ϕ, and the mesh parameters h, H, and δ.

Let $P_\Gamma : \mathcal{N}(\Gamma) \to \mathcal{N}(\Gamma)$ denote the additive Schwarz operator defined by (10) with the bilinear form $d(\cdot, \cdot)$ replaced by the interface form $s(\cdot, \cdot)$ and the decomposition of $\mathcal{N}(\Omega)$ replaced by the decomposition of $\mathcal{N}(\Gamma)$ described above. We now prove that the condition number of $\mathcal{N}_\Gamma$ for the nonconforming space has the same bound given in [4] for the similar operator for the conforming finite element space.

Theorem 10 *The condition number of the additive Schwarz operator P_Γ for Smith's decomposition for the nonconforming finite element discretization satisfies*

$$\kappa(P_\Gamma) \leq C((1 + \log(H/\delta))^2. \tag{22}$$

The constant C is independent of the mesh parameters h, H, and δ.

Proof. As in the proof of Theorem 6, P_Γ may be written as the sum of an a priori bounded number of disjoint projections, and so the largest eigenvalue of P_Γ is bounded by a constant.

To bound the smallest eigenvalue, we also proceed as in the proof of Theorem 6. For $p \in \mathcal{N}(\Gamma)$, set $p_G = \mathcal{I}_\Gamma^{\mathcal{N}}((\widetilde{\mathcal{I}}^\Gamma p)_G)$, $G \in \{H, E_i, V_j, F_k\}$, where $\mathcal{I}_\Gamma^{\mathcal{N}}$ is interpolation at the nodes on Γ into $\mathcal{N}(\Gamma)$ and $(\widetilde{\mathcal{I}}^\Gamma p)_G$ is the decomposition of $\widetilde{\mathcal{I}}^\Gamma p \in U_h(\Gamma)$ that arises in Lemma 9. Since $\widetilde{\mathcal{I}}^\Gamma p$ and p agree at the nodal degrees of freedom of $\mathcal{N}(\Gamma)$, and

$$\mathcal{N}(\Gamma_\delta^H) = \mathcal{I}_\Gamma^{\mathcal{N}}(U_H(\Gamma)), \qquad \mathcal{N}(\Gamma_\delta^G) = \mathcal{I}_\Gamma^{\mathcal{N}}(U_h(\Gamma_\delta^G)) \quad \forall G \in \{E_i, V_j, F_k\},$$

it is easy to check that

$$p = \sum_{G \in \{H, E_i, V_j, F_k\}} p_G.$$

Working one subdomain at a time and using Theorem 8 and Lemma 3, we see that for $G = H$ and for $G \in \{E_i, V_j, F_k\}$ such that $\Gamma_\delta^G \cap \partial\Omega_i \neq \emptyset$ we have

$$s_i(p_G, p_G) \leq C|\widetilde{\mathcal{I}}^{\partial\Omega_i} p_G|^2_{1/2, \partial\Omega_i} = C|\widetilde{\mathcal{I}}^{\partial\Omega_i}((\widetilde{\mathcal{I}}^\Gamma p)_G)|^2_{1/2, \partial\Omega_i} \leq C|(\widetilde{\mathcal{I}}^\Gamma p)_G|^2_{1/2, \partial\Omega_i}. \tag{23}$$

Assume that we can prove that there exists a constant independent of h, H and δ such that

$$\sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Gamma} p|_{1/2,\partial\Omega_i}^2 \leq C \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\partial\Omega_i} p|_{1/2,\partial\Omega_i}^2 \quad \forall p \in \mathcal{N}(\Gamma). \tag{24}$$

Then by summing (23) over subdomains and subspaces, noting that $s_i(p_G, p_G) = 0$ if $\Gamma_\delta^G \cap \partial\Omega_i = \emptyset$, and applying Lemma 9, Equation (24), and Theorem 8, we see that

$$\begin{aligned} \sum_{G\in\{H,E_i,V_j,F_k\}} s(p_G,p_G) &= \sum_{G\in\{H,E_i,V_j,F_k\}} \sum_{i=1}^{M} s_i(p_G,p_G) \leq C\,(1+\log(H/\delta))^2 \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Gamma} p|_{1/2,\partial\Omega_i}^2 \\ &\leq C\,(1+\log(H/\delta))^2 \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\partial\Omega_i} p|_{1/2,\partial\Omega_i}^2 \leq C\,(1+\log(H/\delta))^2\, s(p,p). \end{aligned}$$

The proof of the condition number bound now follows from an application of Theorem 4, and we are only left to verify (24).

Define a pseudo-interpolant $\tilde{\mathcal{I}}^{\Omega\setminus\Gamma} : \mathcal{N}(\Omega) \to U_h(\Omega)$ by (5), noting that the boundary of $\Omega \setminus \Gamma$ is $\partial\Omega \cup \Gamma$. Using the techniques in the proof of Theorem 2, it is easy to show that there exists a constant C_1 depending only on the regularity of the mesh and the degree of the nonconforming space such that

$$\sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Omega\setminus\Gamma} p|_{1,\Omega_i}^2 \leq C_1 \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Omega_i} p|_{1,\Omega_i}^2 \quad \forall p \in \mathcal{N}(\Omega).$$

By Lemma 7, for each $p \in \mathcal{N}(\Gamma)$ there exists an extension $p^E \in \mathcal{N}(\Omega)$ that agrees with p at the nodal points on Γ such that

$$|\tilde{\mathcal{I}}^{\Omega_i} p^E|_{1,\Omega_i}^2 \leq C |\tilde{\mathcal{I}}^{\partial\Omega_i} p|_{1/2,\partial\Omega_i}^2 \quad i = 1, \ldots, M.$$

Combining these results after another application of Lemma 7 with $\hat{\phi} = \tilde{\mathcal{I}}^{\Gamma} p$, we conclude that

$$\sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Gamma} p|_{1/2,\partial\Omega_i}^2 \leq C \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Omega\setminus\Gamma} p^E|_{1,\Omega_i}^2 \leq C \sum_{i=1}^{M} |\tilde{\mathcal{I}}^{\Omega_i} p^E|_{1,\Omega_i}^2 \leq C |\tilde{\mathcal{I}}^{\partial\Omega_i} p|_{1/2,\partial\Omega_i}^2,$$

which verifies (24). □

In [6], the interface form arising from the discretization by mixed finite elements of (1) was shown to satisfy Theorem 8 with $\mathcal{N}(\Gamma)$ replaced by the appropriate space of interelement multipliers. Hence, the proof given above is applicable to discretization by mixed finite elements, and we arrive at the following corollary.

Corollary 11 *The application of Smith's decomposition method to the dual-variable mixed finite element formulation discussed in [6] results in an operator whose condition number grows at worst like* $O((1+\log(H/\delta))^2)$.

BALANCING DOMAIN DECOMPOSITION

As the final domain decomposition method considered in this paper, we investigate the balancing domain decomposition method of Mandel [3] applied to nonconforming finite elements. The method involves the iterative solution (usually by conjugate gradients) of (16) preconditioned by the balancing preconditioner described in Algorithm 1 below. Each iteration involves the solution of a local problem with Dirichlet data, a local problem with Neumann data, and a "coarse-grid" problem to propagate information globally and to insure the consistency of the Neumann problem. The theory and practical performance of balancing domain decomposition for the standard conforming Galerkin finite element method and mixed finite element method are the subjects of [5] and [6], respectively. As in previous sections, we will deduce the convergence theory for the nonconforming spaces from the conforming theory in [5] using the isomorphism introduced in the fifth section of this paper.

One remarkable property of balancing domain decomposition is that the bound on the condition number of the preconditioned operator is independent of jumps in coefficients across subdomains. Specifically, let the tensor A in (1) be written as $A(x) = \alpha(x)\widehat{A}(x)$, where α is a positive function that is piecewise constant with constant value α_i on Ω_i. The uniform ellipticity then implies that there exists positive constants $\widehat{c}_*, \widehat{c}^*$ such that

$$\widehat{c}_*\alpha_i\xi^T\xi \leq \xi^T A(x)\xi \leq \widehat{c}^*\alpha_i\xi^T\xi \quad \forall\xi \in \mathbb{R}^n, \forall x \in \Omega_i. \tag{25}$$

The bound on the condition number of the operator that arises in balancing domain decomposition will depend on $\widehat{c}_*$ and $\widehat{c}^*$ but will be independent of α_i and c_* and c^* in (2).

Following Mandel's original exposition in [3], we now recall the balancing preconditioner in terms of matrices. A equivalent variational presentation is given in [6]. By an abuse of notation, we use the same symbol to denote an element in $\mathcal{N}(\Gamma)$ and its associated vector of values attained at the nodal degrees of freedom.

The balancing preconditioner is parameterized by two sets of matrices, a set of weighting matrices $\{W_i\}_{i=1}^M$ and a set of kernel generators $\{Z_i\}_{i=1}^M$. The weighting matrices $W_i : \mathcal{N}(\partial\Omega_i) \rightarrow \mathcal{N}(\partial\Omega_i)$ are chosen such that they form a decomposition of unity on $\mathcal{N}(\Gamma)$, i.e.

$$\sum_{i=1}^M N_i W_i N_i^T p = p \quad \forall p \in \mathcal{N}(\Gamma),$$

where N_i denotes the canonical inclusion mapping $N_i : \mathcal{N}(\partial\Omega_i) \rightarrow \mathcal{N}(\Gamma)$ by extending elements of $\mathcal{N}(\partial\Omega_i)$ by zero at all other degrees of freedom. A prescription for the weighting matrices that guarantees a convergence bound independent of coefficient jumps between subdomains is given in Lemma 12 below. For each subdomain Ω_i, let $n_i = \dim(\mathcal{N}(\partial\Omega_i))$, and select an $n_i \times m_i$ matrix Z_i of full column rank with $0 \leq m_i \leq n_i$, such that

$$\mathrm{Ker}S_i \subset \mathrm{Range}Z_i, \qquad i = 1, \ldots, M. \tag{26}$$

For the scalar, second order, elliptic problems we consider in this paper, $\mathrm{Ker}S_i$ is empty if there is Dirichlet data imposed on any part of $\partial\Omega_i \cap \partial\Omega$, otherwise it is the set of functions that have the same value at all the nodes on $\partial\Omega_i$. From the kernel generators, we construct a "coarse space", $\mathcal{N}_H \subset \mathcal{N}(\Gamma)$, defined by

$$\mathcal{N}_H = \{p \in \mathcal{N}(\Gamma) \; : \; p = \sum_{i=1}^M N_i W_i z, z \in \mathrm{Range}Z_i\}.$$

We say that $q \in \mathcal{N}(\Gamma)$ is *balanced* if it is orthogonal to $\mathcal{N}_H$; that is,

$$Z_i^T W_i^T N_i^T q = 0, \qquad i = 1, \ldots, M. \tag{27}$$

Let $|p_i|^2_{S_i} = s_i(p_i, p_i)$. Considering those p_i that are orthogonal to the range of Z_i, working one glob at a time, and using (36), (19), Lemma 13 and (36) in that order, we have

$$\begin{aligned} |\mathcal{I}^{\mathcal{N}}_\Gamma(E_G\tilde{\mathcal{I}}^{\partial\Omega_i}p_i)|^2_{S_j} &\leq \alpha_j C|\tilde{\mathcal{I}}^{\partial\Omega_j}E_G\tilde{\mathcal{I}}^{\partial\Omega_i}p_i|^2_{1/2,\partial\Omega_j} \leq \alpha_j C|E_G\tilde{\mathcal{I}}^{\partial\Omega_i}p_i|^2_{1/2,\partial\Omega_j} \\ &\leq \alpha_j C(1+\log(H/h))^2|\tilde{\mathcal{I}}^{\partial\Omega_i}p_i|^2_{1/2,\partial\Omega_i} \\ &\leq \frac{\alpha_j}{\alpha_i}C(1+\log(H/h))^2|p_i|^2_{S_i}. \end{aligned} \tag{37}$$

By the construction of the decomposition, there is an a priori maximum number of globs that intersect $\partial\Omega_i \cap \partial\Omega_j$. Summing over such globs, we conclude that

$$s_j(N_j^T N_i p_i, N_j^T N_i p_i) \leq \frac{\alpha_j}{\alpha_i}C(1+\log(H/h))^2 s_i(p_i, p_i).$$

The proof is completed by appealing to the bound in Lemma 12. □

REFERENCES

[1] Maksymilian Dryja and Olof B. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. Technical Report 339, Courant Institute of Mathematical Sciences, 1987.

[2] Barry F. Smith. An optimal domain decomposition preconditioner for the finite element solution of linear elasticity problems. *SIAM J. of Sci. Stat. Comput.*, 13(1):364–378, January 1992.

[3] Jan Mandel. Balancing domain decomposition. To appear in Communications on Applied Numerical Methods.

[4] Maksymilian Dryja and Olof B. Widlund. Domain decomposition algorithms with small overlap. Technical Report 606, Department of Computer Science, Courant Institute, May 1992.

[5] Jan Mandel and Marian Brezina. Balancing domain decomposition: theory and performance in two and three dimensions. Submitted.

[6] Lawrence C. Cowsar, Jan Mandel, and Mary F. Wheeler. Balancing domain decomposition for mixed finite elements. Dept. Comp. and Appl. Math. TR93-08, Rice University, March 1993. Submitted.

[7] Lawrence C. Cowsar. Dual-variable Schwarz methods for mixed finite elements. Dept. Comp. and Appl. Math. TR93-09, Rice University, March 1993.

[8] Marcus Sarkis. Two-level Schwarz methods for nonconforming finite elements and discontinuous coefficients. In *Proceedings of the Sixth Annual Copper Mountain Conference on Multigrid Methods, NASA CP-3224*, 1993. Also appeared as Technical Report 629, Courant Institute, Department of Computer Science, March, 1993.

[9] Maksymilian Dryja and Olof B. Widlund. Towards a unified theory of domain decomposition algorithms for elliptic problems. In Tony Chan et al., editors, *Proceedings of the Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 3–21. SIAM, March 1989.

[10] Jan Mandel. Hybrid domain decomposition with unstructured subdomains. To appear in the *Proceedings of the Sixth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, 1993.

[11] Junping Wang, Ruifeng Xie, and Tarek Mathew. Domain decomposition for elliptic problems with large jumps in coefficients. In preparation, 1993.

[12] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. I. *Math. of Comp.*, 47(175):103–134, July 1986.

[13] James H. Bramble, Joseph E. Pasciak, Junping Wang, and Jinchao Xu. Convergence estimates for product iterative methods with applications to domain decomposition. *Math. of Comp.*, 57(195):1–21, July 1991.

[14] Maksymilian Dryja and Olof B. Widlund. The Neumann-Neumann method as an additive Schwarz method for fintie element elliptic problems. Technical Report TR-626, Department of Computer Science, Courant Institute, March 1993.

[15] Olof B. Widlund. An extension theorem for finite element spaces with three applications. Dept. of Computer Science 233, Courant Institute of Mathematical Sciences, New York, 10012, August 1986.

N94-23681

NESTED KRYLOV METHODS AND PRESERVING THE ORTHOGONALITY

S8-64
197140
p. 15

Eric De Sturler [1]
Delft University of Technology
Delft, The Netherlands

Diederik R. Fokkema [2]
University of Utrecht
Utrecht, The Netherlands

SUMMARY

Recently the GMRESR inner-outer iteration scheme for the solution of linear systems of equations has been proposed by Van der Vorst and Vuik. Similar methods have been proposed by Axelsson and Vassilevski [1] and Saad (FGMRES) [10]. The outer iteration is GCR, which minimizes the residual over a given set of direction vectors. The inner iteration is GMRES, which at each step computes a new direction vector by approximately solving the residual equation. However, the optimality of the approximation over the space of outer search directions is ignored in the inner GMRES iteration. This leads to suboptimal corrections to the solution in the outer iteration, as components of the outer iteration directions may reenter in the inner iteration process. Therefore we propose to preserve the orthogonality relations of GCR in the inner GMRES iteration. This gives optimal corrections; however, it involves working with a singular, non-symmetric operator. We will discuss some important properties and we will show by experiments that, in terms of matrix vector products, this modification (almost) always leads to better convergence. However, because we do more orthogonalizations, it does not always give an improved performance in CPU-time. Furthermore, we will discuss efficient implementations as well as the truncation possibilities of the outer GCR process. The experimental results indicate that for such methods it is advantageous to preserve the orthogonality in the inner iteration. Of course we can also use other iteration schemes than GMRES as the inner method. Especially methods with short recurrences like BICGSTAB seem of interest.

INTRODUCTION

For the solution of systems of linear equations the so-called Krylov subspace methods are very popular. However, for general matrices no Krylov method can satisfy a global optimality requirement and have short recurrences [5]. Therefore either restarted or truncated versions of optimal methods, like GMRES [11], are used or methods with short recurrences, which do not satisfy a global optimality requirement, like BiCG [6], BICGSTAB [14], BICGSTAB(l) [12], CGS [13] or QMR [8]. Recently Van der Vorst and Vuik proposed a class of methods, GMRESR [15], which are nested GMRES methods; see Fig. 2. The GMRESR algorithm is based upon the GCR algorithm [4]; see Fig. 1. For a given initial guess x_0, they both compute approximate solutions x_k, such that $x_k - x_0 \in span\{u_1, u_2, \ldots, u_k\}$ and $\|r_k\|_2 = \|b - Ax_k\|_2$ is minimal.

[1]Delft University of Technology, Faculty of Technical Mathematics and Informatics, P.O. Box 5031, NL-2600 GA Delft, The Netherlands, E-mail: witaeds@dutinfh.tudelft.nl. The author wishes to acknowledge Shell Research B.V. and STIPT for the financial support of his research.

[2]Mathematical Institute, University of Utrecht, P.O. Box 80.010. NL-3508 TA Utrecht, The Netherlands, E-mail: fokkema@math.ruu.nl. This work was supported in part by a NCF/Cray Research University Grant CRG 92.03

```
GCR:
1.  Select $x_0$, $m$, $tol$;
    $r_0 = b - Ax_0$, $k = 0$;
2.  while $\|r_k\|_2 > tol$ do
        $k = k + 1$;
        $u_k = r_{k-1}$; $c_k = Au_k$;
        for $i = 1, k - 1$ do;
            $\alpha_i = c_i^T c_k$;
            $c_k = c_k - \alpha_i c_i$;
            $u_k = u_k - \alpha_i u_i$;
        $c_k = c_k / \|c_k\|$;
        $u_k = u_k / \|c_k\|$;
        $x_k = x_{k-1} + (c_k^T r_{k-1}) u_k$;
        $r_k = r_{k-1} - (c_k^T r_{k-1}) c_k$;
```

Figure 1: The GCR algorithm

```
GMRESR:
1.  Select $x_0$, $m$, $tol$;
    $r_0 = b - Ax_0$, $k = 0$;
2.  while $\|r_k\|_2 > tol$ do
        $k = k + 1$;
        $u_k = \mathcal{P}_{m,k}(A) r_{k-1}$; $c_k = Au_k$;
        for $i = 1, \ldots, k - 1$ do
            $\alpha_i = c_i^T c_k$;
            $c_k = c_k - \alpha_i c_i$;
            $u_k = u_k - \alpha_i u_i$;
        $c_k = c_k / \|c_k\|_2$;
        $u_k = u_k / \|c_k\|_2$;
        $x_k = x_{k-1} + (c_k^T r_{k-1}) u_k$;
        $r_k = r_{k-1} - (c_k^T r_{k-1}) c_k$;
```

$\mathcal{P}_{m,k}(A)$ indicates the GMRES polynomial that is implicitly constructed in m steps of GMRES when solving $Ay = r_{k-1}$.

Figure 2: The GMRESR algorithm

However, they compute different direction vectors u_k. GCR sets u_k simply to r_{k-1}, while GMRESR computes u_k by applying m steps of GMRES to r_{k-1} (represented by $\mathcal{P}_{m,k}(A)r_{k-1}$ in Fig. 2). The inner GMRES iteration computes a new search direction by approximately solving the residual equation and then the outer GCR iteration minimizes the residual over the new search direction and all previous search directions u_i. The algorithm can be explained as follows.

Assume we are given the system of equations $Ax = b$, where A is a real, nonsingular, linear $(n \times n)$-matrix and b is a n-vector. Let U_k and C_k be two $(n \times k)$-matrices for which

$$C_k = AU_k, \quad C_k^T C_k = I_k, \tag{1}$$

and let x_0 be an initial guess. For $x_k - x_0 \in \; range(U_k)$ the minimization problem

$$\|b - Ax_k\|_2 = \min_{x \in range(U_k)} \|r_0 - Ax\|_2. \tag{2}$$

is solved by

$$x_k = x_0 + U_k C_k^T r_0 \tag{3}$$

and $r_k = b - Ax_k$ satisfies

$$r_k = r_0 - C_k C_k^T r_0, \quad r_k \perp \; range(C_k) \; . \tag{4}$$

In fact we have constructed the inverse of the restriction of A to $range(U_k)$ onto $range(C_k)$. This inverse is given by

$$A^{-1} C_k C_k^T = U_k C_k^T. \tag{5}$$

This principle underlies the GCR method. In GCR the matrices $U_k = [u_1 u_2 \ldots u_k]$ and $C_k = [c_1 c_2 \ldots c_k]$ are constructed such, that $range(U_k)$ is equal to the Krylov subspace $K^k(A; r_0) = \; span\{r_0, Ar_0, \ldots, A^{k-1}r_0\}$. Provided GCR does not break down; i.e. if $c_k \not\perp r_{k-1}$, it is a finite method and at step k it solves the minimization problem (2).

Consider the k-th step in GCR. Equations (1)-(3) indicate that if in the update $u_k = r_{k-1}$ (in GCR), we replace r_{k-1} by any other vector, then the algorithm still solves (2); however, the subspace U_k will be different. The optimal choice would be $u_k = e_{k-1}$, where e_{k-1} is the error in x_{k-1}. In order to find approximations to e_{k-1}, we use the relation $Ae_{k-1} = r_{k-1}$ and *any* method which gives an approximate solution to this equation can be used to find acceptable choices for u_k. In the GMRESR algorithm GMRES(m) is chosen to be the method to find such an approximation.

However, since we already have an optimal x_{k-1}, such that $x_{k-1} - x_0 \in \ range(U_{k-1})$, we need an approximation u_k to e_{k-1}, such that $c_k = Au_k$ is orthogonal to $range(C_{k-1})$. Such an approximation is computed explicitly by the orthogonalization loop in the outer GCR iteration. Because in GMRESR this is not taken into account in the inner GMRES iteration, a less than optimal minimization problem is solved, leading to suboptimal corrections [2] to the residual. Another disadvantage of GMRESR is that the inner iteration is essentially a restarted GMRES. It therefore also displays some of the problems of restarted GMRES. Most notably it can have the tendency to stagnate (see NUMERICAL EXPERIMENTS).

From this we infer that we should preserve the orthogonality of the correction to the residual also in the inner GMRES iteration. In order to do this we use $A_{k-1} = (I - C_{k-1}C_{k-1}^T)A$ as the operator in the inner iteration. This gives the proper corrections to the residual: $c_k \in K^m(A_{k-1}; A_{k-1}r_{k-1})$. However, the corresponding corrections to the approximate solution (contrary to ordinary implementations of Krylov methods) are found by $u_k = A^{-1}c_k \in A^{-1}K^m(A_{k-1}; A_{k-1}r_{k-1})$. These corrections can be computed since the inverse of A is known over this space. Equation (5) gives:

$$A^{-1}A_{k-1} = A^{-1}A - A^{-1}C_{k-1}C_{k-1}^T A = I - U_{k-1}C_{k-1}^T A. \tag{6}$$

This leads to a variant of the GMRESR iteration scheme, which has an improved performance for many problems.

In this article we will consider GMRES and BICGSTAB as inner methods. In the next section we will discuss the implications of the orthogonalization in the inner method. It will be proved that this leads to an optimal approximation over the space spanned by both the outer and the inner iteration vectors. It also introduces a potential problem: the possibility of breakdown in the generation of the Krylov space in the inner iteration, since we iterate with a singular operator. We will show, however, that such a breakdown can never happen before a specific (generally large) number of iterations. Furthermore, we will also show how to remedy such a breakdown. We will also discuss the efficient implementation of these methods and how we can truncate the outer GCR iteration. Outlines of the algorithms can be found in [7], [2].

CONSEQUENCES OF INNER ORTHOGONALIZATION

To keep this section concise, we will only give a short indication of the proofs or omit them completely. The proofs can be found in [2]. Throughout the rest of this article we will use the following notations:
- By $U_k = [u_1 \ldots u_k]$ and $C_k = [c_1 \ldots c_k]$ we denote matrices that satisfy the relations (1);
- By x_k and r_k we denote the vectors that satisfy the relations (2)-(4);
- By P_k and Q_k we denote the projections defined as $P_k = C_kC_k^T$ and $Q_k = U_kC_k^T A$;
- By A_k we denote the operator defined as $A_k = (I - P_k)A$;
- By $V_m = [v_1 \ldots, v_m]$ we denote the orthonormal matrix generated by m steps of Arnoldi (GMRES) with A_k and such that $v_1 = r_k/\|r_k\|_2$.

From this and (6) it then follows that

$$AQ_k = P_kA, \quad \text{and} \quad A^{-1}A_k = (I - Q_k). \tag{7}$$

We will describe the $(k+1)$-th step of our variant of the GMRESR iteration scheme, where in the inner GMRES iteration the modified operator A_k is used. We use m (not fixed) steps of the GMRES algorithm to compute the correction to r_{k+1} in the space $K^m(A_k; A_k r_k)$. This leads to the optimal correction to the approximate solution x_{k+1} over the 'global' space $range(U_{k+1}) \oplus A^{-1}K^m(A_k; A_k r_k)$.

Theorem 1 *The Arnoldi process in the inner GMRES iteration defines the relation $A_k V_m = V_{m+1}\bar{H}_m$, with $\bar{H}_m$ an $((m+1) \times m)$ Hessenberg matrix. Let y be defined by*

$$y : \min_{\tilde{y} \in \mathbf{R}^m} \|r_k - A_k V_m \tilde{y}\|_2 = \min_{\tilde{y} \in \mathbf{R}^m} \|r_k - V_{m+1}\bar{H}_m \tilde{y}\|_2. \tag{8}$$

Then the minimal residual solution of the inner GMRES iteration: $(A^{-1}A_k V_m y)$ gives the outer approximation

$$x_{k+1} = x_k + (I - Q_k)V_m y, \tag{9}$$

which is also the solution to the 'global' minimization problem

$$x_{k+1} : \min_{\substack{\tilde{x} \in range(U_k) \oplus \\ range(V_m)}} \|b - A\tilde{x}\|_2 \tag{10}$$

It also follows from this theorem that the GCR optimization (in the outer iteration) is given by (9), so that the residual computed in the inner GMRES iteration equals the residual of the outer GCR iteration: $r_{k+1} = b - Ax_{k+1} = b - Ax_k - A_k V_m y = r_k - A_k V_m y$. From this it follows that in the outer GCR iteration the vectors u_{k+1} and c_{k+1} are given by

$$c_{k+1} = (A_k V_m y)/\|A_k V_m y\|_2, \tag{11}$$
$$u_{k+1} = ((I - Q_k)V_m y)/\|A_k V_m y\|_2. \tag{12}$$

Note that $(I - Q_k)V_m y$ has been computed already as the the approximate solution in the inner GMRES iteration; see (9), and $A_k V_m y$ is easily computed from the relation $A_k V_m y = V_{m+1}\bar{H} y_m$. Moreover, as a result of using GMRES in the inner iteration, the norm of the residual r_{k+1} as well as the norm of $A_k V_m y$ is already known at no extra computational costs. Consequently, the outer GCR iteration becomes very simple.

We will now consider the possibility of breakdown when generating a Krylov space with a singular, nonsymmetric operator. Although GMRES is still optimal in the sense that at each iteration it delivers the minimum residual solution over the generated Krylov subspace, the generation of the Krylov subspace itself, from a singular operator, may terminate too early. The following simple example shows that this may happen before the solution is found, even when the solution and the right hand side are both in the range of the given (singular) operator and in the orthogonal complement of its null-space.

Define the matrix $A = (e_2\ e_3\ e_4\ 0)$, where e_i denotes the i-th Cartesian basis vector. Note that $A = (I - e_1 e_1^T)(e_2\ e_3\ e_4\ e_1)$, which is the same type of operator as A_k, an orthogonal projection times a nonsingular operator. Now consider the system of equations $Ax = e_3$. Then GMRES (or any other Krylov method) will search for a solution in the space

$$span\{e_3, Ae_3, A^2e_3, \ldots\} = span\{e_3, e_4, 0, 0, \ldots\}.$$

So we have a breakdown of the Krylov space and the solution is not contained in it. We remark that the singular unsymmetric case is quite different from the symmetric one.

In the remainder of this section we will prove that a breakdown in the inner GMRES method cannot occur before the total number of iterations exceeds the dimension of the Krylov space $K(A; r_0)$. This means that, in practice, a breakdown will be rare. Furthermore, we will show how such a breakdown can be overcome.

We will now define breakdown of the Krylov space for the inner GMRES iteration more formally.

Definition 1 *We say there is a breakdown of the Krylov subspace in the inner GMRES iteration if $A_k v_m \in range(V_m)$, since this implies we can no longer expand the Krylov subspace. We call it a* **lucky breakdown** *if $v_1 \in range(A_k V_m)$, because we then have found the solution (the inverse of A is known over the space $range(A_k V_m)$). We call it a* **true breakdown** *if $v_1 \notin range(A_k V_m)$, because then the solution is not contained in the Krylov subspace.*

The following theorem relates true breakdown to the invariance of the sequence of subspaces in the inner method for the operator A_k. Part four indicates that it is always known whether a breakdown is true or lucky.

Theorem 2 *The following statements are equivalent:*

1. *A true breakdown occurs in the inner GMRES iteration at step m;*
2. *$range(A_k V_{m-1})$ is an invariant subspace of A_k;*
3. *$A_k v_m \in range(A_k V_{m-1})$;*
4. *$A_k V_m = V_m H_m$, and H_m is a singular $m \times m$ matrix.*

From theorem 1, one can already conclude that a true breakdown occurs if and only if A_k is singular over $K^m(A_k; r_k)$. From the definition of A_k we know $null(A_k) = range(U_k)$. We will make this more explicit in the following theorem, which relates true breakdown to the intersection of the inner search space and the outer search space.

Theorem 3 *A true breakdown occurs if and only if*

$$range(V_m) \cap range(U_k) \neq \{0\}.$$

The following theorem indicates that no true breakdown in the inner GMRES iteration can occur before the total number of iterations exceeds the dimension of the Krylov space $K(A; r_0)$.

Theorem 4 *Let $m = \dim(K(A; r_0))$ and let l be such that $r_k = \mathcal{P}_l(A) r_0$ for some polynomial $\mathcal{P}_l$ of degree l. Then*

$$\dim(K^{j+1}(A_k; r_0)) = j + 1 \quad \textit{for } j + l < m$$

and therefore no true breakdown occurs in the first j steps of the inner GMRES iteration.

We will now show how a true breakdown can be overcome. There are basically two ways to continue:
In the inner iteration: by finding a suitable vector to expand the Krylov space.

In the outer iteration: by computing the solution of the inner iteration just before the true breakdown and then by making one LSQR-step (see below) in the outer iteration.

We will consider the continuation in the inner GMRES iteration first. The following theorem indicates how one can continue the generation of the Krylov space $K(A; r_k)$ if in the inner GMRES iteration a true breakdown occurs.

Theorem 5 *If a true breakdown occurs in the inner GMRES iteration then*

$$\exists c \in \ range(C_k) \ : \ A_k c \notin \ range(A_k V_{m-1}) \tag{13}$$

This implies that one can try the vectors c_i until one of them works. However, one should realize that the minimization problem (8) is slightly more complicated.

Another way to continue after a true breakdown in the inner GMRES iteration is to compute the inner iteration solution just before the breakdown and then apply an LSQR-switch (see below) in the outer GCR iteration. The following theorem states the reason why one has to apply an LSQR-switch.

Theorem 6 *Suppose one computes the solution of the inner GMRES iteration just before a true breakdown. Then stagnation will occur in the next inner iteration, that is $r_{k+1} \perp K(A_{k+1}; r_{k+1})$. This will lead to a breakdown of the outer GCR iteration.*

The reason for this stagnation in the inner GMRES iteration is that the new residual r_{k+1} remains in the same Krylov space $K(A_k; r_k)$, which contains a $u \in \ range(U_k)$. So we have to 'leave' this Krylov space. We can do this using the so-called LSQR-switch, which was introduced in [15], to remedy stagnation in the inner GMRES iteration. Just as in the GMRESR method, stagnation in the inner GMRES iteration will result in a breakdown in the outer GCR iteration, because the residual cannot be updated. The following theorem states that this LSQR-switch actually works.

Theorem 7 *If stagnation occurs in the inner GMRES iteration, that is if* $\min_{\bar{y} \in \mathbb{R}^m} \|r_{k+1} - A_k V_m \bar{y}\|_2 = \|r_{k+1}\|_2$, *then one can continue by setting (LSQR-switch)*

$$\begin{aligned} c_{k+2} &= \gamma A_{k+1} A^T r_{k+1} \quad \textit{and} \qquad (14)\\ u_{k+2} &= \gamma (I - Q_{k+1}) A^T r_{k+1}, \qquad (15) \end{aligned}$$

where $\gamma = \|c_{k+2}\|_2^{-1}$. This leads to

$$\begin{aligned} r_{k+2} &= r_{k+1} - (r_{k+1}^T c_{k+2}) c_{k+2} \quad \textit{and} \qquad (16)\\ x_{k+2} &= x_{k+1} - (r_{k+1}^T c_{k+2}) u_{k+2}, \qquad (17) \end{aligned}$$

which always gives an improved approximation. Therefore, these vectors can be used as the start vectors for a new inner GMRES iteration.

IMPLEMENTATION

We will now describe how to implement these methods efficiently (see also [2],[7]). First we will discuss the outer GCR iteration and then the inner GMRES iteration. The implementation of a method like

BICGSTAB in the inner iteration will then be obvious. Instead of the matrices U_k and C_k we will use in the actual implementation the matrices $\hat{U}_k$, $\bar{C}_k$, N_k, Z_k and the vector d_k which are defined below.

Definition 2 *The matrices $\hat{U}_k$, $\bar{C}_k$, N_k, Z_k and the vector d_k are defined as follows.*

$$\begin{aligned} C_k &= \bar{C}_k N_k, \quad \textit{where} && (18)\\ N_k &= diag(\|\bar{c}_1\|_2^{-1}, \|\bar{c}_2\|_2^{-1}, \ldots, \|\bar{c}_k\|_2^{-1}), && (19)\\ A\hat{U}_k &= \bar{C}_k Z_k, && (20) \end{aligned}$$

where Z_k is assumed to be upper-triangular. Finally d_k is defined by the relation

$$r_k = r_0 - \bar{C}_k d_k \tag{21}$$

From this the approximate solution x_k, corresponding to r_k, is implicitly represented as

$$x_k = x_0 + \hat{U}_k Z_k^{-1} d_k. \tag{22}$$

Using this relation x_k can be computed at the end of the complete iteration or before truncation (see next section). The implicit representation of U_k saves all the intermediate updates of previous u_i to a new u_{k+1}, which is approximately 50% of the computational costs in the outer GCR iteration (see (11) and (12)).

GMRES as inner iteration. After k outer GCR iterations we have $\hat{U}_k$, $\bar{C}_k$ and r_k. Then, in the inner GMRES iteration, the orthogonal matrix V_{m+1} is constructed such that $\bar{C}_k^T V_{m+1} = O$ and

$$\begin{aligned} AV_m &= \bar{C}_k B_m + V_{m+1} \bar{H}_m && (23)\\ B_m &= N_k^2 \bar{C}_k^T A V_m && (24) \end{aligned}$$

This algorithm is equivalent to the usual GMRES algorithm, except that the vectors Av_i are first orthogonalized on $\bar{C}_k$. From (23) and (24) it is obvious that $AV_m - \bar{C}_k B_m = A_k V_m = V_{m+1}\bar{H}_m$ (cf. theorem 1). Next we compute y according to (8) and we set (cf. (11) without normalization):

$$\begin{aligned} \bar{c}_{k+1} &= V_{m+1}\bar{H}_m y && (25)\\ \hat{u}_{k+1} &= V_m y. && (26) \end{aligned}$$

This leads to $A\hat{u}_{k+1} = AV_m y = \bar{C}_k B_m y + V_{m+1}\bar{H}_m y = \bar{C}_k B_m y + \bar{c}_{k+1}$, so that if we set $z_{k+1} = ((B_m y)^T\, 1)^T$ the relation $A\hat{U}_{k+1} = \bar{C}_{k+1} Z_{k+1}$ is again satisfied. It follows from theorem 1 that the new residual of the outer GCR iterations is equal to the final residual of the inner iteration $r_{k+1} = r_m^{inner}$ and is given by $r_{k+1} = r_k - \bar{c}_{k+1}$, so that $d_{k+1} = 1$. Obviously the residual norm only needs to be computed once. If we replace, in the formula above, the new residual of the outer GCR iteration r_{k+1} by the residual of the inner GMRES iteration r_m^{inner}, we see an important relation that holds more generally $\bar{c}_{k+1} = r_k - r_m^{inner}$. This relation is important, since in general (when other Krylov methods are used for the inner iteration) $\bar{c}_{k+1}$ or c_{k+1} cannot be computed from u_{k+1}, because u_{k+1} is not always computed explicitly, nor does a relation like (25) always exist. Finally, we need to compute the new coefficient of N_{k+1}, $\|\bar{c}_{k+1}\|_2^{-1}$ in order to satisfy the relations in definition 2.

TRUNCATION

In practice, since memory space may be limited and since the method becomes increasingly expensive for large k (the number of outer search vectors), we want to truncate the set of outer iteration vectors ($\hat{u}_i$) and

$(\bar{c}_i)$ at $k = k_{max}$, where k_{max} is some positive integer. Basically, there are two ways to do this: one can discard one or more iteration vector(s) (dropping) or one can assemble two or more iteration vectors into one single iteration vector (assembly). We will first discuss the strategy for truncation and then its implementation.

A strategy for Truncation. In each outer GCR iteration step the matrices $\widehat{U}_k$ and $\bar{C}_k$ are augmented with one extra column. To keep the memory requirement constant, at step $k = k_{max}$, it is therefore sufficient to diminish the matrices $\widehat{U}_{k_{max}}$ and $\bar{C}_{k_{max}}$ by one column. From (22) we have $x_k = x_0 + \widehat{U}_k Z_k^{-1} d_k$. Denote $\xi_k = Z_k^{-1} d_k$. Consider the sequence of vectors (ξ_k). The components ${\xi_k}^{(i)}$ of these vectors ξ_k are the coefficients for the updates $\hat{u}_i$ of the approximate solution x_k. These coefficients ${\xi_k}^{(i)}$ converge to the limits $\xi^{(i)}$ as k increases. Moreover, $({\xi_k}^{(1)})$ converges faster than $({\xi_k}^{(2)})$, and $({\xi_k}^{(2)})$ converges faster than $({\xi_k}^{(3)})$ etc. . Suppose that the sequence $({\xi_k}^{(1)})$ has converged to $\xi^{(1)}$ within machine precision. From then on it makes no difference for the computation of x_k when we perform the update $x_0 + \xi^{(1)}\hat{u}_1$. In terms of direction vectors this means that the outer direction vector $\hat{u}_1$ will not reenter as component in the inner iteration process. Therefore one might hope that discarding the vector $\bar{c}_1$ will not spoil the convergence. This leads to the idea of dropping the vector $\bar{c}_1 (= A\hat{u}_1)$ or of assembling $\bar{c}_1$ with $\bar{c}_2$ into $\tilde{c}$ (say) when

$$\delta(k) = \left| \frac{\xi_{k-1}^{(1)} - \xi_k^{(1)}}{\xi_k^{(1)}} \right| < \epsilon, \tag{27}$$

where $\epsilon > 0$ is a small constant. The optimal ϵ, which may depend on k, can be determined from experiments. When $\delta(k) > \epsilon$ we drop $\bar{c}_{k_{max-1}}$ or we assemble $\bar{c}_{k_{max-1}}$ and $\bar{c}_{k_{max}}$ (of course other choices are feasible as well, but we will not consider them in this article). With this strategy we hope to avoid stagnation by keeping the most relevant part of the subspace $range(C_k)$ in store as a subspace of dimension $k-1$. In the next subsections we describe how to implement this strategy and its consequences for the matrices $\bar{C}_k$ and $\widehat{U}_k$.

Dropping a vector. Let $1 \le j \le k = k_{max}$. Dropping the column $\bar{c}_j$ is easy. We can discard it without consequences. So let $\bar{C}'_{k-1}$ be the matrix $\bar{C}_k$ without the column $\bar{c}_j$. Dropping a column from $\widehat{U}_k$ needs more work, since x_k is computed as $x_k = x_0 + \widehat{U}_k Z_k^{-1} d_k$. Moreover, in order to be able to apply the same strategy in the next outer iteration we have to be able to compute x_{k+1} in a similar way. For that purpose, assume that x_k can be computed as

$$x_k = x'_{k-1} = x_0{}' + \widehat{U}'_{k-1}(Z'_{k-1})^{-1} d'_{k-1}, \tag{28}$$

where $\widehat{U}'_{k-1}$ and Z'_{k-1} are matrices such that $A\widehat{U}'_{k-1} = \bar{C}'_{k-1} Z'_{k-1}$ (see (20)). These matrices $\widehat{U}'_{k-1}$ and Z'_{k-1} are easily computed by using the j-th row of (20) to eliminate the j-th column of $\bar{C}_k$ in (20). In order to determine x'_0 and d'_{k-1} we introduce the matrix $\bar{U}_k = A^{-1}\bar{C}_k = \widehat{U}_k Z_k^{-1}$. This enables us to write

$$x_k = (x_0 + d_k^{(j)} \bar{u}_j) + \sum_{\substack{i=1 \\ i \neq j}}^{k} d_k^{(i)} \bar{u}_i \qquad \text{and} \qquad \bar{u}_j = (\hat{u}_j - \sum_{i=1}^{j-1} z_{ij}\bar{u}_i)/z_{jj}. \tag{29}$$

Substituting the equation for $\bar{u}_j$ into the equation for x_k we can compute x_k from

$$x_k = (x_0 + \frac{d_k^{(j)}}{z_{jj}} \hat{u}_j) + \sum_{i=1}^{j-1} (d_k^{(i)} - d_k^{(j)} \frac{z_{ij}}{z_{jj}}) \bar{u}_i + \sum_{i=j+1}^{k} d_k^{(i)} \bar{u}_i. \tag{30}$$

Notice that this equation precisely defines x'_0 and d'_{k-1}:

$$\begin{aligned} x'_0 &= x_0 + (d_k{}^{(j)}/z_{jj}), & & \\ d^{(i)}_{k-1}{}' &= d^{(i)}_k - d^{(j)}_k(z_{ij}/z_{jj}) & &\text{for } i = 1, \ldots, j-1 \text{ and} \\ d^{(i)}_{k-1}{}' &= d^{(i+1)}_k & &\text{for } i = j, \ldots, k-1. \end{aligned} \tag{31}$$

Now we have deallocated two vectors and we compute x_k as in (28). We can continue the algorithm.

Assembly of two vectors. Let $1 \leq j < l \leq k = k_{max}$. Again assembling $\bar{c}_j$ and $\bar{c}_l$ is easy. Let $\tilde{c} = (d^{(j)}_k \bar{c}_j + d^{(l)}_k \bar{c}_l)$ overwrite the l-th column of $\bar{C}_k$. Then, let $\bar{C}'_{k-1}$ be this new matrix $\bar{C}_k$ without j-th column. Analogous to the above, we wish to compute x_k as (28). For the purpose of determining the matrices $\widehat{U}'_{k-1}$ and Z'_{k-1}, let $\tilde{u} = (d^{(j)}_k \bar{u}_j + d^{(l)}_k \bar{u}_l)$ and compute $t^{(m)}_1$ and $t^{(m)}_2$ such that $z_{jm}\bar{u}_j + z_{lm}\bar{u}_l + t^{(m)}_1 \bar{u}_j = t^{(m)}_2 \tilde{u}$, which gives $t^{(m)}_1 = z_{lm}(d^{(j)}_k / d^{(l)}_k) - z_{jm}$ and $t^{(m)}_2 = z_{lm}/d^{(l)}_k$. This enables us to write $\hat{u}_m = \sum_{i=1}^{m} z_{im}\bar{u}_i$, for $m = 1, \ldots, j-1$ and

$$\hat{u}_m = \sum_{\substack{i=1 \\ i \neq j,l}}^{m} z_{im}\bar{u}_i + t^{(m)}_2 \tilde{u} - t^{(m)}_1 \bar{u}_j, \qquad \text{for } m = j, \ldots, k. \tag{32}$$

Substituting $\bar{u}_j = (\hat{u}_j - \sum_{i=1}^{j-1} z_{ij}\bar{u}_i)/z_{jj}$, to eliminate $\bar{u}_j$ from (32) we get $\hat{u}_m = \sum_{i=1}^{m} z_{im}\bar{u}_i$, for $m = 1, \ldots, j-1$ and

$$\hat{u}_m + \frac{t^{(m)}_1}{z_{jj}} \hat{u}_j = \sum_{\substack{i=1 \\ i \neq j,l}}^{m} (z_{im} + t^{(m)}_1 \frac{z_{ij}}{z_{jj}}) \bar{u}_i + t^{(m)}_2 \tilde{u} \qquad \text{for } m = j+1, \ldots, k. \tag{33}$$

This equation determines the matrices $\widehat{U}'_{k-1}$ and Z'_{k-1}. In order to determine x'_0 and d'_{k-1}, note that x_k can be computed as

$$x_k = x_0 + \sum_{\substack{i=1 \\ i \neq j,l}}^{k} d^{(i)}_k \bar{u}_i + \tilde{u}. \tag{34}$$

Therefore x'_0 is just x_0 and d'_{k-1} equals the vector d_k without the j-th element and the l-th element overwritten by 1. Similarly, as before, we have deallocated two vectors from memory. The assembled vectors $\tilde{u}$ and $\tilde{c}$ overwrite $\hat{u}_l$ and $\hat{c}_l$. The locations of $\hat{u}_j$ and $\hat{c}_j$ can therefore be used in the next step. Finally, we remark that these computations can be done with rank one updates.

NUMERICAL EXPERIMENTS

We will discuss the results of some numerical experiments, which concern the solution of two dimensional convection diffusion problems on regular grids, discretized using a finite volume technique, resulting in a pentadiagonal matrix. The system is preconditioned with ILU applied to the scaled system; see [3],[9]. The first two problems are used to illustrate and compare the following solvers:

- (full) GMRES;
- BICGSTAB;
- GMRESR(m), where m indicates the number of inner GMRES iterations between the outer iterations;
- GCRO(m), which is GCR with m adapted GMRES iterations as inner method, using A_k;
- GMRESRSTAB, which is GMRESR with BICGSTAB as inner method;

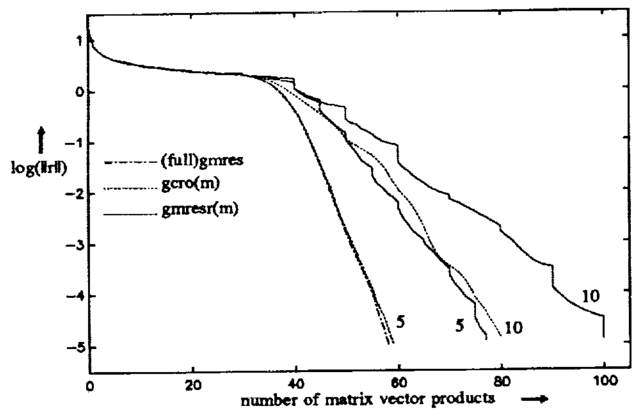


Figure 3: Convergence history for problem 1

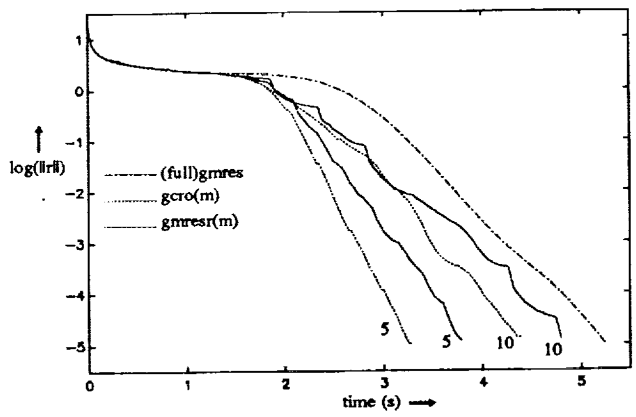


Figure 4: Convergence in time for problem 1

- and GCROSTAB, which is GCR with the adapted BICGSTAB as inner method, using A_k.

We will compare the convergence of these methods both with respect to the number of matrix vector products and with respect to CPU-time on one processor of the Convex 3840. This means e.g. that each step of BICGSTAB (and variants) is counted for two matrix vector products. We give both these convergence rates because the main trade off between (full) GMRES, the GCRO variants and the GMRESR variants is less iterations against more dot products and vector updates per iteration. Any gain in CPU-time then depends on the relative cost of the matrix vector multiplication and preconditioning versus the orthogonalization cost on the one hand and on the difference in iterations on the other hand. We will use our third problem to show the effects of truncation and compare two strategies.

Problem 1. This problem comes from the discretization of

$$-(u_{xx} + u_{yy}) + bu_x + cu_y = 0$$

on $[0,1] \times [0,4]$, where

$$b(x,y) = \begin{cases} 100 & \text{for} \quad 0 \le y < 1 \quad \text{and} \quad 2 \le y < 3 \\ -100 & \text{for} \quad 1 \le y < 2 \quad \text{and} \quad 3 \le y \le 4 \end{cases}$$

and $c = 100$. The boundary conditions are $u = 1$ on $y = 0$, $u = 0$ on $y = 4$, $u' = 0$ on $x = 0$ and $u' = 0$ on $x = 1$, where u' denotes the (outward) normal derivative. The stepsize in x-direction is $1/100$ and in y-direction is $1/50$.

In this example we compare the performances of GMRES, GCRO(m) and GMRESR(m), for $m = 5$ and $m = 10$. The convergence history of problem 1 is given in Fig. 3 and Fig. 4. Fig. 3 shows that GMRES converges fastest (in matrix vector products), which is of course to be expected, followed by GCRO(5), GMRESR(5), GCRO(10) and GMRESR(10). From Fig. 3 we also see that GCRO(m) converges smoother and faster than GMRESR(m). Note that GCRO(5) has practically the same convergence behavior as GMRES. The vertical 'steps' of GMRESR(m) are caused by the optimization in the outer GCR iteration, which does not involve a matrix vector multiplication. We also observe that the GMRESR(m) variants tend to lose their superlinear convergent behavior, at least during certain stages of the convergence history. This seems to be caused by stagnation or slow convergence in the inner GMRES iteration, which (of course) essentially behaves like a restarted GMRES. For GCRO(m), however, we see a much smoother and faster convergence behavior and the superlinearity of (full) GMRES is preserved. This is explained by the 'global' optimization over both the inner and the outer search vectors (the latter form a sample of the entire, previously searched Krylov subspace). So we may view this as a semi-full gmres. Fig. 4 gives the

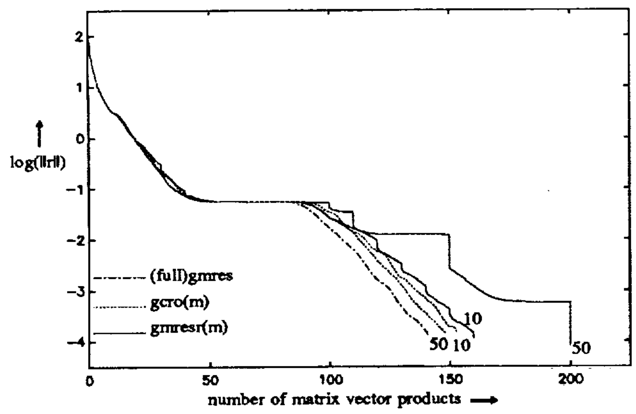


Figure 5: Convergence history for problem 2

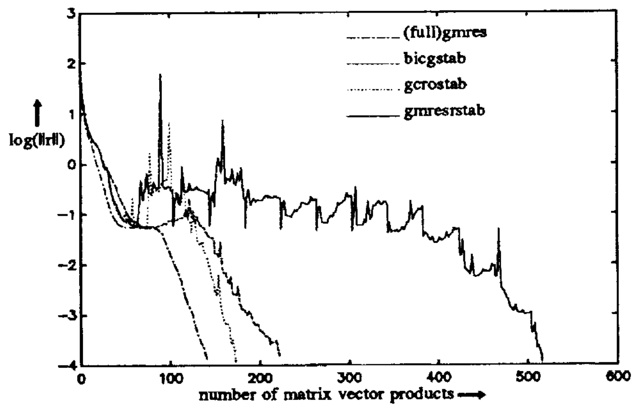


Figure 6: Convergence history for BICGSTAB variants for problem 2

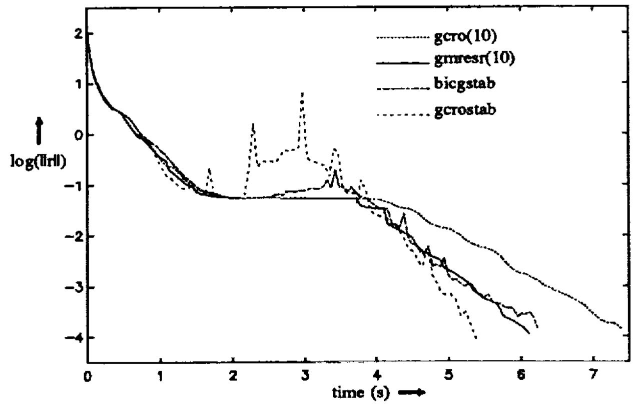


Figure 7: Convergence in time for problem 2

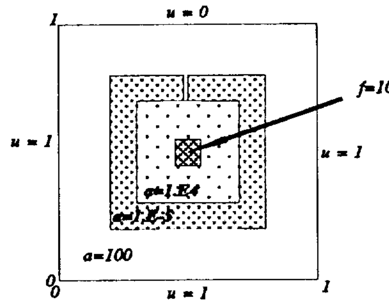


Figure 8: Coefficients for problem 2

convergence with respect to CPU-time. In this example GCRO(5) is the fastest, which is not surprising in view of the fact that it converges almost as fast as GMRES, but against much lower costs. Also, we see that GCRO(10), while slower than GMRESR(5), is still faster than GMRESR(10). In this case the extra orthogonalization costs in GCRO are outweighed by the improved convergence behavior.

Problem 2. This problem is taken from [14]. The linear system comes from the discretization of

$$-(au_x)_x - (au_y)_y + bu_x = f$$

on the unit square, with $b = 2\exp 2(x^2 + y^2)$. Along the boundaries we have Dirichlet conditions: $u = 1$ for $y = 0, x = 0$ and $x = 1$, and $u = 0$ for $y = 1$. The functions a and f are defined as shown in Fig. 8; $f = 0$ everywhere, except for the small subsquare in the center where $f = 100$. The stepsize in x-direction and in y-direction is $1/128$.

If Fig. 5 a convergence plot is given for (full) GMRES, GCRO(m) and GMRESR(m). We used $m = 10$ and $m = 50$ to illustrate the difference in convergence behavior in the inner GMRES iteration of GMRESR(m) and GCRO(m). GMRESR(50) stagnates in the inner GMRES iteration whereas GCRO(50) more or less displays the same convergence behavior as GCRO(10) and full GMRES. For the number of matrix vector products, it seems that for GMRESR(m) small m are the best choice.

In Fig. 6 a convergence plot is given for (full) GMRES, BICGSTAB, and the the BICGSTAB variants, GMRESRSTAB and GCROSTAB. To our experience the following strategy gave the best results for the BICGSTAB variants:

- For GMRESRSTAB we ended an inner iteration after either 20 steps or a relative improvement of the residual of 0.01;
- For GCROSTAB we ended an inner iteration after either after 25 steps or a relative improvement of the residual of 0.01.

The convergence of GMRESRSTAB for this example is somewhat typical for GMRESRSTAB in general (albeit very bad in this case). This might be explained from the fact that the convergence of BICGSTAB depends on a 'shadow' Krylov subspace, which it implicitly generates. Now, if if one restarts, then BICGSTAB also starts to build a new, possibly different, 'shadow' Krylov subspace. This may lead to erratically convergent behavior in the first few steps. Therefore, it may happen that, if in the inner iteration BICGSTAB does not converge (to the relative precision), the 'solution' of the inner iteration is not very good and therefore the outer iteration may not give much improvement either. At the start the same more or less holds for GCROSTAB; however, after a few outer GCR iterations the 'improved' operator (A_k) somehow yields a better convergence than BICGSTAB by itself. This was also observed for more tests, although it also may happen that GCROSTAB converges worse than BICGSTAB.

In Fig. 7 a convergence plot versus the CPU-time is given for GCROSTAB, BICGSTAB, GCRO(10) and GMRESR(10). The fastest convergence in CPU-time is achieved by GCROSTAB(10), which is $\approx 20\%$ faster than BICGSTAB notwithstanding the extra work in orthogonalizations. We also see, that although GCRO(10) takes fewer iterations than GMRESR(10), in CPU-time the latter is faster. So in this case the decrease in iterations does not outweigh the extra work in orthogonalizations. For completeness we mention that GMRESRSTAB took almost 15 seconds to converge, whereas GMRES took almost 20 seconds.

Problem 3. The third problem is taken from [10]. The linear system stems from the discretization of the partial differential equation

$$-u_{xx} - u_{yy} + 1000(xu_x + yu_y) + 10u = f$$

on the unit square with zero Dirichlet boundary conditions. The stepsize in both x-direction and y-direction is 1/65. The right-hand side is selected once the matrix is constructed so that the solution is known to be $x = (1, 1, \ldots, 1)^T$. The zero vector was used as an initial guess.

In Fig. 9 we see a plot of the convergence history of full GMRES, GMRESR(5), GCRO(5) and GCRO(10,5) for two different truncation strategies, where the first parameter gives the dimension of the outer search space and the second the dimension of the inner search space. The number of vectors in the outer GCR iteration is twice the dimension of the search space. For the truncated version:

- 'da' means that we took $\epsilon = 10^{-3}$ and dropped the vectors $\hat{u}_1$ and $\bar{c}_1$ when $\delta(k) < \epsilon$ and assembled the vectors $\hat{u}_9$ and $\hat{u}_{10}$ as well as the vectors $\bar{c}_9$ and $\bar{c}_{10}$ when $\delta(k) > \epsilon$;
- 'tr' means that we dropped the vectors $\hat{u}_9$ and $\bar{c}_9$ each step ($\epsilon = 0$, see also [16]).

Notice that GCRO(5) displays almost the same convergence behavior as full GMRES. GMRESR(5) converges eventually, but only after a long period of stagnation. The truncated versions of GCRO(5) also display stagnation, but for a much shorter period. After that the 'da' version seems to converge as superlinear, whereas the 'tr' version still displays periods of stagnation, most notably at the end. This indicates that the 'da' version is more capable of keeping most of the 'convergence history' than the 'tr' version. This kind of behavior was seen in more tests: 'assembled' truncation strategies seem to work better than just discarding one or more iteration vectors.

In Table 1 we give the number of matrix vector products, the number of memory vectors and the CPU-time on a Sun workstation. From this table we see that GCRO(5) is by far the fastest method and

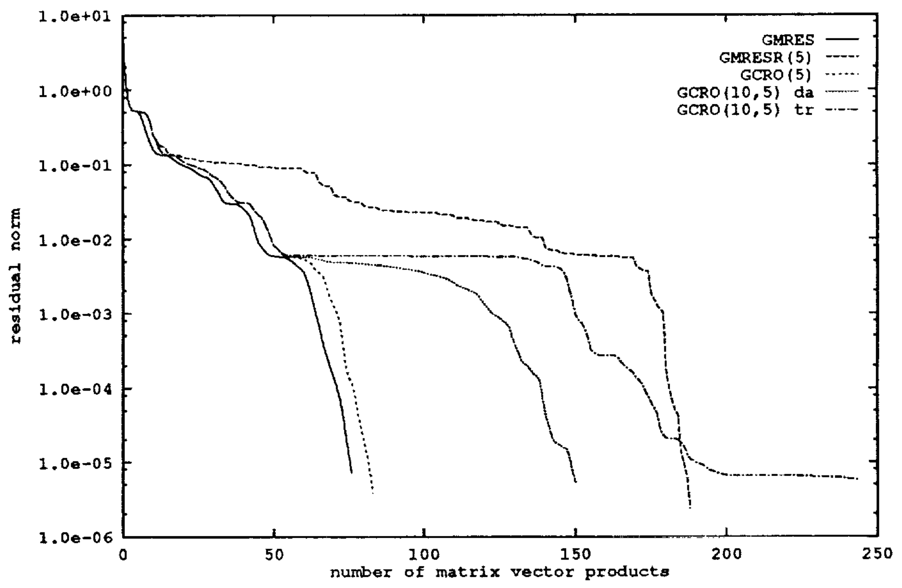


Figure 9: Convergence history for problem 3

uses about half the amount of memory vectors full GMRES and GMRESR(5) use. More interesting is that GCRO(10,5) 'da' converges in the same time as GMRESR(5), but uses only one third of the memory space.

CONCLUSIONS

We have derived from the GMRESR inner-outer iteration schemes a modified set of schemes, which preserve the optimality of the outer iteration. This optimality is lost in GMRESR since it essentially uses 'restarted' inner GMRES iterations, which do not take advantage of the outer 'convergence history'. Therefore, GMRESR may loose superlinear convergence behavior, due to stagnation or slow convergence of the inner GMRES iterations.

Method	Mat-Vec	Memory Vectors	CPU-time
GMRES	77	77	21.3
GMRESR(5)	188	81	18.5
GCRO(5)	83	39	9.4
GCRO(10,5) 'da'	150	25	18.3
GCRO(10,5) 'tr'	244	25	30.3

Table 1: Number of matrix vector products, number of memory vectors and CPU-time in seconds for problem 3

In contrast, the GCRO variants exploit the 'convergence history' to generate a search space that has no components in any of the outer directions in which we have already minimized the error. For GCRO(m) this means we minimize the error over both the inner search space and a sample of the entire previously searched Krylov subspace (the outer search space), resulting in a semi-full GMRES. This probably leads to the smooth convergence (much like GMRES) and the absence of stagnation, which may occur in the inner GMRES iteration of GMRESR. Apparently the small subset of Krylov subspace vectors that is kept approximates the entire Krylov subspace that is generated, sufficiently well. For both GMRESR(m) and GCRO(m) it seems that a small number of inner iterations works well.

We may also say, that the GCRO variants construct a new (improved) operator (of decreasing rank) after each outer GCR iteration. Although there is the possibility of breakdown in the inner method for GCRO, this seems to occur rarely as is indicated by theorem 4 (it has never happened in any of our experiments).

With respect to performance of the discussed methods we see that GCRO(m) (almost) always converges in fewer iterations than GMRESR(m). Because GCRO(m) is on average more expensive per iteration, this does not always lead to faster convergence in CPU-time. This depends on the relative costs of the matrix vector product and preconditioner w.r.t. the cost of the orthogonalizations and the reduction in iterations for GCRO(m) relative to GMRESR(m). Our experiments, with a cheap matrix vector product and preconditioner, show that already in this case the GCRO variants are very competitive with other solvers. However, especially when the matrix vector product and preconditioner are expensive or when not enough memory is available for (full) GMRES, GCRO(m) is very attractive. GCRO with BICGSTAB also seems to be a useful method, especially when a large number of iterations is necessary or when the available memory space is small relative to the problem size. GMRESR with BICGSTAB does not seem to work so well, probably because, to our observation, restarting BICGSTAB does not work so well.

We have derived sophisticated truncation strategies and shown by example that superlinear convergence behavior can be maintained. From our experience, the 'assembled' version seems to have the most promise.

Acknowledgements. The authors are grateful to Gerard Sleijpen and Henk van der Vorst for encouragement, helpful comments and inspiring discussions.

References

[1] O. Axelsson and P.S. Vassilevski. A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM J. Matrix Anal. Appl.*, 12:625–644, 1991.

[2] E. De Sturler. Nested Krylov methods based on GCR. Technical Report 93-.., Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1993.

[3] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H.A. Van der Vorst. *Solving Linear Systems on Vector and Shared Memory Computers.* SIAM Publications, Philadelphia, PA, 1991.

[4] S.C. Eisenstat, H.C. Elman, and M.H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20:345–357, 1983.

[5] V. Faber and T. Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM J. Numer. Anal.*, 21:352–362, 1984.

[6] R. Fletcher. Conjugate gradient methods for indefinite systems. In G.A. Watson, editor, *Numerical Analysis Dundee 1975, Lecture Notes in Mathematics 506*, pages 73–89, Berlin, Heidelberg, New York, 1976. Springer-Verlag.

[7] D.R. Fokkema. Hybrid methods based on the GCR principle (to appear). Technical report, Mathematical Institute, University of Utrecht, Utrecht, The Netherlands, 1993.

[8] R.W. Freund and N.M. Nachtigal. QMR: A quasi minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[9] J.A. Meijerink and H.A. Van der Vorst. An iterative solution method for linear equations systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

[10] Y. Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Statist. Comput.*, 14:461–469, 1993.

[11] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[12] G.L. Sleijpen and D.R. Fokkema. BiCGstab(l) for linear equations involving matrices with complex spectrum. Technical Report 772, Mathematical Institute, University of Utrecht, Utrecht, The Netherlands, 1993.

[13] P. Sonneveld. CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 10:36–52, 1989.

[14] H.A. Van der Vorst. BI-CGSTAB: A fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.

[15] H.A. Van der Vorst and C. Vuik. GMRESR: A family of nested GMRES methods. Technical Report 91-80, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1991.

[16] C. Vuik. Further experiences with GMRESR. Technical Report 92-12, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, The Netherlands, 1992.

59-64
197141
N94-23683

IMPLEMENTING ABSTRACT MULTIGRID OR MULTILEVEL METHODS*

Craig C. Douglas
Department of Computer Science
Yale University
New Haven, Connecticut

SUMMARY

Multigrid can be formulated as an algorithm for an abstract problem that is independent of the partial differential equation, domain, and discretization method. In such an abstract setting, problems not arising from partial differential equations can be treated also (c.f. aggregation-disaggregation methods). Quite general theory exists for linear problems, e.g., C. C. Douglas and J. Douglas, SIAM J. Numer. Anal., 30 (1993), pp. 136–158.

The general theory was motivated by a series of abstract solvers (Madpack). The latest version (4) was motivated instead by the theory. Madpack now allows for a wide variety of iterative and direct solvers, preconditioners, and interpolation and projection schemes, including user callback ones. It allows for sparse, dense, and stencil matrices. Mildly nonlinear problems can be handled. Also, there is a fast, multigrid Poisson solver (two and three dimensions).

The type of solvers and design decisions (including language, data structures, external library support, and callbacks) are discussed here. Based on the author's experiences with two versions of Madpack, a better approach is proposed here. This is based on a mixed language formulation (C and Fortran+preprocessor). Reasons for not just using Fortran, C, or C++ are given. Implementing the proposed strategy is not difficult.

1. INTRODUCTION

The term *abstract multigrid* was coined in [1]. This refers to theory which is quasi-independent of the elliptic boundary value problem. The dependence is introduced by assuming that the (discretized) problem satisfies a very small number of hypotheses which contribute simple expressions to the convergence rate formula. The theory in [1] is general enough to apply to nonnested solution spaces and includes example boundary value problems on general domains, with variable coefficients, and finite difference and finite element discretizations.

The concept of abstract multigrid was pushed to the extreme in [2], where a general theory for linear problems is presented with virtually no constraints on the origin of the problems.

Abstract multigrid is defined in §2. Two implementations of abstract multilevel methods (see [3] and [4]) are discussed in §3. A discussion of what might be the right set of languages to implement

*This work was supported in part by IBM and the Office of Naval Research.

abstract multilevel methods is in §4. Finally, some conclusions are drawn in §5.

2. ABSTRACT MULTIGRID

Assume we are solving some problem, possibly derived from a partial differential equation, possibly not. Assume further that by various means a sequence of (linear) problems

$$A_j x_j = b_j, \quad 1 \le j \le k, \tag{1}$$

are formed which approximate the *real problem*

$$A_k x_k = b_k, \tag{2}$$

where $x_j, b_j \in \mathcal{M}_j$, $1 \le j \le k$. Typically, $\mathcal{M}_j$ is a real or complex vector space when actually computing the solution to the problem. Frequently,

$$\dim(\mathcal{M}_j) \approx C \dim(\mathcal{M}_{j-1}), \quad C > 1.$$

There are typically three mappings between the neighboring solution spaces.

$$\begin{cases} \mathcal{R}_j,\ \mathcal{Q}_j : & \mathcal{M}_j \to \mathcal{M}_{j-1}, \quad 2 \le j \le k, \\ \qquad \mathcal{P}_j : & \mathcal{M}_j \to \mathcal{M}_{j+1}, \quad 1 \le j \le k-1. \end{cases}$$

The $\mathcal{R}_j$ and $\mathcal{Q}_j$ are *restriction* (or projection) matrices and the $\mathcal{P}_j$ are *prolongation* (or interpolation) matrices. Frequently, $\mathcal{P}_j = c\mathcal{R}_{j-1}^T$, where $c \in \mathbb{R}$. The matrices A_j and A_{j-1} are typically related through the relation

$$A_{j-1} = \mathcal{Q}_j A_j \mathcal{P}_{j-1}, \quad 2 \le j \le k.$$

The Galerkin form of multigrid requires that $\mathcal{Q}_j = \mathcal{P}_{j-1}^T$. The $\mathcal{Q}_j$ are frequently injection matrices when a finite difference discretization is applied to a partial differential equation.

A multilevel correction algorithm is simply defined by

```
Algorithm MGC ( $lev$, $\{A_j, x_j, b_j\}_{j=1}^k$, $\{\mathcal{P}_j\}_{j=1}^{k-1}$, $\{\mathcal{R}_j\}_{j=2}^k$ )
1.  $x_{lev} \leftarrow Solver_{lev}(A_{lev}, x_{lev}, b_{lev})$
2.  If $lev > 1$, then repeat 2a–2d until some condition is met:
    2a. $x_{lev-1} \leftarrow 0$, $b_{lev-1} \leftarrow \mathcal{R}_{lev}(b_{lev} - A_{lev}x_{lev})$
    2b. MGC ( $lev-1$, $\{A_j, x_j, b_j\}_{j=1}^k$, $\{\mathcal{P}_j\}_{j=1}^{k-1}$, $\{\mathcal{R}_j\}_{j=2}^k$ )
    2c. $x_{lev} \leftarrow x_{lev} + \mathcal{P}_{lev-1}x_{lev-1}$
    2d. $x_{lev} \leftarrow Solver_{lev}(A_{lev}, x_{lev}, b_{lev})$
```

A common condition in step 2 is to do steps 2a–2d some specified number of times (e.g., 0 for one way multigrid, 1 for a V Cycle, or 2 for a W Cycle).

On the coarsest level, $lev = 1$, the solver is frequently some flavor of Gaussian elimination (e.g., a sparse one). Common solvers on the other levels include relaxation methods (e.g., point, line, plane, or zebra Gauss-Seidel) and conjugate direction methods (e.g., conjugate gradients or residuals, CGS, GMRES, or Orthomin). The latter class of iterative methods is most effective on highly nonuniform meshes with a significant difference between the largest and smallest mesh spacing or diameter on a level.

A general algorithm that provides very good initial guesses is the nested iteration one:

```
Algorithm NIC ( $lev$, $\{A_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=1}^{k-1}$, $\{\mathcal{R}_j\}_{j=2}^{k}$ )
1.  MGC ( 1, $\{A_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=1}^{k-1}$, $\{\mathcal{R}_j\}_{j=2}^{k}$ )
2.  Do steps 2a–2b with $lev = 2, \cdots, k$:
    2a. $x_{lev} \leftarrow \mathcal{P}_{lev-1} x_{lev-1}$
    2b. MGC ( $lev$, $\{A_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=1}^{k-1}$, $\{\mathcal{R}_j\}_{j=2}^{k}$ )
```

A one way multilevel algorithm means that Algorithm MGC never performs any portion of its step 2 as part of its use by Algorithm NIC. Most complexity arguments showing that multigrid is of optimal order are based on Algorithm NIC, not Algorithm MGC.

For nonlinear problems, there are two standard approaches: the Full Approximation Scheme (FAS) and damped Newton multilevel. FAS is similar to Algorithm MGC, but changes two lines:

```
2a. $x_{lev-1} \leftarrow \mathcal{R}_{lev}^{(FAS)} x_{lev}$, $b_{lev-1} \leftarrow \mathcal{R}_{lev}(b_{lev} - A_{lev} x_{lev}) - A_{lev-1} x_{lev-1}$
2c. $x_{lev} \leftarrow x_{lev} + \mathcal{P}_{lev-1}(x_{lev-1} - \mathcal{R}_{lev}^{(FAS)} x_{lev})$
```

Note that in many situations $\mathcal{R}_{lev}^{(FAS)} = \mathcal{R}_{lev}$. Also, the operator A_j is not linear anymore, but involves function evaluations.

The damped Newton algorithm is a modification of Algorithm NIC. Before each reference to Algorithm MGC, a Jacobian is formed and a damped Newton step is performed. The last Jacobian on a level is saved for use in subsequent multilevel correction steps.

The difference between these two nonlinear approaches is easy to categorize. FAS uses a nonlinear iterative method (e.g., nonlinear Gauss-Seidel) while damped Newton uses standard linear solvers. When evaluating the nonlinear function is inexpensive, FAS usually produces an approximate solution faster than the damped Newton multilevel method. However, when the function evaluations are expensive, the damped Newton multilevel method usually produces an approximate solution faster than FAS.

Note that in Algorithms MGC and NIC, there are only two obvious components per level: the solver and the methods for passing information between levels. There are other components hidden by this formulation: a possible set of preconditioners for use by the solvers, a method for computing a matrix-vector product for some set of storage formats, and a set of discretization methods in the partial differential equation case.

For problems not arising from partial differential equations, the only components in Algorithm MGC that can be optimized are the solvers and the restriction matrices $\mathcal{Q}_j$ and $\mathcal{R}_j$. Both theory and practical experience demonstrate rather conclusively that finding better $\mathcal{Q}_j$ matrices is far superior to trying to find an optimal iterative method as the solver (e.g., see [5]).

For partial differential equation problems, using better discretization methods usually makes a bigger impact on the convergence rate than searching for a slightly better interpolation scheme or iterative solver. There are exceptions to this for trivial problems (e.g., Laplace's equation on a square with uniform grids).

3. MADPACK

The term madpack is a mnemonic for *multigrid (multilevel), aggregation-disaggregation package*. It started as a compact set of subroutines for solving problems of the form (1)–(2). The first two versions were released in 1986 and the fourth in 1992. All versions have been written using numerous macros to hide data structures and improve the readability. Currently, version 2 is available through Netlib and MGNet (see [6] and [7] for a description of MGNet). Version 2 is in the public domain. Version 4 is not really compatible with version 2 and is also owned by IBM. It is available through IBM's Internet anonymous ftp server and MGNet. All announcements and bug fixes for version 4 are distributed through MGNet.

Version 2 is discussed in §3.1. Version 4 is discussed in §3.2. A number of issues that these two versions raise are discussed in §4.

3.1. MADPACK, VERSION 2

Version 2 [8] was originally written in an extended flavor of Ratfor. A translator converted this to Fortran-77. This, in turn, is compiled by whatever compiler is available on a given machine. After determining that on some machines (e.g., SUN workstations in 1986) C versions of the subroutines ran up to 40% faster than the Fortran-77 equivalent, the entire code was ported to C. Including comments, there are only 1500–1600 lines in each language version. All three language versions are distributed.

Version 2 consists of 9 subroutines:

Routine	Description
klmg	Algorithm MGC
klni	Algorithm NIC
klax	matrix-vector multiply
kldsnf	factor matrices
kldsss	forward/backward solves
klres	compute residual
klsgs	Symmetric Gauss-Seidel
klsgsc	Preconditioned conjugate gradients
klsgsm	Preconditioned Orthomin(1)

The first two subroutines, klmg and klni, are meant to be the only user callable subroutines, but any can be called directly.

Version 2 supports an odd flavor of sparse matrix storage (see [9]) in the solver routines. The matrices A_j are assumed to have a symmetric nonzero structure, independent of whether or not $A_j = A_j^T$. This means that in some cases, a small number of zeroes are actually stored in the sparse matrix representation of A_j. The main diagonal, the nonzero elements of the columns of the upper triangular part of A_j, and the nonzero elements of the rows of the lower triangular part of A_j are stored independently (the lower part only if A_j is nonsymmetric). This allows for only half of the row or column indices to be stored due to the symmetry of the nonzero structure. It also allows for numerous computational simplifications and some tricks in reducing costs in the direct and iterative solvers (see [10]).

For restriction and prolongation matrices, two additional storage formats are supported. A general sparse matrix format, as implemented in the second Yale Sparse Matrix Package (see [11]) is useful on irregular grids. A stencil format is extremely efficient for uniform or tensor product grids. Typically, $r_j + c$ storage elements are used, where $r_j = \text{Rows}(\mathcal{R}_j)$ and c is a small natural number.

Table 1: Solvers and preconditioners

Solver	Preconditioner					
	None	User	ILU	Diag	SGS	SSOR
NoSolver	*	*	*	*	*	*
User	any	any	*	*	*	*
Factor	GD	*	*	*	*	*
Solve	GD	*	*	*	*	*
Symmetric Gauss-Seidel	G	*	*	*	*	*
Gauss-Seidel	GSD	*	*	*	*	*
Gauss-Seidel, red-black	GSD	*	*	*	*	*
Conjugate gradients	GSD	GSD	G	G	G	G
Minimum residuals	GSD	GSD	*	*	G	*
CGS	G	*	G	G	*	G
CGSTAB	G	*	G	G	*	G
GMRES	G	*	G	G	*	G

*	=	Error
G	=	General sparse matrices
S	=	Stencil matrices
D	=	Dense matrices
any	=	any format

Only Algorithms MGC and NIC are included. There is no support for nonlinear or time dependent problems. Version 2 has been imbedded in other people's nonlinear and time dependent codes, however. There is also no user callback mechanism, so that if the user has some special solver, preconditioner, or change of level subroutine, the source code for version 2 has to modified.

3.2. MADPACK, VERSION 4

This is a complete redesign and rewrite of Madpack. It is incompatible with version 2 in numerous ways. This is actually two quite distinct codes, DAMG [3] and DPMG [4]. DAMG is an abstract solver for linear and mildly nonlinear problems (FAS is supported). DPMG is a fast Poisson solver for two and three dimensional problems on simple uniform or tensor product grids.

DAMG supports dense, stencil, and general sparse matrix formats (this time, the more common first Yale Sparse Matrix Package [12] format was used) in the computational kernels. The dense case rarely occurs in solving partial differential equations; it is more common when solving aggregation-disaggregation problems (see [5]). Table 1 contains a summary of the solvers and preconditioners supported in the IBM version.

Unlike version 2, version 4 requires an external library of solvers (there are some solvers provided, but not many). What is distributed by IBM runs only on machines with their proprietary engineering and scientific subroutine library. Currently, this library only runs on IBM mainframes and RISC System/6000 workstations. Since DAMG was originally written on a machine that is not supported

Table 2: Level independent information data structure

iparm(i)		
i	Symbolic name	Definition
1	*mgfn*	Which multilevel algorithm
2	*l2infm*	Second dimension of *infm* array
3	*bxsize*	Length of *b* and *x* arrays
4	*lndm*	Length of *dm* array
5	*lnim*	Length of *im* array
6	*lnjm*	Length of *jm* array
7	*levelf*	Index of the finest level
8	*levelc*	Index of the coarsest level
9	*startl*	Index of the starting level
10	*presva*	Preserve coarsest level's matrices or not
11	*lastdm*	Index of last element in *dm* in use
12	*lastim*	Index of last element in *im* in use
13	*lastjm*	Index of last element in *jm* in use
14	*info*	Control of debugging information
15	*restart*	Continued computation indicator
20	*assist*	When all else fails

by this library, there is obviously a version which uses other libraries, e.g., LAPACK and the first Yale Sparse Matrix Package. Interfacing DAMG to other libraries is now fairly painless.

DAMG accepts three external subroutine arguments in case the users want to use their own solver(s), preconditioner(s), or change of level subroutine(s). In retrospect, there should have been a fourth for matrix-vector multiplies. These features are used extensively in DPMG to avoid storing matrices.

Both DAMG and DPMG are written in the same extended Ratfor as is version 2. Only the Fortran-77 translation is distributed by IBM, however. The codes assume double precision real data. Changing to single precision only requires changing one line of a file included by each of the Ratfor codes. Changing to complex data is only slightly harder.

DAMG can be restarted after it returns. This allows for coarse levels to be removed from the computational flow. It also allows an external adaptive grid refinement procedure to work with DAMG to add finer levels.

Data is passed to and from DAMG in the standard awkward style imposed by Fortran-77's limitations. Matrices and vectors are piled into a set of five (integer and real) vectors. As a substitute for the more natural pointer data type, indices are stored in information data arrays, indexed by the level number (see Tables 2–4). A language that supports more reasonable data structures, pointers, and dynamic memory allocation and freeing would simplify this.

Table 2 contains information which is level independent. This includes the length and the index of the last used element of certain vectors, which multilevel algorithm to start with, the indices of the finest, coarsest, and starting levels, how much debugging information to print, and whether this is a restart of an earlier computation.

Table 3 contains information relevant to the computational algorithms which is level dependent.

Table 3: Level dependent algorithm information data structure

$infalg(i,j)$ on level j		
i	Symbolic name	Definition
1	*Solver*	Which solution method
2	*SolverIters*	Iterations of *Solver*
3	*Precond*	Which preconditioning method
4	*MGIters*	Iterations of Algorithm MGC or MGFAS
5	*NIIters*	Iterations of Algorithm NIC or NIFAS
6	*IdxXB*	Index of first element of b_j or x_j in b or x
7	*NXB*	Number of elements in b_j and x_j
8	*Colors*	Number of colors in a multicolor ordering

Table 4: Matrix information data structure

$infm(i,k,j)$ on level j					
i/k	1	2	3	4	5
1	AType	RType	PType	NIPType	FASRType
2	ACols	RCols	PCols	NIPCols	FASRCols
3	ARows	RRows	PRows	NIPRows	FASRRows
4	ADim1	RDim1	PDim1	NIPDim1	FASRDim1
5	ADim2	RDim2	PDim2	NIPDim2	FASRDim2
6	IdxA	IdxR	IdxP	IdxNIP	IdxFASR
7	IdxIA	IdxIR	IdxIP	IdxINIP	IdxIFASR
8	IdxJA	IdxJR	IdxJP	IdxJNIP	IdxJFASR

Table 5: How matrices are chosen for changing levels

Wanted	*Order of selection*
$\mathcal{R}_j$	$\mathcal{R}_j$, $\mathcal{P}_{j+1}^T$, *and* $\mathcal{NIP}_{j+1}^T$
$\mathcal{P}_j$	$\mathcal{P}_j$, $\mathcal{R}_{j+1}^T$, *and* $\mathcal{NIP}_j$
$\mathcal{NIP}_j$	$\mathcal{NIP}_j$, $\mathcal{P}_j$, *and* $\mathcal{R}_{j+1}^T$
$\mathcal{R}_j^{(FAS)}$	$\mathcal{R}_j^{(FAS)}$, $\mathcal{R}_j$, $\mathcal{P}_{j+1}^T$, *and* $\mathcal{NIP}_{j+1}^T$

This includes the solver and preconditioner pairing, how many iterations of the algorithms to use on this level, the index into the solution and right hand side vectors for x_j and b_j, and their lengths.

When changing levels, it is very rare that $\mathcal{R}_j$, $\mathcal{P}_j$, $\mathcal{NIP}_j$, and $\mathcal{R}_j^{(FAS)}$ will all be defined. $\mathcal{NIP}_j$ corresponds to a special version of $\mathcal{P}_j$ in step 2a in Algorithm NIC (see §2). Usually only one or two of these will be defined. Further, the matrices are typically related to each other in very particular ways mathematically. An effort has been made to allow users of DAMG the option of generating only one matrix when it can be re-used or is the transpose of another matrix. DAMG determines which operation is wanted and then determines from information in the (three dimensional) *infm* data structure (see Table 4) how to change levels. Table 5 contains the order of choice, as determined by which matrix is wanted. The user callback for changing levels is the last choice unless the matrix type specifies doing this.

DPMG uses DAMG to do multileveling. Specialized solvers, interpolation, and projection subroutines are used throughout the computations, however. This means that DPMG does not store matrices normally, thus saving enormous amounts of memory which can be used instead for solving much larger problems. DPMG solves

$$\left\{\begin{aligned} -\Delta u &= b \text{ in } \Omega, \\ u &= g_0 \text{ on } \partial\Omega_0, \\ u_n &= g_1 \text{ on } \partial\Omega_1, \end{aligned}\right. \tag{3}$$

where $\partial\Omega_0 \cup \partial\Omega_1 = \partial\Omega$ and $\partial\Omega_0 \cap \partial\Omega_1 = \emptyset$.

This is discretized on grids

$$\bar{\Omega} = \Omega \cup \partial\Omega_0 \cup \partial\Omega_1.$$

In essence, linear systems of the form (1)–(2) are solved approximately for a sequence of grids $\bar{\Omega}_j$. The vectors x_j and b_j can be thought of as "grid functions" on $\bar{\Omega}_j$. The values of b, g_0, and g_1 on $\bar{\Omega}_j$ are stored in b_j (multiplied by the square of the mesh spacing when a uniform mesh is used). The values of g_0 on $\partial\Omega_0$ and an initial guess to the solution u in $\Omega \cup \partial\Omega_1$ are stored in x_j before the call to DPMG. DPMG uses a central difference discretization of Poisson's equation, even at Neumann boundary points. Dirichlet boundary points are not eliminated a priori.

Interpolation is either bilinear, trilinear, or a fourth order method based on (3). The latter uses the difference operator, similar to a Gauss-Seidel iteration with a three color ordering and a rotated operator, to improve the order of the interpolation (see [13]).

The three restriction methods are based on stencils. These are described in detail in [14]. The two second order methods are based on $[1,2,1]$ and $[1,4,1]$ weightings in one dimension. Tensor products are used to generate the stencils in higher dimensions. The fourth order stencil is an average of the $[1,4,1]$ tensor product stencil and point injection.

Only Algorithms MGC and NIC are options. The solvers are sparse Gaussian elimination and Gauss-Seidel with either the natural or red-black orderings.

DPMG was designed to run very fast on four quite different architectures:

1. IBM mainframes with vector units.
2. Conventional vector machines.
3. Nonvector machines with multiply-add hardware chaining.
4. Nonvector machines with no fancy hardware.

An example of 2 above is a Cray, an example of 3 is an IBM RISC System/6000 workstation, and an example of 4 is a SUN workstation or a PC.

The Gauss-Seidel with the natural ordering subroutines were rewritten in IBM mainframe vector assembler. These routines are always faster than the Fortran equivalents no matter what size vectors are used. As an interesting aside, a version was produced that completely vectorizes by using an odd re-interpretation of how to compute the updates based on the trailing vector elements that normally do not vectorize. This is described in [15]. The trick does not work in Fortran, C, or C++ unfortunately.

The usual philosophy for vectorizing Gauss-Seidel is to use a red-black ordering. In addition, this allows the interpolation subroutines to ignore half of the fine grid points. However, the red-black ordering has an unfavorable feature. The right hand side and approximate solution vectors pass through cache twice per iteration. Only if a solver is written in a blocked by the cache size manner can this be alleviated. Due to the boundary conditions in (3) and the fact that the matrices are not stored in DPMG, this makes things overly complicated to program. Hence, DPMG uses a traditional implementation for the red-black subroutines.

While the multilevel convergence properties of red-black Gauss-Seidel are better than the naturally ordered one, both solvers provide about equal performance when using Algorithm NIC and a V Cycle.

4. LANGUAGE ISSUES

In this section, advantages and disadvantages of Fortran, C, and C++ will be discussed in the context of an abstract multilevel solver. A mixed solution will be proposed.

4.1. FORTRAN

In §3.2, the disadvantages of Fortran-77 in terms of data structures were discussed. There is no conceivable way to get around this. Even using macros or Ratfor only helps so much. The real problem is that users of the package still have to initialize the data structures. They are not likely to use either my macros or Ratfor.

DAMG uses scratch storage in its solvers. Predicting the amount needed for each (solver, preconditioner) pair is an art which no user should ever have to master. Worse, the formulas given for some popular sparse matrix iterative solvers are wrong (predicting less memory than is required). For all of the solvers used in §3, the amount of scratch storage can be written in terms of N (the number or rows or columns), NZ (the number of nonzeroes in A_j), and a constant:

$$N_{scr} = C_n \cdot N + C_{NZ} \cdot NZ + C_{extra}. \tag{4}$$

While default values can be used, the user should be able to override these.

However, there are some areas where Fortran shines. For one, real and complex data types of various word lengths are part of the language. So, by using a simple preprocessor (e.g., /lib/cpp or m4) that is available on most computer systems used by people who do scientific computation, one source code can be maintained, even if multiple subroutine names are generated, one per data type supported. For example, in the Ratfor source code for DAMG, subroutine mgal is referenced by

```
NameIt(mgal)
```

Table 6: New Matrix Structure

```
struct Matrix {  int   MatrixType;     /* the matrix type */
                 int   MatrixCols;     /* number of columns */
                 int   MatrixRows;     /* number of rows */
                 int   MatrixLDim;     /* leading dimension for dense matrices */
                 void  *MatrixCoeffs;  /* Pointer to matrix elements */
                 int   *MatrixIA;      /* Pointer to IA elements */
                 int   *MatrixJA;      /* Pointer to JA elements */
               };
```

NameIt prepends the letter *d* (double real), *s* (single real), *z* (double complex), or *c* (single complex) depending on the definition of a macro, FLOAT.

Another area where Fortran does well is in optimizing code for certain classes of machines, particularly ones with vector units. The author naively assumed vector machines would go like the dinosaurs with the advent of superscalar, very fast workstations. Unfortunately (or fortunately depending on your view), vector units are being glued onto superscalar workstations by several manufacturers. While some C compilers have made serious inroads on producing very high-quality code, Fortran still holds some advantages in this case.

4.2. C

This language has an obvious disadvantage since complex and double complex are not a part of the language. While either of these can be defined as a structure, computing with them is inexcusably awkward. In particular, maintaining a single set of solvers for real and complex data means writing a set of weird macros to do floating point arithmetic. This is unacceptable.

However, not all of DAMG's or DPMG's subroutines are solvers. In fact, the multilevel algorithm or choose which solver to call subroutines are really doing bookkeeping, not floating point arithmetic. For these subroutines, C provides all of the necessary features to dramatically simplify the entire calling sequence and these subroutines. Just being able to dynamically allocate and free memory would reduce the user's frustration level with trying to guess how much memory to pass to DAMG for scratch storage.

C can easily save addresses of objects, e.g., of subroutines or data objects, in complicated data structures. Hence, routines can be called incrementally to pass very complex data objects to an implementation of an abstract multilevel algorithm without any one call being very complicated. This reduces the aggravation of using a complex program considerably.

4.3. C++

Many of the positive comments about C apply directly to C++. Classes can be constructed instead of structures. Further, C++ usually comes with a complex class (but not necessarily in both single and double precision), alleviating C's worst drawback.

One of C++'s strongest design features is the ability to design classes abstractly. At run time, the

Table 7: External subroutine information structure

```
struct  ExternSubr {
  int   (*Subr)();        /* Pointer to integer function */
  int   *IParms;          /* Pointer to integer parameters */
  void  *FParms;          /* Pointer to floating point parameters */
  float CN;               /* See (4) */
  float CNZ;              /* See (4) */
  float Cextra;           /* See (4) */
  int   SaveScr           /* Save scratch areas between calls? */
  void  **Scrs            /* Vector of pointers to scratch areas */
  int   *NScrs            /* Vector of lengths of scratch areas */
}
```

correct version of some virtual routine is accessed. This feature, while useful, is overkill in the context of abstract multigrid solvers. The data type *void** in C, a pointer to any data type, is sufficient to overcome many of the reasons why C++ would be useful in this context (see §4.4).

A drawback to using C++ is that there is frequently a lot of overhead hidden from the user. This makes C++ programs run unnecessarily slower than the equivalent C or Fortran programs. Interfacing C++ programs to Fortran programs is sometimes challenging, too.

A more serious drawback is that C++ has not yet been standardized. It is evolving with major new versions coming out yearly. This would not be so bad except that features are sometimes dropped or changed in incompatible ways in newer versions of the language. For someone who wants to write a code once and then never have to touch it again, this is not a good point in C++'s favor.

4.4. C AND FORTRAN: MIXED LANGUAGE PROGRAMMING

My personal belief is that mixing Fortran+preprocessor and C is the best choice now. Implement Algorithms MGC and NIC in C and implement the computational solvers in FORTRAN+preprocessor. Numerous people who compute only know one language well and are not comfortable normally with a mixed language set of programs. An interface is described at the end of this section to let these people use what is proposed.

Suppose that we make no assumption about the language of a solver or preconditioned subroutine, other than it really can be called from C. Then we do not know if it can dynamically allocate memory. Hence, some mechanism must be defined for passing a block of memory. One way is to define a structure for externally called subroutines, e.g., Table 7. The subroutine is expected to return some indication of whether or not it worked or produced an error. The IParms and FParms are integer and floating point vectors containing information that the specific subroutine actually needs. Setting CN=CNZ=Cextra=0 could signify "use the defaults." Note that only one ExternSubr structure has to be created per subroutine. In this definition, Subr is a pointer (or external reference) to an integer valued function with a fixed set of arguments. By providing an include file with an abstract solver, a set of default ExternSubr structures can be given to the user (see Table 1).

Consider Table 4. A single structure can be defined that defines everything in a column of Table

4, so that information about matrices can be made easier to define. Also, pointers to the actual floating point and integer vectors or matrices can be defined (instead of indices into a messy vector), placing all of the relevant information in one place (see Table 6).

Information that is in both Tables 3 and 4 can be re-arranged into a single data structure as in Table 8. A NULL pointer can be used to indicate the lack of existence of a matrix.

An implementation of Algorithm MGC can then use the information in LevInfo and the ExterSubr structures to first allocate scratch space (if necessary), then call the solver. Assume *lp* is a pointer to level j's LevInfo structure, that *lap* is a pointer to *lp* $\rightarrow A_j$'s Matrix structure, *lps* is a pointer to *lp* →solver's ExternSubr structure, and *lpp* is a pointer to either *lp* →precond's ExternSubr structure or an empty one. Then the solver is called using the following:

iret = *lps* →Subr(dtype, *lpp* →Subr, *lp* →SolverIters, *lp* →SolverRNorm,
lp →matrix_vec, *lap* →MatrixType, *lap* →MatrixRows,
lap →MatrixCols, *lap* →MatrixCoeffs, *lap* →MatrixIA,
lap →MatrixJA, *lp* $\rightarrow X_j$, *lp* $\rightarrow B_j$, *lps* →IParms,
lps →FParms, resid, scrs, nscrs, scrp, nscrp, oldscr);

Here scrs and scrp are pointers to scratch storage (with lengths nscrs and nscrp) for use by the solver and the preconditioner subroutines. Whether or not this is the same set of scratch areas as a previous call is indicated by oldscr. The resid argument is so that the solver has a place to return the residual, which is used in calculating the next correction problem on a coarser level.

Numerous iterative procedures, based primarily on conjugate direction methods, require a user callback routine to calculate matrix-vector products, thus requiring a matrix_vec argument to be passed. Also, many iterative procedures allow a stopping criterion based on reducing the (possibly scaled) residual norm by some amount, e.g., *lp* →SolverRNorm.

There is an important issue that must be addressed. There are many people who compute who do not know C, but only Fortran. Using the data structures advocated in §4.2 would preclude these people from using the abstract solvers. Some simple subroutines, callable from Fortran (or any language) that build the data structures in a portable manner must be included. For example, a Fortran program can call a C program which returns a *data handle* (a small integer):

mgh=mgini (levels, dtype)

This subroutine allocates space for the structures. The integer argument dtype is used to determine the data type (c.f., the value of FLOAT in §4.1):

Dtype	Data	Floating point data description
1	float	single precision real
2	double	double precision real
3	complex	single precision complex
4	dcomplex	double precision complex
<0	user	−value = length in bytes

While this may seem ugly, this simple mechanism allows the C codes to be written in a "typeless" manner. Note that a mechanism is in place for user defined data types as well.

Matrix structures are defined similarly and return a *matrix handle*:

mat = mgmat (mgh, type, cols, rows, ldim, coeffs, ia, ja)

Matrix handles are coupled to the data handle.

Table 8: Level Information Structure

```
struct LevInfo {
        struct  ExternSubr *solver;       /* Pointer to how to call solver */
        struct  ExternSubr *precond;      /* Pointer to how to call preconditioner */
        struct  ExternSubr *matrix_vec;   /* Pointer to how to call matrix*vector */
        struct  ExternSubr *change_lev;   /* Pointer to how to call level changer */
        int     SolverIters;              /* Number of iterations in solver() */
        float   SolverRNorm;              /* How much to reduce residual norm */
        int     MGIters;                  /* Number of iterators of MGC */
        int     NIIters;                  /* Number of iterators of NIC */
        void    *X_j;                     /* Pointer to x_j */
        void    *B_j;                     /* Pointer to b_j */
        int     NX_j;                     /* Length of x_j */
        int     NB_j;                     /* Length of b_j */
        int     NZA_j;                    /* Number of nonzeroes in A_j */
        struct  Matrix *A_j;              /* Pointer to A_j representation */
        struct  Matrix *R_j;              /* Pointer to R_j representation */
        struct  Matrix *P_j;              /* Pointer to P_j representation */
        struct  Matrix *NIP_j;            /* Pointer to NIP_j representation */
        struct  Matrix *FASR_j;           /* Pointer to R_j^(FAS) representation */
        };
```

Subroutines are declared through another C routine:

```
                real      CN, CNZ, Cextra
                external  rtn
                ...
                (set CN, CNZ, and Cextra)
                isubr = mgsubr ( mgh, rtn, iparms, fparms, CN, CNZ, Cextra, savscr
                )
```

Note that only the addresses of rtn, iparms, and fparms are saved by mgsubr, not the contents. A *subroutine handle* is returned which is coupled to the data handle. Use of the Fortran EXTERNAL declaration allows subroutine addresses to be passed.

Another routine can be called to setup a LevInfo structure for level j:

```
                iret = mglevi ( mgh, j, isolver, iprecond, imatv, ichlev,
             *                nsolviters, rnorm, mgiter, niiter, xj, bj, nxj, nbj,
             *                nza, mata, matr, matp, matnip, matfas )
```

Here, isolver, iprecond, imatv, and ichlev are the return values from mgsubr or 0 if none is wanted. Also, mata–matfas are return valves from mgmat or 0 if no matrix exists. The x_j and b_j are the addresses of the first elements of xj and bj. These may be indexed as X(ixb) and B(ixb), respectively, depending on the user's programming style. A nonzero return value means an error occurred.

Finally, the multilevel subroutines can be called:

```
                iret = mgmeth ( mgh, iparm, resid )
```

where iparm is a simplification of the one in Table 2 (it only needs to contain mgalg, startl, levelc, levelf, and info, but is extendable). The last argument, resid, is an array where the final residual is returned. A nonzero return value means an error occurred.

To free space, a final call can be made:

```
iret = mgdone ( mgh )
```

A nonzero return value means an error occurred. Obviously, this last call is unnecessary if the program immediately ends.

The advantage of this approach is that subroutines can be written in whatever language makes the most sense. Further, people who program in C or C++ will not be penalized by having to construct data structures that only make sense in Fortran.

The worst disadvantage is that to compile the library, some knowledge is needed about how the local compiler treats subroutine names. There are three common methods in use and on many platforms this can be determined automatically. On a very small number of machines, Fortran and C programs cannot be mixed conveniently or at all; these machines will be ignored by this author.

5. CONCLUSIONS

In this paper, abstract multilevel methods were reviewed. Two versions of the author's publicly distributed multilevel codes (Madpack) were discussed. From the experience of these codes, a model of a better approach using a mixed language approach (C and Fortran+preprocessor) was proposed. Implementing such a system, starting from having already working solvers (e.g., [8], [3], and [4]) is a simple exercise for an expert in C and Fortran programming.

REFERENCES

[1] C. C. Douglas. Multi–grid algorithms with applications to elliptic boundary–value problems. *SIAM J. Numer. Anal.*, 21:236–254, 1984.

[2] C. C. Douglas and J. Douglas. A unified convergence theory for abstract multigrid or multilevel algorithms, serial and parallel. *SIAM J. Numer. Anal.*, 30:136–158, 1993.

[3] C. C. Douglas. DAMG: an abstract multilevel solver. Technical Report YALEU/DCS/TR-950, Department of Computer Science, Yale University, New Haven, 1993.

[4] C. C. Douglas. DPMG: a multigrid solver for the poisson equation in two and three dimensions. Technical Report YALEU/DCS/TR-951, Department of Computer Science, Yale University, New Haven, 1993.

[5] F. Chatelin and W. L. Miranker. Acceleration by aggregation of successive approximation methods. *Lin. Alg. Appl.*, 43:17–47, 1982.

[6] C. C. Douglas. Mgnet Digests and Code Repository. Monthly digests subscribed to by sending a message to mgnet-requests@cs.yale.edu and an anonymous ftp site (casper.cs.yale.edu) for codes and papers on multigrid and related topics.

[7] C. C. Douglas. MGNet: a multigrid and domain decomposition network. *ACM SIGNUM Newsletter*, 27:2–8, 1992.

[8] C. C. Douglas. Madpack (version 2) users' guide. Technical Report 16169, IBM Research Division, Yorktown Heights, New York, 1990. The most up to date source code is available through anonymous ftp from casper.cs.yale.edu in the directory mgnet/madpack2. It is also available through the netlib service.

[9] R. E. Bank and R. K. Smith. General sparse elimination requires no permanent integer storage. *SIAM J. Sci. Stat. Comp.*, 8:574–584, 1987.

[10] R. E. Bank and C. C. Douglas. An efficient implementation of the SSOR and ILU preconditionings. *Appl. Numer. Math.*, 1:489–492, 1985.

[11] S. C. Eisenstat, H. E. Elman, M. H. Schultz, and A. H. Sherman. The (new) Yale sparse matrix package. In A. L. Schoenstadt and G. Birkhoff, editors, *Elliptic Problem Solvers II*. Academic Press, New York, 1983.

[12] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman. Yale sparse matrix package: II. the nonsymmetric codes. Technical Report 114, Department of Computer Science, Yale University, New Haven, 1977.

[13] J. H. Hyman. Mesh refinement and local inversion of elliptic differential equations. *J. of Comput. Phys.*, 23:124–134, 1977.

[14] 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.

[15] C. C. Douglas. Some remarks on completely vectorizing point Gauss–Seidel while using the natural ordering. Technical Report YALEU/DCS/TR-943, Department of Computer Science, Yale University, New Haven, 1992.

S/0-25

197142

N94-29683

NUMERICAL SOLUTION OF FLAME SHEET PROBLEMS WITH AND WITHOUT MULTIGRID METHODS*

Craig C. Douglas
Department of Computer Science
Yale University
New Haven, Connecticut

Alexandre Ern
Department of Mechanical Engineering
Yale University
New Haven, Connecticut

SUMMARY

Flame sheet problems are on the natural route to the numerical solution of multidimensional flames, which, in turn, are important in many engineering applications. In order to model the flame structure more accurately, we use the vorticity-velocity formulation of the fluid flow equations instead of the streamfunction-vorticity approach. The numerical solution of the resulting nonlinear coupled elliptic partial differential equations involves a pseudo transient process and a steady state Newton iteration. Rather than working with dimensionless variables, we introduce scale factors that can yield significant savings in the execution time. In this context, we also investigate the applicability and performance of several multigrid methods, focusing on nonlinear damped Newton multigrid, using either one way or correction schemes.

1. INTRODUCTION

Recent advances in the development of computational algorithms and supercomputers have provided new extremely powerful tools with which to investigate chemically reacting systems that were computationally infeasible only a few years ago (see [1], [2], [3], and [4]). The difficulties associated with solving high heat release combustion problems stem from the large number of dependent unknowns, the nonlinear character of the governing partial differential equations and the different length scales present in the problem. Typical combustion problems may involve, in addition to the temperature and the fluid dynamics variables, dozens of species defined at each grid point and require the resolution of curved fronts whose thickness is on the order of thousandths of the domain diameter, across which critical fields vary by orders of magnitude. As a result of the fluid dynamics-thermochemistry interaction and its effect on the flame structure, the governing

*This work was supported in part by CERMICS, ENPC, IBM, the Office of Naval Research, and the Department of Energy.

PRECEDING PAGE BLANK NOT FILMED

PAGE 142 INTENTIONALLY BLANK

equations are strongly coupled together and are also characterized by the presence of stiff source terms and nonlinearities. Hence, Newton methods with sophisticated control strategies, including damping and adaptive continuation techniques, are needed. However, in spite of these difficulties, the numerical modeling of multidimensional laminar (or turbulent) flames has been recently motivated by the growing demand for high fuel efficiency combined with low pollutant emission. While three dimensional turbulent flame simulations still remain infeasible on current supercomputers, axisymmetric laminar diffusion flames constitute a problem of practical importance since they are the flame type of several combustion devices. Hence, new robust numerical models of such a system will provide an efficient tool to probe flame structures and investigate the coupled effects of complex transport phenomena with chemical kinetics.

As part of an ongoing effort to expand combustion modeling capabilities, we investigate computationally the performance of several multigrid techniques (see [5], [6], [7], and [8]) combined with the numerical solution of combustion related problems. In the present work, we consider a flame sheet problem rather than a finite rate chemistry model for an axisymmetric laminar diffusion flame in order to alleviate the memory and CPU requirements on the computer simulations. The numerical techniques presented in this paper, however, also apply to combustion problems with finite rate chemistry [9]. We note that a flame sheet model adds only one field to the hydrodynamic fields that describe the underlying flow. A detailed kinetics model adds as many fields as species considered in the kinetic mechanism, each with its own coupled conservation equation. Since the CPU time and the memory requirements scale with the square of the number of dependent unknowns, the flame sheet model considerably reduces the cost of the computer simulations while still keeping the coupling and nonlinearity features associated with the original problem.

In the flame sheet model, the chemical reactions are described with a single one step irreversible reaction corresponding to infinitely fast conversion of reactants into stable products. This reaction is assumed to be limited to a very thin exothermic reaction zone located at the locus of stoichiometric mixing of fuel and oxidizer, where temperature and products of combustion are maximized. To further simplify the governing equations, one neglects thermal diffusion effects, assumes constant heat capacities and Fick's law for the ordinary mass diffusion velocities, and takes all the Lewis numbers equal to unity [2]. With these approximations, the energy equation and the major species equations take on the same mathematical form and by introducing Schvab-Zeldovich variables, one can derive a source free convective-diffusive equation for a single conserved scalar. Although no information can be recovered about minor or intermediate species in the flame sheet limit, the temperature and the stable major species profiles in the system can be obtained from the solution of the conserved scalar equation coupled to the flow field equations. Further, the location of the physical spatially distributed reaction zone and its temperature distribution can be adequately predicted by the flame sheet model for many important fuel-oxidizer combinations and configurations. Since being studied as a means of obtaining an approximate solution to use as an initial iterate for a one dimensional detailed kinetics computation in [10], flame sheets have been routinely employed to initialize multidimensional diffusion flames.

In §2, a comparison of three possible formulations of the problem is presented, including the governing equations and boundary conditions. In §3, the general solution algorithms are presented, including a damped Newton method, Jacobian evaluation, linear solvers (Bi-CGSTAB or GMRES), and the pseudo transient process. In §4, various multigrid methods are discussed in the context of flame sheets. In §5, numerical experiments are presented. Finally, in §6, some conclusions are reached.

2. VORTICITY-VELOCITY FORMULATION

In diffusion flames the combustion process is primarily controlled by the rate at which the fuel and oxidizer are brought together in stoichiometric proportions. Thus, independently of the submodel used for the chemical kinetics (finite rate vs. flame sheet), the overall accuracy of the numerical solution strongly depends on an accurate representation of the flow field. Hence, a brief discussion on the various formulations of the Navier-Stokes equations in the context of laminar combustion problems is of order.

The first numerical solution of two dimensional axisymmetric laminar diffusion flames was obtained using the streamfunction-vorticity formulation [2]. This approach is attractive for three reasons:

1. It eliminates the coupling associated with the presence of the pressure in the momentum equations.
2. It reduces the number of equations to be solved by one.
3. It also has the important advantage that continuity is explicitly satisfied locally.

However, the specification of boundary conditions meets with difficulties when one attempts to specify vorticity boundary values. In particular, a zero vorticity boundary condition at the inlet of the computational domain results in a rough approximation of the true solution, thus severely altering the resulting velocity field [3]. On the other hand, the specification of vorticity boundary values in terms of the streamfunction requires the discretization of second order derivatives, thus yielding off diagonal terms in the Jacobian matrix which result in having to solve severely ill conditioned linear systems. Another important difficulty associated with the streamfunction-vorticity approach is that the extension to three dimensional configurations through the introduction of a vector potential instead of the scalar streamfunction is cumbersome and computationally expensive since it introduces additional dependent variables.

Alternatively, a primitive form of the Navier-Stokes equations has been recently implemented for several axisymmetric laminar diffusion flames (see [3] and [4]). In this approach, the velocity field is computed using the momentum equations and the pressure field is recovered from the continuity equation. As a result of the difference in nature of the governing equations, the discrete pressure field has to be determined in a manner consistent with the discrete continuity equation. This can be achieved to machine zero on a staggered grid. However, staggered mesh schemes do also have drawbacks in complex geometries configurations where non-orthogonal curvilinear coordinates are used and when using sophisticated numerical techniques such as multigrid methods (see [11] and [12]). Although feasible ([13] and [14]), the development of staggered grid based multigrid solvers is computationally cumbersome since the transfer operators between levels do not coincide for each dependent variable in order to preserve a staggered grid arrangement on all levels. This difficulty may even be further exacerbated in three dimensional configurations. Finally, it is worthwhile to note that two and three dimensional solutions of incompressible viscous flows on a nonstaggered grid have been reported (see [11] and [12]). However, the extension of such procedures to highly compressible systems where the density can vary by several orders of magnitude inside the computational domain may still yield some complications.

The vorticity-velocity formulation constitutes a third approach to the numerical solution of the Navier-Stokes equations. A review of incompressible fluid flow computations using this formulation is well documented in [15]. The vorticity-velocity formulation of the Navier-Stokes equations has been recently extended to two and three dimensional compressible flows and implemented for the numerical solution of flame sheet problems (see [16] and [17]). As motivated in these references, a vorticity-velocity formulation allows replacement of the first order continuity equation with additional second order equations. Whereas the streamfunction-vorticity formulation also accomplishes the same replacement in two dimensions, vorticity-velocity is extensible to three and allows more accurate formulation of boundary conditions in a numerically compact way. Furthermore, off diagonal convective terms in off diagonal blocks that exert a strong influence in a streamfunction-vorticity formulation disappear. Another important attractive feature of the vorticity-velocity formulation is that the governing equations can be discretized on a nonstaggered grid, thus allowing the implementation of a multigrid algorithm at a relatively low overhead in additional programming (see [16], [17], and [18]).

The flame sheet governing equations consist of the conservation of total mass, momentum and a conserved scalar equation. The conservation of total mass and momentum equations constitute the flow field problem and are formulated using the vorticity-velocity formulation of the compressible axisymmetric Navier-Stokes equations. A source free convective-diffusive equation for a conserved scalar is solved coupled together with the flow field equations and the temperature and major stable species profiles in the system can be recovered from the conserved scalar (see [2], [19], and references therein). We introduce the velocity vector $v = (v_r, v_z)$ with radial and axial components v_r and v_z, respectively, and the normal component of the vorticity

$$\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}. \tag{1}$$

The vorticity transport equation is formed by taking the curl of the momentum equations, which eliminates the partial derivatives of the pressure field. A Laplace equation is obtained for each velocity component by taking the gradient of (1) and using the continuity equation. This yields the governing equations in the following form:

$$\begin{aligned}
\frac{\partial^2 v_r}{\partial r^2} + \frac{\partial^2 v_r}{\partial z^2} &= \frac{\partial \omega}{\partial z} - \frac{1}{r}\frac{\partial v_r}{\partial r} + \frac{v_r}{r^2} - \frac{\partial}{\partial r}\left(\frac{v\cdot\nabla\rho}{\rho}\right),\\
\frac{\partial^2 v_z}{\partial r^2} + \frac{\partial^2 v_z}{\partial z^2} &= -\frac{\partial \omega}{\partial r} - \frac{1}{r}\frac{\partial v_r}{\partial z} - \frac{\partial}{\partial z}\left(\frac{v\cdot\nabla\rho}{\rho}\right),\\
\frac{\partial^2 \mu\omega}{\partial r^2} + \frac{\partial^2 \mu\omega}{\partial z^2} + \frac{\partial}{\partial r}\left(\frac{\mu\omega}{r}\right) &= \rho v_r\frac{\partial \omega}{\partial r} + \rho v_z\frac{\partial \omega}{\partial z} - \frac{\rho v_r}{r}\omega + \overline{\nabla}\rho\cdot\nabla\frac{v^2}{2} - \overline{\nabla}\rho\cdot g +\\
&\quad 2\left(\overline{\nabla}(\mathrm{div}(v))\cdot\nabla\mu - \nabla v_r\cdot\overline{\nabla}\frac{\partial\mu}{\partial r} - \nabla v_z\cdot\overline{\nabla}\frac{\partial\mu}{\partial z}\right),\\
\frac{1}{r}\frac{\partial}{\partial r}\left(r\rho D\frac{\partial S}{\partial r}\right) + \frac{\partial}{\partial z}\left(\rho D\frac{\partial S}{\partial z}\right) &= \rho v_r\frac{\partial S}{\partial r} + \rho v_z\frac{\partial S}{\partial z},
\end{aligned} \tag{2}$$

where ρ is the density, μ the viscosity, g the gravity vector, div(v) the cylindrical divergence of the velocity vector, S the conserved scalar, D a diffusion coefficient, and the components of $\overline{\nabla}\beta$ are $(\frac{\partial\beta}{\partial z}, -\frac{\partial\beta}{\partial r})$. The density is computed using the perfect gas law and, in the low Mach numbers approximation valid for these flame configurations, one can use the outlet (constant) pressure.

Table 1: Boundary conditions

Axis of symmetry $(r = 0)$	$v_r = 0$	$\frac{\partial v_z}{\partial r} = 0$	$\omega = 0$	$\frac{\partial S}{\partial r} = 0$
Outer zone $(r = R_{max})$	$\frac{\partial v_r}{\partial r} = 0$	$\frac{\partial v_z}{\partial r} = 0$	$\omega = \frac{\partial v_r}{\partial z}$	$S = 0$
Inlet $(z = 0)$	$v_r = 0$	$v_z = v_z^0(r)$	$\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}$	$S = S^0(r)$
Exit $(z = L)$	$v_r = 0$	$\frac{\partial v_z}{\partial z} = 0$	$\frac{\partial \omega}{\partial z} = 0$	$\frac{\partial S}{\partial z} = 0$

Consequently, in the above formulation, the pressure field is eliminated from the governing equations as a dependent unknown and can be recovered, once a computed numerical solution of (2) is obtained, by solving a Laplace type equation derived by taking the divergence of the momentum equations [15].

Recalling that all of the Lewis numbers are taken equal to unity, the quantity ρD is given by the viscosity coefficient μ divided by a reference Prandtl number and we use an approximate value for air, $Pr = 0.75$. Hence, in this model, the determination of all the transport coefficients is reduced to the specification of a transport relation for the viscosity and we use the same power law as the one given in [2]. We also note that, due to the high temperature gradients present in the system, the viscosity derivatives in the right hand side of the vorticity transport equation (2) can not be neglected. Our numerical experiments show that such an approximation leads to significant differences in the numerical solution, especially for the radial velocity profile. Finally, a conservative form of the convective terms can also be considered but it yields slower convergence rates without any significant changes in the computed solution.

A schematic of the physical configuration is given in Figure 1. It consists of an inner cylindrical fuel jet (radius R_I =0.2cm), an outer co-flowing annular oxidizer jet (radius R_O =2.5cm) and a dead zone extending to R_{max} =7.5cm. The inlet velocity profile of the fuel and oxidizer are a plug flow of 35cm/s. This yields a typical value for the Reynolds number of 550. Further, the flame length is approximately L_f =3cm [19] and the length of the computational domain is set to L =30cm. Although the fuel and oxidizer reservoirs are at room temperature (300 °Kelvin), we need to assume, in the flame sheet model, that the temperature already reaches the peak temperature value along the inlet boundary at $r = R_I$. This peak temperature is estimated for a methane-air configuration to be 2050°K. Hence, the inlet profile of the conserved scalar, $S^0(r)$, is specified in such a way that the resulting temperature distribution blends the room temperature reservoirs and the peak temperature by means of a narrow Gaussian centered at R_I. The narrowness of the Gaussian profile has a relevant influence on the calculated flame length, so that its parameters have to be determined appropriately [19]. The boundary conditions are summarized in Table 1. Finally, we note that the use of the definition of the vorticity (1) for the vorticity outlet boundary condition does not yield any relevant changes in the computed solution.

3. GENERAL SOLUTION ALGORITHM

The partial differential equations (2) together with the boundary conditions (see Table 1) are discretized on a two dimensional tensor product grid. A solution is first obtained on an initial coarse grid. Additional mesh points are then adaptively inserted in regions of high physical activity by equidistributing weight functions of the local gradient and curvature of the numerical solution

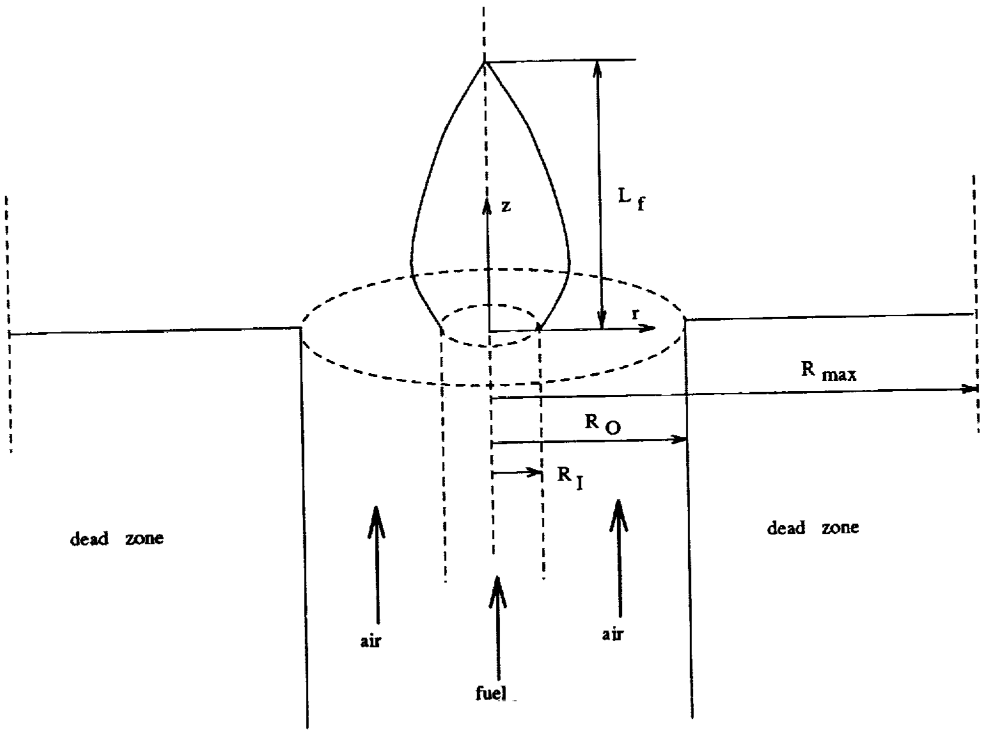


Figure 1: Physical configuration (not in scale)

[2], which yields a 129 × 161 grid. To verify the grid independence of the solution, we refined this grid to 257 × 219 points. The relative error between the two solutions was found to be lower than 2% and differences were only encountered in the outflow region where the grids were still kept somewhat coarse. However, the flame length and the temperature distribution inside the flame were accurately predicted on the 129 × 161 grid. Hence, this grid will be considered as the finest grid in the present work.

The spatial operators in the partial differential equations (2) are approximated with finite difference expressions. Diffusion and source terms are evaluated using centered differences. We adopt a monotonicity preserving upwind scheme for the convective terms (see [20, p. 304]), for instance,

$$v_r \frac{\partial S}{\partial r} = \max\{(v_r)_{i-\frac{1}{2}}, 0\} \frac{S_i - S_{i-1}}{r_i - r_{i-1}} - \max\{-(v_r)_{i+\frac{1}{2}}, 0\} \frac{S_{i+1} - S_i}{r_{i+1} - r_i}. \tag{3}$$

The boundary conditions given in Table 1 involve only zero or first order derivatives. For the latter terms, first order back or forward differences can be used, except for two boundary conditions which require a more accurate treatment. First, as motivated in [17], the vorticity inlet boundary condition is discretized using the vorticity values at the first two lines of the computational domain. More specifically, at an inlet point $(i, 1)$, we discretize the equation $\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}$ as follows:

$$\frac{1}{2}(\omega_1 + \omega_2) = \frac{(v_r)_2}{z_2 - z_1} - \frac{(v_z)_{i+1} - (v_z)_{i-1}}{r_{i+1} - r_{i-1}}. \tag{4}$$

It is also of critical importance for the accuracy of the numerical solution that the axial velocity boundary condition on the axis of symmetry be evaluated using a second order scheme. At any point $(1,j)$, we have

$$(v_z)_2 - (v_z)_1 = \frac{(r_2 - r_1)^2}{2}\frac{\partial^2 v_z}{\partial r^2} + \mathcal{O}\left((r_2 - r_1)^2\right).$$

The right hand side is evaluated using the Laplace equation for v_z in (2). On the axis of symmetry, this reduces to

$$\frac{\partial^2 v_z}{\partial r^2} = -\frac{\partial^2 v_z}{\partial z^2} - \frac{\partial \omega}{\partial r} - \frac{\partial}{\partial z}\left(\frac{v_z}{\rho}\frac{\partial \rho}{\partial z}\right).$$

The radial derivative of the vorticity can be discretized with a first order difference while still yielding an overall second order accuracy for v_z. By comparing our numerical solutions with a primitive variable solution of the same problem [19], we found that these two boundary conditions exerted a strong influence on the overall accuracy of the numerical solution.

The discretization of the partial differential equations (2) together with the boundary conditions (Table 1) yields a set of algebraic equations of the form $F(U) = 0$, which is solved using a damped Newton method

$$J(U^n)\Delta U^n = -\lambda^n F(U^n), \qquad n = 0, 1, \ldots, \tag{5}$$

with convergence tolerance $\|\Delta U^n\|_S < 10^{-5}$. The Jacobian matrix $J(U^n)$ is computed numerically using vector function evaluations and the grid nodes are split into nine independent groups which are perturbed simultaneously (see [2] for more details). Selected cases were rerun with a more stringent convergence tolerance of 10^{-6}, without any significant changes in the numerical solution. Rather than working with dimensionless variables, we introduce a scale factor α_l, $l \in [1, n_c]$, for each dependent variable ($n_c = 4$ for the flame sheet problem). The norm of the discrete vector ΔU^n is then given by

$$\|\Delta U^n\|_S = \sqrt{\sum_{i\in[1,n_r]}\sum_{j\in[1,n_z]}\sum_{l\in[1,n_c]}\left(\alpha_l \Delta U^n(l,i,j)\right)^2}. \tag{6}$$

It is worthwhile to point out that an appropriate choice of the scale factors can yield significant savings in the execution time. This point will be further illustrated with numerical experiments in §5.1.

The linear system (5) is inverted at each Newton step through an inner iteration. This inner iteration may consist of either the Bi-CGSTAB algorithm [21] or a restarted version of GMRES [22] combined with a Gauss-Seidel (GS) left preconditioner. This choice is motivated in [16] through various numerical simulations of flame sheet problems. Although a single Bi-CGSTAB/GS iteration requires approximately 1.5 times more time than an average GMRES/GS iteration, both algorithms yield total execution times which are in general within a few percent of each other. The former has lower memory requirements (see the end of §5.2 for more details). The convergence of the inner iteration is based on the norm of the left preconditioned linear residual using an absolute tolerance equal to one-tenth of the Newton tolerance. Such termination criterion brings enough information on the update vector ΔU^n back to the Newton iteration (see [16] for more details).

Due to the nonlinearity of the original problem, a pseudo transient process is used to produce a parabolic in time problem and bring the starting estimate into the convergence domain of the steady Newton method. The original nonlinear elliptic problem is cast into a parabolic form by appending a pseudo transient term $\frac{\partial U}{\partial t}$ to the original set of algebraic equations $F(U) = 0$, and a fully implicit scheme solves (again with Newton method)

$$\mathcal{F}(U^{n+1}) = F(U^{n+1}) + \frac{U^{n+1} - U^n}{\Delta t^{n+1}} = 0, \tag{7}$$

where Δt^{n+1} is the $(n+1)^{\text{st}}$ time step. The number of time steps needed to bring the initial guessed solution into the convergence domain of the steady Newton iteration depends on the size of the grid, and the coarser the grid, the fewer relaxation steps are necessary. This point will be further discussed in §5.2.

4. MULTIGRID TECHNIQUES

The multigrid philosophy applied to our model problem is derived from [5], [7], and [8]. We assume that there is a sequence of spaces $\mathcal{M}_i$, $i = 1, ..., k$, where the $\mathcal{M}_i$ approximate $\mathcal{M}_1$. We further suppose there exist *restriction* and *prolongation* mappings

$$\begin{cases} \mathcal{R}_i : \ \mathcal{M}_i \to \mathcal{M}_{i+1}, & 1 \le i \le k-1, \\ \mathcal{P}_i : \ \mathcal{M}_i \to \mathcal{M}_{i-1}, & 2 \le i \le k. \end{cases}$$

between neighboring spaces. We also assume there is a sequence of problems (5) represented by J_i.

A multilevel correction algorithm, where the finest level is level 1 and the coarsest level is level k, is simply defined by

```
Algorithm MGC ( $lev$, $\{J_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=2}^{k}$, $\{\mathcal{R}_j\}_{j=1}^{k-1}$ )
1.  $x_{lev} \leftarrow Solver_{lev}(J_{lev}, x_{lev}, b_{lev})$
2.  If $lev < k$, then repeat 2a–2d until some condition is met:
    2a. $x_{lev+1} \leftarrow 0$, $b_{lev+1} \leftarrow \mathcal{R}_{lev}(b_{lev} - J_{lev}x_{lev})$
    2b. MGC ( $lev+1$, $\{J_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=2}^{k}$, $\{\mathcal{R}_j\}_{j=1}^{k-1}$ )
    2c. $x_{lev} \leftarrow x_{lev} + \mathcal{P}_{lev+1}x_{lev+1}$
    2d. $x_{lev} \leftarrow Solver_{lev}(J_{lev}, x_{lev}, b_{lev})$
```

In our case, the solver on every level is either Bi-CGSTAB/GS or GMRES/GS. In Step 1 on level k, our stopping criterion was that the linear residual was adequately reduced (see §3). On the other levels, the stopping criteria was either an upper limit on the number of iterations or that the linear residual was adequately reduced.

A common condition in step 2 is to do steps 2a–2d some specified number of times (e.g., 0 for one way multigrid, 1 for a V Cycle, or 2 for a W Cycle). In §5.2, a V Cycle took less overall time than any other choice for a condition in step 2. However, many V Cycles were necessary, starting from the finest level (see the definition of Algorithm NIC below).

Brandt's FAS algorithm [6] is a nonlinear variant of Algorithm MGC. A nonlinear smoother is used in steps 1 and 2d, the actual solution is computed on every level, and corrections are computed before interpolation in step 2c (see [23] for more details).

We use a nested iteration multilevel algorithm since we do not have an adequate initial guess to the solution initially.

```
Algorithm NIC ( $lev$, $\{J_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=2}^{k}$, $\{\mathcal{R}_j\}_{j=1}^{k-1}$ )
1.  MGC ( $k$, $\{J_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=2}^{k}$, $\{\mathcal{R}_j\}_{j=1}^{k-1}$ )
2.  Do steps 2a–2b with $lev = k-1, \cdots, 1$:
    2a. $x_{lev} \leftarrow \mathcal{P}_{lev+1}x_{lev+1}$
    2b. MGC ( $lev$, $\{J_j, x_j, b_j\}_{j=1}^{k}$, $\{\mathcal{P}_j\}_{j=2}^{k}$, $\{\mathcal{R}_j\}_{j=1}^{k-1}$ )
```

A *damped Newton multilevel algorithm* is defined by introducing an additional step before each reference to Algorithm MGC in just Algorithm NIC. Before each reference to Algorithm MGC, a Jacobian is formed and a damped Newton step is performed. The last Jacobian on a level is saved for use in multilevel correction steps. A *one way multilevel algorithm* means that Algorithm MGC never performs any portion of its step 2 as part of its use by Algorithm NIC. We always use a damped Newton iteration, but we drop the term damped Newton when referring to one way multilevel methods.

The difference between FAS and damped Newton multilevel methods is easy to categorize. FAS uses a nonlinear iterative method (e.g., nonlinear Gauss-Seidel) while damped Newton uses standard linear solvers. When evaluating the nonlinear function is inexpensive, FAS usually produces an approximate solution faster than the damped Newton multilevel method. However, when the function evaluations are expensive, the damped Newton multilevel method usually produces an approximate solution faster than FAS. In a typical diffusion flame problem with finite rate chemistry [9], the function evaluations are horrendously expensive, so we did not explore FAS. For a flame sheet problem solved using FAS, see [24].

5. NUMERICAL RESULTS

In this section, we present several numerical results obtained on an IBM RISC System/6000 (model 560). In §5.1, we focus on unigrid calculations and emphasize the importance of the scale factors α_l in (6) in order to appropriately monitor the convergence of the outer damped Newton iteration. Our numerical experiments show that the overall execution times can be decreased by up to an order of magnitude by taking a large scale factor for all of the vorticity corrections in the computational domain. The execution times can be decreased by an additional factor of six and ten by combining the unigrid numerical procedure with damped Newton multilevel iterations, using either one way or correction schemes, respectively. The corresponding numerical results are presented in §5.2.

5.1. Unigrid tests

In this section, we discuss the influence of the scale factors α_l in (6) on the whole convergence history of the numerical solution. By modifying these scale factors, we shift the balance of work required in the outer Newton iteration and in the inner linear iterations between the different degrees of freedom present in the system. In particular, a large scale factor for the vorticity component asks for less accuracy in the computed vorticity corrections that are brought back to the Newton iteration, thus reducing considerably the amount of work at each Newton step. As indicated in our numerical experiments, this does not yield any loss of accuracy for the other components of the numerical solution (the radial and axial velocity and the conserved scalar). Another important consequence is that much larger time steps can be taken, even at the beginning of the pseudo transient process when the solution is approximated with a very "coarse" initial guess. Furthermore, only a few time steps are required (typically 20) before the numerical solution already lies in the convergence domain of the steady Newton iteration (5). With lower scale factors for the vorticity, most of the CPU time is spent during the pseudo transient iterations, since much smaller time steps need to be taken and the convergence domain of the iteration scheme (5) becomes much narrower. Our numerical experiments indicate that a scale factor for the vorticity of 10^3 can yield savings in CPU time of up to an order of magnitude without altering the velocity and temperature profiles of the numerical solution.

5.2. Multigrid acceleration

In this section, we present further improvements in the total execution times obtained by combining the numerical procedure described in §3 and §5.1 with damped Newton multilevel iterations, using either one way or correction schemes. In all of the results, the speedups represent ratios of CPU times.

We consider the finest level to be a 129×161 grid and we construct three additional coarser grids by successively discarding every other node from one grid to the coarser one. This yields a coarsest grid of 17×21 points. It is worthwhile to note that the use of even coarser grids in these problems meets with difficulties since the calculated flame speeds become excessively large due to the influence of numerical diffusion and/or conduction (see [25]) and the Newton iteration (5) fails to converge.

In the one way nonlinear multigrid approach, we solve the nonlinear problem $F(U) = 0$ in one cycle, starting at the coarsest level and ending at the finest. Asymptotically, as the mesh spacing approaches zero, the interpolant of the computed solution on one grid lies in the convergence domain of Newton method on the next finer grid [26]. In our numerical calculations, this was found to be the case for all levels considered, when using either cubic or linear interpolation between levels. As a consequence, the pseudo transient process needs only to be performed on the coarsest level, in order to bring the initial guess into the convergence domain of the steady Newton iteration on this level. This procedure is particularly attractive for two reasons:

1. By time stepping on the coarsest level, we reduce considerably the amount of work spent in the pseudo transient phase.

2. On coarser grids, less computer time is needed to solve (5).

The first set of numerical experiments was performed using Bi-CGSTAB/GS as the linear smoother. The numerical results obtained during the pseudo transient phase are presented in Table 2. On our workstation, the time stepping requires 15 seconds on the coarsest level as opposed to over 40 minutes on the finest, thus yielding a speedup of 166. Table 3 breaks down the numerical results for the steady state Newton iterations. Note that the CPU time spent during the pseudo transient process has been included in the computation of the speedups presented in Table 3. A speedup of a factor of four is achieved using the one way nonlinear multigrid on two levels, which is due to the significant decrease of smoothing steps done on the finest level. With three and four levels, we obtained speedups of 5.4 and 5.8, respectively. The four level multigrid improves only marginally the execution times, since it decreases the CPU time spent on the third level, while most of the work is already concentrated in the smoothing iterations on the finest level. Finally, it is interesting to note that linear interpolation between levels yields lower execution times than cubic interpolation when Bi-CGSTAB/GS is used as the linear smoother.

We also implemented the one way nonlinear multigrid algorithm using GMRES/GS as the linear smoother with 25 Krylov vectors. This requires 15 Mb of additional storage for the Krylov space.

Table 2: Numerical results for one way nonlinear multigrid during the pseudo transient phase with Bi-CGSTAB/GS as the linear smoother

Operation	Levels			
	1	2	3	4
BiCGSTAB/GS iterations	634	352	217	160
Speedup in time	1.0	6.6	34.6	166.0

Table 3: Numerical results for one way nonlinear multigrid

Operation	Levels			
	1	2	3	4
smooth(1)	1632	371	384	378
smooth(2)	–	723	390	380
smooth(3)	–	–	326	346
smooth(4)	–	–	–	192
Speedup in time	1.0	4.2	5.4	5.8

Smooth(i) represents the total number of Bi-CGSTAB/GS steps done on level i during the steady state Newton iterations.

We found in our numerical experiments that the use of cubic interpolation between levels yielded lower execution times than linear interpolation and that it was more efficient to adaptively increase the time step slightly faster during the pseudo transient phase with respect to the Bi-CGSTAB/GS calculations. The numerical results are given in Tables 4 and 5. We obtain a speedup of 160 for the pseudo transient phase on four levels. As indicated in Table 5, the total execution times delivered are greater than the ones obtained with Bi-CGSTAB/GS. This latter algorithm seems therefore to be a preferable linear smoother when using one way nonlinear multigrid. Note also that the unigrid calculation fails to converge since GMRES/GS stagnates.

In order to solve the linear systems more efficiently, especially the one on the finest level, we perform damped Newton multilevel iterations, making use of the Jacobians computed on all levels coarser than the current one (see algorithm MGC in §4 for more details). The numerical results

Table 4: Numerical results for one way nonlinear multigrid during the pseudo transient phase with GMRES/GS as the linear smoother

Operation	Levels		
	2	3	4
GMRES/GS iterations	572	367	258
Speedup in time	7.2	34.6	159.6

Table 5: Numerical results for one way nonlinear multigrid

Operation	Levels		
	2	3	4
smooth(1)	530	945	945
smooth(2)	1559	592	590
smooth(3)	–	481	825
smooth(4)	–	–	161
Speedup in time	3.2	4.2	4.2

Smooth(i) represents the total number of GMRES/GS steps done on level i during the steady state Newton iterations. The speedups are with respect to the unigrid solution time in Table 3.

Table 6: Numerical results for damped Newton multilevel iterations

Operation	Levels			
	1	2	3	4
smooth(1)	1632	238	268	243
smooth(2)	–	1096	645	673
smooth(3)	–	–	861	1243
smooth(4)	–	–	–	799
Speedup in time	1.0	4.8	6.2	6.6

Smooth(i) represents the total number of Bi-CGSTAB/GS steps done on level i during the steady state Newton iterations.

presented in Table 6 are obtained using 30 steps of Bi-CGSTAB/GS as the linear smoother, which may seem at first glance to be an excessive number of iterations. We obtain a speedup of 6.6 when using four levels. A comparison of Tables 3 and 6 shows that the balance of smoothing iterations is shifted towards the coarsest levels when using damped Newton multilevel iterations, thus yielding lower execution times (approximately 12%) than the ones obtained with the one way nonlinear multigrid. However, it is worthwhile to point out that this improvement comes at the expense of storage since the one way nonlinear multigrid requires 39 Mb and the damped Newton multilevel iterations require up to 62 Mb. This difference is due mainly to the fact that damped Newton multilevel correction methods require saving a Jacobian on every level instead of just one.

Finally, we also performed damped Newton multilevel iterations using GMRES/GS as the linear smoother. In our numerical experiments, we found that the choice of 25 Krylov vectors delivered lower execution times than 20 or 30. We also used cubic and linear interpolation in algorithm NIC and MGC, respectively (see §4). The numerical results are presented in Table 7. We obtain a speedup of a factor of 10.5 when using four levels, thus significantly improving the maximum speedup obtained with Bi-CGSTAB/GS. Using damped Newton multilevel iterations and GMRES/GS as the linear smoother, the whole numerical solution for the flame sheet problem on a 129×161 grid is obtained in about 9 minutes on our workstation. On a supercomputer, the CPU

Table 7: Numerical results for damped Newton multilevel iterations

Operation	Levels		
	2	3	4
smooth(1)	218	216	219
smooth(2)	2272	565	585
smooth(3)	–	1179	1159
smooth(4)	–	–	1020
Speedup in time	5.1	9.9	10.5

Smooth(i) represents the total number of GMRES/GS steps done on level i during the steady state Newton iterations. The speedups are with respect to the unigrid solution time in Table 3.

times will drop dramatically.

6. CONCLUSIONS

In this paper, we presented a new numerical procedure to solve flame sheet problems. The governing equations use the vorticity-velocity formulation of the Navier-Stokes equations coupled together with a conserved scalar equation. By appropriately monitoring the norm of the correction vector in the damped Newton iteration, significant savings in the overall execution time can be obtained. These performances can be further improved by combining the above numerical procedure with one way nonlinear multigrid and damped Newton multilevel iterations. The latter approach yields lower execution times than the former but at a higher cost in storage. With four levels of grids, a speedup of 5.8 is obtained with a one way nonlinear multigrid and Bi-CGSTAB/GS as the linear smoother. Similarly, damped Newton multilevel iterations and GMRES/GS as the linear smoother obtain a speedup of more than a factor of 10. For three dimensional problems, we should obtain speedups much greater than 10.

REFERENCES

[1] M. D. Smooke and V. Giovangigli. Numerical modeling of axisymmetric laminar diffusion flames. *IMPACT Comput. Sci. Engng.*, 4:46–79, 1992.

[2] M. D. Smooke, R. E. Mitchell, and D. E. Keyes. Numerical solution of two-dimensional axisymmetric laminar diffusion flames. *Combust. Sci. and Tech.*, 67:85–122, 1989.

[3] Y. Xu and M. D. Smooke. Application of a primitive variable Newton's method for the calculation of an axisymmetric laminar diffusion flame. *J. Comput. Phys.*, 104:99–109, 1993.

[4] Y. Xu and M. D. Smooke. Primitive variable modeling of multidimensional laminar flames. *Combust. Sci. and Tech.*, 88:1–25, 1993.

[5] R. E. Bank and D. J. Rose. Analysis of a multilevel iterative method for nonlinear finite element equations. *Math. Comp.*, 39:453–465, 1982.

[6] A. Brandt. Multi–level adaptive solution to boundary–value problems. *Math. Comp.*, 31:333–390, 1977.

[7] C. C. Douglas. Multi–grid algorithms with applications to elliptic boundary–value problems. *SIAM J. Numer. Anal.*, 21:236–254, 1984.

[8] C. C. Douglas and J. Douglas. A unified convergence theory for abstract multigrid or multilevel algorithms, serial and parallel. *SIAM J. Numer. Anal.*, 30:136–158, 1993.

[9] A. Ern, C. C. Douglas, and M. D. Smooke. Numerical simulation of laminar diffusion flames with multigrid methods. In preparation.

[10] D. E. Keyes and M. D. Smooke. Flame sheet starting estimates for counterflow diffusion flame problems. *J. Comput. Phys.*, 73:267–288, 1987.

[11] S. W. Armfield. Finite difference solutions of the Navier-Stokes equations on staggered and non-staggered grids. *Computer Fluids*, 20:1–17, 1991.

[12] F. Sotiropoulos and S. Abdallah. A primitive variable method for the solution of three-dimensional incompressible viscous flows. *J. Comput. Phys.*, 103:336–349, 1992.

[13] Wei Shyy and Chia-Sheng Sun. Development of a pressure-correction/staggered-grid based multigrid solver for incompressible recirculating flows. *Computer Fluids*, 22:51–76, 1993.

[14] Z. Zhu and C. A. J. Fletcher. A study of sequential solutions for the reduced/complete Navier-Stokes equations with multigrid acceleration. *Computer Fluids*, 19:43–60, 1991.

[15] T. B. Gatski. Review of incompressible fluid flow computations using the vorticity-velocity formulation. *Appl. Numer. Math.*, 7:227–239, 1991.

[16] A. Ern, V. Giovangigli, D. E. Keyes, and M. D. Smooke. Towards polyalgorithmic linear system solvers for nonlinear elliptic problems. *SIAM J. Sci. Comput.*, 15:to appear, 1994. Also available as Yale University Department of Mechanical Engineering Research Report ME–101–93, New Haven, CT, March, 1993.

[17] A. Ern and M. D. Smooke. Vorticity-velocity formulation for three-dimensional steady compressible flows. *J. Comput. Phys.*, 105:58–71, 1993.

[18] M. Napolitano and L. A. Catalano. A multigrid solver for the vorticity-velocity Navier-Stokes equations. *Int. J. Numer. Methods Fluids*, 13:49–59, 1991.

[19] Y. Xu. *Numerical calculations of an axisymmetric laminar diffusion flame with detailed and reduced reaction mechanisms.* PhD thesis, Yale University, December 1991. Mechanical Engineering Department.

[20] G. A. Sod. *Numerical methods in fluid dynamics.* Cambridge Univ. Press, London, 1985.

[21] H. A. Van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

[22] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[23] C. C. Douglas. Implementing abstract multigrid or multilevel methods. Technical Report YALEU/DCS/TR-952, Department of Computer Science, Yale University, New Haven, CT, 1993.

[24] C. Liu, Z. Liu, and S. F. McCormick. Multigrid methods for numerical simulation of laminar diffusion flames. *AIAA*, 93-0236:1–11, 1993.

[25] M. D. Smooke, J. A. Miller, and R. J. Kee. On the use of adaptive grids in numerically calculating adiabatic flame speeds. In N. Peters and J. Warnatz, editors, *Numerical Methods in Laminar Flame Propagation*, pages 65–70. Friedr. Vieweg & Sohn, Braunschweig, 1982.

[26] M. D. Smooke and R. M. M. Mattheij. On the solution of nonlinear two-point boundary value problems on successively refined grids. *Applied Numer. Math.*, 1:463–487, 1985.

S11-901

197143

N94-23684

A MIXED METHOD POISSON SOLVER FOR THREE-DIMENSIONAL SELF-GRAVITATING ASTROPHYSICAL FLUID DYNAMICAL SYSTEMS

Comer Duncan
Department of Physics and Astronomy
Bowling Green State University
Bowling Green, OH 43403
and
Jim Jones
Computational Mathematics Group
Department of Mathematics
University of Colorado at Denver
Denver, CO 80217

SUMMARY

A key ingredient in the simulation of self-gravitating astrophysical fluid dynamical systems is the gravitational potential and its gradient. This paper focuses on the development of a mixed method multigrid solver of the Poisson equation formulated so that both the potential and the Cartesian components of its gradient are self-consistently and accurately generated. The method achieves this goal by formulating the problem as a system of four equations for the gravitational potential and the three Cartesian components of the gradient and solves them using a distributed relaxation technique combined with conventional full multigrid V-cyles. The method is described, some tests are presented, and the accuracy of the method is assessed. We also describe how the method has been incorporated into our three-dimensional hydrodynamics code and give an example of an application to the collision of two stars. We end with some remarks about the future developments of the method and some of the applications in which it will be used in astrophysics.

1. Introduction

In recent years a number of astrophysicists [1]-[7] have developed simulation tools which build in increasingly realistic physics. The present work grew out of an ongoing effort by us to incorporate enough physics and to realize that physics with robust algorithms so that we can simulate both existing observed phenomena and make reliable predictions which the astronomers can utilize in making better observations and interpreting those observations. The ubiquitous existence of fluids and gravitation in the universe demands that, if we are to have even the most rudimentary simulation code, it must incorporate at least interacting fluids and gravitational physics. In this work, we restrict our attention to the weak-field, Newtonian limit of gravitation. The hydrodynamics code we have created also builds in the effects due to the Special Theory of Relativity, so the description of high speed phenomena is included. The restriction to weak-field gravity implies that the gravitational field is determined by the gravitational potential, which must be a solution to Poisson's equation in three dimensions subject to Dirichlet boundary conditions at the edges of the computational volume. In the coupled hydrodynamic-gravitational system, not only the potential but also its gradient is needed. The gradient contributes to the fluid's acceleration due to its self-gravity, inducing the momentum components to change.

The traditional procedure is to determine the potential by solving the Poisson equation with given Dirichlet boundary condition, then construct approximations to the components of the gradient via finite differencing the potential. However, in simulations of astrophysical gravitating fluids, the development of quite complex flows must be anticipated. Examples from astrophysics include supernova explosions, gravitational collapse, propagation of high-speed jets from active galactic nuclei, star collisions and disruptions in dense star clusters, and realistic models of the early universe. For most of these simulations, we need to compute the gradients of the gravitational potential as accurately as possible, which has motivated our development of an alternate approach to the gradient computation. Here we describe a method which can yield more robust gradients in systems that exhibit large variability in space. This is done using a distributed relaxation procedure coupled with full multigrid V-cycles and is described in Section 2. In Section 3 we present some tests of the method on three-dimensional systems. Section 4 presents our incorporation of it into the three-dimensional relativistic hydrodynamics code. Finally, we briefly describe an application of the code to the collision of two stars and comment on the applications for which the code can be used.

2. The Mixed Method Algorithm

The problems we are interested in are three-dimensional, and the results that we present in later sections are for such problems. However, in presenting the method, we will consider its two-dimensional version to make the description easier to understand and visualize. All components of the method, of both the discretization process and the multigrid algorithm, have natural three-dimensional analogs.

a. The Finite Volume Element Discretization Consider the following partial differential equation defined on some square domain Ω in $\mathcal{R}^2$:

$$\begin{cases} -\ \nabla \cdot \nabla\phi &= f \quad \text{in} \quad \Omega, \\ \phi &= g \quad \text{on} \quad \partial\Omega. \end{cases} \tag{1}$$

We let u and v denote the components of the gradient of -ϕ:

$$(u, v)^t = -\nabla\phi$$

Then the partial differential equation may be written in the form of a first-order system in Ω

$$\begin{cases} u &+ \quad \phi_x &= 0 \quad \text{(u equation)} \\ v &+ \quad \phi_y &= 0 \quad \text{(v equation)} \\ u_x &+ \quad v_y &= f \quad \text{(p equation),} \end{cases} \tag{2}$$

with boundary condition

$$\phi = g \text{ on } \partial\Omega.$$

Here the labels u,v, and p for the equations are introduced simply for convenience. To discretize this system, we follow the Finite Volume Element principles developed in [8]. Consider a uniform square mesh Ω^h with mesh size h that covers Ω. We introduce three sets of control volumes, one for each of the three equations in Eq.**2**. These volumes are shown in Fig. **1**. We denote by $\mathcal{U}$ the set of all volumes $\mathbf{U}$ that will be used to discretize the u equation in Eq.**2**. Similarly, we will use the notation $\mathcal{V}$ and $\mathcal{P}$ for the sets of volumes $\mathbf{V}$ and $\mathbf{P}$ for the v and p equations respectively. For our finite element space we consider the lowest order Raviart-Thomas elements on the triangulation given by the volumes $\mathcal{P}$:

u^h is linear in x and constant in y on each $\mathbf{P} \in \mathcal{P}$,
v^h is linear in y and constant in x on each $\mathbf{P} \in \mathcal{P}$,
ϕ^h is constant on each $\mathbf{P} \in \mathcal{P}$.

The location of the nodes for each of the unknowns with their indexing is also shown in Fig. **1**. We can now disretize the equations. We take the u equation in Eq.**2** and integrate it over each $\mathbf{U} \in \mathcal{U}$. As an example, let $\mathbf{U}_{i,j}$ be the volume in $\mathcal{U}$ that is centered at the interior u^h node (i,j). We then have

$$\int_{\mathrm{U}_{i,j}} (u + \phi_x)\, dxdy \qquad = 0,$$

which implies

$$\tfrac{h^2}{8}(u^h_{i-1,j} + 6u^h_{i,j} + u^h_{i+1,j}) + h(\phi^h_{i+1,j+1} - \phi^h_{i,j+1}) \quad = 0.$$

Integrating the v equation in Eq.**2** over an interior **V** volume yields a similar discrete expression involving nodal values of v^h and ϕ^h. Integrating the p equation in Eq.**2** over the volume in $\mathcal{P}$ centered at the interior ϕ^h node (i,j); denote this volume by $\mathbf{P}_{i,j}$; we get

$$\int_{\mathrm{P}_{i,j}} (u_x + v_y)\, dxdy \qquad = \int_{\mathrm{P}_{i,j}} f dxdy$$

which implies

$$h(u^h_{i,j-1} - u^h_{i-1,j-1} + v^h_{i-1,j} - v^h_{i-1,j-1}) \quad = h^2 f_{i,j}.$$

Here, $f_{i,j}$ is the value of f at the ϕ node (i,j), which results from assuming that f is (approximated by) a piecewise constant function on $\mathcal{P}$. The only remaining part of the discretization involves integrating the u equation in Eq.**2** over the "half size" **U** volumes on the left and right boundaries, and similarly integrating the v equation in Eq.**2** over the "half size" **V** volumes on the lower and upper boundaries. We illustrate this process by integrating the u equation in Eq.**2** over the volume $\mathbf{U}_{1,j}$ that has the boundary u^h node $(1,j)$ as the midpoint of its left edge. We have

$$\int_{\mathrm{U}_{1,j}} (u + \phi_x)\, dxdy \qquad = 0,$$

which implies

$$\tfrac{h^2}{8}(3u^h_{1,j} + u^h_{2,j}) + h(\phi^h_{2,j+1} - \phi^h_{1,j+1}) \quad = 0$$

or

$$\tfrac{h^2}{8}(3u^h_{1,j} + u^h_{2,j}) + h(\phi^h_{2,j+1}) \qquad = h\phi^h_{1,j+1}.$$

Note that $\phi^h_{1,j+1}$ is on the boundary and hence is known. To summarize, the discretization has produced for each **U** volume a discrete version of the u equation in Eq.**2**, for each **V** volume a discrete version of the v equation in Eq.**2**, and for each **P** volume a discrete version of the p equation in Eq.**2**.

b. The Multigrid Algorithm We assume that the reader is familiar with the fundamentals of multigrid methods; good references are [9],[8], [10]. We consider a family of uniform square grids Ω^h that cover our region Ω, where h denotes the mesh size. Fig. **2** shows a coarse grid Ω^{2h}, with twice the mesh size of the grid Ω^h in Fig. **1**. On each grid Ω^h, we can apply the Finite Volume Element discretization process, and we write the discrete set of equations that this process generates as

$$L^h z^h = F^h, \tag{3}$$

where $z^h = (u^h, v^h, \phi^h)^t$ and $F^h = (fu^h, fv^h, f^h)^t$ and the unknowns, u^h, v^h, and ϕ^h, are the nodal values of the corresponding functions on the grid Ω^h. Note that the values of ϕ at nodes on the boundary are known so they are not included in ϕ^h; however, as mentioned in the last section, these boundary values of ϕ do appear in the equations generated by integration over the **U** and **V** volumes near boundaries, resulting in the possibly nonzero terms fu^h and fv^h in Eq.**3**. In this section, we now define the basic components of relaxation, interpolation, and restriction that are necessary to implement a multigrid algorithm.

For the equations on a grid Ω^h, we use a distributive relaxation process similar to that presented in [10]. We can think of relaxation as a three step process. First, we sweep over all of the u^h nodes, change the value of $u^h_{i,j}$ so that the U equation at (i,j) is satisfied. Second, we perform a similar Gauss-Seidel relaxation of all of the V equations. Note that these two steps, the U and V relaxation, are independent of each other and could be performed in parallel. Finally, we step over the ϕ^h nodes and change the value of $\phi^h_{i,j}$ and the values of u^h and v^h that lie on the edge of the volume $\mathbf{P}_{i,j}$, namely $u^h_{i,j-1}$, $u^h_{i-1,j-1}$,$v^h_{i-1,j}$, and $v^h_{i-1,j-1}$. We change these five unknowns so that the P equation at (i,j) is satisfied and so that the residuals of the U equations at $(i,j-1)$ and $(i-1,j-1)$ and of the V equations at $(i-1,j)$ and $(i-1,j-1)$ are unchanged. To allow vectorization, the Gauss-Seidel relaxation performed in each step is done in a red/black ordering.

For defining interpolation operators, we use the same principles as outlined in [8]. The Finte Volume Element discretization is based on finite element spaces for the variables u^h,v^h, and ϕ^h, so we can simply use the relationship between the finite element spaces on the different grids to define interpolation. To define the interpolation operator for ϕ, which we denote as $I(\phi)^h_{2h}$, we note that ϕ^{2h} is constant on the grid $2h$ volume $\mathbf{P}_{I,J}$.

Referring to Figs. **1** and **2**, we thus have the following characterization of $\phi^h = I(\phi)_{2h}^h \phi^{2h}$:

$$\phi_{i,j}^h = \phi_{i+1,j}^h = \phi_{i,j+1}^h = \phi_{i+1,j+1}^h = \phi_{I,J}^{2h}.$$

To define the interpolation operator for u, which we denote as $I(u)_{2h}^h$, we note that u^{2h} is linear in x and constant in y on the grid $2h$ volume $\mathbf{P}_{I,J}$. We thus have the following characterization of $u^h = I(u)_{2h}^h u^{2h}$. (See Figs. **1** and **2**)

$$\begin{aligned} u_{i-1,j-1}^h &= u_{i-1,j}^h = u_{I-1,J-1}^{2h} \\ u_{i+1,j-1}^h &= u_{i+1,j}^h = u_{I,J-1}^{2h} \\ u_{i,j-1}^h &= u_{i,j}^h = 1/2(u_{I-1,J-1}^{2h} + u_{I,J-1}^{2h}). \end{aligned}$$

The definition of the interpolation operator for v is similar.

For defining restriction operators, we again use the same principles as outlined in [8]. In the correction scheme multigrid algorithm, which we use here, restriction operators are used to transfer right-hand sides and residuals of equations, not the unknowns themselves. The definitions of the restriction operators are based on the relationship between the volumes on the various grids. The idea is to lump several of the grid h right-hand sides to produce the grid $2h$ right hand sides. To define the restriction operator for the P equation, which we denote as $I(P)_h^{2h}$, we note that a grid $2h$ volume $\mathbf{P}_{I,J}$ wholly contains four grid h **P** volumes. We thus have the following characterization of $f^{2h} = I(P)_h^{2h} f^h$, referring again to Figs. **1** and **2**:

$$f_{I,J}^{2h} = f_{i,j}^h + f_{i+1,j}^h + f_{i,j+1}^h + f_{i+1,j+1}^h.$$

To define the restriction operator for the U equation, which we denote as $I(U)_h^{2h}$, we note that a grid $2h$ volume $\mathbf{U}_{I,J}$ in the interior of Ω wholly contains two grid h **U** volumes and half of four others. We thus have the following characterization of $fu^{2h} = I(U)_h^{2h} fu^h$, again referring to Figs. **1** and **2**:

$$fu_{I,J-1}^{2h} = fu_{i+1,j}^h + fu_{i+1,j-1}^h + 1/2(fu_{i,j}^h + fu_{i,j-1}^h + fu_{i+2,j}^h + fu_{i+2,j-1}^h).$$

The relationship between **U** volumes at boundaries is different; for example, the grid $2h$ **U** volumes on the left boundary of Ω wholly contain two of the grid h **U** volumes and half of two others, yielding Figs. **1** and **2**:

$$fu_{1,J-1}^{2h} = fu_{1,j}^h + fu_{1,j-1}^h + 1/2(fu_{2,j}^h + fu_{2,j-1}^h).$$

The definition of the restriction operator for the V equation is done in a similar fashion.

3. Tests of the Mixed Method Algorithm

A standard approach to Eq.1 is to solve a discrete equation based on cell-centered finite differences for approximating ϕ, then to use simple differencing of this approximation to get the components of its gradient. We performed some numerical tests to investigate what advantage, in terms of accuracy, the mixed method provides over this standard approach. These tests were for problems with exact solution $\phi(x, y, z) = \sin(k_1 x)\sin(k_2 y)\sin(k_3 z)$ with $\Omega = [0, \pi]^3$. By varying k_1,k_2, and k_3, we were able to see the effect that oscillations in the solution had on the accuracy of the methods. Below are results for some of these tests on a grid with 32 cells in each direction.

			MIXED METHOD		STANDARD METHOD	
k_1	k_2	k_3	ϕ_{err}	$(\phi_x)_{err}$	ϕ_{err}	$(\phi_x)_{err}$
1	1	1	$7.90E-4$	$8.15E-4$	$1.58E-3$	$8.15E-4$
1	16	16	$1.47E-1$	$1.50E-1$	$4.59E-1$	$4.72E-1$
16	1	1	$1.46E-1$	$3.61E-0$	$4.56E-1$	$3.53E-0$
16	16	16	$1.47E-1$	$3.60E-0$	$4.60E-1$	$3.60E-0$

Here, ϕ_{err} and $(\phi_x)_{err}$ are pointwise l_2 norms of the error in ϕ and its x derivative scaled by the volume term h^3. These results are indicative of results seen for other combinations of k_1,k_2, and k_3. For smooth solutions, the methods give nearly identical results. However, for oscillatory solutions, the mixed method gives more accurate results, particularly for ϕ.

4. Incorporation of the Mixed Method Solver into the Three-Dimensional Hydrodynamics Code

a.The Physics and the Code The physics included in the present code consists of a perfect fluid with an adiabatic equation of state formulated in a generally covariant manner. The interval between events in spacetime is represented in the present work in the form

$$ds^2 = -(\alpha^2 - \beta_i\beta^i)dt^2 + \gamma_{ij}(dx^i + \beta^i dt)(dx^j + \beta^j dt). \qquad (4)$$

The function α is called the lapse and represents the lapse of proper time at a given spatial point. The vector field β^i is called the shift vector and determines how much the spatial coordinates shift from one $t = constant$ slice to the next infinitesimally later one. The second rank symmetric tensor field γ_{ij} is the metric tensor of the spatial geometry. In the general theory of relativity[12], the four-dimensional geometry of spacetime is dynamic and the lapse, shift, and three metric are related to the kinematical description of the coordinates of the observer and the spatial geometry. The fluid energy-momentum tensor must obey a local conservation law in order to be consistent with Einstein's theory. When supplemented with the conservation of Baryons, the conservation laws can be written in the following form:

Rest-mass conservation

$$\frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial t}(\gamma^{\frac{1}{2}}d) + \frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial x^i}(\gamma^{\frac{1}{2}}dv^i) = 0 \tag{5}$$

Internal energy equation

$$\frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial t}(\gamma^{\frac{1}{2}}e) + \frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial x^i}(\gamma^{\frac{1}{2}}ev^i) = -P(\frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial t}(\gamma^{\frac{1}{2}}W) + \frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial x^i}(\gamma^{\frac{1}{2}}Wv^i)) \tag{6}$$

Momentum equation

$$\begin{aligned}\frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial t}(\gamma^{\frac{1}{2}}S_j) + \frac{1}{\alpha\gamma^{\frac{1}{2}}}\frac{\partial}{\partial x^j}(\gamma^{\frac{1}{2}}S_jv^i) &= -\frac{\partial P}{\partial x^j} + (d+e+PW)W\frac{\ln\alpha}{\partial x^j} \\ &- \frac{S_i}{\alpha}\frac{\partial\beta^i}{\partial x^j} - \frac{1}{2W}\frac{S_kS_l}{(d+e+PW)}\frac{\partial\gamma^{kl}}{\partial x^j}\end{aligned} \tag{7}$$

The variables d, e, and S_i, which are used in the code, are defined as follows: $d = \rho W$, $e = \rho\epsilon W$, and $S_i = (\rho + \rho\epsilon + P)u_i$. Here d, e, and S_i are respectively the coordinate mass density, internal energy density, and covariant components of the relativistic momentum density. Eq.(**5-7**) are the equations of general relativistic fluid dynamics in a general background spacetime. Since the present paper is restricted to the study of phenomena with weak gravitational fields, we introduce the following Newtonian approximations to the lapse, shift, and three-metric in Cartesian coordinates:

$$\alpha \simeq 1 + \phi \tag{8}$$

$$\beta^i \simeq 0 \tag{9}$$
$$\gamma_{ij} \simeq \delta_{ij}. \tag{10}$$

The scalar field ϕ is the Newtonian gravitational potential and must satisfy the Poisson equation

$$\nabla^2\phi = 4\pi G\rho, \tag{11}$$

in the computational volume and Dirichlet boundary conditions on the volume edges.

With the Newtonian approximation to the geometric variables it then follows that the self-gravity of the fluid contributes to the change in the momentum density through the term $\rho\nabla\alpha$. The value of α itself enters several places in the fluid equations. Thus, a complete characterization of the self-gravitating fluid dynamics requires both the lapse and its gradient vector. It is these quantities that our mixed method computes in a robust manner. Concerning the elements that constitute the hydrodynamics part of the code, the methods used may be characterized as explicit finite volume schemes. The physical variables d, e, and S_i are the fundamental quantities. These variables are discretized on a staggered grid system with the conventions that scalar variables such as density are stored at zone centers, while vector variables are centered on the faces of the zones. The biggest challenge is by far to treat the advection of the physical variables as accurately as possible. This is especially true for the astrophysical applications, since complex flows abound. We want the code to be able to detect and track shocks adequately. The advection method implemented in the code is based on a monotonic advection algorithm due originally to Van Leer [11]. It is robust and tracks shocks reasonably well. The code uses artificial viscosity to smooth developing discontinuities over a few zones. For this we use an artificial viscosity pressure, which is a combination of linear and quadratic functions of the monotonized four-velocity differences. The code uses an adiabatic equation of state of the form $P = (\Gamma - 1)\rho\epsilon$, where Γ is the parameter that characterizes the equation of state and can itself be a function of the thermodynamic variables and position. For the model stars we discuss here, Γ is chosen to be a constant. The overall structure of a single computational step of the code is described in [7] and illustrated as follows:

$$\begin{tikzcd}
\boxed{\textbf{Initial\_Data}} \arrow[d] \\
\boxed{\textbf{Time\_Step\_Constraint}} \arrow[d] \\
\boxed{\textbf{Acceleration}} \arrow[d] \\
\boxed{\textbf{Artificial\_Viscosity}} \arrow[d] \\
\boxed{\textbf{Velocity}} \arrow[d] \\
\boxed{\textbf{Density\_and\_Energy\_Transport}} \arrow[d] \\
\boxed{\textbf{Momentum\_Transport}} \arrow[d] \\
\boxed{\textbf{Poisson\_Solver}}
\end{tikzcd}$$

At the end of the computational step the fully updated physical variables are available. The **Poisson_Solver** routine is invoked and it is here that we utilize our mixed method solver, which returns ϕ and $\nabla\phi$.

b.Application to Collision of Stars As a nontrivial application of the code, we present a summary of the results of using the mixed method Poisson solver in the simulation of the collision of two stars which are initially in equilibrium. The initial data were chosen so that the mass density and energy density correspond to two equilibrium spherical stars. We have chosen the $n = 1$ polytropic equation of state. This equation of state has the following functional forms for the initial mass density and energy density: $d = d_0 \frac{sin(\xi)}{\xi}$ and $e = e_0 \left(\frac{\sin\xi}{\xi}\right)^2$, where $\xi = \pi r/r_0$ and r_0 is the equilibrium radius of the star. The two model stars were placed with their centers displaced in the $z = 0$ plane. We show here the results of simulations in which the radii were chosen equal to $0.26R_{solar}$ and the central mass density d_0 equal to $6.6g/cm^3$. The central temperature of each star was chosen to be 4.0e06 K. The simulations shown here were all done with a $(66)^3$ grid. All computations were performed on the Ohio Supercomputer Center's Cray YMP8/864. The hydrodynamics part of the code has been highly vectorized.

Fig. **3a** shows the contours for the initial potential and its gradient components in the $z = 0$ plane for a run of an off-center collision. The stars were chosen initially to have a relative velocity comparable to the orbital velocity. Fig. **3b** is a plot of the density contours and velocity field in the z=0 plane. Subsequent motion is induced by the combined effects of the initial momentum and the self-gravity of the two stars. Because the stars attract each other, they develop accelerations toward each other and the hydrodynamics that results alters the density and energy distributions. Typical simulations were run for at least on the order of the gravitational free-fall time. Given the combined interactions of the hydrodynamics with self-gravity, we expect disruption of the two stars if the collision is sufficiently violent. Figs. **4a,b** show respective snapshots of the potential contours and gradient and density contours and velocities for late times in the off-center collision.

We conclude from these simulations and others that the mixed method Poisson solver produces physically acceptable results when combined with the three-dimensional hydrodynamics. This code is currently being used to simulate higher resolution runs and other multiple-star systems. We will be using the present code to treat the collision of two neutron stars and compute its final state and the amount of gravitational radiation emitted by such systems. Such computations are of importance because they can shed light on the astrophysics of the mergers of neutron stars as well as provide potentially important benchmarks of how much gravitational radiation should be expected from such encounters.

Acknowledgements
The authors wish to thank Steve McCormick for many useful conversations about the mixed method algorithm. We thank Jeyakumari Khan for her assistance with the merging of the mixed method Poisson solver and the hydro code. C. Duncan acknowledges support from Cray Research and the Ohio Supercomputer Center, where the computations were performed.

References

[1] R. L. Bowers and J. R. Wilson, *Numerical Modeling in Applied Physics and Astrophysics* , Jones & Bartlett, Boston, 1991.

[2] J. M. Stone and M. L. Norman, Astrophysical Journal Supplement Series **80**, 753 (1992).

[3] A. Abrahams, D. Bernstein, D. W. Hobill, E. Seidel, and L. Smarr, Physical Review **D45**, 3544 (1992).

[4] P. Laguna, H. Kurki-Suonio, and R. A. Matzner, Physical Review **D44**, 3077 (1991).

[5] A. Abrahams and C. R. Evans, Physical Review **D46**, R4117 (1992).

[6] T. Nakamura, *Proceedings of the Sixth Marcel Grossmann Meeting on General Relativity*, 1992.

[7] G. Comer Duncan, Proceedings of the First Midwest Relativity Conference, National Center for Supercomputing Applications, March, 1992; Bull. Am. Phys. Soc. (with G. Shastri), May 1992 OSAPS, Cincinnati.

[8] S. F McCormick, *Multilevel Adaptive Methods for Partial Differential Equations*, Vol. 6 in Frontiers in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, 1989.

[9] William L. Briggs, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[10] A. Brandt, *Multigrid Techniques : 1984 Guide*, The Weizmann Institute of Science, Rehovot, Israel.

[11] B. Van Leer, J. Computational Physics **32**, 101 (1979).

[12] C. Misner, K. Thorne, and J. Wheeler *Gravitation*, Freeman, San Francisco, 1973.

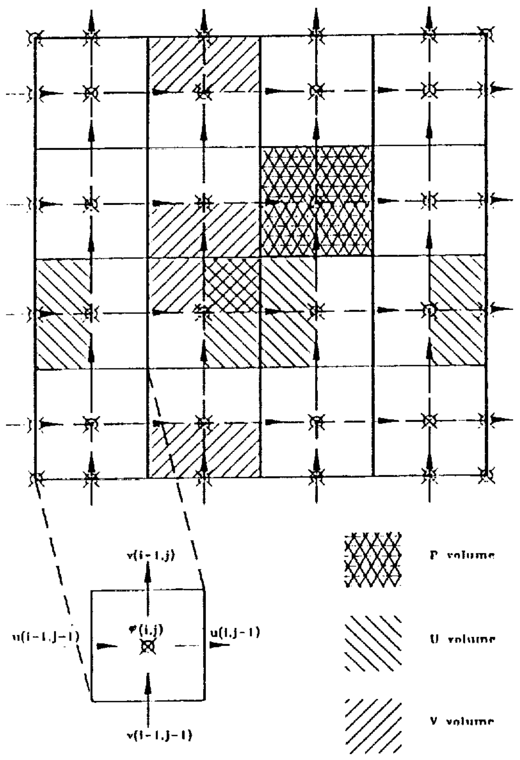


Figure 1

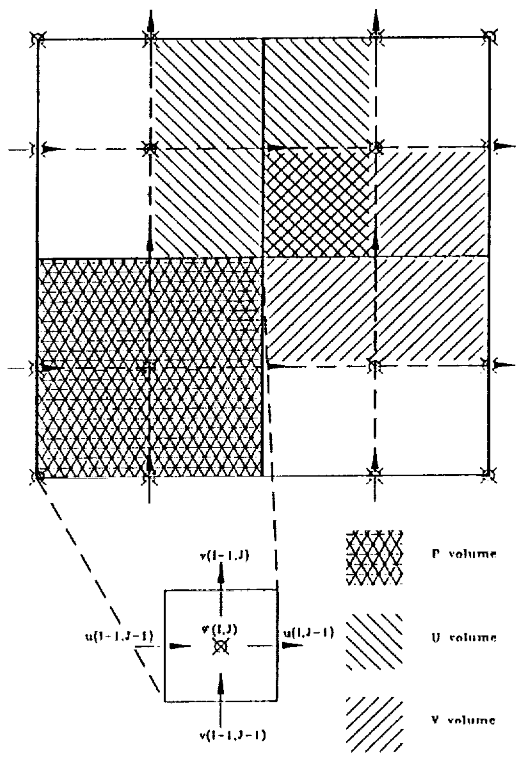


Figure 2

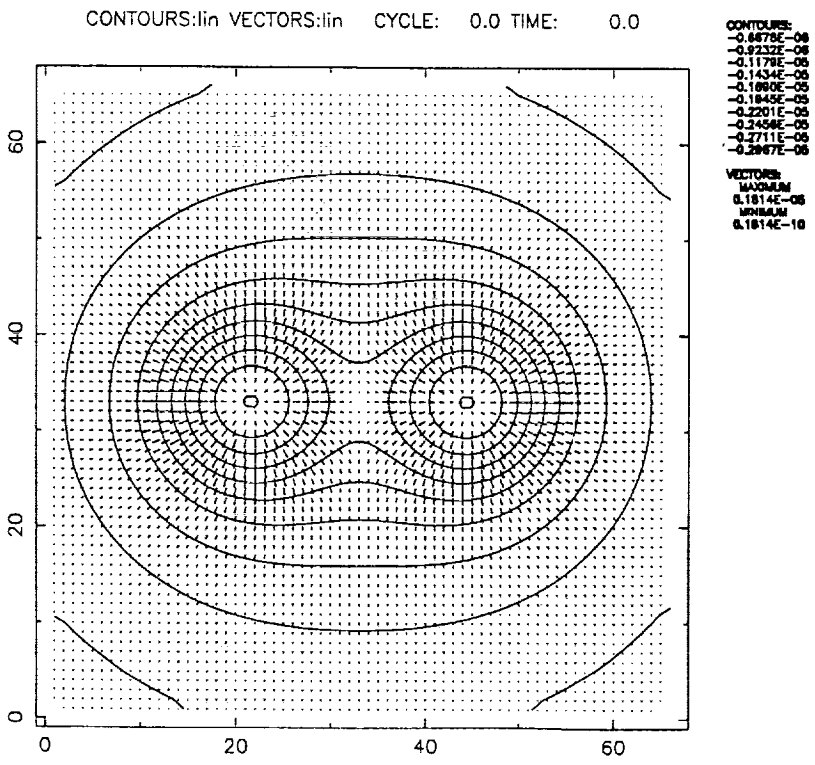


Figure 3a. Initial Gravitational Potential and Gradient

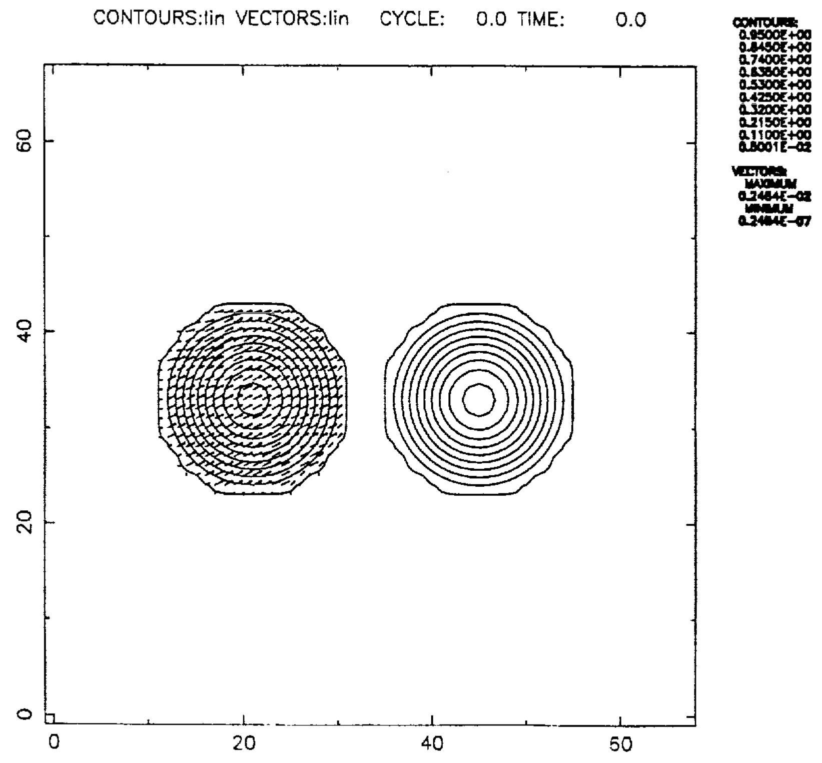


Figure 3b. Initial Density and Velocity

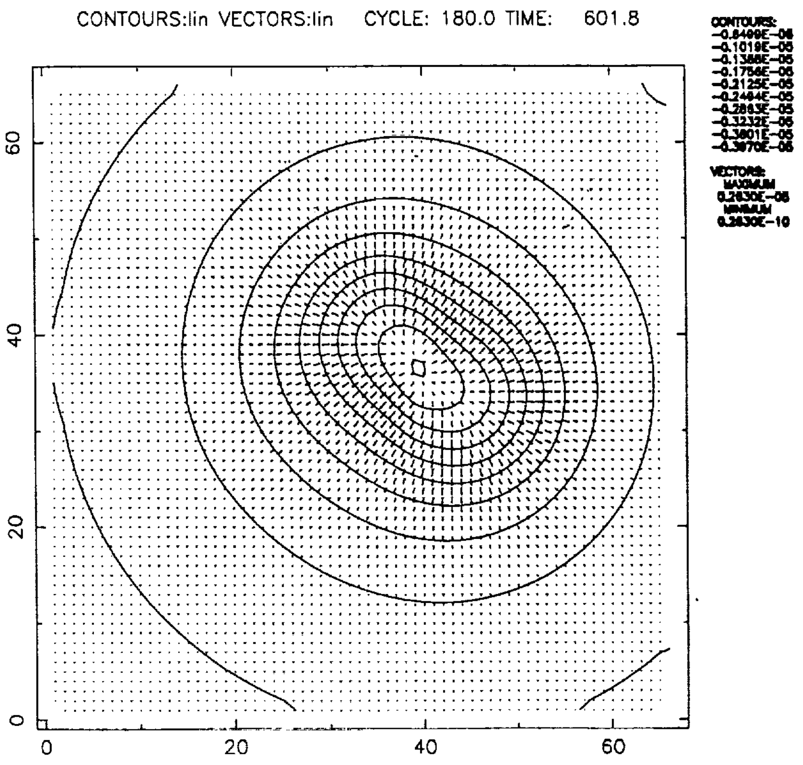


Figure 4a. Gravitational Potential and Gradient

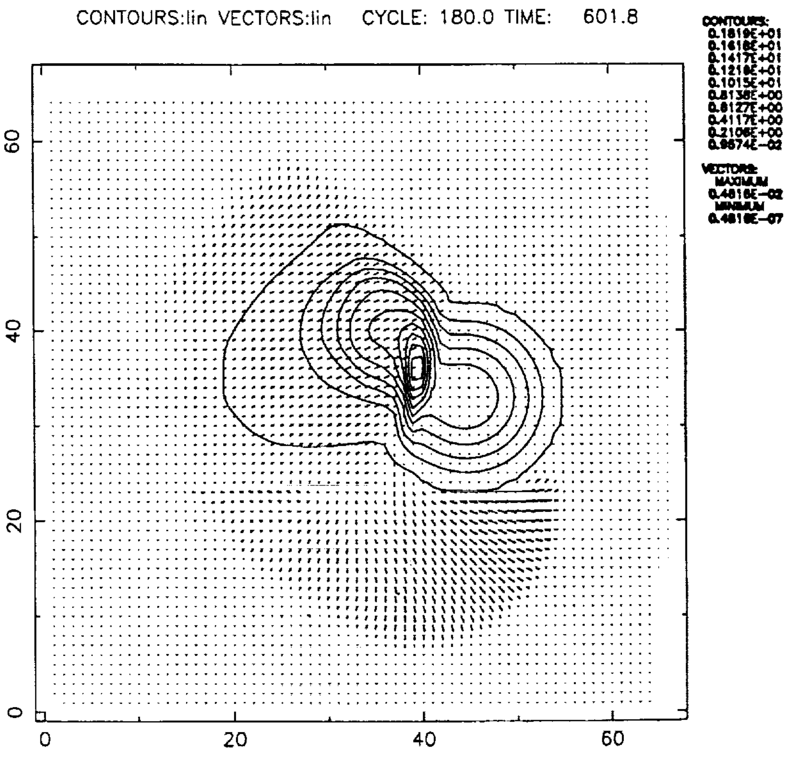


Figure 4b. Density and Velocity

S12-64
197144
p. 15
N94-23685

MULTIGRID METHODS FOR DIFFERENTIAL EQUATIONS WITH HIGHLY OSCILLATORY COEFFICIENTS*

Bjorn Engquist   Erding Luo
Dept. of Math., UCLA, CA 90024

SUMMARY

New coarse grid multigrid operators for problems with highly oscillatory coefficients are developed. These types of operators are necessary when the characters of the differential equations on coarser grids or longer wavelengths are different from that on the fine grid. Elliptic problems for composite materials and different classes of hyperbolic problems are practical examples.

The new coarse grid operators can be constructed directly based on the homogenized differential operators or hierarchally computed from the finest grid. Convergence analysis based on the homogenization theory is given for elliptic problems with periodic coefficients and some hyperbolic problems. These are classes of equations for which there exists a fairly complete theory for the interaction between shorter and longer wavelengths in the problems. Numerical examples are presented.

INTRODUCTION

Multigrid methods are usually not so effective when applied to problems for which the standard coarse grid operators have significantly different properties from those of the fine grid operators [1,3,7-9,11-12]. In some of these problems the coarse grid operators should be constructed based on other principles than just simple restriction from the finest grid. Elliptic and parabolic equations with strongly variable coefficients and some hyperbolic equations are such problems. One feature in these problems is that the smallest eigenvalues

*This work was partially supported by grants from NSF: DMS 91-03104, DARPA: ONR N00014-92-J-1890, and ARO: ARM DAAL03-91-G0162.

do not correspond to very smooth eigenfunctions. It is thus not easy to represent these eigenfunctions of the coarser grids.

We shall investigate elliptic equations with highly oscillatory coefficients,

$$\sum_j \frac{\partial}{\partial x_j} a_j^\epsilon(x) \frac{\partial}{\partial x_j} u_\epsilon(x) = f(x), \quad a_j^\epsilon(x) = a_j(x, \frac{x}{\epsilon}) \tag{1}$$

with $a_j(x,y)$ strictly positive, continuous and 1-periodic in y. This is one class of the problems discussed above for which there exists a fairly complete analytic theory such that a rigorous treatment is possible. This homogenization theory describes the dependence of the large scale features in the solutions on the smaller scales in the coefficients [2,11]. We shall consider model problems but there are also important practical applications of these equations in the study of elasticity and heat conduction for composite materials.

In this paper we analyse the convergence of multigrid methods for equation (1) by introducing new coarse grid operators, based on local or global homogenized forms of the equation. We consider only two level multigrid methods. For full multigrid or with more general coefficients the homogenized operator can be numerically calculated from the finer grids based on local solution of the so called cell problem [2].

In a number of numerical tests we compare the convergence rate for different choices of parameter and coarse grid operators applied to a two dimensional elliptic model problem.

The convergence rate is also analyzed theoretically for a one dimensional problem. If, for example, the oscillatory coefficient is replaced by its average, the direct estimate for multigrid convergence rate is not asymptotically better than just using the damped Jacobi smoothing operator. The homogenized coefficient reduces the number of smoothing operations from $O(h^{-2})$ to $O(h^{-10/7}\log h)$. When h/ϵ belongs to the set of *Diophantine numbers* [4], ergodic mixing improves the estimate to $O(h^{-6/5}\log h)$. The step size is h and the wave length in the oscillating coefficient ϵ.

These results carry over to some but not all hyperbolic problems. A numerical study of using hyperbolic time stepping with multigrid in order to compute a steady state gives similar results to the elliptic case.

TWO DIMENSIONAL ELLIPTIC PROBLEMS

Elliptic problems on the form (1) will be considered,

$$-\nabla \cdot a^\epsilon(x,y)\nabla u_\epsilon = f(x,y), \quad (x,y) \in \Omega = [0,1] \times [0,1], \tag{2}$$

subject to Dirichlet boundary condition $u_\epsilon|_{\partial\Omega} = 0$. The function $a^\epsilon(x,y) = a(x/\epsilon, y/\epsilon)$ is strictly positive and 1-periodic in x and y. From homogenization theory [2] follows,

$$\max_{(x,y)\in\Omega} |u_\epsilon - u| \to 0, \quad as \quad \epsilon \to 0.$$

where u satisfies the following effective equation,

$$-A_{11}\frac{\partial^2 u}{\partial x^2} - (A_{12}+A_{21})\frac{\partial^2 u}{\partial y \partial x} - A_{22}\frac{\partial^2 u}{\partial y^2} = f(x,y), \quad (x,y)\in\Omega, \tag{3}$$

subject to the same boundary condition. Here,

$$A_{ij} = \int a(s_1,s_2)(\delta_{ij} - \frac{\partial \kappa_j}{\partial s_i})ds_1 ds_2, \quad i,j = 1,2,$$

and the periodic functions κ_j are given by,

$$-\nabla_s \cdot a(s_1,s_2)\nabla_s \kappa_j = \frac{\partial a(s_1,s_2)}{\partial s_j}, \quad j = 1,2.$$

For the numerical examples we shall choose a special case with diagonal oscillatory coefficient,

$$a^\epsilon(x,y) = a(\frac{x-y}{\epsilon}). \tag{4}$$

From (3), we know that the corresponding homogenized equation is,

$$-\frac{(\mu+\bar{a})}{2}\frac{\partial^2 u}{\partial^2 x} + (\mu - \bar{a})\frac{\partial^2 u}{\partial x \partial y} - \frac{(\mu+\bar{a})}{2}\frac{\partial^2 u}{\partial^2 y} = f(x,y), \tag{5}$$

where $\mu = m(1/a^\epsilon)^{-1}$ and $\bar{a} = m(a^\epsilon)$. Here, the mean value $m(f)$ of a $\epsilon-$periodic function is defined as,

$$m(f) = \frac{1}{\epsilon}\int_0^\epsilon f(x)dx.$$

For convenience, we introduce a brief notation of a $N \times N$ block tridiagonal matrix T,

$$T = Tridiag[T_{i-1}, T_i, T_{i+1}]_{(N\times N)} = \begin{bmatrix} T_{11} & T_{12} & & \\ T_{21} & T_{22} & T_{23} & \\ & \ddots & \ddots & \ddots & \\ & & & T_{NN-1} & T_{NN} \end{bmatrix}.$$

Numerical Algorithm

The discretization of (2) combined with (4) is

$$-D_{+}^{x}a_{ij}^{h}D_{-}^{x}u_{ij}^{h}-D_{+}^{y}b_{ij}^{h}D_{-}^{y}u_{ij}^{h}=f_{ij}^{h}. \tag{6}$$

where $a_{ij}^{h}=a^{\epsilon}(x_i-\frac{h}{2}-y_j)$, $b_{ij}^{h}=a^{\epsilon}(x_i-y_j+\frac{h}{2})$, $i,j=0,\cdots,N$. D_+ and D_- are forward and backward divided differences, respectively; $h=\frac{1}{N}$ denotes the grid size. Using vector notation, we can rewrite (6) as

$$L_{\epsilon,h}U_{\epsilon,h}=F_{\epsilon,h}$$

where

$$L_{\epsilon,h}=\frac{1}{h^2}Tridiag[B_{j-1}^{h},A_j^h,B_j^h]_{(N-1)\times(N-1)} \tag{7}$$

$$A_j^h=Tridiag[-a_{i-1j}^{h},a_{i-1j}^{h}+a_{ij}^{h}+b_{ij}^{h}+b_{ij-1}^{h},-a_{ij}^{h}]_{(N-1)\times(N-1)}$$

B_j^h is a diagonial matrix, denoted by $B_j^h=Diag[-b_{i,j}^{h}]_{(N-1)\times(N-1)}$ and

$$U_{\epsilon,h}=(u_{11}^h,u_{21}^h,\cdots,u_{N-11}^h,\cdots,u_{1N-1}^h,u_{2N-1}^h,\cdots,u_{N-1N-1}^h)^T$$

$$F_{\epsilon,h}=(f_{11}^h,f_{21}^h,\cdots,f_{N-11}^h,\cdots,f_{1N-1}^h,f_{2N-1}^h,\cdots,f_{N-1N-1}^h)^T$$

For simplicity, we only consider the two-grid method. Denote the full iteration operator of this method by M. It is defined by,

$$M=S^{\gamma}(I-I_H^hL_H^{-1}I_h^HL_{\epsilon,h})S^{\gamma}, \tag{8}$$

where the restriction and interpolation operators are given, as denoted below, by the weighting restriction and bilinear interpolation operators, respectively,

$$I_h^H=\frac{1}{16}\left]\begin{matrix}1&2&1\\2&4&2\\1&2&1\end{matrix}\right[_h^H,\quad I_H^h=\frac{1}{4}\left]\begin{matrix}1&2&1\\2&4&2\\1&2&1\end{matrix}\right[_H^h.$$

The smoothing iteration operator S is based on the damped Jacobi iteration,

$$S=I-\omega h^2L_{\epsilon,h}. \tag{9}$$

The coarse grid operators L_H is one of the following operators:

Global Homogenized operator: which is the discretized form of (5)

$$-0.5(\mu+\bar{a})D_+^x D_-^x u_{ij} + (\mu-\bar{a})D_0^x D_0^y u_{ij} - 0.5(\mu+\bar{a})D_+^y D_-^y u_{ij} = f_{ij}.$$

Written in matrix form,

$$L_H = \frac{1}{H^2} Tridiag[B_{j-1}^H, A_j^H, B_j^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)}, \tag{10}$$

where

$$A_j^H = Tridiag[-\frac{\mu+\bar{a}}{2}, 2(\mu+\bar{a}), -\frac{\mu+\bar{a}}{2}]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)},$$
$$B_j^H = Tridiag[-\frac{\bar{a}-\mu}{4}, -\frac{\mu+\bar{a}}{2}, \frac{\bar{a}-\mu}{4}]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)}.$$

Local Homogenized operator: L_H has the same form as (10), except the entries for A_j^H, B_j^H coming from the local discretized values of $a_\epsilon(x-y)$,

$$L_H = \frac{1}{H^2} Tridiag[B_{j-1}^H, A_j^H, B_j^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)}, \tag{11}$$

where

$$A_j^H = Tridiag[-a_{i-1j}^H, a_{i-1j}^H + a_{ij}^H + b_{ij-1}^H + b_{ij}^H, -a_{ij}^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)},$$
$$B_j^H = Tridiag[-c_{i-1}^H, -b_{ij}^H, c_i^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)},$$

with

$$b_{ij}^H = \frac{b_{ij}^h + b_{ij-1}^h + 2\delta(b_{ij}^h, b_{ij-1}^h)}{4}, \quad a_{ij}^H = \frac{a_{ij}^h + a_{ij-1}^h + 2\delta(a_{ij}^h, a_{i-1j}^h)}{4}, \quad c_i^H = \frac{\mu-\bar{a}}{2}.$$

$\delta(c_1, c_2)$ is defined to be $\frac{2c_1c_2}{c_1+c_2}$.

Reduced Local Homogenized operator: L_H has the same form as in (11), except here we ignore the cross term $D_0^x D_0^y$. That means B_j^H is a diagonal matrix, $B_j^H = Diag[-b_{ij}^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)}$,

$$L_H = \frac{1}{H^2} Tridiag[B_{j-1}^H, A_j^H, B_j^H]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)} \tag{12}$$

Sampling operator: $L_{\epsilon,H}$ has the exact form as $L_{\epsilon,h}$, but values a_{ij}, b_{ij} are defined on the coarse grids,

$$L_H = L_{\epsilon,H} \tag{13}$$

Variational operator:

$$L_H = I_h^H L_{\epsilon,h} I_H^h \tag{14}$$

Numerical Results

In practice, it is not always easy to calculate the spectral radius $\rho(M)$. Therefore, we study the mean rate [14] of convergence under different coarse grid operators L_H. The mean rate of convergence is defined by

$$\rho = \left(\frac{\|L_{\epsilon,h}u^i - f_h\|_h}{\|L_{\epsilon,h}u^1 - f_h\|_h}\right)^{\frac{1}{i-1}} \tag{15}$$

where i is the smallest integer satisfying $\|L_{\epsilon,h}u^i - f_h\|_h \leq 1 \times 10^{-6}$.

In Figure 1, $a^\epsilon(x-y) = 2.1 + 2\sin(2\pi(x-y)/\epsilon)$. We plot ρ defined by (15) as a function of γ by taking $\epsilon = \sqrt{2}h$, and ω in (9) is 0.095.

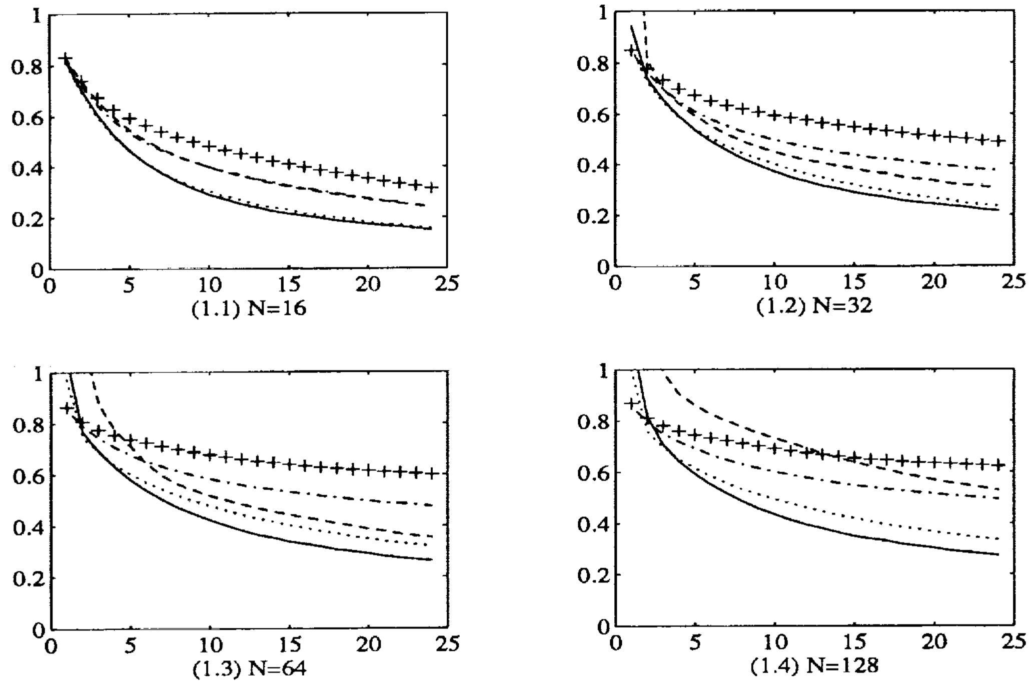


Figure 1: ρ as a function of γ. Dotted line is for (10), solid line for (11), dashed line for (13), and dashdot for (12), + for (14). (1.1)-(1.4) are for different number of grid points N.

It is clear that the coarse grid operators derived from the homogenized forms (10) and (11) are superior. The effect is more pronounced for large γ when the eigenspace corresponding to large eigenvalues of $L_{\epsilon,h}$ is essentially eliminated. For the practical low γ case, a study of the impact of the choices of I_H^h and I_h^H is needed. In this paper we are concentrating on the asymptotic behavior (large γ). Different I_H^h and I_h^H operators are briefly discussed for the one-dimensional problem.

In Figure 2, we plot ρ as a function of the variable α, where $L_H(\alpha)$ comes from the discretized operator $-a_{ij}^H D_+^x D_-^x + \alpha(\mu - \bar{a}) D_0^x D_0^y - b_{ij}^H D_+^y D_-^y$, $\omega = 0.095$ and $\epsilon = \sqrt{2}h$.

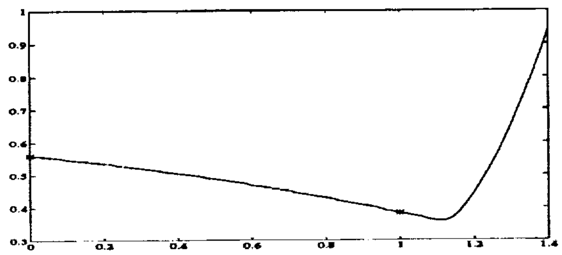


Figure 2: ρ as a function of α. Here $N = 64$ and $\gamma = 12$. "*" denote ρ under the different choice of Normal and Local Homogenized coarse grid operator, respectively.

From Figure 2, we get further evidence of the importance of using the correct homogenized operator. Techniques based on one-dimensional analysis does not contain the mixed derivative term [1].

In order to isolate the influence of the coarse grid approximation we have kept the smoothing operator fixed. It obviously also affects the performance. If we use Gauss Seidel iteration method in (9), the convergence rate can be improved. In Table 1, we test the same coefficient $a^\epsilon(x-y) = 2.1 + 2\sin(2\pi(x-y)/\epsilon)$. Taking $N = 128, \epsilon = \sqrt{2}h$, we compare the convergence rate by choosing damped Jacobi iteration and Gauss Seidel iteration.

γ	5	6	7	8	9	10	11	12
Jacobi	.5929	.5519	.5173	.4863	.4579	.4349	.4140	.3950
G-S	.4545	.4221	.3922	.3703	.3491	.3304	.3158	.3008

Table 1: Spectral radius, two dimensional case

ONE DIMENSIONAL PROBLEMS

The one dimensional equation is useful as a model for which a more complete mathematical analysis is possible.

Consider the following one-dimensional elliptic boundary value problem with a periodic oscillatory coefficient,

$$-\frac{d}{dx}a^{\epsilon}(x)\frac{du_{\epsilon}}{dx}=1, \quad 0<x<1, \quad u_{\epsilon}(0)=u_{\epsilon}(1)=0, \tag{16}$$

where $a^{\epsilon}(x)=a(\frac{x}{\epsilon})$ and satisfies the same assumption as above. As $\epsilon \to 0$, u_{ϵ} converges strongly in L_{∞} to the solution u of the homogenized equation,

$$-\alpha\frac{d^2u}{dx^2}=1, \quad 0<x<1, \quad \alpha=m(1/a^{\epsilon})^{-1}. \tag{17}$$

Subject to the boundary conditions $\phi(0)=\phi(1)=0$.

Numerical Algorithm

Let the difference approximation of (16) be of the form:

$$-a^{\epsilon}(x_{j+\frac{1}{2}})(u^h_{j+1}-u^h_j)+a^{\epsilon}(x_{j-\frac{1}{2}})(u^h_j-u^h_{j-1})=1, \quad j=1,\cdots,N-1 \tag{18}$$

In matrix form, (18) can be written as

$$L_{\epsilon,h}u^h=1, \quad u^h=(u^h_1\cdots,u^h_{N-1})^T$$

where

$$L_{\epsilon,h}=\frac{1}{h^2}Tridiag[-a_{i-1},a_{i-1}+a_i,a_{i+1}]_{(N-1)\times(N-1)} \tag{19}$$

with $a_j=a^{\epsilon}(x_j-\frac{h}{2})$.

We first consider a two-grid method by applying standard restriction, standard interpolation operators and Jacobi smoothing iteration.

The coarse grid operator L_H will be one of the following:

Homogenized operator:

$$L_H=\frac{m(1/a^{\epsilon})^{-1}}{H^2}Tridiag[-1,2,-1]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)} \tag{20}$$

Averaged operator:

$$L_H=\frac{m(a^{\epsilon})}{H^2}Tridiag[-1,2,-1]_{(\frac{N}{2}-1)\times(\frac{N}{2}-1)} \tag{21}$$

Sampling operator: $L_{\epsilon,H}$ has the exact form as $L_{\epsilon,h}$, but only every second a_j value is used,

$$L_H = L_{\epsilon,H} \tag{22}$$

Variational operator:

$$L_H = I_h^H L_{\epsilon,h} I_H^h. \tag{23}$$

Convergence Theory

The theorem below on the convergence rate is too pessimistic in the number of smoothing iterations necessary. However, the analysis still gives insight into the convergence process and the role of homogenization. With L_H replaced by averaging (21) the same analysis results in $\gamma = O(h^{-2})$ which means that multigird does not improve the rate of convergence over just using Jacobi iterations. This follows from the effect of the oscillations on the lower eigenmodes. It should also be noticed that in the case (ii), the solution of $L_{\epsilon,h}$ is much closer to those of L_H, see [11].

Theorem 1 *Let M be defined as in (8), with L_H defined by (11). There exists a constant C such that,*

$$\rho(M) \leq \rho_0 < 1,$$

when either one of the following conditions is satisfied:

(i) $\gamma \geq Ch^{-1-3/7} lnh$

(ii) the ratio of h to ϵ belongs to the set of Diaphantine number, and $\gamma \geq Ch^{-1-1/5} lnh$.

For details of the proof, see [10]. An outline is as follows . Separate the complete eigenspace of $L_{\epsilon,h}$ into two orthogonal subspaces, the space of low eigenmodes and that of high eigenmodes. After several Jacobi smoothing iterations in the fine grid level, the high eigenmodes of the error are reduced, and only the low eigenmodes are left. Combining eigenvalue analysis with homogenization theory [11], one may realize that the low eigenmodes of the original discrete operator are close to those of the corresponding homogenized operator. We then approximate them by the corresponding homogenized eigenmodes and correct these in the coarse grid level.

Numerical Results

In Figure 3.1 and Figure 3.2, $a^\epsilon(x) = 2.1 + 2\sin(2\pi x/\epsilon)$. We plot the analogous graph to Figure 1. Here $\epsilon = \sqrt{2}h$ and ω in (9) is 0.1829. In Figure 3.3 and Figure 3.4, $a^\epsilon(x) = 2.1 + 2\sin(2\pi x/\epsilon + \pi/4)$. Here $\epsilon = 4h$ and $\omega = 0.1585$.

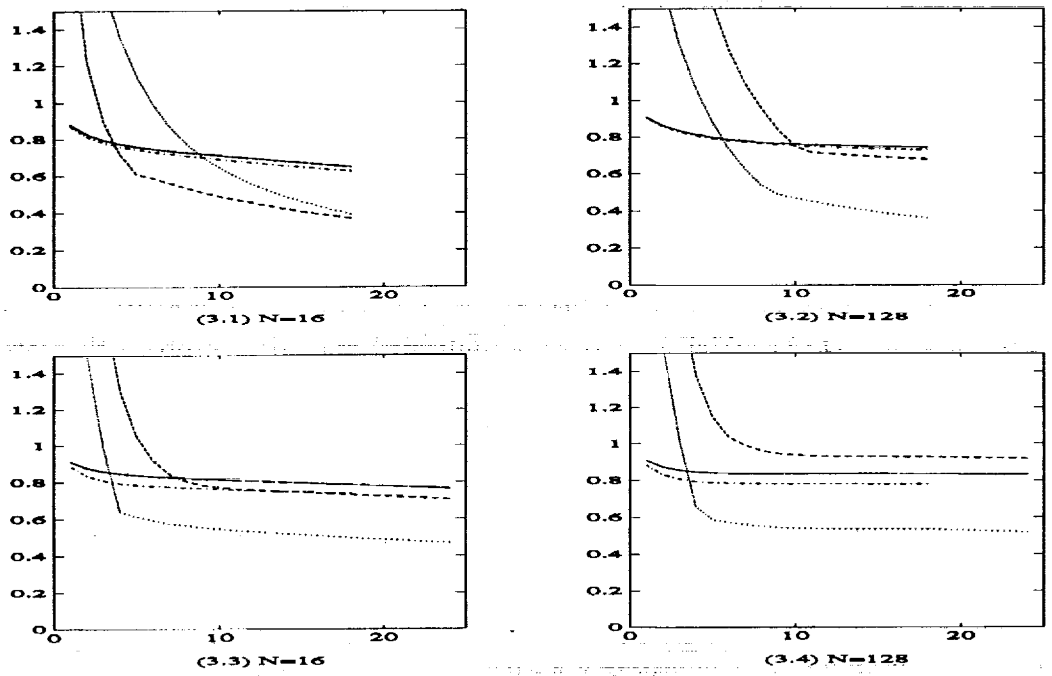


Figure 3: ρ as a function of γ. Dotted line is for (20), solid line for (21), dashed line for (22), and dashdot for (23). (3.1)-(3.4) are for different number grid points N.

In Figure 4, with the assumptions in Figure 3.3-3.4, we plot $a^\epsilon(x)$ and the approximation of (18) under the choices of coefficients in Figure 3.

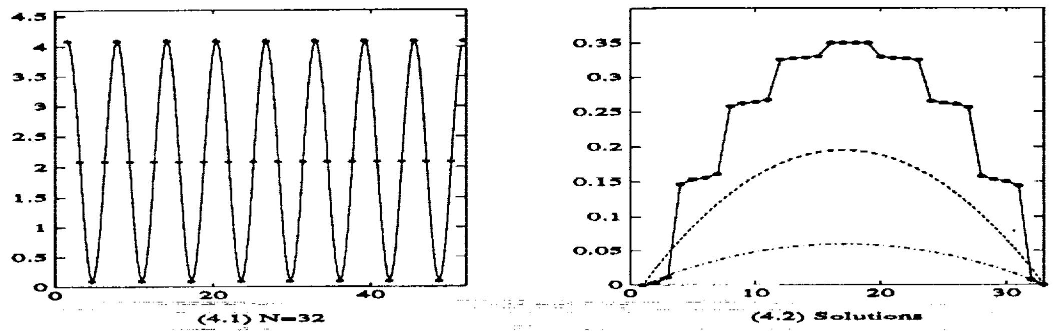


Figure 4: (4.1) and (4.2)are the graphs for $a^\epsilon(x)$, where * are the discretized values. (4.2) is the solution. Dashed line is for (17). Dashdot line is for $-m(a^\epsilon)u_{xx} = 1$ and line with circles is for (18).

In Figure 5, we plot ρ as a function of the variable α, where $L_H = \alpha\Delta_H$. In (5.1), $a^\epsilon(x) = 2.1 + 2sin(2\pi x/\epsilon)$, $\omega = 0.1829$ and $\epsilon = \sqrt{2}h$; In (5.2), $a^\epsilon(x) = 20.1\, if\; x/\epsilon - [x/\epsilon] \in (0.7, 0.9)$; $otherwise, 0.1$, $\omega = 0.0373$ and $\epsilon = 4h$.

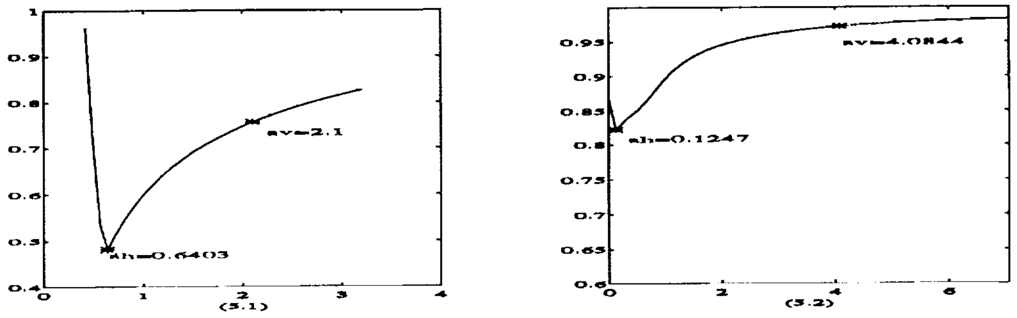


Figure 5: ρ as a function of α. The homogenized value $ah = m(1/a^\epsilon)^{-1}$ and the arithmetic value $av = m(a^\epsilon)$ are given. Here $\gamma = 10$ and $N = 256$.

In Figure 6, we present the convergence $u_\epsilon \rightarrow u$, as $\epsilon \rightarrow 0$ by giving the numerical solutions of (16) and (17). Recall that our goal is to solve the oscillatory problem and to use the homogenized operator only for the coarse grids.

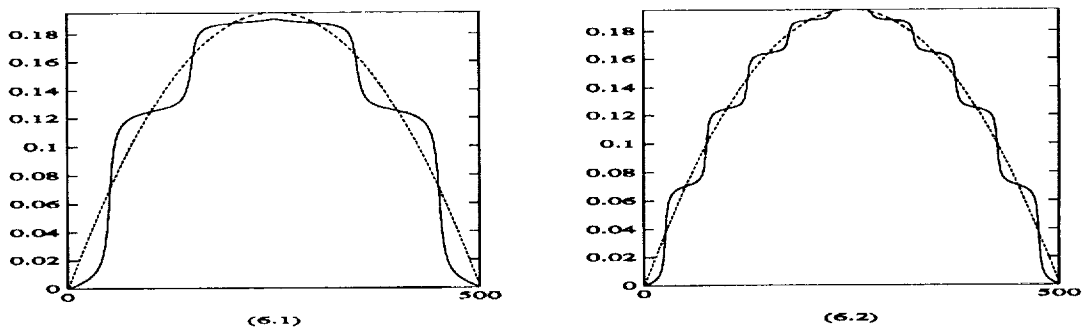


Figure 6: Solid lines are the approximations for (18), dashed lines are the solutions for (17), respectively, when $\epsilon = 0.2$ in (6.1) and $\epsilon = 0.1$ in (6.2). Here $N = 500$.

C-3

HYPERBOLIC PROBLEMS

Time evolution of a hyperbolic differential equation can be used for steady state computations. This is common in computational fluid dynamics, [6]. In multigrid this means that hyperbolic timestepping replaces the smoothing step. There are fundamental differences with standard multigrid for elliptic problems but some of our earlier discussions carry over to the hyperbolic case. The dissipative mechanisms for hyperbolic problems are mainly the boundary conditions. Consider using the model problem,

$$\frac{\partial^2 u_\epsilon}{\partial t^2} - \frac{\partial}{\partial x} a^\epsilon(x) \frac{\partial u_\epsilon}{\partial x} = f(x), \quad 0 \le x \le 1 \tag{24}$$

as the smoothing equation in multigrid for the numerical solution of (16), subject to the boundary conditions

$$u_\epsilon(0) = 0, \quad \frac{\partial u_\epsilon(1)}{\partial x} = 0. \tag{25}$$

The equation (24) must have boundary conditions which are dissipative but reduce to (25) at steady state, see [5],

$$u_\epsilon(0,t) = 0, \quad \frac{\partial u_\epsilon(1,t)}{\partial t} + \sqrt{a^\epsilon(x)} \frac{\partial u_\epsilon(1,t)}{\partial x} = 0. \tag{26}$$

The initial condition should support the transport of the residual to the dissipative boundary $x = 1$,

$$u_\epsilon(x, \tilde{t}) = u^0_\epsilon(x) \quad given,$$

$$u_\epsilon(x, \tilde{t} + \Delta t) = u^0_\epsilon(x) - \Delta t \sqrt{a^\epsilon(x)} D^x_0 u^0_\epsilon(x).$$

Note that the initial condition approximates the transport equation $u_t + \sqrt{a} u_x = 0$. The difference approximation of (24) needs a low level of numerical dissipation.

The homogenization theory of [2] is also valid for equation of the type (24). A numerical indication is seen in Figure 7.

The positive effect of multigrid on the convergence rate does not carry over to problems for which the steady state is hyperbolic or contains hyperbolic components. If,

$$\frac{\partial u_\epsilon}{\partial t} + \alpha \frac{\partial u_\epsilon}{\partial x} + \beta \frac{\partial u_\epsilon}{\partial y} = 0$$

is used for the equation,

$$\alpha \frac{\partial u_\epsilon}{\partial x} + \beta \frac{\partial u_\epsilon}{\partial y} = 0, \quad x, y \in [0,1],$$

$$u_\epsilon \quad is \quad 1-periodic \quad in \quad y, \quad u_\epsilon(0,y,t) = a^\epsilon(y).$$

The coarse grid operator must resolve all scales of a^ϵ to required accuracy in order to produce multigrid speed up. More on this phenomena will be reported elsewhere.

Numerical Results

In Figure 7, take 50 smoothing steps. Coefficient $a(x/\epsilon)$ is the same as in Figure 1.

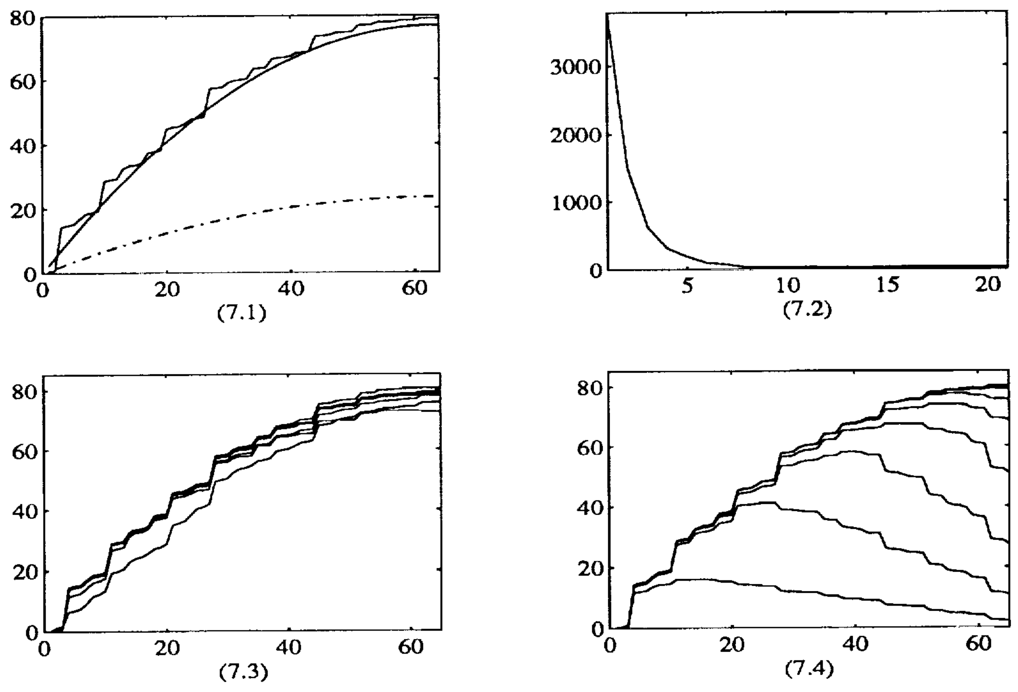


Figure 7: (7.1) Solutions: Solid line is the solution of steady state; Dashed line for homogenized solution; Dashed dot line for average solution. (7.2) Residue as function of two level multigrid cycles. (7.3) Approximate solutions after each two level cycle. (7.4) Approximate solutions for time evolution equation.

CONCLUSION

Elliptic equations and some hyperbolic equations with highly oscillatory coefficients have been studied. We have shown that the homogenized form of the equations are very useful in the design of coarse grid operators for multigrid.

The evidence is from a sequence of numerical examples with strongly variable coefficients and to some extent from theoretical analysis. The result is clear in the asymptotic regime of many smoothing iterations.

The impact on the coarse grid operator from the numerical truncation error and the interpolation operator needs to be asessed in order to improve the performance in the regime of very few smoothing iterations per cycle.

References

[1] Alcouffe, R.E.; Brandt, A.; Dendy, J.E.; and Painter, J.W.: The Multi-Grid Method for the Diffusion Equation with Strongly Discontinuous Coefficients. *SIAM J. Sci. Stat. Comput.*, vol. 2, no. 4, 1981, pp. 430-454.

[2] Bensonssan, A.; Lions, J.L.; and Papanicolaou, G.: *Asymptotic Analysis for Periodic Structure*, North-Holland, Amsterdam, 1987.

[3] Brandt, A: Multi-level Adaptive Solutions to Boundary-Value Problems. *Mathematics of Comput.*, vol. 31, no. 138, 1981, pp. 333-390.

[4] Engquist, B.: Computation of Oscillatory Solutions for Partial Differential Equations. *Lecture Notes 1270.* Springer Verlag, 1989, pp. 10-22.

[5] Engquist, B.; and Halpern, L.: Far Field Boundary Conditions For Computation Over Long Time. *Applied Numerical Mathematics*, no. 4, 1988, pp. 21-45.

[6] Jameson, A.: Solution of the Euler Equations for Two-Dimensional Transonic Flow by a Multigrid Method. *Appl. Math. comput.*, no. 13, 1983, pp. 327-355.

[7] Hackbusch, W.; and Trottenburg, U., eds.: *Multigrid Methods.* Lecture notes in mathematics 960. Springer Verlag, 1981.

[8] Khalil, M.; and Wessseling, P.: Vertex-Centered and Cell-Centered Multigrid for Interface Problems. *J. of Comput. Physics*, vol. 98, no. 1, 1992, pp. 1-10.

[9] Liu, C.; Liu, Z.; and McCormic, S.F.: An Efficient Multigrid Scheme for Elliptic Equations with Discontinuous Coefficients. *Communications in Applied Numerical Methods*, vol. 8, no. 9, 1992, pp. 621-631.

[10] Luo, E.: *Multigrid Method for Elliptic Equation with Oscillatory Coefficients.* Ph.D. Thesis, UCLA, 1993.

[11] Santosa, F.: and Vogelius, M.: *First Order Corrections to the Homogenized Eigenvalues of a Periodic Composite Medium.* To appear.

[12] Wesseling, P.: Two Remarks on Multigrid Methods. *Robust Multi-Grid Methods*, Hackbusch, W., eds., Wiesbaden: Vieweg Publ., 1988, pp. 209-216.

[13] Wesseling, P.: Cell-Centered Multigrid for Interface Problems. *J. Comput. Phys.*, vol. 79, no. 85, 1988, pp. 85-91.

[14] Varga, R.S.: *Matrix Iterative Analysis*, Prentice-Hall. Englewood Cliffs, NJ., 1962.

5/3-76'

197145

N94-23686

p 16

APPLICATION OF MULTIGRID METHODS TO THE SOLUTION OF LIQUID CRYSTAL EQUATIONS ON A SIMD COMPUTER

Paul A. Farrell[a], Arden Ruttan and Reinhardt R. Zeller
Department of Mathematics and Computer Science, Kent State University
Kent, OH

SUMMARY

We will describe a finite difference code for computing equilibrium configurations, of the order-parameter tensor field for nematic liquid crystals, in rectangular regions, by minimization of the Landau-de Gennes Free Energy functional. The implementation of the free energy functional described here includes magnetic fields, quadratic gradient terms, and scalar bulk terms through fourth order. Boundary conditions include the effects of strong surface anchoring. The target architectures for our implementation are SIMD machines, with interconnection networks which can be configured as 2 or 3 dimensional grids, such as the Wavetracer DTC. We also discuss the relative efficiency of a number of iterative methods for the solution of the linear systems arising from this discretization on such architectures.

INTRODUCTION: LIQUID CRYSTALS

Liquid crystal based technology plays a key role in many devices such digital watches and calculators, active and passive matrix liquid crystal displays in laptop computers, switchable windows using Polymer Dispersed Liquid Crystals (PDLCs), thermometers, temperature sensitive films and materials such as Kevlar which employ high-strength liquid crystal polymers. In addition they are likely to play a key role in developments such as High Definition Television (HDTV) and optical communications and computing.

Liquid crystals are so called because they exhibit some of the properties of both the liquid and crystalline states. In fact they are substances which, over certain ranges of temperatures, can exist in one or more *mesophases* somewhere between the rigid lattices of crystalline solids, which exhibit both orientational and positional order, and the isotropic liquid phase, which exhibits neither. Liquid crystals resemble liquids in that their molecules are free to flow and thus can assume the shape of a containment vessel. On the other hand they exhibit orientational and possibly some positional order. This is due to the intermolecular forces which are stronger than those in liquids

This work was supported in part by the Advanced Liquid Crystalline Optical Materials (ALCOM) Science and Technology Center at Kent State University under DMR89-20147.
[a]This author was supported in part by The Research Council of Kent State University.

and which cause the molecules to have, on average, a preferred direction. Liquid crystals may exist in a number of mesophases, such as the *nematic, smectic, cholesteric* phases (see [3]).

In this paper we shall confine ourselves to nematic liquid crystals, which exhibit orientational but no positional order. We wish to study the orientational order and the inter-molecular forces that are present in a nematic liquid crystal material. To do this we need a quantitative measure of the degree of order and of the total *free-energy* (sum of inter-molecular forces) in the system. A typical liquid crystal molecule is long, rod-like and rigid. Its direction in space is given by the unit vector $n = (n1, n2, n3)$. The molecule points in the n or $-n$ direction with equal probability; therefore, there is no up or down direction. The *director* $\hat{n} = (\hat{n1}, \hat{n2}, \hat{n3})$ is also a unit vector showing the preferred average direction of the molecules at a point in the sample. The *degree of order* of a liquid crystal material at a particular point in the sample can be measured in terms of the statistical average of the angles θ, which molecules make with the director. A more common measure that is used is $S :=< 3\cos^2\theta - 1 > /2$, where $<>$ is a thermodynamic or temporal average. A value close to 1 indicates a strong ordering of the molecules as is present in a crystalline solid. Values near zero indicate random ordering, such as exist in an isotropic liquid. The order parameter S depends on the temperature T.

Most early theoretical and computational results on liquid crystals employed the Oseen-Frank theory. This assumes that the degree of order S is uniform throughout the material and seeks to calculate the equilibrium configuration of the material by obtaining the director field which minimizes the *free energy functional*

$$F(n) := \frac{1}{2}\int_\Omega \{K_1(\nabla \cdot n)^2 + K_2(n \cdot \nabla \times n)^2 + K_3|n \times \nabla \times n|^2\}.$$

In an infinite bulk the preferred configuration for the director field is one of uniform parallel alignment. This will not normally be the case in practice, however, due to the effects of boundaries and external fields. This theory, while instrumental in predicting many important phenomena in liquid crystal physics, has some deficiencies. In particular, it is inadequate to model behavior close to a defect, where the order may not be uniform and the director may not be well defined. For example, in the presence of a radial field about a line defect this theory will exhibit a singularity at the core. For this reason there is increased emphasis on the more computationally complex Landau-de Gennes formulation.

THE LANDAU DE-GENNES FORMULATION

The Landau-de Gennes formulation describes nematic liquid crystals by a 3×3 symmetric, traceless tensor order parameter Q. The local orientational information is given by the eigenvectors and eigenvalues of Q at each point. Several behaviors can be distinguished by considering the relative magnitudes of the eigenvalues. The material is said to be *uniaxial* if Q has a unique largest eigenvalue, with the two other eigenvalues equal to minus half the largest one. The corresponding eigenvector gives the locally preferred direction . Thus this is the case which can be represented by the Oseen-Frank theory and in fact in this case Q can be represented in the form

$$Q = \frac{1}{2}S(3\hat{n}\hat{n}^T - I)$$

where S is the value of the maximum eigenvalue and $\hat{n}$ is the normalized eigenvector associated with it. The Landau-de Gennes formulation, however, is capable of representing more complex behaviors, such as the *biaxial* case, where all three eigenvalues are distinct and the *isotropic* case, where all three eigenvalues are equal and hence, because Q is traceless, all three are 0.

To obtain the equilibrium tensor field again seek a tensor field Q that minimizes the free energy of the system. In this case, the free energy can be expressed as

$$F(Q) = F_{\text{vol}}(Q) + F_{\text{surf}}(Q) = \int_{\Omega} f_{\text{vol}}(Q) + \int_{\partial\Omega} f_{\text{surf}}(Q)\,,$$

where Ω and $\partial\Omega$ represent the interior and surface of the slab respectively. In this implementation we limit ourselves to strong anchoring on the surface of Ω.

The term $F_{\text{vol}}(Q)$ gives an approximation of the interior free energy and is given by the following expression, (see, for instance [18]):

$$\begin{aligned} f_{\text{vol}}(Q) \;:=\; & \frac{1}{2}L_1 Q_{\alpha\beta,\gamma}Q_{\alpha\beta,\gamma} + \frac{1}{2}L_2 Q_{\alpha\beta,\beta}Q_{\alpha\gamma,\gamma} + \frac{1}{2}L_3 Q_{\alpha\beta,\gamma}Q_{\alpha\gamma,\beta} + \frac{1}{2}A\,\text{trace}(Q^2) \\ & -\frac{1}{3}B\,\text{trace}(Q^3) + \frac{1}{4}C\,\text{trace}(Q^2)^2 + \frac{1}{5}D\,\text{trace}(Q^2)\text{trace}(Q^3) \\ & +\frac{1}{6}M\,\text{trace}(Q^2)^3 + \frac{1}{6}M'\,\text{trace}(Q^3)^2 - \Delta\chi_{\max}H_\alpha Q_{\alpha\beta}H_\beta \end{aligned} \tag{1}$$

where L_1, L_2, and L_3 are elastic constants, A, B, C, D, M, and M' are bulk constants, and H, $\Delta\chi_{\max}$, and E are the field terms and constants associated with the magnetic field respectively, and the convention is used that summation over repeated indices is implied and that indices separated by commas represent partial derivatives. The surface free density f_{surf} has the form

$$f_{\text{surf}}(Q) := \frac{1}{2}V\text{trace}((Q - Q_0)^2) \tag{2}$$

where Q_0 is a tensor associated with the type of anchoring of the surface elements and V is prescribed constant. In the strong anchoring case presented here Q cannot vary from Q_0 and hence $\int_{\partial\Omega} f_{\text{surf}}(Q) = 0$.

For $P \in \Omega$, the tensor $Q(P)$ will be represented in the form,

$$\begin{aligned} Q(P) &= (Q_{\alpha\beta})_{\alpha,\beta=1}^{3} \\ &= q_1(P)\phi_1 + q_2(P)\phi_2 + q_3(P)\phi_3 + q_4(P)\phi_4 + q_5(P)\phi_5 \\ &= q_1(P)\begin{pmatrix} \frac{\sqrt{3}-3}{6} & 0 & 0 \\ 0 & \frac{\sqrt{3}+3}{6} & 0 \\ 0 & 0 & \frac{-\sqrt{3}}{3} \end{pmatrix} + q_2(P)\begin{pmatrix} \frac{\sqrt{3}+3}{6} & 0 & 0 \\ 0 & \frac{\sqrt{3}-3}{6} & 0 \\ 0 & 0 & \frac{-\sqrt{3}}{3} \end{pmatrix} \\ &\quad + q_3(P)\begin{pmatrix} 0 & \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + q_4(P)\begin{pmatrix} 0 & 0 & \frac{\sqrt{2}}{2} \\ 0 & 0 & 0 \\ \frac{\sqrt{2}}{2} & 0 & 0 \end{pmatrix} + q_5(P)\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \\ 0 & \frac{\sqrt{2}}{2} & 0 \end{pmatrix}, \end{aligned}$$

similar to that in Gartland [12], where $\{q_\ell(P)\}_{\ell=1}^{5}$ are real-valued functions on Ω.

The discretization of the full slab problem in which a finite difference approximation of the equilibrium configuration of liquid crystals in a slab

$$\Omega = \{(x, y, z) : 0 \le x \le a, 0 \le y \le b, 0 \le z \le c\}$$

is given in [7]. In this paper we shall confine our consideration to the case of an infinite slab. Assuming the slab is infinite in the z-direction and imposing boundary conditions, which do not vary with z, effectively reduces the problem to a two dimensional problem on a rectangle:

$$\Omega := \{(x, y) : 0 \le x \le a,\ 0 \le y \le b\}.$$

The region is discretized in the standard manner by dividing the rectangle Ω into $I \times J$ regions

$$v(i, j) = \{(x, y) : i\Delta x \le x \le (i+1)\Delta x, j\Delta y \le y \le (j+1)\Delta y\}$$

for $0 \le i \le I-1$, and $0 \le j \le J-1$, where $\Delta x = a/I, \Delta y = b/J$.

The discrete interior free energy integral is now represented by

$$\int_\Omega f_{\text{vol}}(Q) \approx \sum_{i,j} f_{\text{vol}}(Q(x_i, y_j)) \times volume(v(i, j)), \tag{3}$$

where the points $P = (x_i, y_j)$, for $x_i = i\Delta x$ and $y_j = j\Delta y$, are located in the lower left-hand corner of the rectangle $v(i, j)$. The derivatives with respect to x and y in (1) are approximated using central difference approximations.

With the assumption of strong anchoring, a second order accurate approximation of the Landau-de Gennes free energy density given by

$$F(Q) \approx \sum_{i,j} f_{\text{vol}}(Q(x_i, y_j)) \times volume(v(i, j)) = \sum_{i,j} h(x_i, y_j) \tag{4}$$

is obtained. With the discretization (4), the problem is reduced to one of minimizing $\sum_{i,j} h(x_i, y_j)$ overall choices of $\{q_\ell(x_i, y_j)\}_{\ell=1}^5$. This unconstrained discrete minimization problem can be attacked in the standard way. That is, seek a solution of the non-linear system of equations

$$g(\hat{\ell}, \hat{i}, \hat{j}) := \frac{\partial \sum_{i,j,k} h(x_i, y_j)}{\partial q_{\hat{\ell}}(x_{\hat{i}}, y_{\hat{j}})} = 0, \tag{5}$$

for $0 \le \hat{i} \le I$, $0 \le \hat{j} \le J$, and $\hat{\ell} = 1 \ldots 5$. A standard approach to solving non-linear systems such as these is to use a modified Newton method (see [6]).

Each iteration of the modified Newton method involves solving a linear system, whose matrix is the Jacobian of (5), and then using that solution to update the iterate and the Jacobian, after which the process is repeated. The system in question is a large symmetric system, but for certain values of the temperature it may become indefinite. In addition, it may be expected to exhibit multiple solutions, which may be either stable or unstable. The ultimate aim of this research is to track the minimal energy states as the temperature varies and to model the resulting bifurcations and phase transitions.

THE WAVETRACER DTC ARCHITECTURE

The target architecture for this application is a massively parallel SIMD computer. A SIMD computer uses multiple synchronized processing elements that operate in a lock-step fashion to achieve parallelism. Each processing element (PE) performs the same operation at the same time on its local data which is either stored in its own local memory or in a shared memory. A control unit (CU) broadcasts instructions to the processing elements for execution. Each PE can be either active or inactive during a particular operation. The control unit determines which PEs are to participate by means of a masking function that either turns a PE on or off. Only the selected processors execute the instruction, while the masked processors remain idle. The control unit normally buffers data and instructions that will be broadcast to the processor array. A front-end computer provides the programming environment along with the usual programming utilities such as a debugger and a compiler. Program code is compiled and separated into scalar and parallel instructions. Scalar operations are usually executed on the front-end, thus freeing the processor array to perform only parallel computations. This architecture is considerably simpler to implement and program than the alternative Multiple Instruction Multiple Data stream (MIMD) machines, in which each processor can execute a different instruction. The SIMD architecture is normally used for massively parallel machines, having between 4096 and 65536 processors, each with local memory, connected by a special purpose high-capacity communication network. Early examples of this architecture included the MASPAR MP-1 and MP-2 and the Thinking Machines Corporation Connection Machine CM-1 and CM-2.

The platform chosen for this implementation was the Wavetracer Data Transport Computer (DTC), situated in the Department of Mathematics and Computer Science at Kent State. This has a number of unique features compared with previous SIMD computers. It was designed as a low cost massively parallel processor, which can deliver "super-computing" levels of performance at relatively low cost. Unlike previous SIMD machines, which had dedicated front-end processors for storing scalar data and performing uni-variable (scalar) computations, the DTC uses a standard workstation for this purpose as well as for compilation and storage of the program. Among front-ends supported were the Sun 3, Sparc and Hewlett-Packard/Apollo workstations.

The DTC is connected to the front-end by means of the industry standard Small Computer System Interface (SCSI), which is normally used to connect hard disks. The maximum bandwidth of this interface is 5 Mbytes per second. The front-end sends instructions and data to a control unit, which decodes these instructions and broadcasts both instructions and data to the processor array. The array processors are semi-custom 1.5 micron standard cell chips. Each chip contains 32 one-bit processors together with 2 kilobits of fast RAM for each processor, and associated control and memory error-detection circuitry. In addition, each processor has access to between 8 and 32 kilobytes of private external dynamic memory depending on the configuration. Each circuit board consists of 128 chips. The minimal configuration, the DTC-4, has one circuit board and thus 4096 processors. Other configurations are the DTC-8, with 2 circuit boards and 8192 processors, and the DTC-16, with 4 circuit boards and 16384 processors.

The processors on each circuit board of the DTC-4 can be configured either as a $16 \times 16 \times 16$ cube, for three dimensional application, or a 64×64 square, for two dimensional applications. The DTC-8, can be configured as $16 \times 32 \times 16$ cube or a 64×128 square, and the DTC-16 as a $32 \times 32 \times 16$ cube or a 128×128 square. The assumption here is that most applications correspond to physical problems in 2 or 3 dimensions, and thus a 2 and 3 dimensional interconnection network is the most efficient for their solution. This is in contrast to the Connection Machine, in which the processors are connected by a hypercube network.

There are a number of factors which affect the DTC's performance. Firstly, the speed of the front-end is a determining factor in the overall performance of the DTC, since all uni-variable expressions are processed on the front-end and, in addition, all instructions are passed from the front-end to the control unit. In addition, although the DTC provides efficient data movement along the grid, the results of propagating data to the left, for example, are undefined at the right boundary nodes. In addition, for problems with periodic boundary conditions it is desirable that the interconnection network have *wraparound*, in which one can propagate values from one boundary to the other. This is not provided. This also poses a problem for periodic geometries such as spherical or ellipsoidal. One other inconvenience is that there is no microsecond timer on the DTC and all timings must be done on the front end.

The traditional mode of solution of problems on a SIMD machine involves assigning one processor of the array per node in the problem space. To provide the ability to consider problems with more nodes than are available in the array, the DTC provides the ability to partition the memory of each processor to provide a larger number of *virtual processors*. There must be the same number of virtual processors for each physical processor. The number of virtual processors per physical processor is called the *virtual processor ratio*. The controller automatically issues instructions to the array once for each partition. Thus the execution time may be expected to increase linearly with the virtual processor ratio.

The Wavetracer used in the results presented here was a Wavetracer DTC-4 with a Sun 3/50 front end. Current codes are bring run on a Wavetracer DTC-16 with a Hewlett-Packard/Apollo 705 front end. For the minimization problem we are considering, each discretization point, P, of the slab is associated with a virtual processor. Since the virtual processors are arranged in a rectangle or cube, similar to the actual processors, this provides an entirely natural mapping of the domain onto the rectangular grid of the DTC, provided an equal number of grid points are used in each direction.

At each point P of the slab the tensor order parameter Q is defined in terms of the 5 unknowns $\{q_\ell(P)\}_{\ell=1,5}$. In our implementation, each set of 5 unknowns $\{q_\ell(P)\}_{\ell=1,5}$ is stored in a single virtual processor. Associated with each unknown $q_\ell(P)$ there is also a corresponding row of the Jacobian matrix. The nonzero constants of that row are also stored in the memory of the processor associated with P. Each non-zero constant, in a row of the Jacobian associated with P, also corresponds to another virtual processor (which in turn corresponds to a discretization point) to which the values of $\{q_\ell(P)\}_{\ell=1,5}$ at P must be communicated when the Jacobian matrix is updated. The set of processors with which a given processor, P, must communicate in order to update its row of the Jacobian is called the stencil of P. If the stencil of any processor is large, then the process of updating the Jacobian at each step of Newton's method will be expensive. Fortunately, the finite difference approximation described here yields a relatively small and compact stencil. In the problem discussed below, the stencil will at worst consist of the nine points which correspond to processors at most two steps away from the given processor.

SOLUTION OF THE MINIMIZATION PROBLEM

Non-linear iterations

The minimization of the free energy can be carried out by solution of the corresponding discrete Euler-Lagrange equations given by (5). These give rise to a coupled system of five non-linear elliptic partial differential equations.

An alternative approach is to compute the Euler-Lagrange equations from $\int_\Omega f_{\text{vol}}(Q)$. Discretizing these produces a system similar to (5). In this case central differences are used for the unidirectional partial derivatives. Two alternative choices of discretizations for the mixed derivatives, both having the same accuracy, are considered. One produces a seven and the other a nine point stencil at each nodal point in the domain. Since nearest-neighbor communications are efficient on the Wavetracer's mesh array of processors, the communication costs are minimal. A reduced model in which $L_2 = L_3 = D = M = M'$ was also considered. This is significantly less complex and gives rise to a five point scheme. Results for this case were considered in [7].

In all cases the resulting non-linear system of equations was solved using a (modified) inexact Newton method. Let $G : R^n \to R^n$ be a function representing the discrete Euler-Lagrange equations. There are a total of $5(I-1)(J-1)$ non-linear equations in this system. The function G depends on the $5n$ unknowns

$$G(\mathbf{x}) = G\ (q_1^1, \ldots, q_5^1, q_1^2, \ldots, q_5^2, \ldots, q_1^n, \ldots q_5^n),$$

where $n = (I-1)(J-1)$ is the number of nodal points. Let $G'(\mathbf{x})$ be the Jacobian of the system of equations. Newton type methods require solving a large sparse linear system $G'(\mathbf{x}_k)\mathbf{s}_k = -G(\mathbf{x}_k)$ and then updating the unknowns appropriately.

In theory, Newton's method requires the exact solution to the linear system for each Newton iteration. Inexact Newton methods use some form of iterative procedure to solve the linear system approximately. Several iterative techniques such as SOR and multigrid were tested on this problem with varying success. Note that the matrix $A := G'(\mathbf{x}_k)$ is singular at bifurcation and turning points and can be indefinite near these points. This can cause convergence problems when solving the inner linear system. It is well known that in the early stages of the Newton or *outer* iteration process, the linear system need not be solved to full accuracy, since $\mathbf{x}_k$ is relatively far from the true solution $\mathbf{x}^*$. Thus only a few *inner* iterations of the linear solver need to be performed. In later stages, the inner system will need to be solved more accurately. This is precisely the philosophy of the inexact (modified) Newton method. A common criterion used to determine how many inner iterations are needed is as follows. In the k^{th} iteration, compute a value $n_k \epsilon [0,1)$ which is an acceptable bound on the relative residual. Common choices for this are $n_k := \frac{1}{2^{k+1}}$, $n_k := \frac{1}{k+2}$, and

$n_k := \min\{\|G(\mathbf{x}_k)\|, \frac{1}{k+2}\}$. For these problems the second of the above choices proved on average to give the best results. The update $\mathbf{s}_k$ was then determined by:

$$\frac{\|G(\mathbf{x}_k) + G'(\mathbf{x}_k)\mathbf{s}_k\|}{\|G(\mathbf{x}_k)\|} \leq n_k \tag{6}$$

Expression (6) may be interpreted intuitively as indicating that one should iterate until the inner residual becomes "small" enough, then do an update.

Linear System Solvers

Several classical iterative schemes were used to solve the inner sparse linear system for $q_1, \ldots, q_5$ at each nodal point. Each method had certain advantages and disadvantages when used as a solver on the Wavetracer. The following schemes were evaluated :

1. Multi-color SOR
2. Nested (multilevel) multi-color SOR
3. Preconditioned conjugate gradient
4. Multigrid (V-cycle)
5. Nested (multilevel) multigrid

All were implemented as both point-iterative and as block-iterative methods by blocking the $q_1, \ldots, q_5$ at each nodal point. In the point iterative methods one solves for each q_i sequentially, using the best available values for the $q_j, j \neq i$. The block method involves solving a 5×5 dense linear system at each node.

A multi-coloring scheme was used for the SOR iterations [17] in order to introduce parallelism into the method. One should recall that, with red-black ordering, the Gauss-Seidel method decomposes into two Jacobi steps on the half size systems resulting from the coloring. Unlike the original Gauss-Seidel method, the Jacobi method is highly parallelizable. The multi-colored SOR produces similar benefits. In the case of the reduced model with the five point stencil only two colors were needed. Results for this case are given in [7]. In the full model, three colors are required for the seven point stencil and four colors for the nine point stencil. The parameter ω for the SOR method was chosen as the *optimal* parameter for the simple Laplacian model since the matrix in our linear system has a similar structure to the Laplacian matrix. Numerical experimentation showed that this was a good choice for our reduced model and gave good convergence results.

Preconditioned conjugate gradient [17] using several pre-conditioners was tried and the performance of all were essentially similar. The results are presented here for symmetric multi-colored SSOR [17], which is simple to implement and easily parallelizable.

Multigrid methods [2, 14, 16] were also implemented for these problems. The multigrid implementation discussed here uses a single V-cycle in the inner iteration for each Newton outer iteration. The Gauss-Seidel iteration is used as the relaxation method on the fine and intermediate coarse grids. The Gauss-Seidel method was chosen over the SOR method for the fine and intermediate grids because of its better *smoothing* property; that is, it eliminates the high frequency components quicker in the early iterations than the SOR iteration. This is important because a few iterations are performed on these grids per cycle of the multigrid algorithm. The relaxation parameters ν_1 and ν_2 were usually taken to be equal to 3. Multicolored SOR iteration was used to solve the problem on the coarsest grid which was usually taken to be of size $n = 4$. The problem was solved to the level of the truncation error with usually just a few iterations. The numerical simulations were mostly done on the two-dimensional problem of size $n = 64$, meaning 65 grid points in both the x and y directions. Some smaller and larger problems were also examined, but with the minimal configuration of the *Wavetracer*, the DTC-4, available at the time, the $n = 64$ size problem was the largest that could be simulated for the full liquid crystal problem using the multigrid method.

The implementation of the multigrid algorithm on the *Wavetracer* assigns a processor (virtual or physical) to each grid point on the finest mesh, including the boundary grid points. The model simulations all assume Dirichlet (strong anchoring) boundary conditions, so the boundary processors are used mainly to store the boundary data. The *Wavetracer* uses a multi-array data structure to hold the values for each grid level. Because of the restriction in the MultiC language that each multi-variable in the executing program must be of the same size, this implementation was deemed to be the most efficient and easiest to implement. One problem with this implementation is that many processors are idle when solving on the coarser grids. The multigrid is thus not a fully parallelizable method using this implementation because not all processors are being utilized. Alternative variations have been proposed to overcome this problem. Data transfers between grids are fast since they are handled within processor memory and no communications between processors is required. Communications are required when computing the weighted averages for the restriction operation, but the actual transfer of data to the coarser grid is all done within processor memory. Another drawback to this implementation of the multigrid method is evident when one solves the $n = 64$ size problem in two-dimensions. The physical two-dimensional processor grid on the *Wavetracer* contains 64 processors in each dimension for a total of 4096 processors. The $n = 64$ multigrid problem requires 65 mesh points in each of the x and y directions. This causes the *Wavetracer* to operate in *virtual memory* mode. Since each physical processor must contain the same amount of virtual processors, many virtual processors will remain idle during the iterations, resulting in a great loss in efficiency. In addition, since the available memory associated with each physical processor is divided into two halves, one for each of the virtual processors, the maximum problem size, which can be solved, is diminished. Naturally, the solution would be to define a slightly smaller problem of $n = 63$ that would not have this difficulty. The problem then becomes one of how to define the series of coarser grids. In our original definition of the coarse grids we let each grid size be a power of two. This greatly simplifies the construction of the grids and provides the necessary symmetry to allow us to assign processors to the different grid levels in the manner described. Data transfer between grids is also extremely simple, since it is all handled within processor memory. Defining the coarse grids in any other way would greatly complicate the programming process and would require many more computations and inter-processor communications.

Another solution to this problem would be to use the boundary processors to not only store the boundary data but also to take part in the iteration process. This means that now each boundary processor would really represent two grid points in the mesh instead of only one. This would solve the virtual processing problem because one would actually need only 63 physical processors in each direction for the $n = 64$ problem. However, another problem presents itself because of the *SIMD* nature of the *Wavetracer*. In a *SIMD* environment each processor must perform exactly the same operation as all the other processors, except on a different set of data. The boundary processors as defined above would have to be treated separately from the interior processors because in the communications stage of the algorithm they are not performing the same operation. An interior processor must communicate with its four nearest-neighbors in a five-point stencil scheme, whereas a boundary processor would only have to communicate with a subset of its neighbors since the boundary data that it needs to do its update is stored in its own memory. In the two-dimensional mesh the processors on each of the four edges of the grid must be treated separately as must the four corner processors. In a naive implementation these sets of processors would be handled sequentially in the iteration process, greatly slowing down the computations. In fact, if the obvious choice is made, this could increase the update time nine times, which is considerably more than the increase incurred by virtual processing. Unfortunately this problem is not so easily avoided when one considers general boundary conditions rather than Dirichlet conditions.

Another alternative approach is to use a Black Box multigrid method similar to that in Dendy [4, 5]. This eliminates the restriction that the number of unknowns in the finest grid should be $2^k + 1$; for some k. In addition, by storing the interpolation operators explicitly, it allows the incorporation of the boundary conditions, for example, by using extrapolation at the points closest to the boundaries. Thus the boundary conditions are incorporated algebraically rather than by using the difference equations directly. This does involve extra storage and in the SIMD case loss of parallelism due to grid point dependent code. However judicious coding, involving initialized multipliers, can reduce the latter effect at the expense of some further storage. There is reason to believe that, for most problems of this type and most geometries, the increased storage will be less than 100% and thus that a code of this type will consume less storage overhead than one involving virtual processing.

The philosophy behind the nested or multilevel schemes [1, 13] is as follows. The problem is solved on a coarse grid to a certain precision. The results are then interpolated to a finer grid and used as initial starting values for the solution process there. A sequence of successively finer grids is used, the finest is the one on which the result is required. It is hoped that providing good initial guesses will reduce the amount of work needed to obtain the desired accuracy on the finer grids. This effect is observed in the numerical simulations. The multilevel methods suffer the same kinds of problems that the multigrid iterations suffered when implemented on the *Wavetracer*. The different levels are implemented using a multi-variable array (in the MultiC language) with the physical (or virtual) processors assigned to the grid points on the finest grid level. This means that when one iterates on the coarsest level, many virtual processors will be idle. The interpolation of results between grids is fast because it is all done within the processor and no inter-processor communications are necessary.

NUMERICAL RESULTS

Laplacian and Scalar Liquid Crystal Problem

Laplacian in Two Dimensions

The model Laplacian problem in two-dimensions is given by:

$$-u_{xx} - u_{yy} = f(x, y), \qquad u = g(x, y) \quad \textit{on the boundary of } \Omega. \tag{7}$$

Dirichlet boundary conditions are assumed and Ω is taken to be the *unit square*. The performances of the various iterative methods previously discussed are compared for problems of size $n = 63$ for the one-level schemes and $n = 64$ for the methods using more than one level. For these simulations we also assume a known true solution given by

$$u = x^2y^2 \tag{8}$$

which makes the right-hand side of equation (7)

$$f(x, y) = -2.0 * (x^2 + y^2). \tag{9}$$

With this known solution one can compute the *error* as well as the *residual* after each iteration in order to observe the convergence. The boundary values are set to the known true solution and an initial guess of $u = 0.0$ is used at all interior grid points to start the iterations. At each iteration the maximum absolute error and residual (infinity norms) calculated over all interior grid points are monitored.

The Wavetracer DTC does not itself contain a micro-second timer. Consequently, all timings must be performed on the Sun 3/50 front end. The columns real, user and syst give the real (wall clock) time, the time spent in systems tasks related to the program, including input/output, and the time spent in executing user code on the front end. The input/output time includes time spent accessing the SCSI bus and thus time spent sending instructions from the front end processor to the sequencer of the Wavetracer. User time includes time spent executing the sequential parts of the program. The majority of the remaining real time is time elapsed while the DTC is executing parallel instructions.

The results of these simulations are given in Table 1. Given the initial guess $u = 0.0$, the maximum initial error is 1.0 and the maximum initial residuals are approximately 7934 and 7684 for the $n = 64$ and $n = 63$ size problems, respectively. The iterations are continued until the maximum absolute error is reduced by about a factor of 10^5. A red-black scheme is implemented for all the iterations (except Jacobi) to induce parallelism into the methods. The red-black coloring scheme is appropriate since the model Laplacian problem uses a 5-point stencil for processor communications. The iterations are done on the *Wavetracer* using a 64×64 physical two-dimensional grid of processors.

Table 1. Timings for the Model Laplacian Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	iterations
Jacobi (n=64)	150.8	9.9	67.0	1.5(-3)	3.5(-5)	6901
Jacobi (n=63)	132.6	13.5	78.4	2.1(-3)	3.5(-5)	6689
Gauss-Seidel (n=64)	89.1	7.5	40.5	1.5(-3)	3.5(-5)	3451
Gauss-Seidel (n=63)	67.3	8.2	39.3	1.5(-3)	3.5(-5)	3345
SOR (n=64)	3.2	0.4	1.5	6.2(-2)	3.4(-5)	113
SOR (n=63)	3.2	0.4	2.0	5.8(-2)	3.5(-5)	111
Pre-cg (n=64)	6.1	0.5	1.0	7.2(-2)	2.5(-5)	32
Pre-cg (n=63)	2.7	0.5	1.0	6.8(-2)	3.1(-5)	31
Multigrid (n=64)	2.5	0.3	0.5	3.4(-2)	3.5(-5)	3 V-cycles

As expected, the Jacobi iteration is the slowest to converge. Even though it is completely parallelizable on the *Wavetracer*, its slow rate of convergence does not make it competitive. The Gauss-Seidel method converges in about half as many iterations as the Jacobi method. This is expected for the model Laplacian problem. Since the Gauss-Seidel iterations are implemented in a red-black ordering, each iteration takes slightly longer than a Jacobi iteration. For both the Jacobi and Gauss-Seidel iterations the real running times for the $n = 63$ size problem are faster than those for the $n = 64$ problem. This is because the $n = 64$ problem uses virtual processors whereas the $n = 63$ problem fits the physical grid of processors precisely.

The SOR method greatly improved the convergence of the problem. It needed only 113 iterations to get to the same level of error as the previous two iterative methods (for the $n = 64$ problem). The real times, user, and systems have also been significantly reduced. This agrees with the theoretical results for the behaviour of these three iterative methods on the model problem.

The preconditioned conjugate gradient iteration was implemented using a red-black coloring scheme and *Symmetric SOR* as the preconditioner. The method is competitive with the SOR iteration for the $n = 63$ problem. It is, however, slower than SOR for the slightly larger problem.

To make a fair comparison, one must compare the multigrid algorithm with the $n = 64$ size problems of the other four iterative methods, since multigrid was implemented using a finest grid of this size. As one can see from the table, multigrid converges significantly faster than Jacobi, Gauss-Seidel and preconditioned conjugate gradient, and slightly faster than SOR (in real time). It even beats the other four methods when they are run on the smaller problem. This shows that multigrid is a very competitive method even with its limitations as discussed previously. Only three V-cycles are needed to reduce the error to the desired level. Five levels were used ($n = 4$ at the coarsest level) with $\nu_1 = \nu_2 = 3$.

Scalar Liquid Crystal Problem

The scalar analog to the full liquid crystal problem is of interest because it has a similar structure to the full model. Various algorithms for solving the full model are first developed for the scalar problem. The relative performances of these algorithms were basically the same for both models.

The free-energy density for the scalar-field analogue to the full systems model is given by:

$$f(q) = \frac{1}{2}L_1|\nabla q|^2 + \frac{1}{2}Aq^2 - \frac{1}{3}Bq^3 + \frac{1}{4}Cq^4 - H^2 q \tag{10}$$

where L_1 is an elastic constant, A,B,C are bulk constants and H is a field term representing an outside field such as a magnetic field. To minimize the free-energy of the system one needs to solve the *Euler-Lagrange* equation

$$-L_1\nabla^2 q + Aq - Bq^2 + Cq^3 = H^2. \tag{11}$$

Equation (11) is non-linear in the scalar variable q. The resulting linear system that needs to be solved at each Newton step is very similar in structure to the *Laplacian* problem. The only difference is the additional terms on the diagonal elements of the A matrix that is a result of the non-linearity of the scalar problem. The discretization of the scalar Euler-Lagrange equation produces a 5-point stencil at each mesh point. The communications pattern is thus the same as it was for the Laplacian problem. A red-black coloring scheme is sufficient to induce parallelism into the iterative solvers used.

For the problem used in these tests $L_1 = 1.0$, $A = B = C = 1.0$, $H = 0.0$, with Dirichlet boundary conditions given by :

$$q = 1 \text{ on } x = 1 \text{ and } y = 1\,, \quad q = x \text{ on } y = 0\,, q = y \text{ on } x = 0\,.$$

The true solution to this problem is not known, therefore the error cannot be computed. The maximum absolute residual at each iteration is used to monitor the convergence. The initial guess is given by q=0.0 at each interior mesh point and iteration proceeds until the maximum absolute residual is reduced by approximately factor of 10^6. The initial residuals for the $n = 64$ and $n = 63$ size problems are 8192 and 7938, respectively. Table 2 gives the results of the simulations.

Table 2. Timings for the Scalar Liquid Crystal Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	outer iter.
SOR (n=64)	6.2	0.3	1.4	2.7(-3)	—	9
SOR (n=63)	3.3	0.3	1.4	2.4(-3)	—	9
Pre-cg (n=64)	12.0	0.8	1.2	2.5(-3)	—	8
Pre-cg (n=63)	5.2	0.8	0.9	2.1(-3)	—	8
Multigrid (n=64)	4.0	0.5	0.6	2.3(-3)	—	4
Nested SOR (n=64)	6.8	0.5	1.7	5.9(-3)	—	19(4,3,4,4,4)
Nested Multigrid (n=64)	4.0	0.4	0.7	5.9(-3)	—	8(4,1,1,1,1)

Comparing the real execution times of these algorithms shows again that the preconditioned conjugate gradient method is not competitive on this kind of architecture. The nested (multilevel) methods use five levels with the coarsest level being of size $n = 4$. The numbers in parentheses in the last column of the table are the number of outer Newton iterations needed to achieve convergence at each level.

We refer to a nonlinear Newton based analogue of the Full Multigrid (FMG) method as nested (multilevel) multigrid. It employs Newton iteration on each mesh level until the desired accuracy is attained. In the case of the example considered here, this required 4 outer iterations on the coarsest mesh and one each of the finer mesh levels. With the exception of the coarsest level, a V-cycle with $\nu_1 = \nu_2 = 3$ is applied at each level to solve the linear system arising from the Newton process. It is considerably faster than nested SOR in achieving the same reduction in the residual. Thus with the exception of the initial 4 Newton cycles, it is a natural nonlinear analogue of the Full Multigrid method (FMG) [16, p.22]. Multigrid (non-nested) seems to perform the best since its timings are essentially the same as nested multigrid but it has greater residual reduction. The SOR ($n = 63$) iteration has the fastest real time but on a smaller problem where no virtual processing is involved.

Note that the *optimal* ω from the Laplacian model was used in the SOR iterations for the scalar problem. The experimental results showed that this was a good choice and gave the best convergence over any other choice. The stopping criteria used to terminate the inner iterations for each Newton step was $n_k = 1/(k+2)$.

Full Liquid Crystal Problem

Table 3 gives the results of the numerical simulations for the full systems model. The same test problem was used as in the case of the reduced model together with the appropriate Dirichlet boundary conditions. Only the size $n = 64$ problem was considered for this set of runs. The following set of parameter values was used: $L_1 = 10.0$, $L_2 = L_3 = 1.0$, $A = B = C = D = M = M' = 1.0$ and outside field parameters are set to zero. Results from both the 7-point and 9-point discretizations are given. Both point and block iterative methods were compared. The initial maximum absolute residuals for the 7-point and 9-point schemes are 8.2(4) and 8.95(4), respectively. The iterations were continued until the maximum residual was reduced by approximately a factor of 10^6. The initial maximum error is 1.0 since initial guesses of $q_i = 0.0,\ i = 1, \cdots, 5$ were used for the interior mesh points. The simulations were all done in single precision.

The SOR methods used 10 inner iterations for each Newton outer iteration. The stopping criteria used for the reduced model ($n_k = 1/(k+2)$) was too restrictive in some cases and caused convergence problems. Using 10 inner iterations avoided these problem areas. As before, the multigrid methods outperformed their SOR counterparts. The 7-point iterative scheme (point method) was competitive with the 9-point scheme for both multigrid and nested (multilevel) multigrid. This was not the case for the SOR methods. The 9-point scheme performed better for the one-level SOR case but did worse for the multilevel iteration. Block methods were not competitive for either multigrid or nested multigrid. The block method performed best for the single-level SOR iterations, and was also competitive in the nested case. The best algorithm for solving the test problem was again nested (multilevel) multigrid using the point iterative approach. The 9-point scheme performed marginally better than the 7-point, producing a slightly smaller residual, upon convergence, in about the same amount of real time. The pre-conditioned conjugate gradient methods were not implemented for the full model since they showed to be not competitive in the reduced model case [7].

Table 3. Timings for the Full Systems Liquid Crystal Problem on the Wavetracer DTC

	real	user	syst	max. residual	max. error	outer iter.
SOR (7p)	277.0	8.2	15.6	8.17(-2)	1.70(-5)	42(10 inner/out)
SOR (9p)	148.7	7.6	12.9	9.10(-2)	2.39(-5)	34(10 inner/out)
Block-SOR (9p)	106.3	6.4	6.5	7.95(-2)	1.27(-5)	17(10 inner/out)
Multigrid (7p)	47.2	2.0	1.6	6.55(-2)	5.36(-5)	4(1 V-cyc/out)
Multigrid (9p)	42.2	2.2	1.9	3.40(-2)	4.72(-5)	4(1 V-cyc/out)
Block-Multigrid (9p)	60.9	4.1	1.6	2.99(-2)	4.15(-5)	4(1 V-cyc/out)
Nested SOR (7p)	85.9	3.6	7.5	8.26(-2)	1.36(-5)	18(3,1,2,4,8)
Nested SOR (9p)	100.5	6.2	9.7	6.14(-2)	1.98(-5)	25(3,1,2,5,14)
Block-Nested SOR (9p)	105.0	7.4	7.6	8.23(-2)	1.05(-5)	18(3,1,2,4,8)
Nested Multigrid (7p)	37.4	2.4	2.8	8.44(-2)	6.88(-6)	8(4,1,1,1,1)
Nested Multigrid (9p)	36.3	2.8	2.8	4.62(-2)	3.86(-6)	8(4,1,1,1,1)
Block-Nested Multigrid (9p)	50.5	3.8	2.7	4.62(-2)	3.89(-6)	8(4,1,1,1,1)

CONCLUDING REMARKS

Multigrid methods work well as inner solvers for liquid crystal problems when implemented on SIMD computers with 2-D grid architectures. Multi-colored SOR methods are also effective, but due to the cost of inner products on such machines pre-conditioned conjugate gradient methods are not. The multigrid algorithms (one-level and multilevel) perform better than their SOR counterparts for the larger $n = 64$ problem.

Although the Wavetracer's mesh architecture fits the problem (discretization) well thereby making communications between nearest neighbors efficient, it is not as well suited for multigrid algorithms. This is due to the fact that the machine has a physical 2-D grid structure with 64 processors in each dimension. For multigrid and multilevel iterative schemes a grid size of 65×65 is required for an efficient implementation, because of the way the grid refinements are defined. So for an $n = 64$ size problem, the machine must to go into virtual processing mode, thus slowing down the execution time of the algorithm and increasing the storage overhead. One solution would be to generate grids that would not suffer this problem, but this involves considerably more complex coding, which would also increase execution time and storage overhead but not to the same extent as virtual processing. We emphasize that the multigrid implementation employed here is effectively the sequential version of the multigrid method. Thus on the coarsest mesh only 0.4% of the processors were active. Despite this disadvantage multigrid proved the fastest of the algorithms tested. We remark that although these methods worked well for the test problems, where the iteration matrix was positive definite symmetric, convergence problems can be expected, when the system becomes indefinite, due to the coarseness of the coarsest mesh. Use of a coarsest mesh with more points can be expected to remove this problem as well as improving the performance due to higher processor utilization. Further improvements in performance can be anticipated if parallelism were introduced using the method of [4, 5, 8, 9, 15].

REFERENCES

[1] R. E. Bank and D. J. Rose, Analysis of a Multilevel Iterative Method for Nonlinear Finite Element Equations, Math. Comp., 39, no. 160, pp. 453–465, 1982.

[2] William L. Briggs, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.

[3] P. J. Collings, *Liquid Crystals, Nature's Delicate Phase of Matter*, Princeton Science Library, Princeton University Press, New Jersey, 1990.

[4] J. E. Dendy, Jr., *Black Box Multigrid*, J. Comp. Phys., 48, pp.366–386, 1982.

[5] J. E. Dendy, Jr., M. P. Ida and J. M. Rutledge, *A Semicoarsening Multigrid Algorithm for SIMD Machines*, SIAM J. Sci. Statist. Comput., 13, no. 6, pp. 1460–1469, 1992.

[6] J. E. Dennis Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood, NJ, 1983.

[7] Paul A. Farrell, Arden Ruttan, and Reinhardt R. Zeller, *Finite Difference Minimization of the Landau-de Gennes Free Energy for Liquid Crystals in Rectangular Regions*, Computational and Applied Mathematics, I, C. Brezinski and U. Kulish eds., North–Holland, pp. 137–146, 1992.

[8] Paul O. Frederickson and Oliver A. McBryan, *Parallel Superconvergent Multigrid*, Lecture Notes in Pure. and Appl. Math., 110, Dekker, New York, 1988.

[9] Paul O. Frederickson and Oliver A. McBryan, *Normalized Convergence Rates for the PSMG Method*, SIAM J. Sci. Statist. Comput., 12, no. 1, pp. 221–229, 1991.

[10] Eugene C. Gartland, An Introduction to Liquid Crystals, SIAM News, 25, no. 6, 1992.

[11] Eugene C. Gartland, On Some Mathematical and Numerical Aspects of the Landau-de Gennes Minimization Problem for Liquid Crystals, Inst. for Computational Mathematics Preprint, Kent State University, Ohio.

[12] E. C. Gartland, Jr., P. Palffy-Muhoray, and R. S. Varga, Numerical Minimization of the Landau-de Gennes free energy: Defects in cylindrical capillaries, Mol. Cryst. Liq. Cryst., 199, pp. 429–452, 1991.

[13] W. Hackbusch, Multigrid Solution of Continuation Problems, in *Iterative Solution of Nonlinear Systems of Equations*, Lect. Notes in Math., 953, pp. 20–45, Springer, Berlin, 1982.

[14] Wolfgang Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1985.

[15] Oliver A. McBryan and Paul O. Frederickson, *Multigrid Methods on Parallel Computers a Survey of Recent Developments*, Impact Comput. Sci. Engrg., 3, no. 1, pp. 1–75, 1991.

[16] Stephen F. McCormick, ed., *Multigrid Methods*, SIAM, Philadelphia, 1987.

[17] James M. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1989.

[18] E. B. Priestley, P. J. Wojyowicz, and P. Sheng, eds., *Introduction to Liquid Crystals*. Plenum Press, New York, London, 1975.

[19] Reinhardt R. Zeller, *Ph. D. thesis*, Kent State University, in preparation.

S14-47
197146

AN ADAPTIVE MULTIGRID MODEL FOR HURRICANE TRACK PREDICTION

P. 8

N94-23687

Scott R. Fulton
Department of Mathematics and Computer Science
Clarkson University
Potsdam, NY

SUMMARY

This paper describes a simple numerical model for hurricane track prediction which uses a multigrid method to adapt the model resolution as the vortex moves. The model is based on the modified barotropic vorticity equation, discretized in space by conservative finite differences and in time by a Runge-Kutta scheme. A multigrid method is used to solve an elliptic problem for the streamfunction at each time step. Nonuniform resolution is obtained by superimposing uniform grids of different spatial extent; these grids move with the vortex as it moves. Preliminary numerical results indicate that the local mesh refinement allows accurate prediction of the hurricane track with substantially less computer time than required on a single uniform grid.

INTRODUCTION

Accurately predicting the track of a moving hurricane is a problem of great practical importance. One approach is to treat the problem as one in computational fluid dynamics, taking observed meteorological data as initial values for a numerical model. Many factors influence the accuracy of this approach, including the initial data (or lack thereof), the dynamical and physical processes included in the model, and the numerical scheme employed. While the relative importance of these three factors is a subject of considerable debate, in this paper we focus on the third.

Our premise is that predicting the track of a moving hurricane accurately requires resolving the flow field adequately on both the large scale surrounding the vortex and the small scale within the vortex itself. Since the spatial scales involved may differ by more than an order of magnitude, models using uniform resolution are inherently less efficient than what should be possible. Here, we use a simple dynamical model which has been used successfully by many authors (ref. 1, 2, 3), namely, the modified barotropic vorticity equation. However, rather than use a single uniform grid as in those studies, we investigate the use of adaptive multigrid techniques, with the goal of obtaining high accuracy at low computational cost. In the following sections we detail the formulation of the model, describe the mesh refinement scheme, and present some preliminary numerical results.

*Work supported by the National Science Foundation under Grant No. ATM-9118966

MODEL FORMULATION

Governing Equations

We formulate the model on a section of the sphere using a Mercator projection (true at latitude $\phi = \phi_c$). The model consists of the modified barotropic vorticity equation

$$\frac{\partial \zeta}{\partial t} + m^2 J(\psi, \zeta) + \beta m \frac{\partial \psi}{\partial x} = \nu m^2 \nabla^2 \zeta, \tag{1}$$

where the relative vorticity ζ and streamfunction ψ are related by

$$\left(m^2 \nabla^2 - \gamma^2\right) \psi = \zeta. \tag{2}$$

Here $\nabla^2 = \partial^2/\partial x^2 + \partial^2/\partial y^2$, $J(\psi, \zeta)$ is the Jacobian of (ψ, ζ) with respect to (x, y), $\beta = 2\Omega \cos\phi / a$ (with a and Ω the radius and rotation rate of the earth), and $m = \cos\phi_c / \cos\phi$ is the map factor. There are two quasi-physical parameters: the diffusion coefficient ν, and the parameter γ (inverse of the effective Rossby radius) which helps prevent retrogression of ultralong Rossby waves (ref. 4). We also consider versions on the f-plane ($m = 1$ and $\beta = 0$) and β-plane ($m = 1$ and $\beta = 2\Omega \cos\phi_c / a$). The model domain is a rectangle in x and y centered at $(x, y) = (0, 0)$, where $(\lambda, \phi) = (\lambda_c, \phi_c)$. At the boundaries we specify the streamfunction ψ (and thus the normal component of the velocity); where there is inflow, we also specify the vorticity ζ.

Space Discretization

On a single uniform rectangular grid Ω^h consisting of gridpoints (x_i, y_j) with mesh spacing h in x and y, we discretize (1) and (2) in space by finite differences as

$$\frac{d\zeta_{i,j}}{dt} + m_j^2 \mathcal{J}_{i,j}(\psi, \zeta) + \beta_j m_j \partial_x^{2h} \psi_{i,j} = \nu m_j^2 \nabla_{i,j}^2 \zeta_{i,j} \tag{3}$$

and

$$\left(m_j^2 \nabla_{i,j}^2 - \gamma^2\right) \psi_{i,j} = \zeta_{i,j}, \tag{4}$$

respectively. Here $\mathcal{J}_{i,j}(\psi, \zeta)$ is the discrete Jacobian of Arakawa (ref. 5), and $\partial_x^{2h} \psi_{i,j}$ and $\nabla_{i,j}^2 \psi_{i,j}$ are the $O(h^2)$ centered difference approximations to $\partial\psi/\partial x$ and the Laplacian operator, respectively. We apply (3) and (4) at the interior points. At boundary points where there is inflow, ζ is specified; otherwise, we predict ζ on the boundaries by applying an equation of the form (3) but using appropriate one-sided differences. It should be noted that using the Arakawa Jacobian is crucial here: the fact that it conserves discrete analogues of vorticity, enstrophy (mean square vorticity), and kinetic energy implies that the model is not subject to nonlinear computational instability.

To write the space-discretized equations in a more compact form, we collect the values $\psi_{i,j}$ and $\zeta_{i,j}$ into grid functions ψ^h and ζ^h, respectively, defined on the grid Ω^h. We can then write (3) and (4) as

$$\frac{d\zeta^h}{dt} = F^h(\psi^h, \zeta^h) \tag{5}$$

and

$$G^h(\psi^h) = \zeta^h, \tag{6}$$

where the operators F^h and G^h express the space discretization described above.

Time Discretization

To discretize (5) and (6) in time we use the classical fourth-order Runge-Kutta (RK4) scheme. To describe it, we specify a time step $\Delta t > 0$ and introduce time levels $t_k = k\Delta t$ for $k = 0, \ldots$. Suppressing the superscript h for simplicity, we now use the superscript k to denote values at time level k, e.g., $\psi^k \approx \psi^h(t_k)$. With this notation, the RK4 scheme can be written as

$$\begin{aligned}
\frac{\tilde{\zeta}^{k+\frac{1}{2}} - \zeta^k}{\frac{1}{2}\Delta t} &= F^k &&:= F(\psi^k, \zeta^k), && G(\tilde{\psi}^{k+\frac{1}{2}}) = \tilde{\zeta}^{k+\frac{1}{2}}, \\
\frac{\zeta^{k+\frac{1}{2}} - \zeta^k}{\frac{1}{2}\Delta t} &= \tilde{F}^{k+\frac{1}{2}} &&:= F(\tilde{\psi}^{k+\frac{1}{2}}, \tilde{\zeta}^{k+\frac{1}{2}}), && G(\psi^{k+\frac{1}{2}}) = \zeta^{k+\frac{1}{2}}, \\
\frac{\tilde{\zeta}^{k+1} - \zeta^k}{\Delta t} &= F^{k+\frac{1}{2}} &&:= F(\psi^{k+\frac{1}{2}}, \zeta^{k+\frac{1}{2}}), && G(\tilde{\psi}^{k+1}) = \tilde{\zeta}^{k+1}, \\
\frac{\zeta^{k+1} - \zeta^k}{\Delta t} &= \bar{F}^{k+1}, && && G(\psi^{k+1}) = \zeta^{k+1},
\end{aligned} \tag{7}$$

where

$$\bar{F}^{k+1} = \frac{1}{6}\left(F^k + 2\tilde{F}^{k+\frac{1}{2}} + 2F^{k+\frac{1}{2}} + \tilde{F}^{k+1}\right). \tag{8}$$

Thus, to execute a single time step $t_k \to t_{k+1}$, we perform the four stages indicated in (7); each of these stages consists of computing F based on known values of ψ and ζ, predicting a new vorticity ζ, and solving the diagnostic equation for the corresponding streamfunction ψ.

Although it requires four times as much work (per time step) as the second-order Adams-Bashforth scheme commonly used in such models, this RK4 scheme has several advantages. First, it allows time steps at least four times as large, so in fact it is more efficient. Second, it is more accurate, so time discretization errors are less likely to distort the conservation properties of the Arakawa Jacobian. Finally, since it is a one-step scheme, it has no computational modes and needs no other method for the initial time step.

Multigrid Solution

To solve the diagnostic equation at each stage for the streamfunction ψ, we use a multigrid method. For the relaxation scheme we use a point Gauss-Seidel method formulated as follows. The discrete (interior) equation (4) can be written as

$$(L\psi)_{i,j} = \frac{1}{h^2}\left(\sigma_j \psi_{i,j} - S_{i,j}\right) = -\frac{\zeta_{i,j}}{m_j^2} = F_{i,j}, \tag{9}$$

where

$$S_{i,j} := \psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j-1} + \psi_{i,j+1} \tag{10}$$

is the sum of the neighboring values of ψ and

$$\sigma_j := 4 + \frac{\gamma^2 h^2}{m_j^2} \tag{11}$$

is the diagonal term of the discrete Helmholtz operator. Given an approximate solution $\tilde{\psi}$ of (9), we relax at a point (i, j) by changing the value there to satisfy the corresponding equation (9); this results in the new value

$$\bar{\psi}_{i,j} = \frac{h^2 F_{i,j} + \tilde{S}_{i,j}}{\sigma_j}, \tag{12}$$

where $\tilde{S}_{i,j}$ is defined using the current surrounding values in (10). The corresponding residual (if needed) is given by

$$r_{i,j} := F_{i,j} - \frac{1}{h^2}\left(\sigma_j \tilde{\psi}_{i,j} - \tilde{S}_{i,j}\right) = \frac{\sigma_j}{h^2}\left(\bar{\psi}_{i,j} - \tilde{\psi}_{i,j}\right). \tag{13}$$

We use this relaxation (with red-black ordering) as a smoother in a multigrid method, using half-injection for the fine-to-coarse transfer of residuals and bilinear interpolation for the coarse-to-fine transfer of corrections. For the control algorithm we use repeated V(1,1)-cycles.

LOCAL MESH REFINEMENT

Given the premise that the flow near the center of the vortex requires much higher resolution than the flow surrounding the vortex, we now consider how to provide such variable resolution. Our basic method is essentially that of (ref. 6), constructing nonuniform resolution by superimposing uniform grids of varying spatial extent. Since all calculations are carried out on the uniform grids, programming remains relatively easy.

To illustrate the method, let us consider first the case of two grids: a coarse grid Ω^{2h} covering the whole domain Ω, and a fine grid Ω^h which covers only a portion of the domain (i.e., enclosing the vortex). We assume that the boundaries of the fine grid coincide with coarse grid lines. The model variables ζ and ψ are carried on both the coarse and fine grids (denoted by ζ^{2h}, ψ^{2h} and ζ^h, ψ^h, respectively). Noting that the coarse grid allows time steps twice as large as those on the fine grid, we use the following basic procedure for stepping the model from time t_k to t_{k+1}:

1. Execute one time step of length Δt on the coarse grid to produce $\zeta^{2h,k+1}$, $\psi^{2h,k+1}$;

2. Execute two time steps of length $\Delta t/2$ on the fine grid to produce $\zeta^{h,k+1}$, $\psi^{h,k+1}$, using boundary values for ψ interpolated from the coarse grid (in space and time);

3. Copy the fine-grid solution to the coarse grid at points common to both.

Several points deserve mention here. First, in solving the implicit problem for ψ on either grid, we use the multigrid method outlined above. This introduces additional coarse grids, e.g., a grid with mesh spacing $2h$ covering only the region of the local fine grid Ω^h. In fact, the "underlying" part of the coarse grid Ω^{2h} could be used for this; however, the resulting complications of preserving interface values (for fine-grid boundary values) and restricting relaxation to only part of Ω^{2h} seem too high a price to pay for the relatively small savings in storage which would be achieved. Second, after completing the above three steps, the resulting solution on the composite grid $\Omega^{\underline{h}} = \Omega^h \cup \Omega^{2h}$ could be further refined by applying a composite-grid discretization of the governing equations; this FAC (Fast Adaptive Composite grid) method and several variants are described in (ref. 7), and will be explored in future work. Finally, the above approach generalizes immediately to more than two grids.

For the initial work reported here, we have made the following simplifying assumptions. First, we require the grids to be rectangular and strictly nested (i.e., any fine grid is contained wholly within the interior of the next coarser grid), with one grid per level (i.e., the refinement occurs in one region only, surrounding the vortex). Second, we use a constant mesh ratio of two (i.e., the mesh spacing h on any grid is twice that of the next finer grid, if any). Finally, we will specify the number of grids and their sizes in advance but allow them to move following the vortex as the solution is computed.

Since the problem to be solved has an easily identifiable region of interest surrounding the vortex, we take the following simple approach to moving the grids. First, we locate the vortex center on the finest grid. Then for each grid in turn, from the next-to-coarsest to the finest, we decide whether or not to move the grid. This decision is based on the distance of the vortex center from the center of the grid: if it is more than a specified fraction α of the distance L to the boundary, we move the grid. The move is calculated so as to "overshoot" a bit, i.e., aiming to put the vortex center beyond the (new) grid center by a specified fraction δ of the distance to the grid boundary. Note that care must be taken at this stage to ensure the strict nesting of grids assumed above. Finally, the grid is moved by shifting the values which remain on the grid and filling in the rest by interpolation from the next coarser grid. For the results presented here, we check for possible grid moves after each time step on the coarsest grid, and use the parameters $\alpha = 0.4$ and $\delta = 0.2$.

To locate the vortex center (needed both for moving the grids as described above and for determining the vortex track), we first locate the point of maximum vorticity on the finest grid. We then interpolate the vorticity at that point and its nearest neighbors in x and y (five points total) by a quadratic function, and define the vortex center to be the location of the maximum of that quadratic.

RESULTS

The initial conditions for the test problem consist of an axisymmetric vortex superimposed on an environmental flow, as considered in (ref. 1). The environmental flow is given by

$$\bar{\psi}(y) = \left(\frac{\bar{u}_0 L}{2\pi}\right) \cos\left(\frac{2\pi y}{L}\right), \tag{14}$$

which corresponds to the zonal current

$$\bar{u}(y) = -\frac{d\bar{\psi}}{dy} = \bar{u}_0 \sin\left(\frac{2\pi y}{L}\right). \tag{15}$$

The tangential wind in the initial vortex is given by

$$V(r) = 2V_m \left(\frac{r}{r_m}\right) \frac{\exp[-a(r/r_m)^b]}{1 + (r/r_m)^2}, \tag{16}$$

where $r = [(x - x_0)^2 + (y - y_0)^2]^{1/2}$ is the radial distance from the vortex center (x_0, y_0). Note that V has the approximate maximum value V_m near $r = r_m$ (exact when $a = 0$); the exponential factor is included to make V vanish quickly for large r. The vorticity corresponding to (16) is

$$\zeta(r) = \frac{\partial(rV)}{r\partial r} = \frac{V}{r}\left[\frac{2}{1 + (r/r_m)^2} - ab\left(\frac{r}{r_m}\right)^b\right]. \tag{17}$$

We will use the following parameter values: $\bar{u}_0 = 10\,\mathrm{ms}^{-1}$ and $L = 4000\,\mathrm{km}$ for the environmental flow, and $V_m = 30\,\mathrm{ms}^{-1}$, $r_m = 80\,\mathrm{km}$, $a = 10^{-6}$, and $b = 6$ for the initial vortex. The computational domain is a square of side length 4000 km on a β-plane, using β for the latitude 20° N; the vortex is initially centered at $x_0 = 750\,\mathrm{km}$ and $y_0 = -750\,\mathrm{km}$. The model was run from $t = 0$ to $t = 72\,\mathrm{hr}$; for simplicity we have set $\nu = 0$ and $\gamma = 0$ here.

To establish a standard for comparison, we ran the model with high resolution (384 × 384 grid with spacing $h = 10.42\,\mathrm{km}$ and time step 10 s). We then ran the model with a variety of grid configurations (using up to four grids) and compared the vortex track to that of the reference run. Table I summarizes these results, with the runs listed in order of increasing execution time (on a SUN SPARCstation2). All of the cases in this table use only square grids, with $N_x = N_y = N$. The forecast error is defined as the distance between the predicted vortex location at a given time and that in the reference run. These results show that the local refinement process has the potential to substantially reduce the execution time required to achieve a given accuracy. For example, a single grid with $h = 31.25\,\mathrm{km}$ (run 6) achieves errors on the order of 10–20 km; with local refinement (run 2) comparable accuracy is obtained with only about 36% as much computer time. Similarly, a single grid with $h = 20.83\,\mathrm{km}$ (run 8) achieves errors on the order of 1–5 km; with local refinement (run 7) comparable accuracy is obtained with only about 42% as much computer time. In fact, run 7 with local refinement achieved about the same accuracy as did the single-grid run with $h = 15.625\,\mathrm{km}$ (run 9) but with only about 18% of the computer time. In addition, the solution fields produced with local refinement (run 7) are smooth, as shown in Figures 1–5, with no indication of any problem due to the change of resolution at the grid interfaces.

CONCLUDING REMARKS

The preliminary results reported here show that adaptive multigrid techniques can substantially reduce the computer time required to make accurate hurricane track forecasts. In addition to ongoing testing of the existing model, we plan to investigate the following possible improvements. First, we plan to include the FAC method as discussed above. This should have the advantage of more precise conservation of vorticity, enstrophy, and kinetic energy at the grid interfaces. Second, we plan to construct a fully adaptive version of the model by using the Full Approximation Scheme (FAS) to produce estimates of the local truncation error to be used in an automatic grid refinement scheme (as proposed in ref. 8). Finally, we plan to test the model using real data, and compare its performance to that of models currently in operational use.

REFERENCES

1. DeMaria, M.: Tropical cyclone track prediction with a barotropic model. *Mon. Wea. Rev.*, vol. 113, 1985, pp. 1199–1210.

2. DeMaria, M.: Tropical cyclone track prediction with a barotropic spectral model. *Mon. Wea. Rev.*, vol. 115, 1987, pp. 2346–2357.

3. Smith, R. K., W. Ulrich, and G. Dietachmayer: A numerical study of tropical cyclone motion using a barotropic model. Part I: The role of vortex asymmetries. *Quart. J. Roy. Meteor. Soc.*, vol. 116, 1990, pp. 337-362.

4. Holton, J. R.: *An Introduction to Dynamic Meteorology*, second edition. Academic Press, 1979.

5. Arakawa, A.: Computational design for long-term numerical integration of the equations of fluid motion: Two-dimensional incompressible flow. Part I. *J. Comp. Phys.*, vol. 1, 1966, pp 119–143.

6. Berger, M., and J. Oliger: Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Phys.*, vol. 53, 1984, pp. 484–512.

7. McCormick, S. F.: *Multilevel Adaptive Methods for Partial Differential Equations.* SIAM, 1989.

8. Brandt, A.: Multi-level adaptive solutions to boundary-value problems. *Math. Comp.*, vol. 31, 1977, pp. 333–390.

Table I. Results of Model Runs

Run	Grid size(s) N	h (km)	Δt (min)	Forecast error (km) at: 24 hr	48 hr	72 hr	Execution time (sec)
1	64	62.50	60	110	143	47	170
2	64	62.50	60	11	8	17	504
	64	31.25	30				
3	96	41.67	30	53	12	25	799
4	32	125.0	120	14	24	39	916
	32	62.50	60				
	48	31.25	30				
	64	15.62	15				
5	64	62.50	60	1	6	10	1,174
	64	31.25	30				
	64	15.62	15				
6	128	31.25	30	11	8	19	1,409
7	64	62.50	60	1	5	5	2,047
	64	31.25	30				
	96	15.62	15				
8	192	20.83	20	1	3	5	4,860
9	256	15.62	15	2	3	4	11,405
10	384	10.42	10	–	–	–	41,716

315-64
197147

RELAXATION SCHEMES FOR CHEBYSHEV SPECTRAL MULTIGRID METHODS*

N94-23688

Yimin Kang and Scott R. Fulton
Department of Mathematics and Computer Science
Clarkson University
Potsdam, NY

SUMMARY

Two relaxation schemes for Chebyshev spectral multigrid methods are presented for elliptic equations with Dirichlet boundary conditions. The first scheme is a pointwise-preconditioned Richardson relaxation scheme and the second is a line relaxation scheme. The line relaxation scheme provides an efficient and relatively simple approach for solving two-dimensional spectral equations. Numerical examples and comparisons with other methods are given.

INTRODUCTION

For limited-area problems with general (non-periodic) boundary conditions, Chebyshev spectral methods give exponential convergence for smooth solutions. However, except in some very simple cases (e.g., one-dimensional constant-coefficient problems), Chebyshev approximations usually lead to full linear systems which cannot be solved efficiently by direct methods, and iterative methods must be used. Unfortunately, designing efficient iterative methods for discrete spectral equations has proven difficult, especially for problems with non-constant coefficients (ref. 1). Perhaps the most promising technique to date for solving spectral discretizations of elliptic problems is the spectral multigrid method (ref. 2, 3). However, the best relaxation schemes known today are complicated to apply. In this paper we introduce two simpler relaxation schemes and investigate their performance.

As prototype problems we consider one- and two-dimensional elliptic equations with Dirichlet boundary conditions on simple geometric domains. In one dimension we consider

$$\begin{aligned} -u''(x) &= f(x), & |x| &< 1, \\ u(+1) &= a, & u(-1) &= b. \end{aligned} \tag{1}$$

The two-dimensional prototype problem is

$$\begin{aligned} -\Delta u(x,y) &= f(x,y), & |x|,|y| &< 1, \\ u(x,y) &= g(x,y), & |x| &= 1, |y| = 1. \end{aligned} \tag{2}$$

We discretize these problems by Chebyshev collocation. For example, for the two-dimensional problem (2), the solution $u(x,y)$ is approximated by a set of discrete values $\bar{u}_{j,k}$ on the Chebyshev

*Work supported by the National Science Foundation under Grant No. ATM-9118966.

grid $\{(\bar{x}_j, \bar{y}_k) = (\cos(j\pi/N_x), \cos(k\pi/N_y)) \,|\, 0 \le j \le N_x, 0 \le k \le N_y\}$, with the requirement that problem (2) be satisfied on this grid, i.e.,

$$\begin{aligned} -(\bar{u}^{(xx)}_{j,k} + \bar{u}^{(yy)}_{j,k}) &= f(\bar{x}_j, \bar{y}_k), & 1 < j < N_x, 0 < k < N_y \\ \bar{u}_{j,k} &= g(\bar{x}_j, \bar{y}_k), & j = 0, j = N_x, k = 0, k = N_y \end{aligned} \tag{3}$$

where $\bar{u}^{(xx)}_{j,k}$ and $\bar{u}^{(yy)}_{j,k}$ are values of the second-order derivatives of the Chebyshev approximation $\sum_{m=0}^{N_x} \sum_{n=0}^{N_y} \hat{u}_{mn} T_m(x) T_n(y)$ to $u(x, y)$ on the Chebyshev grid. For simplicity, we will assume here that $N_x = N_y = N$; however, the codes described in this paper do not require this.

The discrete problem (3) can be expressed in form of a linear system

$$A\bar{U} = \bar{F} \tag{4}$$

Unfortunately, the matrix A, formulated by Chebyshev collocation approximations, is full and non-symmetric. For two-dimensional problems, direct methods (like Gaussian elimination) would require $O(N^6)$ operations for factorization and $O(N^4)$ for the subsequent solution, which is far too much work to be practical. Thus, iterative methods must be used.

THE POINTWISE PRECONDITIONED RICHARDSON RELAXATION SCHEME

The most efficient method available today for solving (4) and its generalizations to other elliptic problems is the spectral multigrid method of Zang et al. (ref. 2, 3), which employs finite-difference preconditioned Richardson iteration as the relaxation scheme in a multigrid context. Preconditioned Richardson relaxation for (4) takes the form

$$V \longleftarrow V + \omega H(F - AV), \tag{5}$$

where V is the current approximation to $\bar{U}$, ω is a relaxation parameter, and H is the preconditioner. The criteria for choosing a preconditioner H are:

- H should give fast multigrid convergence,
- H should be easy and cheap to generate or apply.

The finite-difference preconditioning of Zang et al. (ref. 2, 3) gives fast convergence, but applying it requires solving (or nearly solving) a finite-difference discretization on the nonuniform Chebyshev grid. This procedure is complicated and expensive. Are there alternatives which are simpler and still effective? Achi Brandt (personal communication, 1983) has suggested that pointwise preconditioning based on the (variable) Chebyshev mesh spacing might work well. In this section, we investigate the performance of this simple preconditioner when applied to the problem (4).

The One-Dimensional Case

Formulation

As an analogue of the Gauss-Seidel relaxation for a finite-difference method, the pointwise preconditioning for the Chebyshev discretization takes the form

$$v_j \longleftarrow v_j + \omega \frac{h_j^2}{2} r_j, \tag{6}$$

where $h_j = (\bar{x}_{j-1} - \bar{x}_{j+1})/2$ is the effective grid size at the point $\bar{x}_j$, r_j is the the residual $R = F - AV$ at $\bar{x}_j$, and ω is a relaxation parameter to be chosen to accelerate the convergence. Note that (6) is equivalent to choosing the preconditioning matrix H in (5) as a diagonal matrix

$$H = \operatorname{diag}\left(1, \frac{h_1^2}{2}, \ldots, \frac{h_{N-1}^2}{2}, 1\right). \tag{7}$$

Analysis

The evolution of the error $E = V - \bar{U}$ in the Richardson relaxation (5) is described by

$$E \longleftarrow (I - \omega HA)E. \tag{8}$$

Therefore, the convergence factor for (5) on a single grid is

$$\sigma_{SG} = \rho(I - \omega HA),$$

where ρ denotes the spectral radius. Likewise, the multigrid smoothing factor for (5), when used as a smoother in a multigrid method (e.g., ref. 4), is

$$\bar{\mu} = \rho(G(I - \omega HA)), \tag{9}$$

where G represents the perfect coarse-grid correction, i.e., set all low modes of the error to zero.

For the simple preconditioning (7), our numerical computations show that the eigenvalues of the matrix HA are all positive real numbers. The maximum eigenvalue is $\lambda_{\max} \approx 5.0$, the middle is $\lambda_{\text{mid}} \approx 1.5$, and the minimum is $\lambda_{\min} \approx O(N^{-2})$. The formulas of Zang et. al. (ref. 2, 3) then give a good approximation to the optimal ω and $\bar{\mu}$, namely,

$$\omega \approx \frac{2}{\lambda_{\max} + \lambda_{\text{mid}}} \approx 0.325, \qquad \bar{\mu} \approx \frac{\lambda_{\max} - \lambda_{\text{mid}}}{\lambda_{\max} + \lambda_{\text{mid}}} \approx 0.6. \tag{10}$$

Indeed, computing the smoothing factor directly from (9) using $\omega = 0.325$, we find that $\bar{\mu} \leq 0.6$ for all $N \leq 512$.

To take into account the effects of grid transfers (omitted in the smoothing analysis above), we use the following two-grid analysis. The evolution of the error E in one two-grid $V(n_1, n_2)$-cycle (where n_1 and n_2 specify the number of relaxation sweeps before and after the coarse-grid correction, respectively) is described by the matrix

$$T = (I - \omega H A_f)^{n_2}(I - P A_c^{-1} R A)(I - \omega H A_f)^{n_1}. \tag{11}$$

Here, R represents the fine-to-coarse grid transfer (we use injection), P represents the coarse-to-fine grid transfer (we use Chebyshev interpolation), and A_f and A_c represent the discrete operator matrix in (4) on the fine and coarse grids, respectively. Note that (11) assumes that the coarse-grid problem is solved exactly.

We computed the two-grid convergence factor $\sigma_{TG} = \rho(T)$ for $N \leq 512$ using different values of ω, and the numerical results show that $\omega = 0.325$ again gives the optimal convergence factor (or very close to it). Using that constant value, we find that the smoothing factor per sweep $\mu_s = (\sigma_{TG})^{1/(n_1+n_2)}$ satisfies

$$0.5 \leq \mu_s \leq 0.6$$

for all $N \leq 512$. A similar analysis for the one-dimensional Helmholtz problem

$$\lambda\, u(x) - u''(x) = f(x) \tag{12}$$

shows that with various choices of λ and boundary conditions (Dirichlet, Neumann and mixed), an appropriate pointwise preconditioner also yields the smoothing factor per sweep $\mu_s \leq 0.6$.

We have developed FORTRAN-77 routines to implement the Chebyshev multigrid method using the pointwise preconditioner as described above. The code has been used to solve the problem (12) with various choices of $u(x)$, λ, and boundary conditions. The observed convergence factor per sweep μ_s is smaller than 0.60 for all cases tested, in agreement with the analysis presented above.

The Two-Dimensional Case

Formulation

We note that Gauss-Seidel relaxation for the second-order centered finite difference approximation to (2) can be written as

$$u_{j,k} \longleftarrow u_{j,k} + \frac{h^2}{4} r_{j,k},$$

where $r_{j,k}$ is the finite-difference residual. A natural analogue for the Chebyshev collocation discretization (3) is

$$\bar{v}_{j,k} \longleftarrow \bar{v}_{j,k} + \omega \left(\frac{1}{2/h_j^2 + 2/h_k^2} \right) \bar{r}_{j,k}, \tag{13}$$

where h_j and h_k are the grid sizes at the point $(\bar{x}_j, \bar{y}_k)$, $\bar{r}_{j,k} := \bar{f}_{j,k} - [-(\bar{v}_{j,k}^{(xx)} + \bar{v}_{j,k}^{(yy)})]$ is the residual of Chebyshev discretization, and ω is a relaxation parameter to be chosen to accelerate the convergence. Clearly, the iteration (13) is a special case of the Richardson iteration (5), with a diagonal preconditioner H with diagonal entries $(H)_{jk,jk} = (2/h_j^2 + 2/h_k^2)^{-1}$. This preconditioner is easy and fast to apply. Does it gives a fast convergence? Unfortunately, the following analysis shows that the answer is no.

Analysis

Computational results indicate that the eigenvalues of the matrix HA are all positive real numbers. Again, good approximations to the optimal ω and $\bar{\mu}$ can be obtained by

$$\omega \approx \frac{2}{\lambda_{\max} + \lambda_{\text{qua}}}, \qquad \bar{\mu} \approx \frac{\lambda_{\max} - \lambda_{\text{qua}}}{\lambda_{\max} + \lambda_{\text{qua}}}, \tag{14}$$

where $\lambda_{\max}$ is the maximum eigenvalue and λ_{qua} is the quarter eigenvalue (ref. 1). More precise values of the optimal ω and $\bar{\mu}$, can be obtained by actually computing the spectral radius $\rho(G(I - \omega HA))$ for different choices of ω and comparing the results. For $N \leq 32$, the eigenvalues $\lambda_{\max}$ and λ_{qua}, ω and $\bar{\mu}$ computed by (14) and the optimal ω and $\bar{\mu}$ are listed in Table I. Since $\bar{\mu}$ is large and increases with N, these results suggest that the pointwise preconditioner (13) will not be a good multigrid smoother.

Table I also lists the two-grid smoothing factors per sweep $\mu_s = (\rho(T))^{1/(n_1+n_2)}$ computed from the matrices in (11) for $N \leq 32$ using $\omega = 0.36$. These results again show that the pointwise preconditioning (13) does not give fast convergence.

We have implemented the pointwise preconditioning (13) in a multigrid solver written in Fortran 77. Computational results from a number of test cases confirm the above analysis: we conclude that the pointwise preconditioning does not give fast convergence.

Table I. Multigrid Analysis of Two-Dimensional Pointwise Preconditioning

	Eigenvalues of HA		By (14)		By computation		
N	$\lambda_{\max}$	λ_{qua}	ω	$\bar{\mu}$	ω_{opt}	$\bar{\mu}$	μ_s
4	3.00	1.83	0.41	0.24	0.35	0.28	0.51
8	4.10	1.26	0.37	0.53	0.35	0.52	0.68
16	4.57	0.95	0.36	0.66	0.36	0.75	0.80
32	4.76	0.78	0.36	0.72	0.36	0.82	0.88

THE LINE RELAXATION METHOD

The poor performance of pointwise preconditioning in two dimensions can be understood in terms of the anisotropy introduced by the nonuniform Chebyshev collocation grid. Since the mesh spacing varies with x and y, at any given point (x, y) the coupling in the discrete operator in (3) may be stronger in x or in y. In finite-difference multigrid methods, point relaxation performs poorly in such anisotropic cases, and the cure is to use alternating direction line relaxation. Thus, it is reasonable to try an analogous approach for the Chebyshev discretization.

Formulation

To formulate the line relaxation method, we express the discrete problem (3) in the matrix form

$$(\mathcal{H} + \mathcal{V})\bar{U} = \bar{F}, \tag{15}$$

where $\mathcal{H}$ and $\mathcal{V}$ correspond to the horizontal part $(-\partial^2/\partial x^2)$ and vertical part $(-\partial^2/\partial y^2)$ of the Laplacian operator, respectively. Starting from an approximation V^{old} to the solution $\bar{U}$, one sweep of (alternating direction) line relaxation based on (15) consists of the following two parts:

1. Sweep along the x-direction. On each grid line parallel to x-axis, use the values of V^{old} except those on the current line, and solve for values on the current line by solving (15). This can be expressed in the matrix form as

   $$(\mathcal{H} + \mathcal{V}_d)V^{\text{mid}} = \bar{F} - \mathcal{V}_o V^{\text{old}}, \tag{16}$$

   where $\mathcal{V}_d$ and $\mathcal{V}_o$ denote the diagonal and off-diagonal parts of the matrix $\mathcal{V}$, respectively. Note that the entries of $\mathcal{V}_d$ are known (ref. 1) and $\mathcal{V}_d$ is a constant on each grid line parallel to the x-axis. Thus, the system (16) can be decoupled into $(N - 1)$ one-dimensional discrete problems, each of which is a Chebyshev collocation approximation to a Helmholtz equation on an interior grid line parallel to x-axis; the x-directional sweep consists of solving these equations.

2. Sweep along the y-direction. The y-direction sweep is basically the same as the x-direction sweep except that we now work on grid lines that are parallel to y-axis and use values of V^{mid} instead of V^{old}. The equation we need to solve is

   $$(\mathcal{H}_d + \mathcal{V})V^{\text{new}} = \bar{F} - \mathcal{H}_o V^{\text{mid}}, \tag{17}$$

   where $\mathcal{H}_d$ and $\mathcal{H}_o$ are the diagonal and off-diagonal parts of $\mathcal{H}$. As in the x-direction sweep, the two-dimensional problem (17) is solved by solving $(N-1)$ one-dimensional Helmholtz equations.

It turns out that as it stands, the line relaxation (16)–(17) is not a good multigrid smoother; however, this can be fixed as follows. Let $C^{\text{mid}} = V^{\text{mid}} - V^{\text{old}}$ and $C^{\text{new}} = V^{\text{new}} - V^{\text{mid}}$ denote the corrections for V^{old} and V^{mid}, and $R^{\text{old}} = \bar{F} - AV^{\text{old}}$ and $R^{\text{mid}} = \bar{F} - AV^{\text{mid}}$ denote the residuals of V^{old} and V^{mid}, respectively. Rewriting equations (16) and (17) as correction equations and introducing a relaxation parameter ω (to be determined by analysis to accelerate the convergence), we obtain

$$(\mathcal{H} + \mathcal{V}_d)C^{\text{mid}} = \omega R^{\text{old}}, \qquad (\mathcal{H}_d + \mathcal{V})C^{\text{new}} = \omega R^{\text{mid}} \tag{18}$$

We refer to (18) as the collocation version of the line relaxation method.

It is not practical to implement the collocation version because there are no fast solvers available for the collocation approximations, even for one-dimensional problems. However, in the multigrid context, a relaxation scheme functions as a smoother rather than a solver: instead of solving each problem exactly, we only need to smooth out the error, i.e., reduce high modes in the error. Therefore, it is reasonable to replace the one-dimensional problems in (18) by approximate versions which can be solved efficiently. We consider two alternatives as follows.

In the first, we replace the collocation discretizations of the one-dimensional Helmholtz equations in (18) by tau discretizations. Tau approximations have the same exponential convergence as collocation method, but can be solved directly in $O(N \log N)$ operations. This leads to the tau version of the line relaxation method, and the total work of one x or y-direction sweep is $O(N^2 \log N)$. As we will see below, this tau version turns out to be an efficient multigrid smoother.

In the second, we replace the collocation discretizations of the one-dimensional Helmholtz equations in (18) by finite-difference discretizations. This leads to the finite-difference version of the line relaxation method, which has two obvious advantages over the tau version. First, it is faster because it eliminates the transforms required in tau version, thus reducing the operation count for solving each one-dimensional problem from $O(N \log N)$ to $O(N)$. Second, it can be extended to solve more generalized problems, e.g., problems with variable coefficients. As we will see below, this finite-difference version also turns out to be an efficient multigrid smoother, even in the case of variable coefficients.

Analysis

As in the case of the pointwise preconditioned Richardson relaxation, we can analyze the performance of the line relaxation methods described above by computing the eigenvalues of the corresponding interation matrices. Because the tau version cannot be expressed in matrix form like (18), we will only do the analysis for the collocation and finite-difference versions. Note that the tau and collocation versions are nearly the same, so the analysis for collocation version should give a good prediction for the performance of the tau version. In this section, we will give details of the analysis for finite-difference version and only list results for collocation version.

Smoothing Analysis

For the finite-difference version of the line relaxation iteration, the error evolution is described by

$$E^{\text{mid}} \longleftarrow [I - \omega(\mathcal{H}^{fd} + \mathcal{V}_d)^{-1}(\mathcal{H} + \mathcal{V})]E^{\text{old}}, \tag{19}$$

$$E^{\text{new}} \longleftarrow [I - \omega(\mathcal{H}_d + \mathcal{V}^{fd})^{-1}(\mathcal{H} + \mathcal{V})]E^{\text{mid}}. \tag{20}$$

where $\mathcal{H}^{fd}$ and $\mathcal{V}^{fd}$ are the finite-difference analogues of the collocation discretization matrices $\mathcal{H}$ and $\mathcal{V}$, respectively. Therefore, the error evolution matrix for one relaxation is

$$S = [I - \omega(\mathcal{H}_d + \mathcal{V}^{fd})^{-1}(\mathcal{H} + \mathcal{V})][I - \omega(\mathcal{H}^{fd} + \mathcal{V}_d)^{-1}(\mathcal{H} + \mathcal{V})]. \tag{21}$$

The matrices $S_{\mathcal{H}} = (\mathcal{H}^{fd} + \mathcal{V}_d)^{-1}(\mathcal{H} + \mathcal{V})$ and $S_{\mathcal{V}} = (\mathcal{H}_d + \mathcal{V}^{fd})^{-1}(\mathcal{H} + \mathcal{V})$ have the same eigenvalues (since x and y can be interchanged in the Laplacian operator), so we can focus on just the x-direction sweep (19). The eigenvalues of $S_{\mathcal{H}}$ are all positive real numbers, so we can use formulas (14) to obtain approximate values of ω and $\bar{\mu}$ (squaring $\bar{\mu}$ to represent the effect of both the x and y sweeps). These values are listed in Table II for $N \leq 32$, along with the optimal relaxation parameter ω and corresponding multigrid smoothing factor $\bar{\mu} = \rho(GS)$ computed directly. These results suggest that for large values of truncation number N, $\omega_{opt} \approx 0.6$ and $\bar{\mu} \leq 0.5$, independent of the grid size. Corresponding results for the collocation version are listed in Table III.

Multigrid Analysis

For a multigrid $V(n_1, n_2)$-cycle, if we use zeros as initial guesses on all coarse grids (which is a natural choice because the coarse-grid solution is a correction to the solution on the next finer grid), then we can write out the error evolution matrix explicitly as

$$M = S^{n_2}\,[I - P\,G\,R\,(\mathcal{H} + \mathcal{V})]\,S^{n_1}. \tag{22}$$

This represents a procedure of n_1 pre-relaxations (S^{n_1}) followed by a coarse-grid-correction ($I - P\,G\,R\,(\mathcal{H} + \mathcal{V})$) and then n_2 post-relaxations (S^{n_2}). The matrix S is the error evolution matrix of one relaxation on the finest grid defined in (21). The central part $I - PGR(\mathcal{H} + \mathcal{V})$ represents the coarse-grid-correction, where R represents the fine-to-coarse grid transfer (we use injection) and P represents the coarse-to-fine grid transfer (we use Chebyshev interpolation). The matrix G is defined on the next coarser grid as follows: on the coarsest grid, $G = (\mathcal{H} + \mathcal{V})^{-1}$ (which means the coarsest grid problem is solved exactly); otherwise,

$$G = [I - M] * (\mathcal{H} + \mathcal{V})^{-1}, \tag{23}$$

which represents a multigrid solution procedure on that grid. Note that (23) is actually a recursive definition, since the matrix M in (23) includes another matrix G on the next coarser grid.

Tables II and III also list computed values of smoothing factor per sweep $\mu_s = (\rho(M))^{1/(n_1+n_2)}$ for the case $\omega = 0.6$, $n_1 = 2$, and $n_2 = 1$. These results suggest that the smoothing factor of the line relaxation method is less than 0.5, independent of the grid size. Note that while we could also use Chebyshev restriction instead of injection for the fine-to-coarse grid transfer R, our numerical experience shows very little difference between these two choices.

Table II. Analysis of the Finite-Difference Version

	Eigenvalues of $S_{\mathcal{H}}$		By (14)		By computation		
N	$\lambda_{\max}$	λ_{qua}	ω	$\bar{\mu}$	ω_{opt}	$\bar{\mu}$	μ_s
4	1.995	1.000	0.669	0.110	0.58	0.110	0.181
8	2.513	1.000	0.569	0.186	0.60	0.168	0.293
16	2.780	0.995	0.530	0.224	0.60	0.271	0.364
32	2.898	0.815	0.539	0.315	0.60	0.366	0.421

Table III. Analysis of the Collocation Version

	Eigenvalues of $S_{\mathcal{H}}$		By (14)		By computation		
N	$\lambda_{\max}$	λ_{qua}	ω	$\bar{\mu}$	ω_{opt}	$\bar{\mu}$	μ_s
4	1.651	1.000	0.754	0.060	0.68	0.120	0.302
8	2.322	0.922	0.616	0.186	0.60	0.216	0.328
16	2.701	0.810	0.570	0.290	0.58	0.326	0.380
32	2.869	0.700	0.560	0.370	0.60	0.410	0.428

Computational Results

We have implemented the tau and finite-difference versions of the line relaxation scheme described above in a Chebyshev collocation multigrid solver for the two-dimensional Helmholtz problem

$$\begin{aligned} \lambda u(x,y) - \Delta u(x,y) &= f(x,y), \qquad |x|,|y| < 1, \\ u(x,y) &= g(x,y), \qquad |x| = 1, |y| = 1, \end{aligned}$$

with various choices of f, g, and λ. For both versions, the observed convergence factor per sweep is less than 0.5 for all cases tested, in agreement with the analysis above. The finite-difference version turns out to have slightly better convergence factors than the tau version, but the difference is minor.

Comparisons with Other Methods

In this section we compare the line relaxation spectral multigrid method developed above to two other methods for solving the two-dimensional prototype problem (2). The first is a conventional finite-difference multigrid method; the second is a matrix diagonalization technique. We do not compare with the method of Zang et. al. (ref. 3) since the details presented in that paper were not enough to allow programing the method. All computations are done on a SUN SPARCstation2 using double precision; the machine round-off error is about 2.22×10^{-16}.

Conventional Finite-Difference Multigrid Method

The finite-difference discretization is the usual second-order five-point scheme on a uniform grid. The finite-difference multigrid method uses Gauss-Seidel (Red-Black) iteration as a relaxation scheme, the fine-to-coarse grid transfer is half-injection, the coarse-to-fine grid transfer is bilinear interpolation, and the multigrid V-cycle algorithm is used.

According to computations, the average execution time of one $V(2,1)$-cycle of the finite-difference multigrid method is approximately $(0.56 \times 10^{-4})\, N^2$ seconds, and $(0.21 \times 10^{-3})\, N^2 \log_2 N$ seconds for line relaxation spectral multigrid method. Therefore, for the same grid sizes, one $V(2,1)$-cycle of the finite-difference multigrid method is approximately $3.75 \log_2 N$ times faster than the line relaxation spectral multigrid method.

However, because spectral methods have exponential convergence and finite-difference methods only have polynomial convergence, when high accuracy is required, finite-difference multigrid methods must use much bigger grid sizes than spectral methods. The result is that the line relaxation spectral multigrid method is faster than finite-difference when high accuracy is required. As a specific example, consider the prototype problem (2) with true solution $u(x,y) = e^{2x+y} \cos(\pi(x + 4y + 0.25))$. The relation between accuracy and execution time required to achieve that accuracy is plotted in Figure 1 for both methods. We can see that when low accuracy is required, the finite-difference multigrid method is much faster than the line relaxation spectral multigrid method, but the situation is reversed when high accuracy is required. The crossover point for this problem is at an accuracy of about one percent error. The same conclusion would hold for finite-difference methods of higher (fixed) orders, although the crossover point would shift. Variable-order finite-difference methods could be expected to perform more like the spectral method, at a cost of considerable complexity.

Matrix Diagonalization Technique

The matrix diagonalization technique is introduced in (ref. 5) as a direct solver for the Chebyshev spectral approximation to the Poisson equation with Dirichlet boundary conditions. This technique requires a preprocessing step, which involves computing the eigenvalues and eigenvectors of a one-dimensional operator matrix ($O(N^3)$ operations), and a solution step, which involves one-dimensional matrix multiplications ($O(N^3)$ operations).

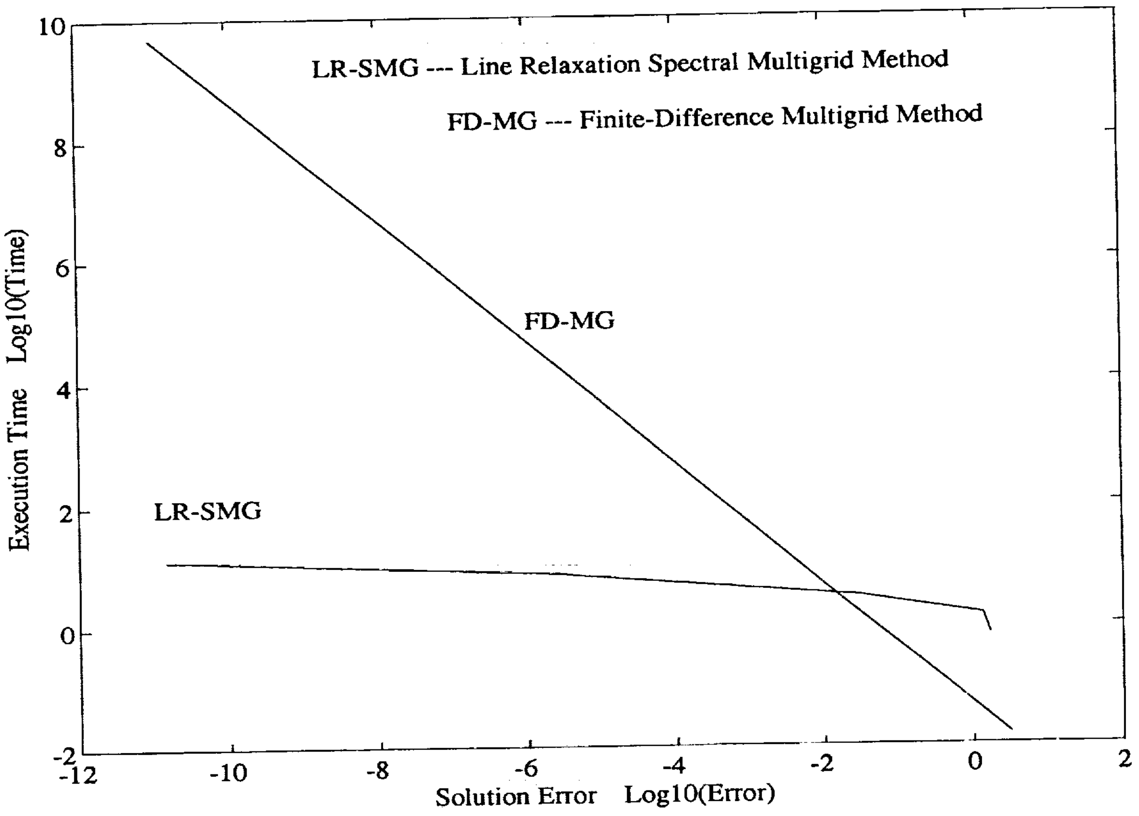


Figure 1. Execution time: LR-SMG vs FD-MG

To compare execution times, we note that the line relaxation spectral multigrid method usually takes approximately 10 V-cycles to solve to the level of machine precision. Thus, Figure 2 compares the execution time of 10 V-cycles of the line relaxation spectral multigrid method with the execution time of solving the same problem directly by the matrix diagonalization method (including the preprocessing step). These results show that the matrix diagonalization method is quite fast for small grid sizes, but as the grid size grows, it becomes slower than the line relaxation spectral multigrid method. This is because the line relaxation spectral multigrid method is an $O(N^2 \log N)$ method, while the matrix diagonalization method requires $O(N^3)$ operations (even without the preprocessing step).

The matrix diagonalization technique is very efficient for problems with constant coefficients, especially when repeated solutions are required. However, this technique can only handle problems with constant coefficients. As shown below, the line relaxation spectral multigrid method is able to solve problems with non-constant coefficients.

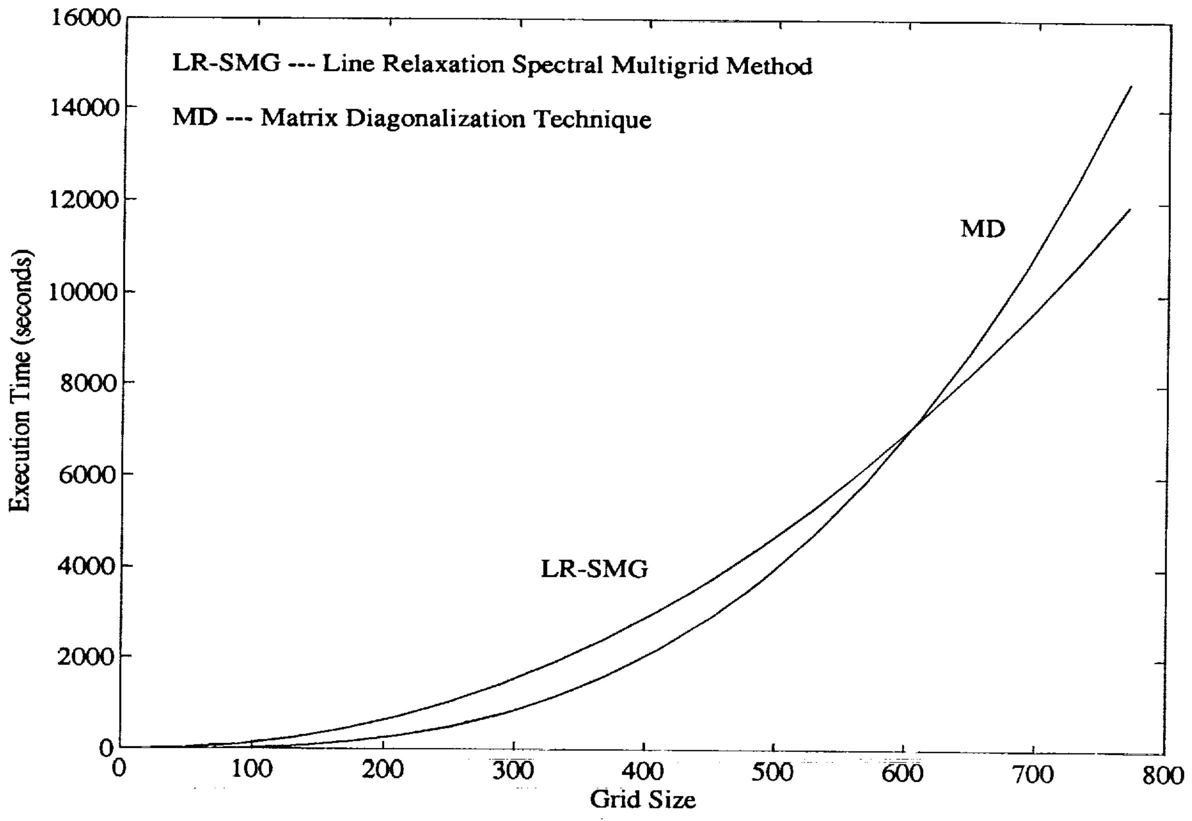


Figure 2. Execution time: LR-SMG vs MD

Extension to Problems With Variable Coefficients

As a test problem with variable coefficients we consider

$$\begin{aligned} -\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial}{\partial x}u(x,y)\right) - \frac{\partial}{\partial y}\left(b(x,y)\frac{\partial}{\partial y}u(x,y)\right) &= f(x,y), \qquad |x|,|y|<1, \\ u(x,y) &= g(x,y), \qquad |x|=1,|y|=1, \end{aligned} \tag{24}$$

where the coefficient functions and the true solution are

$$a(x,y) = b(x,y) = 1 + \varepsilon e^{\cos(\beta\pi(x+y))}, \tag{25}$$

$$u(x,y) = \sin(\alpha\pi x + \frac{\pi}{4})\sin(\alpha\pi y + \frac{\pi}{4}). \tag{26}$$

The parameter ε measures how far the coefficients are away from the constant 1, β measures the oscillation of the coefficients, and α measures the oscillation of the solution.

Implementation of the Line Relaxation Spectral Multigrid Method

The implementation of the finite-difference version of the line relaxation method is basically the same as for the constant coefficient case except for the following:

1. On each grid line, the one-dimensional problem is not a Helmholtz equation anymore. For example, on a gird line $y = \bar{y}_k$ which is parallel to x-axis, we now solve a problem like

$$-\frac{\partial}{\partial x}\left(a(x,\bar{y}_k)\frac{\partial}{\partial x}v(x,\bar{y}_k)\right) - \mathcal{V}_d(x,\bar{y}_k)v(x,\bar{y}_k) = h(x,\bar{y}_k) \tag{27}$$

   by using a second-order finite-difference approximation on the Chebyshev grid.

2. To compute values of $\mathcal{V}_d(\bar{x}_j,\bar{y}_k)$, note that the interior equation in (24) can be rewritten as

$$-a\frac{\partial^2 u}{\partial x^2} - \frac{\partial a}{\partial x}\frac{\partial u}{\partial x} - b\frac{\partial^2 u}{\partial y^2} - \frac{\partial b}{\partial y}\frac{\partial u}{\partial y} = f, \quad |x|,|y| < 1 \tag{28}$$

   and the Chebyshev collocation approximation to (28) can be written as

$$\{-\mathcal{A}\mathcal{D}_{xx} - \mathcal{A}_x\mathcal{D}_x\}\bar{U} - \{-\mathcal{B}\mathcal{D}_{yy} - \mathcal{B}_y\mathcal{D}_y\}\bar{U} = \bar{F}, \tag{29}$$

   where $\mathcal{A}$ and $\mathcal{B}$ are diagonal matrices containing the values of the coefficients $a(\bar{x}_j,\bar{y}_k)$ and $b(\bar{x}_j,\bar{y}_k)$, $\mathcal{A}_x$ and $\mathcal{B}_x$ are diagonal matrices containing the values of the derivatives $\frac{\partial}{\partial x}a(\bar{x}_j,\bar{y}_k)$ and $\frac{\partial}{\partial y}b(\bar{x}_j,\bar{y}_k)$ (which can be computed from values $a(\bar{x}_j,\bar{y}_k)$ and $b(\bar{x}_j,\bar{y}_k)$), and $\mathcal{D}_x$, $\mathcal{D}_{xx}$, $\mathcal{D}_y$, and $\mathcal{D}_{yy}$ are Chebyshev differentiation matrices. Therefore, $\mathcal{H} = -\mathcal{A}\mathcal{D}_{xx} - \mathcal{A}_x\mathcal{D}_x$ and $\mathcal{V} = -\mathcal{B}\mathcal{D}_{yy} - \mathcal{B}_y\mathcal{D}_y$; generating the diagonal entries of $\mathcal{H}$ and $\mathcal{V}$ is straightforward.

3. On coarse grids, we need to use so-called "filtered" coefficients $a(x,y)$ and $b(x,y)$ to formulate the coarse grid problems; i.e., the coefficients $a(x,y)$ and $b(x,y)$ are evaluated on the finest grid and then transferred to the coarser grids by Chebyshev restriction (ref. 3).

Computational Results

We have run the line relaxation spectral multigrid method for different values of parameters ε, α and β. For $\alpha = 1.0$ and $N_x = N_y = 32$, the smoothing factor is graphed in Figure 3 as a function of ε and α. Here we have chosen to measure the smoothing by the "smoothing factor per work unit" defined by $\mu_w = (r_2/r_1)^{\tau_0/\tau}$, where r_1 and r_2 are residual norms before and after one multigrid V-cycle, τ is the execution time of one cycle and τ_0 is the execution time of one relaxation. These results show that for a wide range of ε and β, the method converges relatively quickly.

In (ref. 3) the same test problem (24) was solved using the Richardson relaxation (5) using two-dimensional finite-difference preconditioning; incomplete LU decomposition was used to approximately solve the finite difference approximation on the Chebyshev grid. With only limited details of the formulation and results of this method, it is difficult to make a complete comparison to the line relaxation method considered here. However, it appears that the line relaxation method gives convergence factors at least as small as those in (ref. 3); moreover, it is simpler.

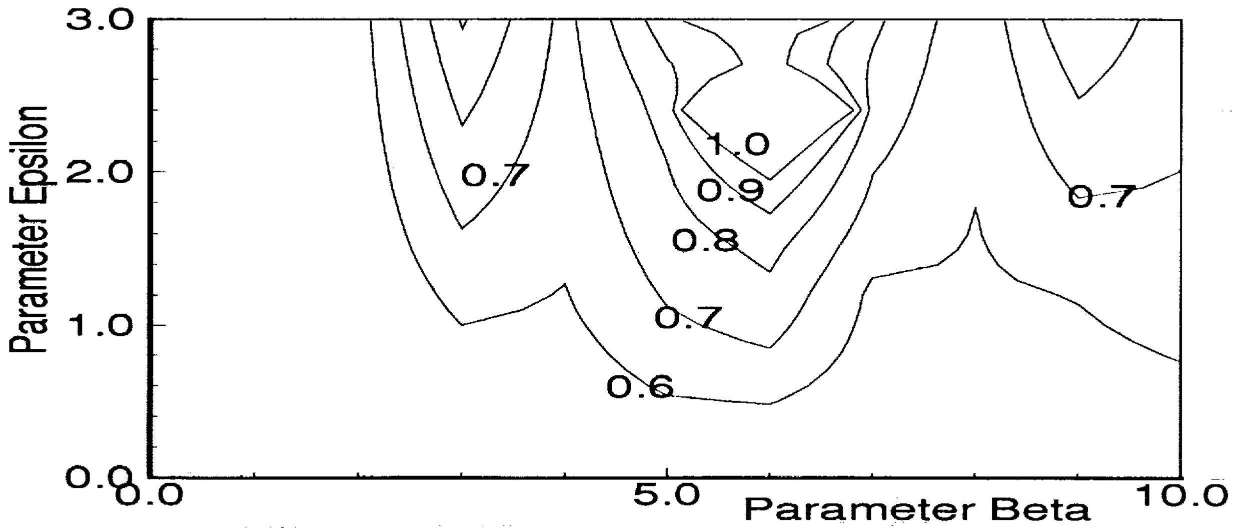


Figure 3. Smoothing factors for problems with variable coefficients.

CONCLUSIONS

The pointwise preconditioning is simple and fast to apply. It is very efficient for one-dimensional problems. Unfortunately, it does not give fast multigrid convergence for two-dimensional problems.

The line relaxation method provides a new approach to accelerate the multigrid Chebyshev spectral method for solving two-dimensional elliptic problems. It is efficient (yielding multigrid smoothing factors no larger than 0.5 per sweep) and inexpensive (requiring $O(N^2 \log N)$ operations per sweep).

When high accuracy is required, the spectral multigrid method using line relaxation is orders of magnitude faster than a conventional finite-difference multigrid method, due primarily to the exponential convergence of the spectral discretization. Compared to other methods for solving the discrete spectral equations, the line relaxation method also has advantages: it is comparable in efficiency to matrix diagonalization and finite-difference preconditioned Richardson relaxation, but can solve problems with variable coefficients which the former cannot, and is simpler than the latter.

REFERENCES

1. Canuto; Hussaini; Quarteroni; and Zang: *Spectral Methods in Fluid Dynamics.* Springer-Verlag, 1988.

2. Zang, T. A.; Wong, Y. S.; and Hussaini, M. Y.: Spectral multigrid methods for elliptic equations. *J. Comp. Phys.*, vol. 48, 1982, pp. 485–501.

3. Zang, T. A.; Wong, Y. S.; and Hussaini, M. Y.: Spectral multigrid methods for elliptic equations II. *J. Comp. Phys.*, vol. 54, 1984, pp. 489–507.

4. Brandt, A.: *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, 1984. [Available from GMD, Postfach 1240, D-5202 St. Augustin, 1, F.R.G.]

5. Haidvogel, Dale B.; Zang, Thomas: The Accurate Solution of Poisson's Equation by Expansion in Chebyshev Polynomials. *J. Comp. Phys.*, vol. 30, 1997, pp. 167-180.

5/6 34
197148
P- 12

MULTIGRID METHODS FOR A SEMILINEAR PDE IN THE THEORY OF PSEUDOPLASTIC FLUIDS*

N94-29689

Van Emden Henson
Department of Mathematics
Naval Postgraduate School
Monterey, CA

A. W. Shaker
Department of Mathematics
Naval Postgraduate School
Monterey, CA

SUMMARY

We show that by certain transformations the boundary layer equations for the class of non-Newtonian fluids named *pseudoplastic* can be generalized in the form

$$\Delta u + p(x)u^{-\lambda} = 0, \quad x \in \Omega \subseteq R^n, \quad n \geq 1$$

under the classical conditions for steady flow over a semi-infinite flat plate. We provide a survey of the existence, uniqueness, and analyticity of the solutions for this problem. We also establish numerical solutions in one- and two-dimensional regions using multigrid methods.

INTRODUCTION

In the last two decades, solutions of the singular semilinear equation

$$\Delta u + p(x)u^{-\lambda} = 0, \quad x \in \Omega \subseteq R^n \tag{1}$$

have been extensively studied. Various existence and uniqueness results are given in [1], [2], and [3], to name a few. More recently, in [4], it is shown that by certain transformations the boundary layer equations for the class of non-Newtonian fluids named *pseudoplastic* can be generalized in the above form for the ODE case $n = 1$. Under this physical interpretation the above equation, considered in the context of partial differential equations ($n > 1$), has been the subject of much study. The equation has a unique classical solution with a bounded domain Ω, where $p(x)$ is a sufficiently regular function which is positive on $\bar{\Omega}$ [5]. There exist entire solutions with $\lambda \in (0,1)$ for $p(x)$ sufficiently regular ([6], [7]). This is generalized to all $\lambda > 0$ via the upper and lower solution method ([8]) or other methods ([9]).

*This work was supported in part by the the Naval Postgraduate School Research Council under grant No. ZZ867-ZZ899/5986

PAGE 230 INTENTIONALLY BLANK

PRECEDING PAGE BLANK NOT FILMED

The following sections provide a survey of both theoretical and numerical results in this area including a physical derivation [4], existence theorems for both the ODE and PDE cases with a proof of a main result [8], and our numerical results. We conclude with a discussion of a new technique and some open questions for further research.

PRELIMINARIES

A non-Newtonian fluid is called *pseudoplastic* if the shear stress τ and the strain rate $\dfrac{\partial u}{\partial y}$ are related as

$$|\tau| = k\left|\frac{\partial u}{\partial y}\right|^{\alpha}, \quad 0 < \alpha < 1$$

where k is a positive constant. That is, the absolute value of the shear stress increases with respect to the absolute value of the strain rate less than linearly.

In this paper, we study solutions of the singular semilinear equation (1) where $\lambda > 0$ and Ω is a domain in R^n, $n \geq 1$. In the following section we show that through a series of transformations the boundary layer equations for the class of pseudoplastic fluids under the classical conditions for a steady flow over a semi-infinite flat plate can be generalized into the well-known Blasius problem

$$f''' + ff' = 0,$$

$$f(0) = f'(0) = 0, \quad f'(\infty) = 1$$

for the shear function, which arises from the standard Newtonian fluid case.

DERIVATION OF THE PROBLEM

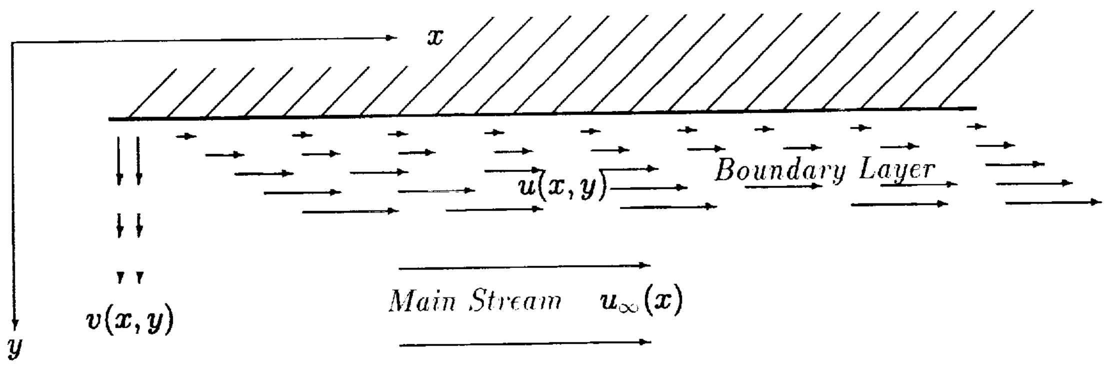
For $n = 1$ equation (1) arises in the study of pseudoplastic fluids. We consider a two-dimensional incompressible flow of low viscosity along a plane wall. We denote by $\vec{v} = (u, v)$ as the fluid velocity in the boundary layer and $u_\infty(x)$ in the main stream. Since there is no velocity on the wall and the fluid takes the velocity of the main stream $u_\infty(x)$ outside the boundary layer, we see that $\dfrac{\partial u}{\partial y}$ is large near the wall which causes a significant transfer of momentum in the x direction.

The boundary layer equations for this model include a continuity equation and a momentum equation in the x direction.

$$\begin{gathered} \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \frac{1}{\rho}\frac{\partial \tau_{xy}}{\partial y} \end{gathered} \tag{2}$$

with boundary condtions

$$u(x,0) = v(x,0) = 0,$$
$$u(x,\infty) = u_\infty(x)$$



$$\vec{v} = \begin{cases} (u,v) & \text{in boundary layer} \\ u_\infty(x) & \text{in main stream} \end{cases}$$

Figure 1: *2-D flow of low viscosity along a plane wall.*

where $\tau_{xy} = K\left|\frac{\partial u}{\partial y}\right|^\alpha$ is the shear stress.

Note that (2) has 2 coupled equations in 3 dependent variables, u, v, and τ_{xy}. To reduce it to a single higher-order equation in only 1 dependent variable, we introduce the Lagrange stream function $\Phi(x,y)$ such that

$$u = \frac{\partial \Phi}{\partial y}, \quad v = -\frac{\partial \Phi}{\partial x}.$$

Then the momentum equation becomes

$$\frac{\partial \Phi}{\partial y} \cdot \frac{\partial^2 \Phi}{\partial x \partial y} - \frac{\partial \Phi}{\partial x} \cdot \frac{\partial^2 \Phi}{\partial y^2} = \nu \frac{\partial}{\partial y}(\frac{\partial^2 \Phi}{\partial y^2})^\alpha. \tag{3}$$

where $\nu = \frac{K}{\rho}$, while the continuity equation is clearly satisfied by Φ.

Let $f = \frac{a\Phi}{\sqrt{u_\infty \nu x}}$; $\eta = by\sqrt{\frac{u_\infty}{\nu x}}$, for some a, b. Then (3) becomes

$$f''' + f(f'')^{2-\alpha} = 0 \tag{4}$$

with

$$f(0) = f'(0) = 0$$
$$f'(\infty) = 1$$

where $f = f(\eta)$. (Observe that if $\alpha = 1$ (4) is the well-known Blasius equation.) Employing the Crocco-like transformation

$$u = f'(\eta), \quad g(u) = u' = f''(\eta)$$

(4) becomes

$$g^\alpha g'' + (\alpha - 1)g^{\alpha-1}(g')^2 + u = 0$$

with $g'(0) = 0$, $g(1) = 0$, where $g = g(u)$. Finally the transformation $G = g^\alpha$ leads to the singular boundary value problem

$$\begin{aligned} &G'' + \alpha u G^{-1/\alpha} = 0, \quad 0 < u, \alpha < 1, \\ &G'(0) = G(1) = 0 \end{aligned}$$

of the form (e1) with $\lambda = \dfrac{1}{\alpha}$, $u = x$, $p = \dfrac{x}{\lambda}$.

EXISTENCE AND UNIQUENESS RESULTS

In the first part of this section we study the results in finite and infinite domains; in the second we discuss methods that are commonly used to approach the problem.

Theorems

Let Ω be a bounded domain in R^n, $n \geq 1$ with smooth boundary $\partial\Omega$ (of class $C^{2+\alpha}$, $0 < \alpha < 1$). Let $p(x)$ be of $C^\alpha(\bar{\Omega})$ and positive on $\bar{\Omega}$, $\lambda > 0$.

Theorem 1 *(Lazer-McKenna [5]). The problem*

$$\begin{aligned} &\Delta u + p(x)u^{-\lambda} = 0, \quad x \in \Omega \\ &u\,|_{\partial\Omega} = 0 \end{aligned}$$

has a unique positive solution $u(x)$ in Ω with $u \in C^{2+\alpha}(\Omega) \cap (\bar{\Omega})$. Furthermore let ϕ be an eigenfunction corresponding to the smallest eigenvalue λ_1 of the problem

$$\begin{aligned} &\Delta\phi + \lambda\phi = 0, \quad x \in \Omega \\ &\phi\,|_{\partial\Omega} = 0 \end{aligned}$$

such that $\phi_1(x) > 0$, $x \in \Omega$ and $\lambda > 1$. Then there exists a unique $b_1, b_2 > 0$ such that

$$b_1\phi_1^{2/(1+\lambda)} \leq u \leq b_2\phi_1^{2/(1+\lambda)}$$

on $\bar{\Omega}$.

In the case $\Omega = R^n$, $n \geq 1$, we study the results under conditions $n = 1$, $n = 2$, $n \geq 3$. Observe that if $n = 1$, since $p, y > 0$, $y'' + py^{-\lambda} = 0$ we have $y'' > 0$ and thus $y' \downarrow$. Hence $0 < y'(\infty) < \infty$.

Theorem 2 *(Taliaferro [3]) The problem*

$$\begin{aligned} &y'' + p(x)u^{-\lambda} = 0 \\ &y(c) = \alpha \\ &y'(\infty) = 0 \end{aligned}$$

has a unique positive solution $y(x)$ if

$$\int_1^\infty x^{-\lambda}p(x)\,dx < \infty$$

where α, $c \in R^1$, $\alpha > 0$. Furthermore $y(\infty) < \infty$ if and only if $\int_0^\infty xp(x)\,dx < \infty$.

The following theorem describes the asymptotic behaviors of the solution.

Theorem 3 *(Taliaferro [3])*

- *If* $0 < y'(\infty) < \infty$ *and* $\int_1^\infty x^{-\lambda+1}p(x)dx < \infty$, $a, b > 0$ *then*

$$y(x) = ax + b - a{-}\lambda(1 + o(1)) \int_x^\infty (\xi - x)\xi^{-\lambda}p(\xi)d\xi.$$

- *If* $y'(\infty) = 0$ *and* $\int_0^\infty xp(x)dx < \infty$, $a > 0$ *then*

$$y(x) = a - a{-}\lambda(1 + o(1)) \int_x^\infty (\xi - x)p(\xi)d\xi.$$

- *if* $p, q > 0$ *are continuous on* $[0, \infty)$, $\lim_{x\to\infty} \frac{q(x)}{p(x)} = R > 0$ *and*

$$\begin{array}{ll} z'' + p(x)z^{-\lambda} = 0, & z'(\infty) = 0; \\ w'' + q(x)w^{-\lambda} = 0, & w'(\infty) = 0 \end{array}$$

and $\int_0^\infty xp(x)dx = \infty$, *then* $\lim_{x\to\infty} w/z = R^{\frac{1}{1+\lambda}}$.

Theorem 4 *(Kusano-Swanson [7]). The problem*

$$\Delta u = f(|x|)u^{-\lambda} = 0, \quad x \in R^2,\ 0 < \lambda < 1$$

has an entire positive solution in R^2 *with logarithmic growth at* ∞ *if* $f(t) > 0$, $t > 0$, $f(t) \in C(0, \infty)$, *and*

$$\int_e^\infty t(\log t)^{-\lambda}f(t)dt < \infty.$$

A function $u(x)$ is said to be an *entire* solution of (1) if $u \in C^2_{loc}(R^n)$ and u satisfies the equation pointwise in R^n.

Theorem 5 *(Shaker [8]) The problem*

$$\Delta u + p(x)u^{-\lambda} = 0, \quad x \in R^n,\ \lambda > 0$$

has an entire positive solution $u(x)$ *such that*

$$c_1 \leq u(x)|x|^{q|n-2|} \leq c_2$$

for some c_1, c_2 *and* $0 < q < 1$ *as* $x \to \infty$ *if*

1. $p(x) \in C^\alpha_{loc}(R^n)$, $p(x) > 0$ *for* $x \in R^n\backslash\{0\}$;
2. *there exists* $0 < c < 1$ *such that* $c\phi(|x|) \leq p(x) \leq \phi(|x|)$ *where* $\phi(t) \equiv \max_{|x|=t} p(x)$. $t \in [0, \infty)$;
3. $\int_1^\infty t^{n-1+\lambda(n-2)}\phi(t)dt < \infty$.

Methods

In general there are two methods that are commonly used in proving existence and uniqueness of solutions for equations of type (1), namely *Schauder's fixed point theorem* and *Barrier Methods*. Since the former is standard we elaborate here only on the latter.

Let Ω be a smoothly bounded domain in R^n. $\phi(x)$ is said to be an upper (lower) solution of the problem

$$\begin{aligned} \Delta u + f(x,u) = 0, \quad x \in \Omega \\ u\,|_{\partial\Omega} = 0 \end{aligned} \tag{5}$$

if $\Delta\phi + f(x,\phi) \leq 0,\ x \in \Omega,\ \phi(x) \geq 0\ x \in \partial\Omega\ (\phi + f(x,\phi) \geq 0,\ x \in \Omega,\ \phi(x) \leq 0\ x \in \partial\Omega)$.

Theorem 6 *(Sattinger [10]). Let ϕ_1 be an upper solution and ϕ_2 be a lower solution of (5), and let f be locally Hölder continuous in Ω. If $\phi_1(x) \geq \phi_2(x)$ in Ω, then (5) has a solution u such that $\phi_2(x) \leq u(x) \leq \phi_1(x),\ x \in \bar{\Omega}$.*

In the case when $\Omega = R^n$ we say ϕ is an upper (lower) solution of

$$\Delta u + f(x,u) = 0 \tag{6}$$

if $\Delta\phi + f(x,\phi) \leq 0\ x \in R^n$ (for lower solution, $\Delta\phi + f(x,\phi) \geq 0$).

Theorem 7 *(Ni [11]). Let ϕ_1 and ϕ_2 be an upper and a lower solution of equation (6), such that $\phi_1 \geq \phi_2\ x \in R^n$. If f is locally Hölder continuous in x and locally Lipschitz continuous in u, then (6) has a solution u with $\phi_2(x) \leq u(x) \leq \phi_1(x),\ x \in R^n$.*

An Example.

Consider the problem

$$\begin{aligned} &u'' + \lambda u - u^3 = 0, \quad && x \in (0,\pi) \\ &u = 0, && x = 0, \pi. \end{aligned}$$

It is easy to show that $\phi_1(x) = Rx^{1/2}$ for some R large is an upper solution, and $\phi_2(x) = \epsilon \sin x$ for some ϵ small is a lower solution of this problem. Clearly $\phi_1(x) \geq \phi_2(x)$ for $x \in [0,\pi]$. Thus by the above theorem there is a solution $u(x)$ such that $\epsilon \sin x \leq u(x) \leq Rx^{\frac{1}{2}},\ x \in [0,\pi]$. Since the problem is homogeneous we conclude that the problem has at least three solutions, namely, u, $-u$ and the trivial solution.

MULTIGRID SOLUTION OF THE PROBLEM

In this section we present some numerical results for solving the problem

$$\begin{aligned} \Delta u + p(x)\,u^{-\lambda} &= 0 \quad && x \in \Omega \\ u(x) &= 0 && x \in \partial\Omega. \end{aligned}$$

Specifically, we describe Newton's method for non-linear systems to solutions and multigrid $V - cycle$ and FMV methods. We have implemented all of these methods for both the one- and two-dimensional cases, using (respectively) the unit interval and the unit square for Ω. In each case we use a straightforward finite-difference discretization, employing the standard second-order difference approximation for the second derivative operator. For the one-dimensional problem we define the grid of $(N+1)$ points $x_k = jh$, for $k = 0, 1, \ldots N$, where h is the mesh parameter $1/N$. The second derivative operator is then approximated by

$$\left.\frac{d^2u}{dx^2}\right|_{x_k} = \frac{u_{k-1} - 2u_k + u_{k+1}}{h^2} + O(h^2), \tag{7}$$

where u_k approximates $u(x_k)$. For the non-linear term $p(x)u(x)^{-\lambda}$ we use the nodal values, $p_k\, u_k^{-\lambda}$. Since $u_0 = u_N = 0$, this results in the non-linear system of equations

$$\frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} p_1\, u_1^{-\lambda} \\ p_2\, u_2^{-\lambda} \\ \vdots \\ p_{N-2}\, u_{N-2}^{-\lambda} \\ p_{N-1}\, u_{N-1}^{-\lambda} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{8}$$

Letting u represent the vector of unknowns, we may write the system as $Hu + g(u) = 0$, where H is the tridiagonal matrix and g is the non-linear vector function.

For the two-dimensional case we take the tensor product of the $(N+1)$-point grid in the x direction with an identical $(N+1)$-point grid in the y direction, yielding an $(N+1)^2$-point regular grid covering the unit square. The difference operator for the two-dimensional problem is

$$\left.\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)\right|_{x_{j,k}} = \frac{u_{j-1,k} - 2u_{j,k} + u_{j+1,k}}{h^2} + \frac{u_{j,k-1} - 2u_{j,k} + u_{j,k+1}}{h^2} + O(h^2). \tag{9}$$

Numbering the unknowns lexicographically by lines of constant y, we obtain the nonlinear system

$$\begin{bmatrix} A & B & & & \\ B & A & B & & \\ & \ddots & \ddots & \ddots & \\ & & B & A & B \\ & & & B & A \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-2} \\ u_{N-1} \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{N-2} \\ w_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{10}$$

where here u_j denotes the $(N-1)$-length vector of unknowns $u_{j,k}$ for $k = 1, 2, \ldots, N-1$ corresponding to the j^{th} grid-line in the y direction, and A and B are $(N-1) \times (N-1)$ matrices

$$A = \frac{1}{h^2}\begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{bmatrix} \qquad B = \frac{1}{h^2}\begin{bmatrix} -1 & & & & \\ & -1 & & & \\ & & \ddots & & \\ & & & -1 & \\ & & & & -1 \end{bmatrix}.$$

The $(N-1)$-length vectors w_j contain the non-linear entries $p_{j,k}\, u_{j,k}^{-\lambda}$, for $k = 1, 2, \ldots, N-1$. Once again, we may write the system as $Hu + g(u) = 0$, where H is the block tri-diagonal matrix and g is the non-linear vector function containing the w_j's.

Solution techniques

The classical solution technique for (8) or (10) is to apply Newton's method for non-linear systems. We write the system as $F(u) = 0$, where $F(u) = Hu + g(u)$. Each step of the iteration is then given by

$$u \leftarrow u - [J_F(u)]^{-1} F(u)$$

where the Jacobian of the system is given by

$$[J_F(u)] = H + D$$

with H the linear part of F and D a diagonal matrix whose diagonal entries are the derivatives of the entries of g, for example $-\lambda p(x_{j,k}) u_{j,k}^{-\lambda-1}$.

Naturally, the Jacobian is not inverted at each step, but rather, we solve the system $[J_F(u)]y = -F(u)$ and then make the correction $u \leftarrow u + y$. We examined two methods for solving the system at each step, namely LU decomposition and a multigrid FMV cycle.

Newton's method converges quadratically. However, since each step involves inverting a system, it tends to be very slow. While the use of the FMV solver speeds the method up somewhat, it still is slower than the techniques we present next. It has long been known ([12], [13]) that on certain problems non-linear analogs to the classical Jacobi or Gauss-Seidel iteration methods could be employed with some success. Technically, one sweep of such a method means that for $j = 1, 2, \ldots, N-1$ (or $(N-1)^2$ for the two-dimensional problem) one solves, via the scalar Newton's method, the j^{th} non-linear equation in the system $F(u) = 0$ for the j^{th} unknown. As in the linear case, if the old values u are used throughout the sweep this is the Newton-Jacobi method, while if the updated values are employed as they become available it is the Newton-Gauss-Seidel method. In practice the j^{th} equation is not actually solved, but rather, a few (one or two) steps of the scalar Newton's method is performed on each equation in turn.

The Newton-Jacobi and Newton-Gauss-Seidel iterations, however, typically behave in the same fashion that is observed in their linear counterparts. That is, the iteration generally progresses rapidly toward a solution with the first few sweeps, but then stalls out so that each additional sweep produces very little improvement. The reason behind this is the same as that seen in the linear case. The method stalls after the non-linear relaxation has successfully eliminated the oscillatory portion of the error, which it eliminates rapidly, but is unable to effectively treat the smooth portion of the error. This is precisely the difficulty that multigrid methods were devised to overcome.

At the heart of multigrid is the coarse-grid correction [14]. Many common relaxation iterative relaxation methods for solving a *linear* problem $Au = f$ have the property that the relaxation effectively eliminates the high-frequency (oscillatory) components of the error but leave the low frequency (smooth) components essentially unaffected. However, because the error is smooth after the relaxation, it may be represented accurately on a coarser grid, on which it also appears more oscillatory (relatively). Relaxation on this coarser grid then eliminates the oscillatory components of the coarse-grid error, which cannot be eliminated on the fine grid. The coarse-grid correction for a linear problem may be written as

$$u^h \leftarrow P^\nu u^h + I_{2h}^h (A^{2h})^{-1} I_h^{2h} (f^h - A^h P^\nu u^h) \tag{11}$$

where P is the relaxation matrix, ν is the number of relaxations, I_{2h}^h is a *prolongation* or *interpolation* matrix mapping coarse-grid vectors to the fine grid, I_h^{2h} is a *restriction* matrix mapping fine-grid vectors to the coarse grid, and A^{2h} is a coarse-grid version of the original matrix A. A crucial feature is that on the coarse grid Ω^{2h}, the problem to be solved is the *residual* equation $Ae = r$, where the residual is defined $r = f - Au$ and e is the *error*. That is, if u^* is the exact solution, then $Ae = A(u^* - u) = f - Au = r$.

For *nonlinear* problems the residual equation doesn't hold. Instead, we write the nonlinear equivalent of the residual equation,

$$F(u + e) - F(u) = r.$$

This equation is to be solved on the coarse grid, so we write

$$F^{2h}(I_h^{2h} u^h + e^{2h}) - F^{2h}(I_h^{2h} u^h) = I_h^{2h}(f^h - F^h(u^h)), \tag{12}$$

or

$$F^{2h}(u^{2h}) = I_h^{2h}(f^h - F^h(u^h)) + F^{2h}(I_h^{2h}).$$

The coarse-grid correction is then performed by solving (12) for $u^{2h} = I_h^{2h} u^h + e^{2h}$, and then making the correction $u^h \leftarrow u^h + I_{2h}^h(u^{2h} - I_h^{2h} u^h)$. This gives the full approximation scheme [15]

$$u^h \leftarrow P^\nu(u^h) + I_{2h}^h((F^{2h})^{-1}(I_h^{2h}(f^h - F^h(P^\nu(u^h))) + F^{2h}(I_h^{2h} P^\nu(u^h))) - I_h^{2h} P^\nu(u^h)),$$

where P is a nonlinear relaxation scheme.

For both the linear and nonlinear problems, the solution of the coarse-grid problem is computed using the same coarse-grid correction scheme as is being employed to solve the fine-grid problem. This leads to the multigrid *V-cycle* scheme, which (for the nonlinear problem using *FAS*) is described recursively as follows.

$$u^h \leftarrow FASV^h(u^h, f^h, \nu_1, \nu_2)$$

```
1. Perform $\nu_1$ non-linear relaxation sweeps times on $F^h(u^h) = f^h$ with initial guess $u^h$.
2. If $\Omega^h$ is the coarsest grid, then go to 4. Else:
       $f^{2h} = I_h^{2h}(f^h - F^h(u^h)) + F^{2h}(I_h^{2h} u^h)$
       $u^{2h} \leftarrow 0$
       $u^{2h} \leftarrow FASV^{2h}(u^{2h}, f^{2h}, \nu_1, \nu_2)$.
3. Correct $u^h \leftarrow u^h + I_{2h}^h(u^{2h} - I_h^{2h} u^h)$.
4. Perform $\nu_2$ non-linear relaxation sweeps times on $F^h(u^h) = f^h$ with initial guess $u^h$.
```

An important consideration for this (or any) iterative method is the choice of a good initial guess. Clearly a better initial guess will reduce the overall effort required to obtain an acceptable solution. A standard approach in multigrid is to obtain a good initial guess by first solving the problem on a coarse grid, and then interpolating that solution to the fine-grid for use as an initial guess. Solving this coarse-grid problem, in turn, will be easier if an initial guess is obtained by first solving the problem on a still coarser grid. Applying this idea recursively leads the Full Multigrid *FMG* scheme, which (applied to the non-linear *FASV* scheme) may be described as follows:

$$u^h \leftarrow FASFMG^h(u^h, \nu_1, \nu_2)$$

1. If Ω^h is the coarsest grid, then go to 3. Else:
   $f^{2h} = I_h^{2h}(f^h - F^h(u^h)) + F^{2h}(I_h^{2h}u^h)$
   $u^{2h} \leftarrow 0$
   $u^{2h} \leftarrow FASFMG^{2h}(u^{2h}, f^{2h}, \nu_1, \nu_2)$.
2. Correct $u^h \leftarrow u^h + I_{2h}^h u^{2h}$.
3. $u^h \leftarrow FASV^h(u^h, f^h, \nu_1, \nu_2)$.

Numerical results for multigrid methods

We have implemented the $FASV$ using Newton-Jacobi and Red-black Newton-Gauss-Seidel iteration schemes. (Our implementation was in *Matlab* using vector arithmetic. We elected not to analyse Newton-Gauss-Seidel since it is not vectorizable. We did encode it, however, and found that the slowness of the *for* loops overwhelmed the speed of convergence.) Several different choices for λ, $p(x)$ and $p(x,y)$ were used, as were several sets of relaxation parameters.

Table 1 gives some quantitative information regarding the performance of the method, comparing convergence rates for various choices of parameters. The results shown were obtained using the Red-black Newton-Gauss-Seidel relaxation. We find that for this problem we are able to obtain convergence rates that are similar to those obtained on the linear elliptic model problems for which multigrid is best known ([14], [16], [17]). Data for the one-dimensional problem are not shown, however, they are very similar to the two-dimensional case.

Dimension	$p(\vec{x})$	λ	Fine-grid size	Average V-cycle convergence factor
2	$2xy$	2	63×63	0.051
		5		0.050
		8		0.078
2	$2\sin(2\pi x)\sin(\pi y)$	2	63×63	0.060
		5		0.063
		8		0.104
2	x/y	2	63×63	0.059
		2		0.060
		8		0.086

Table 1

Additionally, we have implemented the $FASFMG$ using Newton-Jacobi and Red-black Newton-Gauss-Seidel iteration schemes. Again, we find that the performance of the method is compatible with that found for FMG applied to the linear model problems ([15], [17]).

CONCLUSIONS

Our survey of existence and uniqueness results has shown the problem

$$\Delta u + p(x)\, u^{-\lambda} = 0 \quad x \in \Omega$$

is guaranteed to have unique solutions under certain conditions, although these solutions will not be known in closed form. The problem arises in certain non-Newtonian fluids problems, so there is some interest in actually computing solutions. We have shown that for homogeneous Dirichlet boundary conditions on the unit interval and the unit square, multigrid methods appear to provide an efficient means of solution for reasonable choices of $p(x)$.

We note, however, that an actual convergence proof for the FAS method would be very difficult to obtain, in that such proofs normally require that we be able to decompose the space of grid functions into two operator-subspaces. Error components in one are annihilated by relaxation, while those in the other subspace are annihilated by coarse-grid correction. While such analysis is achieved for linear problems, non-linear problems generally can only be treated by linearization near a solution. In point of fact, the literature is remarkably sparse in the area of founding theory for the FAS method.

A new technique, called *multilevel projection methods (PML)* has recently been introduced, [18] in an effort to provide a unifying, thematic approach to the design of a multilevel solver for a given problem. The main feature of PML methods is that the only basic choices that must be made concern the subspaces that will be used in relaxation and coarsening. All other components of the method, such as interlevel transfers, scaling, coarse-level problems, etc., are determined by projection between appropriate subspaces. In [18], several prototypical problems are developed to illustrate the principals involved. It now appears that the best hope of obtaining a strong founding theory for multilevel treatment of nonlinear problems may well be through careful and judicious application of PML, and our future research into solution methods for the problems we have discussed here will be aimed in that direction.

ACKNOWLEDGEMENTS

The authors wish to thank Steve McCormick for his helpful suggestions throughout the project. This work was supported, in part, by the Research Council of the Naval Postgraduate School.

REFERENCES

[1] M. G. Crandall, P. H. Rabinowitz, and L. Tartar. On a Dirichlet problem with a singular nonlinearity. *Comm. Part. Diff. Eq.*, 2(2):193–222, 1977.

[2] W. Fulks and L. S. Maybee. A singular nonlinear equation. *Osaka Math. J.*, 12:1–19, 1960.

[3] S. Taliaferro. On the positive solution of $y'' + \phi(t)y^{-\lambda} = 0$. *Nonlinear Analysis, Theory, Methods & Applications*, 2(4):437–446, 1978.

[4] A. J. Callegari and A. Nachman. A nonlinear singular boundary value problem in the theory of pseudoplastic fluids. *SIAM J. Appl. Math*, 30:275–281, 1980.

[5] A. C. Lazer and P. J. McKenna. On a singular nonlinear elliptic boundary value problem. *Proceedings of the AMS*, 111:721–730, 1991.

[6] A. Edelson. Entire solutions of singular elliptic equations. *J. Math. Anal. Appl.*, 139:523–532, 1989.

[7] T. Kusano and A. Swanson. Entire positive solutions of singular semilinear elliptic equations. *Japan J. Math.*, 11, 1985.

[8] A. W. Shaker. On singular semilinear elliptic equations. *J. Math. Anal. and Appl.*, 173(1):222–228, 1993.

[9] R. Dalmasso. Solutions d'equations elliptiques semilineaires singulieres. *Annali di matematica pura et applicata serie quarta*, 153:191–201, 1988.

[10] D. H. Sattinger. *Topics in stability and bifurcation theory.* Springer-Verlag, Berlin, 1973.

[11] W. Ni. On the elliptic equation $\delta u + k(x)u^{(n+2)/(n-2)} = 0$, its generalization and application in geometry. *Indiana Univ. Math. J.*, 31(optional):493–525, 1982.

[12] Louis B. Rall. *Computational solution of nonlinear operator equations.* Robert E. Krieger Publishing Company, Huntington, New York, 1979.

[13] James M. Ortega. *Numerical analysis, a second course.* Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1979.

[14] William L. Briggs. *A Multigrid Tutorial.* Society for Industrial and Applied Mathematics, Philadelphia, 1987.

[15] Achi Brandt. *Mulitgrid techniques: 1984 guide with application to fluid dynamics.* GMD-Studien Nr. 85. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, 1984.

[16] Jan Mandel, Stephen F. McCormick, and R. Bank. Variational multigrid theory. In Stephen F. McCormick, editor, *Multigrid methods*, volume 3 of *Frontiers in applied mathematics*, pages 131–177, Philadelphia, PA, 1987. Society for Industrial and Applied Mathematics.

[17] K. Stüben and Ullrich Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. In W. Hackbusch and U. Trottenberg, editors, *Multigrid methods, proceedings of a conference held at Köln-Porz, November 23-27, 1981*, volume 960 of *Lecture notes in mathematics*, pages 1–176, Berlin, 1982. Springer-Verlag.

[18] Stephen F. McCormick. *Multilevel projection methods for partial differential equations*, volume 62 of *CBMS-NSF regional conference series in applied mathematics.* Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

S17-34
197149
P. 15
N94-29690

A Multilevel Adaptive Projection Method for Unsteady Incompressible Flow*

Louis H. Howell
Lawrence Livermore National Laboratory
Livermore, CA 94550

ABSTRACT

There are two main requirements for practical simulation of unsteady flow at high Reynolds number: the algorithm must accurately propagate discontinuous flow fields without excessive artificial viscosity, and it must have some adaptive capability to concentrate computational effort where it is most needed. We satisfy the first of these requirements with a second-order Godunov method similar to those used for high-speed flows with shocks, and the second with a grid-based refinement scheme which avoids some of the drawbacks associated with unstructured meshes.

These two features of our algorithm place certain constraints on the projection method used to enforce incompressibility. Velocities are cell-based, leading to a Laplacian stencil for the projection which decouples adjacent grid points. We discuss features of the multigrid and multilevel iteration schemes required for solution of the resulting decoupled problem. Variable-density flows require use of a modified projection operator—we have found a multigrid method for this modified projection that successfully handles density jumps of thousands to one. Numerical results are shown for the 2D adaptive and 3D variable-density algorithms.

INTRODUCTION

The incompressible flow algorithm presented by Bell, Colella and Glaz [3] combines the original projection method of Chorin [9, 10] with the Godunov methodology developed by Colella [11] to yield a robust scheme which is second-order in both space and time. In [5] Bell and Marcus extend this method to handle flows involving spatial density variations.

Originally developed for gas dynamics problems with strong shocks, the second-order Godunov technology gives the algorithm the ability to propagate discontinuous

*This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48. Support was provided by the Applied Mathematical Sciences Program of the Office of Energy Research under contract No. W-7405-Eng-48, and by the Defense Nuclear Agency under IACRO 93-817.

flow fields or density jumps without introducing nonphysical oscillations, violating conservation laws, or employing unnecessary dissipation. The resulting schemes are therefore appropriate for studying unsteady flows with little or no viscosity. The projection portion of the algorithm enforces incompressibility without the need for an artificial pressure boundary condition.

The most natural discretization for Godunov methods involves storing all velocity components at the centers of grid cells. Node-based variants are not difficult to obtain, but the requirement that all components be stored at the same points is a fairly strong one. Formulations of the projection using the staggered grid system of Harlow and Welsh [13] are thus largely incompatible with the Godunov approach. Use of collocated velocities, however, leads to unusual difference stencils for the projection which decouple adjacent grid cells.

We have developed extensions to the algorithms of [3] and [5], the most important of which are a reformulation of the methods on an adaptive hierarchy of grids, and the use of multigrid and multilevel iteration techniques to speed up computation of the projection. While we have made some attempt to keep separate the questions of how to formulate the projection versus how to solve it, there has inevitably been some interplay between these two halves of the problem. The decoupled difference stencils used by the projection in uniform parts of the grid place certain requirements on the multigrid scheme, while the need for efficient convergence of the multilevel iteration influences the choice of derivative stencils across coarse-fine grid interfaces.

These issues, concerning the formulation of the projection and its solution via multigrid methods, are the primary concern of this paper. Most of this material is new, though the need for a decoupled multigrid stencil was discussed briefly in [4]. The detailed formulation of the Godunov module, methods for error estimation and regridding, and the addition of viscous terms to the equations are all discussed in another paper, currently in preparation. These subjects will therefore be given only the most cursory attention in the present work. We will, however, describe the time-stepping procedure, so as to place the projection in its proper context as a component of the algorithm. This will be part of the general overview given in the next section. The section after that discusses the multigrid projection, while the final section presents some examples and numerical results.

OVERVIEW OF THE METHOD

The equations we are attempting to solve are the incompressible Euler equations with finite-amplitude density variation,

$$U_t + (U \cdot \nabla)U = -\frac{\nabla p}{\rho}, \quad (1)$$

$$\rho_t + (U \cdot \nabla)\rho = 0, \quad (2)$$

$$\nabla \cdot U = 0, \quad (3)$$

where U represents the velocity field, p represents the hydrodynamic pressure and ρ represents the local mass density. We will denote the x and y components of velocity by u and v, respectively.

The range of density variation in a problem may be moderate, as in the case of two or more different gases mixing in a combustion chamber, or may be relatively large, as in the 800-to-1 density jump at a water-air interface. Of course, many flows of interest do not involve density variations at all—for these problems (2) may be discarded, or similar equations may be used to advect passive quantities which do not affect the flow field. (Our implementation of the adaptive scheme currently handles only constant-density flows.) Flows with very small density variations are an intermediate case, as they may not require the full variable-density formulation. As described in [5], these flows may be modeled using what amounts to a constant-density projection method with a Boussinesq forcing term added to (1).

From a computational point of view the most problematic term in (1–3) is the pressure gradient. In contrast to the compressible case, pressure in incompressible flow plays no thermodynamic role, and cannot be determined from an equation of state. Its only function in the equations is to indirectly enforce the incompressibility constraint (3). The essential idea of projection methods is to eliminate the pressure entirely, by use of an operator which projects the velocity U onto the space of divergence-free vector fields.

The theory behind the projection operator is based on the Hodge decomposition, which provides that any vector field V can be decomposed into a divergence-free component V^d and the gradient of some scalar ϕ. This decomposition can be made unique through imposition of appropriate boundary conditions, e.g., no flow through boundaries. It is also orthogonal, since divergence and gradient are skew-adjoint with respect to the usual inner products on scalar and vector fields.

Given operators D for divergence and G for gradient, either continuous or discrete, a projection onto the space of divergence-free fields can be written as

$$\mathbf{P} = I - G(DG)^{-1}D. \tag{4}$$

(The numerical inversion of DG takes the place of solving the "pressure Poisson equation" that often appears in incompressible flow algorithms.) A modification of this projection is required for variable-density flows. We want to decompose a field into a divergence-free component and $1/\rho$ times the gradient of a scalar. The appropriate form is

$$\mathbf{P}_\sigma = I - \sigma G(D\sigma G)^{-1}D, \tag{5}$$

where $\sigma = 1/\rho$ and orthogonality is now with respect to a ρ-weighted inner product. In terms of this weighted projection, (1) can be written as

$$U_t = \mathbf{P}_\sigma\left[(-U \cdot \nabla)U\right]. \tag{6}$$

To obtain a second-order temporal discretization of this equation (and (2)), we use a fractional step process. First, the Godunov advection procedure is used to compute $(U \cdot \nabla)U$ and $(U \cdot \nabla)\rho$ at the $n + 1/2$ time level. The density equation can then be advanced immediately, while the projection is applied to $(U \cdot \nabla)U^{n+1/2}$ to give a divergence-free approximation to U_t:

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} = -(U \cdot \nabla)\rho^{n+1/2}, \tag{7}$$

$$\frac{U^{n+1} - U^n}{\Delta t} = \mathbf{P}_\sigma \left[-(U \cdot \nabla)U^{n+1/2}\right]. \tag{8}$$

Since the ρ equation can be advanced first, $\rho^{n+1/2}$ is available for use in the projection. The Godunov method uses $(1/\rho)\nabla p^{n-1/2}$ to approximate the effect of the incompressibility constraint on U_t; the projection in (8) then yields an updated approximation to $(1/\rho)\nabla p^{n+1/2}$ to be used at the next time step.

We will not go into detail on the internal workings of the Godunov procedure here. Suffice it to say that using approximations to time derivatives and limited slopes (U_x, etc.) at cell centers at time n, U and ρ are extrapolated to cell edges (faces in 3D) at time $n + 1/2$. Upwinding rules resolve the choices between values coming from either side of an edge, then these edge values are differenced to yield the $(U \cdot \nabla)$ terms at cell centers at time $n + 1/2$. The detailed procedure we use is very similar to that described in [3], with the variable-density enhancements given in [5], and an improved treatment of the transverse derivative terms (vU_y, etc.) as described in [4].

For a more thorough discussion of the Hodge decomposition, the incompressible Godunov algorithm, and the time-stepping procedure, we refer the reader to [3] and [5]. These papers deal exclusively with the single-grid case, but the adaptive case requires no changes to the time-stepping method and only minimal modification to the Godunov method, e.g., interpolation into ghost cells around the edges of fine grids. An adaptive Godunov method for gas dynamics that is similar to our approach is described in [7]. We describe the adaptive projection at the end of the next section; other aspects of our adaptive incompressible algorithm will be addressed in a forthcoming paper.

MULTIGRID PROJECTION

We now discuss a multigrid algorithm for computing the variable-density projection (5). For simplicity we restrict the notation to two dimensions, but the methods presented are immediately extensible to 3D. A three-dimensional flow example is included in the following section.

Given appropriate divergence and gradient stencils, a projection of the form (5) will yield a velocity field which is discretely divergence-free to the limit imposed by roundoff error. The projection will therefore be idempotent, i.e., repeated application will not further modify the projected vector field. This is a valuable property for an unsteady flow algorithm since the projection will be applied at every time step. If $D = -G^T$ then the projection will also be orthogonal, yielding the nearest—in a ρ-weighted sense—divergence-free field.

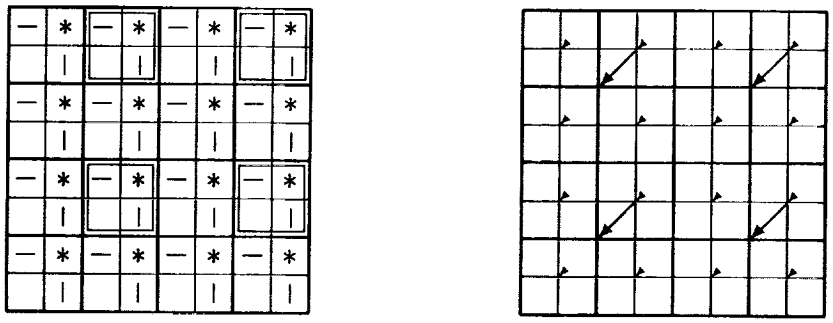


Figure 1: Decoupled grid structure: $D\sigma G\phi$ at cells marked '$*$' depends on ϕ at '$*$' cells, σ at '$-$' cells for x-differences, and σ at '$|$' cells for y-differences. Residuals from '$*$' cells are restricted by averaging to the cells marked with boxes on the next coarser grid. For purposes of restriction and interpolation, these coarse and fine values behave as if they were located at the points indicated by the arrows, rather than at the centers of their respective cells.

The simplest choice is to use centered differences for both divergence and gradient:

$$(DU)_{i,j} = \frac{1}{\Delta x}(u_{i+1,j} - u_{i-1,j}) + \frac{1}{\Delta y}(v_{i,j+1} - v_{i,j-1}), \tag{9}$$

$$(G\phi)_{i,j} = \left(\frac{1}{\Delta x}(\phi_{i+1,j} - \phi_{i-1,j}), \frac{1}{\Delta y}(\phi_{i,j+1} - \phi_{i,j-1})\right). \tag{10}$$

Composition of these then yields the elliptic stencil

$$\begin{aligned}(D\sigma G\phi)_{i,j} &= \frac{1}{(\Delta x)^2}[\sigma_{i-1,j}(\phi_{i-2,j} - \phi_{i,j}) + \sigma_{i+1,j}(\phi_{i+2,j} - \phi_{i,j})] + \\ &\quad \frac{1}{(\Delta y)^2}[\sigma_{i,j-1}(\phi_{i,j-2} - \phi_{i,j}) + \sigma_{i,j+1}(\phi_{i,j+2} - \phi_{i,j})] \end{aligned} \tag{11}$$

which appears in the projection. The main calculation we have to perform is the inversion of this expression—we have to solve $D\sigma G\phi = DV$ for ϕ given an input vector field V. Boundary conditions for ϕ are determined by those for the velocity field. Slip walls (inviscid flow) yield Neumann boundary conditions for ϕ, while in periodic problems all quantities are, naturally, periodic. Though the linear system is singular, solvability is provided by the special structure of the problem: if $D = -G^T$, then the range of G is orthogonal to the null space of D; therefore, any field in the range of D is also in the range of $D\sigma G$.

Ignoring the σ's for the moment, we see that (11) looks like a stretched version of the familiar 5-point stencil for the Laplacian. The difference is that (11) provides for no communication between adjacent grid points. Except for the effect of boundary conditions, four distinct sets of grid points participate in four distinct linear systems. Grids couple in pairs at wall boundaries, but the only local coupling comes from the smoothness of the right hand side DV. Figure 1 illustrates the decoupling pattern, including the role of the σ's.

However smooth the initial right hand side, later residuals in a multigrid scheme tend to have significant components at all wavenumbers. Multigrid depends on the fact that a solution to a coarsened system provides a good approximation to the desired fine solution. It is not surprising, therefore, that every experiment we have tried where the coarsening procedure combined components from decoupled grids proved to be wildly divergent. On the other hand, coarsening schemes which respect the decoupling lead to systems analogous to those arising from the usual 5-point Laplacian, for which multigrid is quite effective.

Let us define transformations between coarse and fine index spaces as follows,

$$I = 2 \cdot \lfloor i/4 \rfloor + i \bmod 2, \tag{12}$$
$$i = 4 \cdot \lfloor I/2 \rfloor + I \bmod 2 \tag{13}$$

and similarly for J, j. Capitals denote indices on the coarse grid, lower case on the fine grid, and $\lfloor\ \rfloor$ reduces its argument to the next lower (or equal) integer. Each coarse point (I, J) then has four fine points associated with it: (i, j), $(i, j+2)$, $(i+2, j)$, $(i+2, j+2)$. These fine points do not appear to be quite centered around the coarse point, which would complicate restriction and interpolation formulas. We observe, however, that a centered pattern results if the points in question are each shifted to the center of their local 2x2 blocks, as illustrated in Figure 1. This shifting does not change the spatial relationship of any coupled points, even at the boundary, so for multigrid purposes we can treat each coarse point as if it were centered among its four associated fine points.

The simplest restriction formula gives a coarse cell the average of the values from its associated fine cells, while the simplest interpolation formula distributes the coarse value to each of the four fine cells (piecewise-constant interpolation). There are both theoretical results and experiments, discussed in [17], which suggest that for second-degree problems at least one of these must be replaced by a higher-order formula in order to give satisfactory convergence rates. Our own experience does not bear out this assertion. However, for difficult problems involving large density jumps we have observed an improvement in robustness from use of a bilinear stencil for interpolation,

$$\phi_{i,j} = \frac{1}{16}(9\phi_{I,J} + 3\phi_{I-2,J} + 3\phi_{I,J-2} + \phi_{I-2,J-2}) \tag{14}$$

and similarly for $\phi_{i,j+2}$, etc. A smaller improvement resulted from the opposite choice, bilinear restriction with piecewise-constant interpolation. Problems without difficult density configurations did not show a consistent improvement in convergence rate with either stencil. We use (14) routinely in our variable-density code, but use the piecewise-constant formula in the constant-density adaptive code. Restriction is by simple averaging in both cases.

We have now satisfactorily dealt with the decoupling problem for ϕ, but what about σ, i.e., how to we form the elliptic stencil on coarser grids? It is apparent from Figure 1 that σ values do not occupy the same decoupled component of the grid as ϕ and the residuals. Moreover, σ values used for x-differences are on a different component from those used for y-differences.

One possibility is to redefine the problem to place σ's at the same points as ϕ's:

$$\begin{aligned}(D\sigma G\phi)_{i,j} = & \frac{1}{2(\Delta x)^2}[(\sigma_{i-2,j}+\sigma_{i,j})(\phi_{i-2,j}-\phi_{i,j})+(\sigma_{i+2,j}+\sigma_{i,j})(\phi_{i+2,j}-\phi_{i,j})]+ \\ & \frac{1}{2(\Delta y)^2}[(\sigma_{i,j-2}+\sigma_{i,j})(\phi_{i,j-2}-\phi_{i,j})+(\sigma_{i,j+2}+\sigma_{i,j})(\phi_{i,j+2}-\phi_{i,j})]. \quad (15)\end{aligned}$$

The hope is that σ could be coarsened by averaging over associated cells, just as ϕ is. Unfortunately, this scheme gives somewhat degraded accuracy, and more importantly, horrible multigrid convergence rates for problems with large density variations.

The convergence rate of the multigrid cycle seems more strongly dependent on the proper coarsening pattern for σ than on any other single feature of the method. The following procedure is in fact the *only* scheme we have tried that gave anything approaching satisfactory results. We keep two different arrays of σ values on coarser grids, one for x-differences and one for y-differences. These are coarsened as follows:

$$\begin{aligned}\sigma^x_{I,J} &= \frac{1}{2}(\sigma^x_{i',j}+\sigma^x_{i',j+2}), \\ \sigma^y_{I,J} &= \frac{1}{2}(\sigma^y_{i,j'}+\sigma^y_{i+2,j'}), \qquad (16)\end{aligned}$$

where $i' = 2I + I \bmod 2$, $j' = 2J + J \bmod 2$ and $\sigma^x = \sigma^y = \sigma$ on the fine grid. Coarse stencils based on (11) and formed with these values perform well even in the presence of sharp density interfaces. They only begin to fail when presented with such nonphysical effects as large sawtooth variations in the density field.

One common approach to deriving coarse grid equations is to use the form RAP, where R is the restriction operator, A is the elliptic stencil, and P is the interpolation operator. Unfortunately, this approach does not give a usable stencil when applied with piecewise-constant formulas for R and P, and higher-order transfer stencils give rise to larger, more complicated coarse grid operators. Use of (16) can be motivated in two ways, however. First, patterns like this one do appear in the RAP stencils, even though those formulas have other drawbacks. Second, if we confine our attention to one decoupled component of the grid, the σ locations can be interpreted as the edges between its cells. An analogy to a diffusion problem with ϕ as heat content and σ as conductivity then suggests an averaging along edges equivalent to (16).

A detailed discussion of multigrid for problems with difficult coefficients can be found in [1]. Our approach seems adequate for configurations likely to arise in practical projection problems, however, and the authors of [1] acknowledge certain pathological cases where even their more complicated schemes will fail.

For our multigrid schedule we use the pattern called FMV in [8]—the F-cycle in [17]—with smoothing by point Gauss-Seidel. Two smoothing steps before each grid transfer operation, up or down, seems to give the best performance. In problems with large density variations the Gauss-Seidel method alone does not give rapid convergence on the coarsest grid, so we have replaced it at that level with an exact solver. A direct method could be used here, but we have found it more convenient to employ

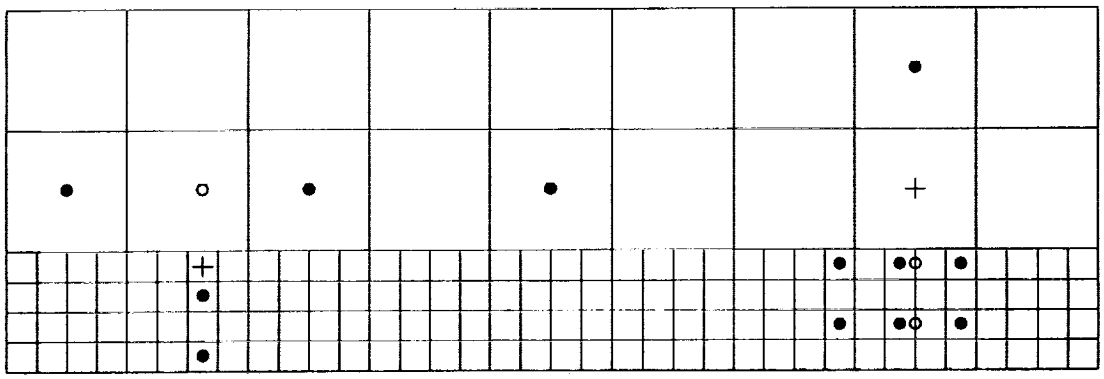


Figure 2: Examples of decoupled derivative stencils across a coarse-fine interface. The crosses indicate a fine cell (left) and a coarse cell (right) at which y-derivatives are evaluated. Bullets show which cells participate in the stencils. In each case, values on the opposite side of the interface are interpolated in the transverse direction to the circled points, giving three values on a line normal to the interface from which the derivative can be computed.

a simple diagonally-preconditioned conjugate gradient method based on algorithm 10.3-1 and equation 10.3-3 from [12]. The conjugate gradient approach has the advantage in that it neither requires explicit storage of a matrix, nor special treatment of the singular linear system.

This completes our description of the variable-density multigrid projection. One variation should be noted in passing. To reformulate the 2D projection in cylindrical (r-z) coordinates, it suffices to redefine σ as x/ρ, where $x = r$ becomes the radial coordinate. No other change is required in the projection portion of the algorithm.

— — —

An adaptive version of the projection method can be described, at least roughly, in terms of a few relatively minor additions to the single-grid algorithm. The details of the implementation, however, are considerably more complicated, and we only have a working program for the 2D constant-density flow case. Our purpose here is not to give a step-by-step breakdown of the entire adaptive procedure, but rather to highlight the ways in which a decoupled Laplacian stencil affects the multilevel projection calculation. For the sake of brevity, we have decided not to burden this discussion with explicit formulas—we trust that all necessary expressions can be easily derived from the descriptions given in the text.

The structure of the grid hierarchy is similar to that used in [7]. A single rectangular grid covers the entire computational domain at the coarsest level. In "interesting" regions of the flow, finer grid patches are laid down, refined from the coarse level by a fixed ratio r. These finer grids are themselves rectangular, both to minimize program overhead and to improve performance on vector architectures. If necessary, more levels of grids can be created, but we impose a "proper-nesting" requirement that each refined level l have a border of cells at level $l-1$ separating it from still coarser levels. The simplest choice for a refinement ratio is 2, but we often use 4 instead in order to reduce both the number of refined levels and the amount of wasted storage allocated to coarse grids underlying fine grids.

In contrast to approaches like that of [15], we have maintained a logical separation between the multilevel iteration for the adaptive scheme, and the multigrid solvers on individual grids. Our multilevel iteration proceeds as follows, where we assume familiarity with the residual-correction formulation discussed in [8] and [17]:

- Start with an initial approximation to ϕ, either 0 or the value obtained at the previous time step.
- Repeat until residuals satisfy tolerance:
    - Compute residual on all grids, including coarse-fine interfaces.
    - Restrict residuals from fine to coarse grids.
    - Set correction array to 0 at coarse level.
    - For each level l, from coarse to fine, do:
        - Execute FMV cycle for residual equation on each grid of level l, using values from adjacent grids as boundary conditions if necessary.
        - Add correction into ϕ at level l.
        - Interpolate correction to next finer level, if any.

The convergence properties of this method depend on a coarse grid solution being a satisfactory approximation to the solution on the composite grid. In order for this to be the case, all interpolation, restriction, and difference stencils have to respect the decoupling pattern. For the grid transfer operations, these formulas are like those we have already discussed. Restriction is by simple averaging of associated cells. For interpolation we have had best results with a higher-order method, a biquadratic formula using coarse cells from the appropriate decoupled grid component. Unlike the single-grid case, effective position shifts like those shown in Figure 1 are no longer valid, so we use the actual positions of cell centers to derive the interpolation stencil.

Difference formulas across the grid interfaces are more problematic. Whereas restriction and interpolation schemes affect only the convergence rate of the iteration, the difference stencils determine the actual converged solution. Stencil outlines for both fine and coarse points near the interface are shown in Figure 2. In both cases we use quadratic interpolation to obtain third-order accurate values on the opposite side of the interface, then a three-point difference formula to give a second-order accurate derivative at the desired point. Composition of second-order derivatives in D and G gives a Laplacian approximation that is first-order accurate along the interface, sufficient for global second-order accuracy of the projected velocity field.

These derivative stencils are used for computing residuals and for obtaining divergence and gradient in the projection formula. Note that D is no longer equal to

	32	64	128	256
1	3.54926 8	0.907518 (3.91) 7	0.228655 (3.97) 7	0.0573795 (3.98) 7
2	4.16279 8	1.14104 (3.65) 7	0.293781 (3.88) 7	0.074023 (3.97) 7
3	9.84036 13	4.7626 (2.07) 9	2.14014 (2.23) 11	0.876724 (2.44) 18
4	1.29866 19	0.378418 (3.43) 16	0.097241 (3.89) 15	0.024058 (4.04) 14
5	7.26845 19	1.84558 (3.94) 20	0.476259 (3.88) 21	0.123554 (3.85) 22
	0.802074	0.196431 (4.08)	0.0487401 (4.03)	0.0121474 (4.01)

Table 1: Convergence results for both variable-density and adaptive implementations of the decoupled projection. For each case the problem was run with square base grids of four different sizes—32x32 through 256x256—to a final residual less than 10^{-10}. The numbers given for each run are the final ∞-norm error in the velocity field (times 1000), the factor of improvement from the next coarser grid, and the number of multigrid cycles required. For the last run (adaptive code), 2-norm error data is also given. A description of each problem is given in the text.

$-G^T$. This means that the adaptive projection is no longer quite orthogonal, and we have to add a slight correction to DV to make the system solvable. The alternative, however, would be to use less accurate stencils for either D or G at the interface, which would seriously degrade the performance of the algorithm.

NUMERICAL EXAMPLES

Table 1 summarizes the convergence behavior of the projection for five different problems. The domain is the unit square with no flow through the boundaries. In each case we start with the divergence-free vector field

$$\begin{aligned} u &= (+0.2)(x+1)(\pi(y+1)\cos\pi y + \sin\pi y)\sin\pi x, \\ v &= (-0.2)(y+1)(\pi(x+1)\cos\pi x + \sin\pi x)\sin\pi y, \end{aligned} \tag{17}$$

add to it $1/\rho$ times the gradient of

$$\phi = \frac{-1}{\pi}\cos\left(\frac{\pi}{2}(x+x^3)\right)\cos\pi y, \tag{18}$$

then apply the projection. This should strip off the gradient portion of each field and return the divergence-free portion (17). The five cases considered are: (1) constant density, (2) mild density variation—$\rho = 1 + 100\sin^2\pi x\sin^2\pi y$, (3) extreme density variation—$\rho = 1 + 100000\sin^2\pi x\sin^2\pi y$, (4) discontinuous jump in density—$\rho = 1$ inside a radius 0.1 circle centered at (0.4, 0.4), $\rho = 10001$ elsewhere, (5) constant density adaptive—the square from 0.25 to 0.75 in x and y is refined by a factor of four from the base grid.

Cases (1) and (2) are smooth, so the multigrid scheme converges rapidly and gives unambiguous second-order convergence. Cases (3) and (4) are more difficult, but the scheme is still clearly better than first order. In the adaptive case (5) the errors are concentrated along the coarse-fine grid interface, where the discretization of DG is only first-order accurate. Convergence is still second-order in the 2-norm, but may be slightly degraded in the ∞-norm. Note that this example is not representative of the intended use of the adaptive method. In normal operation the interfaces are well-separated from complicated regions of the flow field, which dominate the error behavior of the scheme. Slower convergence for the adaptive scheme appears to be due the mismatch between coarse grid stencils and the residuals computed at the interface. Relaxation at interfaces and/or closer integration of the multigrid and multilevel iterations might yield a faster algorithm.

Quantitative analysis of the the flow solver as a whole is beyond the scope of this paper. Our remaining two examples are intended mainly as illustrations, to demonstrate the power of the algorithm for modeling unsteady flow fields with finely detailed structure. In Figure 3 we show an image from a 3D variable-density calculation set up and run by Dan Marcus. A bubble of helium was initially started at rest near the bottom of the domain. The ambient fluid is air, giving a density ratio of 7.25. The calculation was performed on a 64x64x128 grid occupying one quarter of the volume shown—this was filled out to 128^3 for rendering by reflection through the two symmetry planes. At the time of the picture the bubble has risen and developed into a torus, with more complicated flow patterns visible in the outer mixed region. We do not claim that this calculation accurately models a turbulent flow field. However, a more detailed examination of transition to turbulence, using a projection method similar to the one presented here, can be found in [6].

Figure 4 illustrates the adaptive algorithm. A 64x64 base grid is refined twice, by a factor of four each time, so the finest level has resolution equivalent to a single 1024x1024 grid. Every 10 time steps grids are re-allocated according to a procedure based on second derivatives of the velocity field. In the initial conditions, four patches of vorticity with radii 0.025 are placed in the unit square at (0.5, 0.5), (0.5, 0.575), and the two 120° rotations of this position. Each patch has uniform vorticity except for a linear ramp 3/256 wide down to zero vorticity at the edge—the radius of the patch is the distance from the center to the halfway point of the ramp. The initial velocity field is obtained by solving for the stream function associated with the given vorticity field. This is identical to the projection calculation, except that the stream function satisfies a Dirichlet boundary condition. Note how well the Godunov advection scheme preserves fine details of the flow field, even in the highly stretched regions near the vortex core.

CONCLUSIONS AND FUTURE PLANS

Centered difference stencils are the simplest choice for implementing the discrete divergence and gradient, subject to the requirement that velocity components must all be defined at the same points. The decoupled projection stencils arising from this choice require various contortions in the solution algorithm, which raises doubts as to

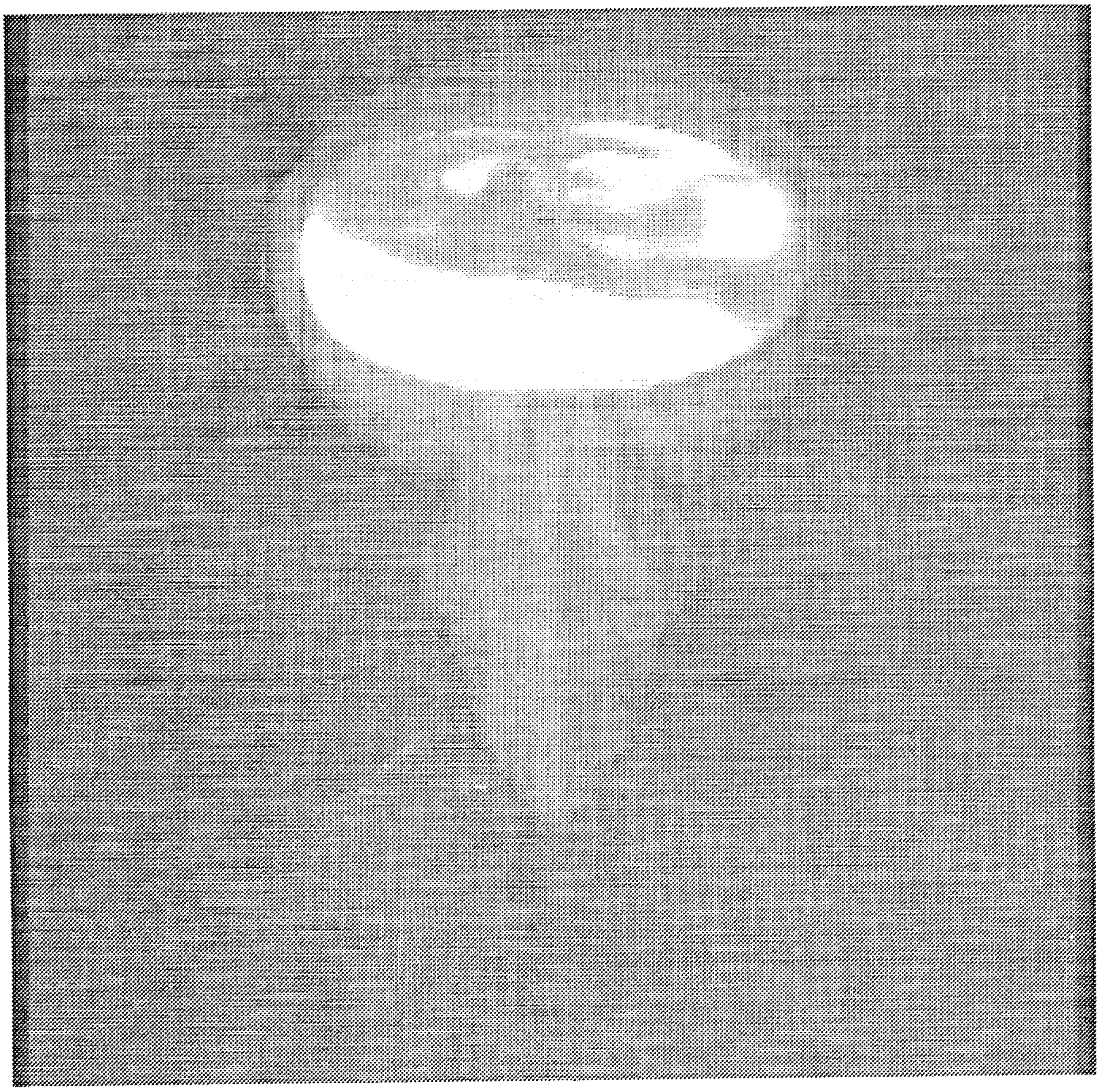


Figure 3: Volume-rendering of a helium bubble rising through air. The central part of the bubble has taken on a simple toroidal shape, but the outlying mixed regions show more complicated flow patterns.

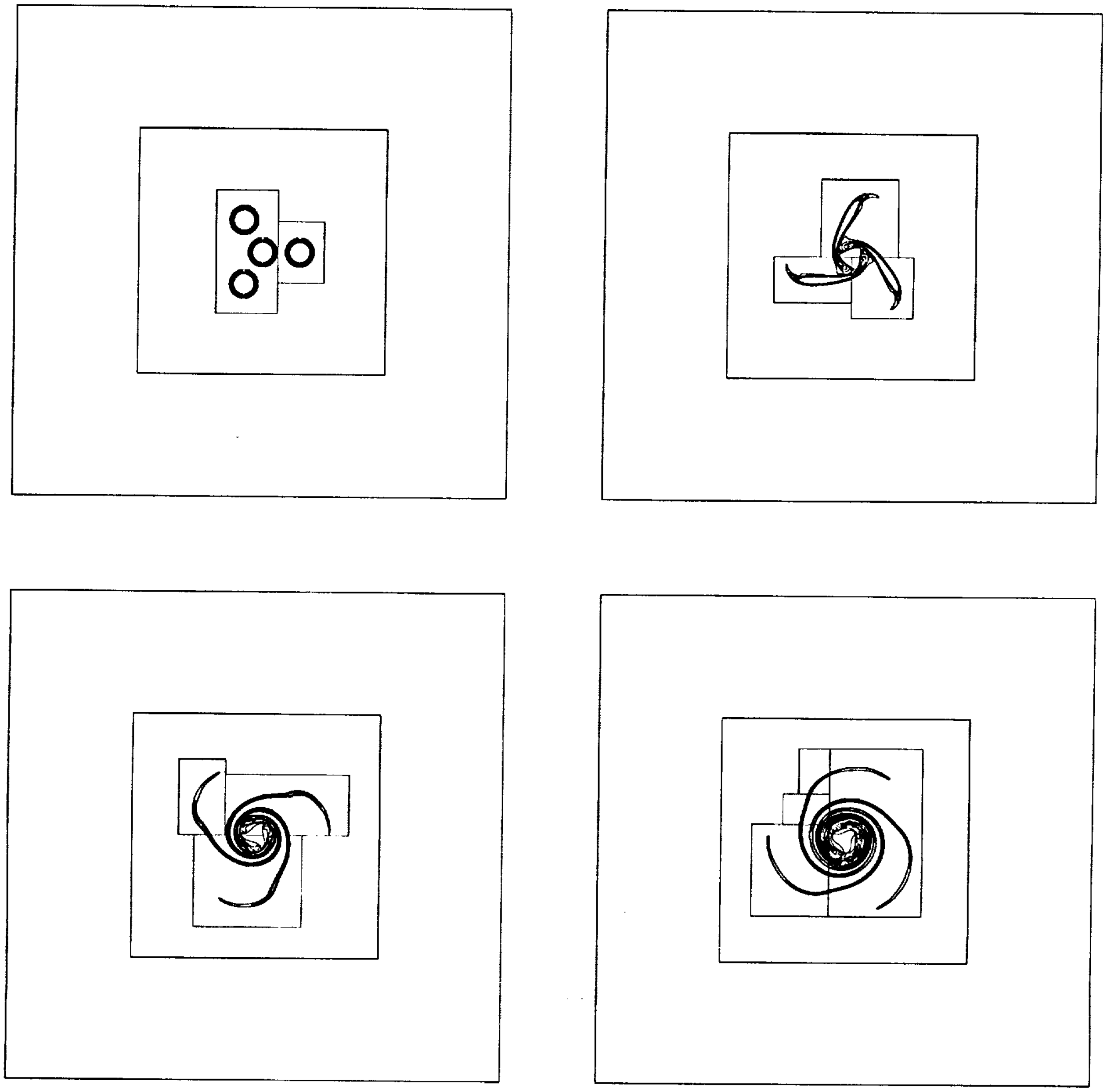


Figure 4: Adaptive simulation of a four-way vortex merger problem, showing contours of vorticity.

the practical utility of the results thus obtained. Despite the unusual behavior of the projection, however, the difficulties have been overcome and the method successfully models a variety of incompressible flow problems.

It seems likely that some flow problems will not be suitable for this type of algorithm. Though the projection does not directly cause high-wavenumber instabilities, neither does it do anything to suppress them when they are excited by other parts of a flow solver. Lai, for example, reports having difficulty using this type of projection for certain combustion problems [14]. We have seen stability problems ourselves in an adaptive version of the algorithm of [4], where a staggered-mesh projection is applied to the edge velocities computed in the Godunov predictor.

While we believe the decoupled method is a worthy contender, these difficulties beg for comparative studies with other types of projections. One alternative is the regularization given by Strikwerda [16]. Though coupled, however, the stencils derived in this work are both large and asymmetrical. A newer approach is that of Almgren, Bell and Szymczak in [2], which is coupled and symmetrical but not quite idempotent. We have recently completed an adaptive version of this projection, early results from which seem quite promising.

ACKNOWLEDGEMENTS

The author wishes to thank John Bell for his help with the design of the Godunov module, Dan Simkins for writing the streamlined Godunov integrator for 3D, and Dan Marcus for supplying the data from his helium bubble calculation.

REFERENCES

[1] R. E. Alcouffe, A. Brandt, J. E. Dendy, and J. W. Painter. The multi-grid method for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, 2:430–454, 1981.

[2] A. S. Almgren, J. B. Bell, and W. G. Szymczak. A numerical method for the incompressible Navier-Stokes equations based on an approximate projection. Technical Report UCRL-JC-112842, LLNL, January 1993.

[3] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85:257–283, December 1989.

[4] J. B. Bell, P. Colella, and L. H. Howell. An efficient second-order projection method for viscous incompressible flow. In *10th AIAA Computational Fluid Dynamics Conference*, Honolulu, June 24–27, 1991.

[5] J. B. Bell and D. L. Marcus. A second-order projection method for variable-density flows. *J. Comput. Phys.*, 101:334–348, 1992.

[6] J. B. Bell and D. L. Marcus. Vorticity intensification and transition to turbulence in the three-dimensional Euler equations. *Commun. Math. Phys.*, 147:371–394, 1992.

[7] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1989.

[8] W. L. Briggs. *A Multigrid Tutorial.* SIAM, Philadelphia, 1987.

[9] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comput.*, 22:742–762, October 1968.

[10] A. J. Chorin. On the convergence of discrete approximations to the Navier-Stokes equations. *Math. Comput.*, 23:341–353, 1969.

[11] P. Colella. A multidimensional second order Godunov scheme for conservation laws. *J. Comput. Phys.*, 87:171–200, 1990.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, 1983.

[13] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces. *Physics of Fluids*, 8:2182–2189, 1965.

[14] M. F. Lai. *A Projection Method for Reacting Flow in the Zero Mach Number Limit.* PhD thesis, University of California at Berkeley, 1993.

[15] S. F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, 1989.

[16] J. C. Strikwerda. Finite difference methods for the Stokes and Navier-Stokes equations. *SIAM J. Sci. Stat. Comput.*, 5:56–67, 1984.

[17] P. Wesseling. *An Introduction to Multigrid Methods.* Wiley, New York, 1992.

S18-64

197150

N94-23691

WAVELET MULTIRESOLUTION ANALYSES ADAPTED FOR THE FAST SOLUTION OF BOUNDARY VALUE ORDINARY DIFFERENTIAL EQUATIONS *

Björn Jawerth
University of South Carolina
Columbia, SC

Wim Sweldens
Katholieke Universteit Leuven, Belgium
University of South Carolina

SUMMARY

We present ideas on how to use wavelets in the solution of boundary value ordinary differential equations. Rather than using classical wavelets, we adapt their construction so that they become (bi)orthogonal with respect to the inner product defined by the operator. The stiffness matrix in a Galerkin method then becomes diagonal and can thus be trivially inverted. We show how one can construct an $\mathcal{O}(N)$ algorithm for various constant and variable coefficient operators.

INTRODUCTION

The purpose of this paper is to use wavelets in the solution of certain linear ordinary differential equations of the form

$$L u(x) = f(x) \quad \text{for} \quad x \in [0,1], \quad \text{where} \quad L = \sum_{j=0}^{m} a_j(x)\, D^j,$$

and with appropriate boundary conditions on $u(x)$ for $x = 0, 1$.

Currently there exist two major solution techniques. First, if the coefficients $a_j(x)$ of the operator are constants, then the Fourier transform is well suited for solving these equations. The underlying reason is that the complex exponentials are eigenfunctions of a constant coefficient operator and they form an orthogonal system. As a result the operator becomes diagonal in the

*The first author is partially supported by DARPA Grant AFOSR 89-0455 and ONR Grant N00014-90-J-1343, the second author is Research Assistant of the National Fund of Scientific Research Belgium and partially supported by ONR Grant N00014-90-J-1343.

Fourier basis and can thus trivially be inverted. The numerical algorithm then boils down to calculating the discrete Fourier transform of the right hand side, dividing each coefficient by its corresponding entry in a diagonal matrix and finally taking the inverse Fourier transform to obtain the solution. This can be done quickly using the fast Fourier transform which has a complexity of $N \log N$, where N is the number of unknowns in the discretization.

If the coefficients are not constant one typically uses finite element or finite difference methods to discretize the problem. We focus here on finite element methods. Define the *operator inner product* associated with an operator L by

$$\langle\langle u,v \rangle\rangle \;=\; \langle Lu, v \rangle .$$

A weak solution u can be found with a Petrov-Galerkin method, i.e. consider two spaces $\mathcal{S}$ and $\mathcal{S}^*$ and look for a solution $u \in \mathcal{S}$ such that

$$\langle\langle u,v \rangle\rangle \;=\; \langle f, v \rangle ,$$

for all v in $\mathcal{S}^*$. If $\mathcal{S}$ and $\mathcal{S}^*$ are finite dimensional spaces with the same dimension, this leads to a linear system of equations. The matrix of this system, also referred to as the *stiffness matrix*, has as elements the operator inner products of the basis functions of $\mathcal{S}$ and $\mathcal{S}^*$.

Traditionally one uses very local finite elements such that the stiffness matrix has a banded structure. The linear system can then be solved efficiently with an iterative method. These classical finite elements however have the disadvantage that the stiffness matrix becomes ill conditioned as the problem size grows. This slows down the convergence speed of the iterative algorithm dramatically. It is well understood by now that this can be solved with multiresolution techniques such as multigrid or hierarchical basis functions [1, 2]. Multiresolution finite element bases can provide preconditioners which result in a uniformly bounded condition number, see e.g. [3, 4, 5]. The convergence of the linear system is then independent of the problem size.

The research presented here is motived by the question of how good wavelets are for the solution of ordinary differential equations. We know that there are basically four main properties of wavelets; namely, they provide a multiresolution basis for a wide variety of function spaces, they are local in both space and frequency, they satisfy (bi)orthogonality conditions and fast transform algorithms are available. Because of these properties, wavelets have already proven to be a valuable substitute for the Fourier transform in many applications.

One possible idea, as proposed by several researchers, is to use wavelets as basis functions in a Galerkin method. This has proven to work and results in a linear system that is sparse because of the compact support of the wavelets, and that, after preconditioning, has a condition number independent of problem size because of the multiresolution structure. However, in this setting the wavelets do not provide significantly better results than more general multiresolution techniques (cfr. supra) and in fact one of their major properties, namely their (bi)orthogonality, is not exploited at all.

Three questions are addressed in this research. The first, how can one make use of the (bi)orthogonality property of the wavelets? The second, which operators can be diagonalized by wavelets? The last, are fast algorithms available and what is their complexity?

PRELIMINARIES

Notation and definitions

Much of the notation will be presented as we go along. Here we just note that the inner product of two square integrable functions $f, g \in L^2(I\!R)$ is defined by

$$\langle f, g \rangle = \int_{-\infty}^{+\infty} f(x)\,\overline{g(x)}\,dx,$$

and that the Fourier transform of a function f is defined as

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x)\,e^{-i\omega x}\,dx.$$

We say that a function w is an *L-spline* if

$$L^* L w = 0 \quad \text{and} \quad w \in C^{2m-2},$$

where L^* is the adjoint of L, a linear differential operator of order m. This definition leads to the classical piecewise polynomial splines in case $L = D^m$.

Multiresolution analysis

We give a brief review of wavelets and multiresolution analysis. For more information one can consult [6, 7, 8, 9]. A *multiresolution analysis* of $L^2(I\!R)$ is defined as a set of closed subspaces V_j, with $j \in Z\!\!Z$, that exhibit the following properties:

1. $V_j \subset V_{j+1}$,
2. $v(x) \in V_j \Leftrightarrow v(2x) \in V_{j+1}$ and $v(x) \in V_0 \Leftrightarrow v(x+1) \in V_0$,
3. $\bigcup_{j=-\infty}^{+\infty} V_j$ is dense in $L^2(I\!R)$ and $\bigcap_{j=-\infty}^{+\infty} V_j = \{0\}$,
4. A *scaling function* $\phi(x) \in V_0$ exists such that the set of functions $\{\phi_{j,l}(x) \mid l \in Z\!\!Z\}$, with $\phi_{j,l}(x) = \sqrt{2^j}\,\phi(2^j x - l)$, is a Riesz basis of V_j.

As a result there is a sequence $\{h_k \mid k \in Z\!\!Z\}$ such that the scaling function satisfies a *refinement equation*

$$\phi(x) = 2\sum_k h_k\,\phi(2x-k). \qquad (1)$$

Define W_j now as a complementary space of V_j in V_{j+1}, such that $V_{j+1} = V_j \oplus W_j$ ($\oplus$ stands for direct sum) and, consequently,

$$\bigoplus_j W_j = L^2(I\!R).$$

Note that this definition of W_j as a complementary space is non unique.

A function $\psi(x)$ is a *wavelet* if the set of functions $\{\psi(x-l) \mid l \in Z\!\!Z\}$ is a Riesz basis of W_0. The set of wavelet functions $\{\psi_{j,l}(x) \mid l, j \in Z\!\!Z\}$ is then a Riesz basis of $L^2(I\!R)$. Since the wavelet is an element of V_1, it too satisfies a refinement relation,

$$\psi(x) = 2\sum_k g_k\, \phi(2x-k). \tag{2}$$

There are dual functions $\tilde{\phi}_{j,l}(x) = \sqrt{2^j}\tilde{\phi}(2^jx - l)$ and $\tilde{\psi}_{j,l}(x) = \sqrt{2^j}\tilde{\psi}(2^jx-l)$ that exist so that the projection operators P_j and Q_j onto V_j and W_j, respectively, are given by

$$P_jf(x) = \sum_l \langle f, \tilde{\phi}_{j,l}\rangle\, \phi_{j,l}(x) \quad \text{and} \quad Q_jf(x) = \sum_l \langle f, \tilde{\psi}_{j,l}\rangle\, \psi_{j,l}(x).$$

The basis functions and dual functions are biorthogonal,

$$\langle \phi_{j,l}, \tilde{\phi}_{j,l}\rangle = \delta_{l-l'} \quad \text{and} \quad \langle \psi_{j,l}, \tilde{\psi}_{j',l'}\rangle = \delta_{j-j'}\delta_{l-l'}. \tag{3}$$

If the basis functions are orthogonal, they coincide with the dual functions and the projections are orthogonal.

The dual scaling function and wavelet satisfy

$$\tilde{\phi}(x) = 2\sum_k \tilde{h}_k\, \tilde{\phi}(2x-k), \qquad \tilde{\psi}(x) = 2\sum_k \tilde{g}_k\, \tilde{\phi}(2x-k), \tag{4}$$

and

$$\tilde{\phi}(2x-k) = \sum_l h_{k-2l}\, \tilde{\phi}(x-l) + \sum_l g_{k-2l}\, \tilde{\psi}(x-l). \tag{5}$$

Taking the Fourier transform of the refinement equations (1) and (2) yields

$$\hat{\phi}(\omega) = h(\omega/2)\,\hat{\phi}(\omega/2) \quad \text{with} \quad h(\omega) = \sum_k h_k\, e^{-ik\omega}$$

and

$$\hat{\psi}(\omega) = g(\omega/2)\,\hat{\psi}(\omega/2), \quad \text{with} \quad g(\omega) = \sum_k g_k\, e^{-ik\omega}.$$

Here $h(\omega)$ and $g(\omega)$ are 2π-periodic functions that correspond to discrete filters. Similar definitions and equations hold for the dual functions. A necessary condition for biorthogonality is then

$$\forall \omega \in I\!R : \ \tilde{m}(\omega)\overline{m^t(\omega)} = 1,$$

where

$$m(\omega) = \begin{bmatrix} h(\omega) & h(\omega+\pi) \\ g(\omega) & g(\omega+\pi) \end{bmatrix}$$

and similarly for $\tilde{m}(\omega)$. The existence of the dual filters is guaranteed by the following lemma:

Lemma 1 *The space generated by the set of functions $\{\psi_{j,l} \mid l \in \mathbb{Z}\}$ complements V_j in V_{j+1} if and only if $\delta(\omega) = \det m(\omega)$ does not vanish.*

The following statements are now equivalent :

- The dual wavelet has M vanishing moments.
- Any polynomial with degree less than M can be written as a linear combination of the functions $\phi_{j,l}(x)$ with $l \in \mathbb{Z}$.
- If $f \in \mathcal{C}^M$, then the error of the approximation $\|f - P_j f\|$ decays as $\mathcal{O}(h^M)$ with $h = 2^{-j}$.

These statements are also equivalent with the Strang-Fix condition [10].

The fast wavelet transform

Since V_j is equal to $V_{j-1} \oplus W_{j-1}$, a function $v_j \in V_j$ can be written uniquely as the sum of a function $v_{j-1} \in V_{j-1}$ and a function $w_{j-1} \in W_{j-1}$:

$$\begin{aligned} v_j(x) &= \sum_k \nu_{j,k}\, \phi_{j,k}(x) = v_{j-1}(x) + w_{j-1}(x) \\ &= \sum_l \nu_{j-1,l}\, \phi_{j-1,l}(x) + \sum_l \mu_{j-1,l}\, \psi_{j-1,l}(x). \end{aligned}$$

There is a one-to-one relationship between the coefficients in the different representations. The decomposition formulae can be found using (4):

$$\nu_{j-1,l} = \sqrt{2} \sum_k \tilde{h}_{k-2l}\, \nu_{j,k}, \quad \text{and} \quad \mu_{j-1,l} = \sqrt{2} \sum_k \tilde{g}_{k-2l}\, \nu_{j,k}.$$

The reconstruction step involves calculating the $\nu_{j,k}$ from the $\nu_{j-1,l}$ and the $\mu_{j-1,l}$. Using (5) we have

$$\nu_{j,k} = \sqrt{2} \sum_l h_{k-2l}\, \nu_{j-1,l} + \sqrt{2} \sum_l g_{k-2l}\, \mu_{j-1,l}.$$

When applied recursively, these formulae define a transformation, the *fast wavelet transform* [8, 11]. The decomposition step consists of applying a low-pass ($\tilde{h}$) and a band-pass ($\tilde{g}$) filter followed by downsampling (i.e. retaining only the even index samples). The reconstruction consists of upsampling (i.e. adding a zero between every two samples) followed by filtering and addition. Note that the filter coefficients of the fast wavelet transform are given by the coefficients of the refinement equations.

There are many constructions of wavelets. Here we shall only consider compactly supported wavelets as in [12, 13]. In this case the filters used in the fast wavelet transform are finite impulse response filters and a fast accurate implementation is assured.

General idea

We shall assume that L is self-adjoint and positive definite and, in particular, we can write

$$L = V^* V,$$

where V^* is the adjoint of V. We call V the *square root operator* of L. Suppose that $\{\Psi_{j,l}\}$ and $\{\Psi^*_{j,l}\}$, for an appropriate range of indices, are bases for $\mathcal{S}$ and $\mathcal{S}^*$ respectively. The entries of the stiffness matrix are then given by

$$\langle\langle \Psi_{j,l}, \Psi^*_{j',l'} \rangle\rangle = \langle L\Psi_{j,l}, \Psi^*_{j',l'} \rangle = \langle V\Psi_{j,l}, V\Psi^*_{j',l'} \rangle .$$

Now, the idea is to let

$$\Psi_{j,l} = V^{-1}\psi_{j,l} \quad \text{and} \quad \Psi^*_{j,l} = V^{-1}\tilde{\psi}_{j,l} ,$$

where ψ and $\tilde{\psi}$ are the wavelets of a classical multiresolution analysis. Because of the biorthogonality (3), the stiffness matrix becomes a diagonal matrix which can trivially be inverted. This avoids the use of an iterative algorithm. We will call the Ψ and Ψ^* functions the *operator wavelets* and the ψ functions the *original wavelets*. The operator wavelets are biorthogonal with respect to the operator inner product, a property we refer to as *operator biorthogonal.*

This idea can be powerful, but there are a few problems. First of all one has to check whether the operator wavelets still provide an multiresolution analysis where the successive approximations to a general function converge sufficiently fast (cfr the Strang-Fix condition). Secondly one has to construct a fast wavelet transform for this operator multiresolution analysis. We want operator wavelets to be compactly supported and to be able to construct compactly supported operator scaling functions $\Phi_{j,l}$. We will see that the latter is not as simple as just applying V^{-1} to the original scaling functions.

The analysis is relatively straightforward for simple constant coefficient operators such as the Laplace and polyharmonic operator. For more general constant coefficient operators, we will show that one needs to modify the construction of the original wavelets for the operator wavelets to satisfy all the desired properties. We will discuss the Helmholz operator as a typical example. At the end of the paper we shall consider a variable coefficient operator.

A similar idea was described in [14, 15]. However there only the operator wavelets of different levels are operator orthogonal and not the ones from the same level. As a result, one does not obtain a full diagonalization, but rather a decoupling of equations corresponding to different levels.

Our idea is different from the technique presented in [16]. There wavelets are used to efficiently compute the inverse of the matrix that comes from a finite difference discretization. It is also shown that the wavelets provide a diagonal preconditioner which yields uniformly bounded condition numbers.

In [17, 18] antiderivates of wavelets are used in a Galerkin method. This parallels our construction in the case of the Laplace or polyharmonic operator.

LAPLACE OPERATOR

The one dimensional Laplace operator and its square root are

$$L = -D^2 \quad \text{and} \quad V = D.$$

The associated *operator inner product* is therefore $\langle\langle u, v \rangle\rangle = \langle u', v' \rangle$. Since the action of V^{-1} is simply taking the antiderivative here, we define the operator wavelets as

$$\Psi(x) = \int_{-\infty}^{x} \psi(t)\, dt, \quad \text{and} \quad \Psi^*(x) = \int_{-\infty}^{x} \tilde{\psi}(t)\, dt.$$

The operator wavelets are compactly supported because the integral of the original wavelets has to vanish. Also translation and dilation invariance is preserved, so we define

$$\Psi_{j,l}(x) = \Psi(2^j x - l) \quad \text{and} \quad \Psi^*_{j,l}(x) = \Psi^*(2^j x - l).$$

It is then easy to see that

$$\langle\langle \Psi^*_{j,l}, \Psi_{j',l'} \rangle\rangle = 2^j\, \delta_{j-j'}\, \delta_{l-l'} \quad \text{for} \quad j, j', l, l' \in \mathbb{Z}.$$

This means that the stiffness matrix is diagonal with powers of 2 on its diagonal.

We now need to find an operator scaling function Φ. The antiderivative of the original scaling function is not compactly supported and hence not suited. We instead construct the operator scaling function Φ by taking the convolution of the original scaling function with the indicator function on $[0, 1]$,

$$\Phi = \phi * \chi_{[0,1]},$$

and similarly for the dual functions. We will show that these functions indeed generate a multiresolution analysis. To this end define

$$V_j = \text{clos span}\, \{\Phi_{j,l} \mid l \in \mathbb{Z}\} \quad \text{and} \quad W_j = \text{clos span}\, \{\Psi_{j,l} \mid l \in \mathbb{Z}\}.$$

We show that the V_j spaces are nested and that W_j complements V_j in V_{j+1}.

In the Fourier domain we have

$$\hat{\Phi}(\omega) = \frac{1 - e^{-i\omega}}{i\omega}\, \hat{\phi}(\omega) \quad \text{and} \quad \hat{\Psi}(\omega) = \frac{1}{i\omega}\, \hat{\psi}(\omega).$$

A simple calculation shows that the operator scaling function satisfies a refinement equation

$$\hat{\Phi}(\omega) = \hat{\Phi}(\omega/2)\, H(\omega/2) \quad \text{with} \quad H(\omega) = \frac{1 + e^{-i\omega}}{2}\, h(\omega).$$

Consequently, the V_j spaces are nested. If we can find a function G such that

$$\hat{\Psi}(\omega) = \hat{\Phi}(\omega/2)\, G(\omega/2),$$

then this implies that W_j is a subset of V_{j+1}. It is easy to see that this holds with

$$G(\omega) = \frac{1}{2(1-e^{-i\omega})} g(\omega).$$

This function is well defined because $g(0) = 0$.

The space W_j complements V_j in V_{j+1} if

$$\Delta(\omega) = \det \begin{bmatrix} H(\omega) & H(\omega+\pi) \\ G(\omega) & G(\omega+\pi) \end{bmatrix}$$

does not vanish, see lemma 1. In fact, we readily see that $\Delta(\omega) = \delta(\omega)/4$, and this cannot vanish since ϕ and ψ generate a multiresolution analysis. The construction of the dual functions Φ^* and Ψ^* from $\tilde{\phi}$ and $\tilde{\psi}$ is competely similar. The coefficients of the trigonometric functions H, H^*, G and G^* now define a fast wavelet transform.

Note that there is no reason why the operator scaling functions should be operator biorthogonal and in fact one can prove that this never happens. Note also that if true, this property would make the use of wavelets superfluous.

Algorithm

We will describe the algorithm in the case of periodic boundary conditions. This implies that the basis functions on the interval $[0,1]$ are just the periodization of the basis functions on the real line.

Let $\mathcal{S} = V^n$ and consider the basis $\{\Phi_{n,l} \mid 0 \leqslant l < 2^n\}$. Define vectors b and x such that

$$b_l = \langle f, \Phi^*_{n,l} \rangle, \quad \text{and} \quad u = \sum_{l=0}^{2^n-1} x_l \, \Phi_{n,l}.$$

The Galerkin method with this basis then yields a system

$$A x = b \quad \text{with} \quad A_{k,l} = \langle\langle \Phi_{n,l}, \Phi_{n,k} \rangle\rangle.$$

As we mentioned earlier, the matrix A cannot be diagonal. Also its condition number grows as $\mathcal{O}(2^{2n})$. Consider now the decomposition

$$V_n = V_0 \oplus W_0 \oplus \cdots \oplus W_{n-1},$$

and the corresponding wavelet basis. The space V_0 has dimension one and contains constant functions. We now switch to a one index notation such that the sets

$$\{1, \Psi_{j,l} \mid 0 \leqslant j < n,\ 0 \leqslant l < 2^j\} \quad \text{and} \quad \{\Psi_k \mid 0 \leqslant k < 2^n\}$$

coincide. Define the vectors $\tilde{b}$ and $\tilde{x}$ such that

$$\tilde{b}_l = \langle f, \Psi^*_l \rangle \quad \text{and} \quad u = \sum_{l=0}^{2^n-1} \tilde{x}_l \, \Psi_l.$$

We know that there exists matrices T and T^* such that

$$\tilde{b} = T^* b \quad \text{and} \quad x = T \tilde{x}$$

The matrix T^* corresponds to the fast wavelet transform decomposition with filters H^* and G^* and T corresponds to reconstruction with filters H and G. The complexity of the matrix vector multiplication is $\mathcal{O}(N)$, $N = 2^n$. In the wavelet basis the system becomes

$$\tilde{A}\,\tilde{x} = \tilde{b} \quad \text{with} \quad \tilde{A} = T^* A T \quad \text{and} \quad \tilde{A}_{k,l} = \langle\langle \Psi_{n,l}, \Psi_{n,k} \rangle\rangle.$$

Since $\tilde{A}$ is diagonal, it can be trivially inverted and the solution is then given by

$$x = T\tilde{A}^{-1}T^* b.$$

This means that one has to calculate the wavelet decomposition of the right hand side, divide each coefficient by its corresponding diagonal element and reconstruct to find the solution. The complexity is $\mathcal{O}(N)$.

The constant basis function of V_0 has a zero as diagonal element and its coefficient is thus undetermined. Note that this leads to an inconsistency if the integral of f does not vanish.

Boundary conditions

Our general idea to deal with boundary conditions is to let the operator wavelets satisfy the homogeneous boundary conditions and to let the component in the V_0 space satisfy the imposed boundary conditions. This requires the use of special boundary wavelets as described in [19]. With only a slight change of basis one can then incorporate Dirichlet, Neumann, mixed and periodic boundary conditions. The details of this construction go beyond the scope of this paper. We will describe the construction in some specific cases.

Example

In this section we shall take a look at a simple example, namely the basis we get by starting from the Haar multiresolution analysis, where

$$\phi = \chi_{[0,1]} \quad \text{and} \quad \psi(x) = \phi(2x) - \phi(2x-1).$$

Define the *hat function* as

$$\Lambda = \chi_{[0,1]} * \chi_{[0,1]}, \quad \text{such that} \quad \Phi = \Lambda \quad \text{and} \quad \Psi(x) = \Lambda(2x).$$

The original wavelets are orthogonal and as a consequence the basis functions and dual functions coincide.

The operator scaling functions can represent linears which means they satisfy the Stang-Fix condition with $M = 2$ and the convergence is of order h^2. One can prove that higher order wavelets with more vanishing moments (M) will in general not yield faster convergence because the solution u is not smooth enough. The underlying reason is that the solution u belongs to the Sobolev space W_2. One can get faster convergence only by imposing extra regularity conditions on the right hand side. So in a way this basis seems to be the most natural one to work with. Note that these piecewise linear basis functions are local solutions of the homogeneous equation such that the

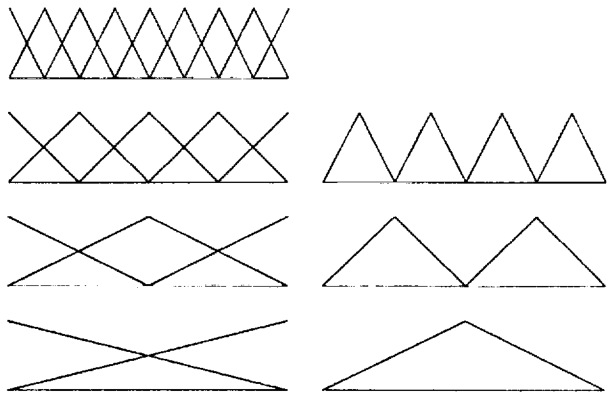


Figure 1: Basis for Dirichlet problem.

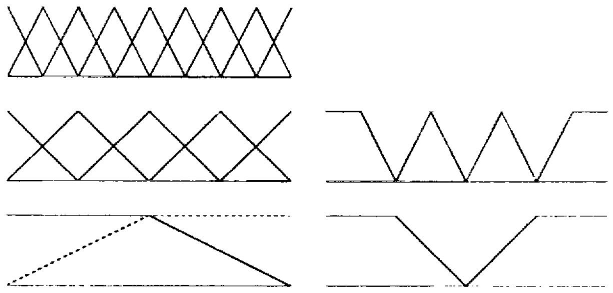


Figure 2: Basis for Neumann problem.

operator scaling functions and wavelets are V-splines. This basis also coincides with Yserentant's hierarchical basis.

Figure 1 shows the basis functions in the case of Dirichlet boundary conditions and $n = 3$. The left part are the bases for the spaces V_0 up to V_3 while the right part are the bases for W_0 up to W_2, which provide the diagonalization. The coefficients of the two functions in the V_0 space are determined by the boundary conditions. The fast wavelet transform differs from the periodic algorithm here in the sense that different coefficients are used for the wavelets at the boundary. Note the "half hat" functions here. The basis in case of the Neumann problem is shown in figure 2. The boundary conditions are handled by the two functions in the V_1 space. Again the coefficient of the constant is undetermined. The algorithm leads to an inconsistency in case the integral of f is not equal to $u'(1) - u'(0)$. Note that in both cases the operator wavelets satisfy the homogeneous boundary conditions.

MORE GENERAL CONSTANT COEFFICIENT OPERATORS

The polyharmonic operator

The polyharmonic equation is defined as

$$-u^{(2m)} = f,$$

and the square root operator is now $V = D^m$. The operator scaling function Φ is now m times the convolution of the original scaling function ϕ with the box function and the operator wavelet Ψ is m times the antiderivative of the original wavelet ψ. In order to get a compactly supported wavelet, the original wavelet now needs to have at least m vanishing moments, a property which can be satisfied by all known wavelet families. The construction and algorithm are then completely similar

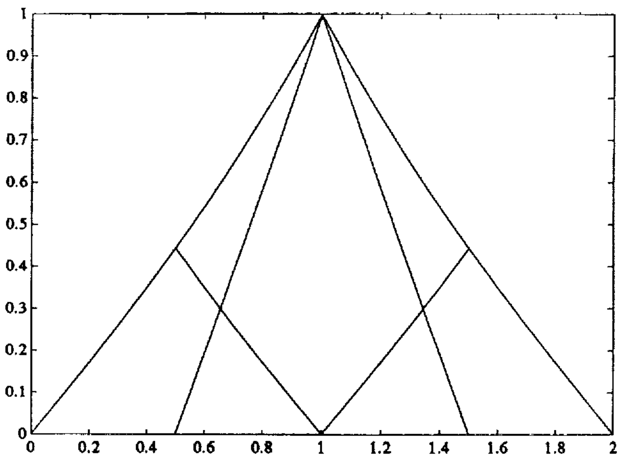


Figure 3: The refinement relation for the piecewise exponentials.

to the case of the Laplace operator.

The Helmholz operator

The general definition of the one dimensional Helmholz operator is:

$$L = -D^2 + k^2 \;\text{ such that }\; V = D + k.$$

Here we shall assume that $k = 1$ which can always be obtained from a simple transformation. Observe that $V = D + I = e^{-x}De^{x}$ and thus $V^{-1} = e^{-x}D^{-1}\,e^{x}$. One easily verifies that applying V^{-1} to a wavelet will not necessarily yield a compactly supported function since $e^{x}\psi_{j,l}$ in general does not have a vanishing integral. Therefore we let $\Psi_{j,l} = V^{-1}e^{-x}\psi_{j,l} = e^{-x}D^{-1}\psi_{j,l}$. If $\psi_{j,l}$ has a vanishing integral, then $\Psi_{j,l}$ is compactly supported.

In order to diagonalize the stiffness matrix, the original wavelets now need to be orthogonal with respect to a weighted inner product with weight function e^{-2x} because

$$\langle\langle\, \Psi_{j,l}, \Psi^{*}_{j',l'} \,\rangle\rangle \;=\; \int_{-\infty}^{+\infty} e^{-2x}\, \psi_{j,l}(x)\, \tilde{\psi}_{j',l'}(x)\, dx.$$

Finding such wavelets is a hard problem to solve in general. Inspired by the Haar basis, we construct a solution where the orthogonality of the wavelets on each level immediately follows from their disjoint support, by letting $\operatorname{supp}\psi_{j,l} \;=\; [2^{-j}l, 2^{-j}(l+1)]$. To get orthogonality between the different levels, we need that V_j is orthogonal to $W_{j'}$ for $j' \geqslant j$ or

$$\int_{-\infty}^{+\infty} e^{-2x}\, \phi_{j,l}(x)\, \tilde{\psi}_{j',l'}(x)\, dx \;=\; 0 \quad \text{for} \quad j' \geqslant j.$$

We now let the scaling function coincide with e^{2x} on the support of the finer scale wavelets,

$$\phi_{j,l} \;=\; e^{2x}\, \chi_{j,l},$$

where $\chi_{j,l}$ is the indicator function on the interval $[2^{-j}l, 2^{-j}(l+1)]$, normalized such that the integral of the scaling functions is a constant. As in the Haar case we choose the wavelets as

$$\psi_{j,l} = \phi_{j+1,2l} - \phi_{j+1,2l+1},$$

so that they have a vanishing integral. The orthogonality between levels now follows from the fact that the scaling functions coincide with e^{2x} on the support of the finer scale wavelets, and from the vanishing integral of the wavelets

$$\int_{-\infty}^{+\infty} e^{-2x}\,\phi_{j,l}(x)\,\tilde{\psi}_{j',l'}(x)\,dx = \int_{-\infty}^{+\infty} \chi_{j,l}(x)\,\tilde{\psi}_{j',l'}(x)\,dx = \int_{-\infty}^{+\infty} \tilde{\psi}_{j',l'}(x)\,dx = 0.$$

One can see that the operator wavelets are now piecewise hyperbolic functions (piecewise combinations of e^{x} and e^{-x}). The scaling functions are chosen as

$$\Phi_{j,l} = e^{-x}D^{-1}(\phi_{j,l} - \phi_{j,l+1}) \text{ so that } \Psi_{j,l} = \Phi_{j+1,2l}.$$

With the right normalization, one gets

$$\Phi_{j,l}(x) = \begin{cases} \dfrac{\sinh(x - l2^{-j})}{\sinh(2^{-j})} & \text{for } x \in [l2^{-j}, (l+1)2^{-j}] \\ \dfrac{\sinh((l+2)2^{-j} - x)}{\sinh(2^{-j})} & \text{for } x \in [(l+1)2^{-j}, (l+2)2^{-j}] \\ 0 & \text{elsewhere.} \end{cases}$$

The operator scaling functions on one level are translates of each other but the ones on different levels are no longer dilates of each other. They are supported on exactly the same sets as the ones in figure 1 and they roughly look similar. The operator scaling functions satisfy a refinement relation

$$\Phi_{j,l} = \sum_{k=0}^{2} H_k^j\,\Phi_{j+1,2l+k},$$

with

$$H_0^j = H_2^j = \sinh(2^{-j-1})/\sinh(2^{-j}) \quad\text{and}\quad H_1^j = 1.$$

Figure 3 shows the refinement relation for the scaling functions. The 3 finer scale functions are not the dilates of the coarse scale one but they still add up to it.

The Helmholz operator in this basis of hyperbolic wavelets again is diagonal and the algorithm is completely similar to the Laplace case. The only difference in implementation is that the filter coefficients H_k^j used in the fast wavelet transform now depend on the level.

Note that these functions again are V-splines and, in a way, are the most natural to work with. Also note that

$$\lim_{j\to\infty} \phi_{j,0}(2^{-j}x) = \Lambda(x).$$

Despite the fact that the Strang-Fix conditions are not satisfied, one can prove that the convergence is still of order h^2.

So we can conclude that a wavelet transform can diagonalize constant coefficient operators similar to the Fourier transform. The resulting algorithm is a little faster ($\mathcal{O}(N)$ instead of

$\mathcal{O}(N \log N)$). This gain in speed is a consequence of the subsampling of the coarser levels in the wavelet transform (the ones that correspond to the low frequency components of the solution) which is not present in the Fourier transform. Also boundary conditions are taken care of more easily than in the Fourier case.

VARIABLE COEFFICIENTS

Naturally, the next question is how to use wavelets for variable coefficient operators. The underlying reason why wavelets can diagonalize constant coefficient operators is their locality in the frequency domain. We want to understand if we can exploit the localization in space to diagonalize variable coefficient operators. The answer is (perhaps quite surprisingly) yes and this really justifies the use of wavelets for differential equations. No other technique (to our knowledge) has been able to accomplish this.

We take a closer look at the following operator

$$L = -D\,p^2(x) D,$$

where p is sufficiently smooth and positive. The square root is now $V = p\,D$ and $V^{-1} = D^{-1}\,1/p$. The rest of the analysis is very similar to the case of the Helmholz operator. Applying V^{-1} directly to a wavelet does not yield a compactly supported function. We therefore take $\Psi_{j,l} = V^{-1}\,p\,\psi_{j,l}$ which implies that the wavelets need to be (bi)orthogonal with respect to a weighted inner product with p^2 as weight function. We use the same trick as for the Helmholz equation to construct such functions. This means that we let the scaling functions $\phi_{j,l}$ coincide with $1/p^2$ on the dyadic interval $[2^{-j}l, 2^{-j}(l+1)]$ and normalize them such that they have a constant integral. We then take the wavelets $\psi_{j,l}$ to be equal to $\phi_{j+1,2l} - \phi_{j+1,2l+1}$ so they have a vanishing integral and the operator wavelets are compactly supported. The operator wavelets are now piecewise functions that locally look like $AP + B$ where P is the antiderivative of $1/p^2$ and again are V-splines. Their support also coincides with the support of the functions of figure 1, and since p is smooth they will converge to hat functions as the level goes to infinity. The operator wavelets are neither dilates nor translates of one function, since their behavior locally depends on p. This is not a problem because they still generate a multiresolution analysis and satisfy refinement relations. The coefficients in the fast wavelet transform are now different everywhere and they depend in a very simple way on the Haar wavelet transform of $1/p^2$. The entries of the diagonal stiffness matrix can be calculated from the wavelet transform of $1/p^2$. The algorithm is completely similar to previous cases and is of order N. Boundary conditions are as easy to handle as in the case of the Laplace operator. Note that the operator scaling functions do not satisfy the Strang-Fix conditions. It is however again possible to prove that the method has a convergence of order h^2. As mentioned earlier, higher convergence orders can not be obtained in general.

NUMERICAL EXAMPLE

We solve the equation

$$-D\,e^{x^2}\,D\,u(x) = e^{x^2}\left(\sin(x)(3x^2-2) + \cos(x)(2x-2x^3)\right)/x^3, \quad \text{with} \quad u(0)=1 \quad \text{and} \quad u(1)=\sin(1),$$

l	L_∞ error
1	1.22e-02
2	3.37e-03
3	8.66e-04
4	2.18e-04
5	5.45e-05
6	1.36e-05
7	3.41e-06
8	8.52e-07
9	2.13e-07

such that the exact solution is given by $u(x) = \sin(x)/x$. The L_∞ error of the numerically computed solution is a function of the number of levels (l) shown in the above table . Each time the number of levels is increased the error is divided almost exactly by a factor of 4, which agrees with the $\mathcal{O}(h^2)$ convergence.

CONCLUSION

In this paper we showed how wavelets can be adapted to be useful in the solution of differential equations. Like the Fourier transform, wavelets can diagonalize constant coefficient operators. The resulting algorithm is slightly faster. The main result however is that even non-constant coefficient operators can be diagonalized with the right choice of basis which evidently yields a much faster algorithm than more classical iterative methods.

This technique can also be applied to the solution of implicit time stepping discretizations of equations of the form $\partial u/\partial t = Lu + f$ even when L is non-linear. Future research includes the study of non self adjoint operators where a splitting $L = V\tilde{V}^*$ is needed and the study of the possible generalization of these ideas to partial differential equations.

ACKNOWLEDGEMENT

We would like to thank Ilona Weinreich for providing unpublished preprints, Marius Mitrea and Gunnar Peters for several interesting discussions and Van Henson for valuable comments on the paper.

REFERENCES

[1] W. Hackbusch. *Multi-Grid Methods and Applications*. Springer-Verlag, Berlin, 1985.

[2] H. Yserentant. On the multi-level splitting of finite element spaces. *Numer. Math.*, 49:379–412, 1986.

[3] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 55:1–22, 1990.

[4] W. Dahmen and A. Kunoth. Multilevel preconditioning. *Numer. Math.*, 63(2):315–344, 1992.

[5] P. Oswald. On a hierarchical basis multilevel method with nonconforming P1 elements. *Numer. Math.*, 62:189–212, 1992.

[6] C. K. Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[7] I. Daubechies. *Ten Lectures on Wavelets*. Number 61 in CBMS-NSF Series in Applied Mathematics. SIAM Publications, Philadelphia, 1992.

[8] S. G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. on Acoust. Signal Speech Process.*, 37(12):2091–2110, 1989.

[9] Y. Meyer. *Ondelettes et Opérateurs I. Ondelettes*. Hermann, Paris, 1990.

[10] G. Strang and G. Fix. A Fourier analysis of the finite element variational method. In *Constructive Aspects of Functional Analysis*, Rome, 1973. Edizione Cremonese.

[11] S. G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(IR)$. *Trans. Amer. Math. Soc.*, 315(1):69–87, 1989.

[12] A. Cohen, I. Daubechies, and J. Feauveau. Bi-orthogonal bases of compactly supported wavelets. To appear in Comm. Pure and Appl. Math.

[13] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.*, 41:909–996, 1988.

[14] S. Dahlke and I. Weinreich. Wavelet bases adapted to pseudo-differential operators. Technical report, RWTH Aachen, 1992.

[15] S. Dahlke and I. Weinreich. Wavelet-Galerkin-methods: An adapted biorthogonal wavelet basis. *Constructive approximation*, 9(2):237–262, 1993.

[16] G. Beylkin. On wavelet-based algorithms for solving differential equations. Preprint University of Colorado at Boulder, ftp from newton.colorado.edu.

[17] R. A. Lorentz and W. R. Madych. Spline wavelets for ordinary differential equations. Preprint Geselschaft für Mathematik und Datenverarbeitung, St. Augustin, Germany, 1990.

[18] J.-C. Xu and W.-C. Shann. Galerkin-wavelet methods for two-point boundary value problems. *Numer. Math.*, 63(1):123–142, 1992.

[19] L. Andersson, N. Hall, B. Jawerth, and G. Peters. Wavelets on closed subsects of the real line. In L. L Schumacher and G. Webb, editors, *Topics in the Theory and Applications of Wavelets*. Academic Press. To be published.

A COMPARISON OF LOCALLY ADAPTIVE MULTIGRID METHODS : L.D.C., F.A.C. AND F.I.C.

N94-23692

197151

p. 18

Khodor Khadra
Modélisation Avancée des Systèmes Thermiques et Ecoulements Réels
and Centre de Recherche en Mathématiques de Bordeaux
Université Bordeaux I, U.A. C.N.R.S. n° 226
351, cours de la Libération, 33405 Talence Cedex - France

Philippe Angot
Institut de Mécanique Statistique de la Turbulence
Université Aix-Marseille II, U.M. C.N.R.S. n° 33
12, avenue du Général Leclerc, 13003 Marseille - France

Jean-Paul Caltagirone
Modélisation Avancée des Systèmes Thermiques et Ecoulements Réels
E.N.S.C.P.B., Université Bordeaux I
351, cours de la Libération, 33405 Talence Cedex - France

SUMMARY

This study is devoted to a comparative analysis of three "Adaptive ZOOM" (ZOom Overlapping Multi-level) methods based on similar concepts of hierarchical multigrid local refinement : L.D.C. (Local Defect Correction), F.A.C. (Fast Adaptive Composite), and F.I.C. (Flux Interface Correction), which we proposed recently. These methods are tested on two examples of a bidimensional elliptic problem. We compare, for V-cycle procedures, the asymptotic evolution of the global error evaluated by discrete norms, the corresponding local errors, and the convergence rates of these algorithms.

INTRODUCTION

The need for local resolution in physical models occurs frequently in practice. Special local features of the operator coefficients, source terms, and boundary conditions can demand resolution in restricted regions of the domain that is much finer than the required global resolution. The multigrid methods with local mesh refinement provide one solution method to achieve efficient local resolution by solving problems on various locally nested grids, and by using these grids as a basis for fast solution and correction on the global basic grid of the calculation domain. Different techniques have been proposed in the literature, such as the pioneering works [1,2,3,4,5].

Therefore, the concept of "Computational Adaptive Zoom" in the context of a "Graphical and Computational Architecture" has been introduced in the field of numerical simulation in order to take the best advantage of the new capabilities of high performance computer architectures [6]. It can be viewed as a generation made automatically (i.e. in an adaptive way) or not, of some multilevel hierarchical local nested zoom grids (ZG), overlapped all over the global basic grid (BG). These grids may move all over the entire computation domain Ω during the solution phase. This concept is supposed to allow both local refinement and global correction of the basic grid solution by a successive transfer of information between the connected grids (BG) and (ZG). So it is well adapted to a graphical vision of Zoom in terms of the creation of local graphical windows where it is needed in the problem (strong gradients, discontinuities, singularities,...), but in an active sense, i.e., the basic grid solution is modified and improved as the computing is performed. This has involved us in the creation of an original engineering software package called *"AQUILON"*, still currently in development [6].

In addition, this strategy offers other interests. The goal is to combine the best features of both multigrid techniques and domain decomposition methods (in the case of overlapping grids) to provide an acceleration of the convergence rate and a good suitability for implementation on parallel computers, thus reducing the ellapse time. Moreover, another advantage is the possibilty to solve different differential problems on the grids (BG) and (ZG), which allows us to optimize both the physical and the numerical model. This can be particularly interesting for the approach of solving problems by "imbedding inside fictitious domains" associated with appropriate "control terms" for expressing the boundary conditions, as proposed in [6]. It is also possible to adopt different kinds of discretization on each grid. Thereby, the multigrid zoom methods share with the domain decomposition techniques the opportunity for obtaining precise solutions by combining solutions to problems posed on physical subdomains, or, more generally, by combining solutions to appropriately constructed continuous and discrete boundary value sub-problems.

From the numerical point of view, the strategy adopted enables us to work only on structured and uniform meshes for each grid separately, on which a moderate number of degrees of freedom is required. On each grid, a "simple and inexpensive" discretization is performed, leading to the same simple form of sparse pattern matrices (e.g. 2D block-tridiagonal). We aim at avoiding solving problems on unstructured or nonuniform composite meshes, which tends to introduce inaccuracies in the discretization, slowness in the solvers, and being surely more expensive in terms of implementation, data structures storage and CPU time. Our choice is expected to be relatively good in terms of duality quality/cost of computation for a lot of cases of moderate complexity.

MULTIGRID ZOOM ALGORITHMS

Different ZOOM algorithms will be examined and compared. We consider first the L.D.C. (Local Defect Correction) algorithm proposed by Hackbush [1]; we choose for the restriction operator a 2D bilinear interpolation one of type "full weighting control volume". The second one belongs to the class of F.A.C. (Fast Adaptive Composite Grid) methods from McCormick [5], for which the analogy with the B.E.P.S. method [4] can be noticed. We use here the "delayed correction" version of F.A.C. Only the third one, the F.I.C. (Flux Interface Correction) algorithm that we proposed more recently [7], will be briefly described hereafter.

All these Multigrid Zoom Algorithms are based on the same general principle : a successive transfer of information level by level, leading to the global correction of the initial discrete solution on each grid, and thus on the global basic grid (BG). The multilevel implementation is made in a recursive way as in the usual multigrid techniques (V-Cycles, W-Cycles, etc .) [1,3]. The resolution on each grid may be performed "exactly" or by using an inexact solve (e.g. a few iterations of a smoothing procedure).

Notations and Definitions

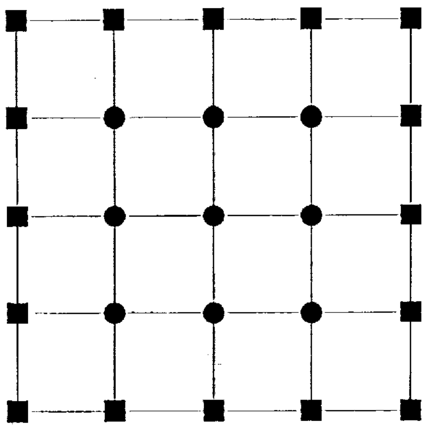
Consider the following second order non-linear elliptic boundary value problem defined on Ω a bounded, open domain in $\mathbb{R}^d$, for d = 2 or 3 :

$$(\mathcal{P})\begin{cases} L(u) \equiv \mathbf{div}(\varphi(u)) + G(u) = f(x) \qquad x \in \Omega & (1) \\ \text{well-posed boundary conditions on } \Gamma = \partial\Omega \text{ symbolically called by (BC)} \end{cases}$$

The equation (1) L(u) ≡ f is so expressed by splitting the nonlinear operator L(u) in the divergent part where φ(u) has the physical meaning of the flux density of the solution u= u(x) and the nonconservative one **G** = G(u). The relation between the solution u and the flux φ can take the general vector form $\varphi(u) = \mathbf{F(u)}$ in many systems of conservation laws, but applications will concern an advection-diffusion equation or a Navier-Stokes problem. For the experiments here, $(\mathcal{P})$ is a diffusion problem and we have $\varphi(u) = -\sigma.\mathbf{grad}u$.

In order not to have too many formal requirements and restrictions, we assume explicitly only that this equation (1) has at least one isolated solution u^* in the space $L^2(\Omega)$. All other assumptions are implicitly contained in the following considerations.

The basic notations will be those classically used in the multigrid framework [1]. We denote by ℓ the current index of the grid level $(0 \leq \ell \leq \ell^*)$, $\ell = 0$ is the level of the global basic grid (BG) which discretizes the entire calculation domain Ω, and $\ell = \ell^* \neq 0$ is the level of the most nested and finest zoom grid (ZG). Each grid of level ℓ can be characterized by :



- the open domain $\Omega_\ell = \{●\}$
- the boundary $\Gamma_\ell = \{■\}$ on which can be defined the unit outside normal vector $\mathbf{n}_\ell$
- the closure $\overline{\Omega}_\ell = \Omega_\ell \cup \Gamma_\ell$
- the mesh size h_ℓ

Each grid of level ℓ is divided into a set of control volumes $\mathcal{V}_x$ associated to the nodes $x \in \overline{\Omega}_\ell$. We denote by $\Gamma_{\ell,\ell+1}$ the interface between two successive grids of level ℓ and $\ell+1$ and we have $\forall \ell$, $\Omega_\ell \cap \Omega_{\ell+1} \neq \emptyset$. The successive mesh sizes will be taken as $h_{\ell+1} = h_\ell / 2^p$, $p \in \mathbb{N}^*$. The following notations will also be used : $A_\ell = \Omega_{\ell+1} \cap \Omega_\ell$ and $\overline{A}_\ell = \overline{\Omega}_{\ell+1} \cap \Omega_\ell$.

The transfer operators between the grids ℓ and $\ell+1$ will be called, respectively, by $R^\ell_{\ell+1}$ for the restriction operator and by $P^{\ell+1}_\ell$ for the prolongation operator. For all three algorithms, we have chosen $P^{\ell+1}_\ell$ as :

$$P^{\ell+1}_\ell : \Gamma_{\ell,\ell+1} \cap \overline{\Omega}_\ell \longrightarrow \Gamma_{\ell+1} \setminus (\Gamma_{\ell+1} \cap \Gamma)$$

which is a monodimensional linear interpolation operator defined on the interface of the grids ℓ and $\ell+1$. Each value $u_{\ell+1}$ at a node $y \in \Gamma_{\ell+1} \setminus (\Gamma_{\ell+1} \cap \Gamma)$ on the interface is obtained by a linear interpolation of the values u_ℓ at the two neighbour nodes $\mathbf{x}$ et $\mathbf{x'}$ belonging to $(\Gamma_{\ell,\ell+1} \cap \overline{\Omega}_\ell)$, and thus verifying $u_{\ell+1}(y) = u_\ell(x)$ if $y = x$.

If we denote by $L_\ell\, u_\ell = f_\ell$ the discretized equation of (1) on the grid of level ℓ , we can define the following discrete boundary value problems on Ω_ℓ :

$$(\mathcal{P}_0)\begin{cases} L_0\, u_0 = f_0 & : \text{in } \Omega_0 \\ \text{on } \Gamma_0 & : (BC) \end{cases} \quad (\mathcal{P}_\ell)\left\{\begin{array}{ll} L_\ell\, u_\ell = f_\ell & : \text{in } \Omega_\ell \\ \text{on } \Gamma_\ell \cap \Gamma & : (BC) \\ \text{on } \Gamma_\ell \setminus (\Gamma_\ell \cap \Gamma) & : u_\ell = P^\ell_{\ell-1}\, u_{\ell-1} \quad (2) \end{array}\right\}\ \ell \neq 0$$

We denote by u^k_ℓ the discrete solution obtained on the grid Ω_ℓ at the k-th iteration of the zoom algorithm, and $e^k_\ell = u^*_\ell - u^k_\ell$ the associated discrete error, where u^*_ℓ is the natural restriction of the exact solution u^* of problem $(\mathcal{P})$ on Ω_ℓ .

For $0 < \ell < \ell^*$, $\gamma(\ell)$ will represent the number of iterations of the zoom algorithm on the grid level ℓ in order to describe a whole cycle : if we have $\gamma(\ell)=1$ (respectively $\gamma(\ell)=2$), $\forall 0< \ell <\ell^*$, then V-cycles (respectively W-cycles) will be described. We have $\gamma(\ell^*) = 1$, and $\gamma(0)$ is the total number of cycles performed from the basic grid (BG) in order to obtain the so-called convergence of the zoom algorithm. When $\ell^*=1$ (i.e. for a two-grid algorithm), only V-cycles are of course carried out. The term **"No Zoom"** will be used for the resolution by "an exact solve" of problem $(\mathcal{P})$ on the basic grid (BG) of mesh size h_0 ($\ell^*=0$, k=0). The term **"Zoom"** will be used to indicate that some iterations of a multilevel zoom algorithm have been performed: $\ell^*\neq 0$, $1\leq k\leq\gamma(0)$.

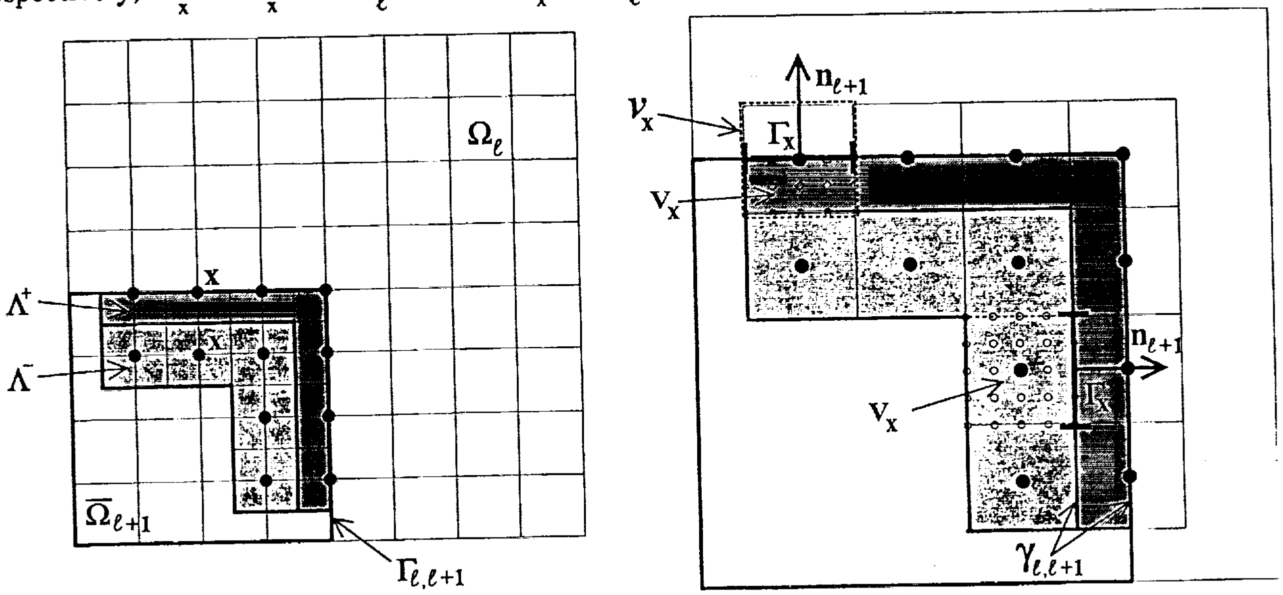
Description of The Multilevel F.I.C. Algorithm

The main idea of the two-grid FIC method for levels ℓ and $\ell+1$ is to give the opportunity to apply the local "flux residual" correction due to the whole patch level $\ell+1$, at each node $\mathbf{x} \in \overline{\Lambda}_\ell$ on the grid level ℓ. This is obtained through the expression of the local flux balance (i.e. integration) of eq.(1) over the volume $V_x = \mathcal{V}_x \cap \Omega_{\ell+1}$, between the grid levels ℓ, on one hand, and $\ell+1$ on the other hand.

Because of the consistency of the conservative discretization of the fluxes by the finite volume method, which must be respected on each grid, the outside normal fluxes of $\varphi(u)$ through an interface of two neighbour control volumes are opposite. By giving more importance to the local "flux residual", that leads to consider for the correction step on the grid level ℓ, the local flux of the defect only at each node of a boundary zone $I_{\ell,\ell+1}$ defined as the "flux correction interface". We can choose for $I_{\ell,\ell+1}$, either the stripe $\Lambda^+= \{\cup\ V_x\ ,\ \mathbf{x} \in \partial\overline{\Lambda}_\ell = \Gamma_{\ell,\ell+1} \cap \Omega_\ell\}$, or Λ^- (see further Figure) if we want the

C-4

boundary $\partial\Lambda$ to correspond to interfaces between control volumes on the grid level ℓ : we will have $V_x = \mathcal{V}_x$ in the latter case. We define on the grid level ℓ, $\partial V_x = \Gamma_x \cup \Gamma_r$, $\forall x \in I_\ell = I_{\ell,\ell+1} \cap \Omega_\ell$, where $\Gamma_x = \partial V_x \cap \partial\overline{A}_\ell \neq \emptyset$, or respectively, $\Gamma_x = \partial V_x \cap \partial A_\ell$, (mes $(\Gamma_x) = h_\ell$).



We then propose the following restriction operator on the outside normal flux through the "interface boundary" $\gamma_{\ell,\ell+1} = \{\cup\, \Gamma_x\ ,\ x \in I_\ell\}$:

$$R^\ell_{\ell+1} : \gamma_{\ell,\ell+1} \cap \overline{\Omega}_{\ell+1} \longrightarrow I_{\ell,\ell+1} \cap \Omega_\ell$$

$$R^\ell_{\ell+1}(\varphi_{\ell+1}(u).\mathbf{n}_{\ell+1})(x) = \frac{1}{\text{mes}(\Gamma_x)} \int_{\Gamma_x} \varphi_{\ell+1}(u).\mathbf{n}_{\ell+1} d\gamma \quad \forall x \in I_{\ell,\ell+1} \cap \Omega_\ell \quad (3)$$

We can then define, as in [7], the local "flux residual" correction at each node $x \in I_\ell = I_{\ell,\ell+1} \cap \Omega_\ell$ on the grid level ℓ by :

$$r_\ell(\varphi)(x) = \frac{\omega(\ell,x)(u)}{\varepsilon(\ell,x)} \left\{ R^\ell_{\ell+1}(\varphi_{\ell+1}(u)\ .\mathbf{n}_{\ell+1}) - \varphi_\ell(u).\mathbf{n}_{\ell+1} \right\}(x) \qquad (4)$$

The control parameter $\varepsilon(\ell,x)$, which has the dimension of a length, has already been encountered in order to assign Neumann and Robin (or Fourier) boundary conditions in the context of "imbedding inside a fictitious domain" [6]. Its expression is given by :

$$\varepsilon(\ell,x) = \frac{\text{mes }(V_x)}{\text{mes }(\Gamma_x)} \qquad (5)$$

A complete calculation, still not published, gives a complex expression for $\omega(\ell,x)(u)$, which is the following one in the case under subject of $\mathbf{G} \equiv 0$:

$$\omega(\ell,x)(u)^* \simeq 1 + \frac{\left(\int_{\Gamma_r} \varphi(u).\mathbf{n}\, d\gamma\right)_{\ell+1} - \left(\int_{\Gamma_r} \varphi(u).\mathbf{n}\, d\gamma\right)_{\ell}}{\left(\int_{\Gamma_x} \varphi(u).\mathbf{n}\, d\gamma\right)_{\ell+1} - \left(\int_{\Gamma_x} \varphi(u).\mathbf{n}\, d\gamma\right)_{\ell}} \tag{6}$$

We can then generate the successive iterates u_0^k by the multilevel FIC algorithm implemented in a recursive way :

```
Initialization : compute $u_0^0$
$u_0^0$ is obtained by resolution of problem $(\mathcal{P}_0)$
Iterations : compute the successive iterates $u_0^k$
for k = 1 to $\gamma(0)$ do FIC(0)
Composite re-actualization : providing $u_0^{\gamma(0)}$ on (BG) by assigning
for $\ell = \ell^* - 1$ to 0 by step of -1 : $u_\ell^{\gamma(0)}(x) = u_{\ell+1}^{\gamma(0)}(x)$  $\forall x \in A_\ell$
Procedure FIC($\ell$)
    If $\ell = \ell^*$ Then solve problem $(\mathcal{P}_{\ell^*})$ Else
    begin
      * 1st step - resolution on the grid level $\ell+1$ :
          - solve problem $(\mathcal{P}_{\ell+1})$ providing $u_{\ell+1}$
          - for k = 1 to $\gamma(\ell+1)$ do FIC($\ell+1$)
      * 2nd step - correction on the grid level $\ell$ :
          - solve problem $(\mathcal{P}_\ell)$ with $\tilde{f}_\ell = f_\ell + \chi_{I_\ell}\, r_\ell(\varphi)$
            where $r_\ell(\varphi)$ is computed by equations (3)(4)(5)(6)
              and $\chi_{I_\ell}$ is the characteristic function of $I_\ell$ in $\Omega_\ell$
    end
```

Remarks :

1) - In any case, in order to avoid the explicit calculation of $\omega(\ell,x)(u)$ by eq.(6), an economical solution is to use an approximate correction for FIC. In that version, called FIC(ω), only the flux integrals on the interface Γ_x will be evaluated by quadrature formulae (Simpson), and an average weighting factor $\omega(\ell)$ will be determined by a semi-empirical way for each grid level. Besides, it can play the role of an average relaxation parameter for the iterative zoom algorithm when $\omega(\ell)=\omega$, $\forall \ell \neq \ell^*$.

2) - In terms of domain decomposition, the two-grid FIC for levels ℓ and $\ell+1$ can be regarded as a full overlapping iterative algorithm that splits the whole composite problem in two Dirichlet/ Neumann boundary value sub-problems:

- the problem on the grid level $\ell+1$ with a Dirichlet boundary condition on the interface $\Gamma_{\ell,\ell+1}$ (2),

- the problem on the level ℓ with a condition of relaxed transmission of the flux on the interface $\gamma_{\ell,\ell+1}$ through (4), which demands the flux continuity at convergence. That condition can be considered as a Neumann boundary condition on $\gamma_{\ell,\ell+1}$ by the technique of "fictitious domain" in [6].

General Comments on the Three Algorithms

i) - The two-grid FAC method for levels ℓ and $\ell+1$ can be regarded as an iterative procedure to solve "exactly" the discrete composite problem coming from an adequate discretization of problem $(\mathcal{P})$ on the composite grid $\underline{\Omega}_\ell$ defined by the association of the grids Ω_ℓ and $\Omega_{\ell+1}$. Therefore, the principle is to apply a multigrid algorithm between the grids Ω_ℓ and $\underline{\Omega}_\ell$ on one hand, and between the grids $\underline{\Omega}_\ell$ and $\Omega_{\ell+1}$ on the other [5,4]. There is therefore a correction phase on both the grid levels ℓ and $\ell+1$ with respect to the discretization on the composite grid. In that sense, FAC can be viewed as an "exact" solver for the composite problem. Because the composite grid stencils agree with the coarse and fine grid stencils, respectively, outside and inside the refinement region, and because the correction equations are solved exactly, the composite grid residual is nonzero only at the interface.

ii) - Due to the attention needed for the nonuniform discretization of the problem on the interface zone of the composite grid, FAC method can prove to be a little difficult to implement in a more than two grids version.

iii) - On the contrary, LDC and FIC methods, which are easier implementing in the multilevel case, are only approximate solvers : they don't use a composite grid and neglect the fine grid residual correction. The former consists in the local correction of the solution defect inside A_ℓ as the latter involves a local flux residual correction through the interface $\gamma_{\ell,\ell+1}$.

iv) - Both FAC and FIC methods provide corrections by balancing fluxes computed from both coarse and fine grids across the interface. They take the best advantage of a conservative discretization of the equations, for example, by a finite volume technique.

NUMERICAL APPLICATIONS

In that context, we propose to compare three types of multigrid zoom algorithms on two examples of a linear elliptic problem $(\mathcal{P})$ presenting, respectively, a discontinuity of the operator coefficients for $(\mathcal{P}1)$ [8], and a singularity of the exact solution for $(\mathcal{P}2)$ [1] :

$$(\mathcal{P}) \begin{cases} L(u) \equiv -\mathbf{div}(\sigma(x).\mathbf{grad}u) + \alpha(x)\, u = f(x) \qquad \text{in } \Omega =]0,1[\times]0,1[\quad (1') \\ \sigma,\ \alpha > 0 \in L^{\infty}(\Omega) \quad \text{et} \quad f \in L^{2}(\Omega) \\ \text{well-posed boundary conditions on } \Gamma = \partial\Omega \text{ symbolically called by (BC)} \end{cases}$$

These problems were already tested successfully on the FIC method in [7]. Problem $(\mathcal{P}1)$ is heterogeneous and defined by $f\equiv0$, $\alpha\equiv0$, $\sigma\equiv100$ inside a disk of radius 0.1 and $\sigma\equiv1$ outside (Fig.1a). A solution computed on a very fine basic mesh (512^2) will be used as the reference solution u^*. Problem $(\mathcal{P}2)$ is defined by $f\equiv0$, $\alpha\equiv0$, $\sigma\equiv1$ (Fig.1b); the exact solution is $u^* = \ln(r)$ with $r = \sqrt{x^2 + y^2}$.

Numerical Implementation and Procedures

The discretization on each grid, independant of the geometry of the problem, is made in a conservative way by a finite volume method on a uniform Cartesian mesh. The classical five-point scheme is used providing a second order precision. The resolution of the linear systems, which are block-tridiagonal and symmetric positive definite, is performed by a fast and efficient solver : a preconditionned conjugate gradient (PCG) method (CG-SSOR) vectorized by a Red and Black numbering of unknowns. The results for two grids are obtained by an "exact" solve on each grid. The results for multilevel LDC or FIC ($\ell^* \geq 1$) are given for an "inexact" solve on each grid (including Fig.5b), i.e., a fixed number *itcg* of iterations of PCG on each grid with :

itcg=2 for $h_0 = 1/8$ *itcg*=4 for $h_0 = 1/16$ *itcg*=8 for $h_0 = 1/32$

The results are analyzed with different norms (L^{∞}, L^2, L-energy norm) of the discrete error evaluated on the global basic grid (BG, ℓ=0). We study the asymptotic evolution of the relative error norms $\xi_0^0 = \|e_0^0\| / \|u_0^*\|$ (No Zoom) and $\xi_0^{\gamma(0)} = \|e_0^{\gamma(0)}\| / \|u_0^*\|$ (after $\gamma(0)$ Zoom iterations) with $e_0^k = u_0^k - u_0^*$, which allows us to estimate an asymptotic average rate τ :

* for $\ell^* = 1$, as function of h_1 or p (for a fixed h_0)

$$\tau = \left(\frac{\xi_0^0}{\xi_0^{\gamma(0)}(p=m)} \right)^{1/m} \quad , \text{with} \quad m = \max\left\{ p \in \mathbb{N}^* \right\}$$

Here m=3 and $\gamma(0)=2$, see Tab.1, Tab.2, and Fig.2a, Fig.3, and Fig.4.

* for $\ell^* \geq 1$, as function of ℓ^* (for a fixed h_0 and p=1)

$$\tau = \left(\frac{\xi_0^0}{\xi_0^{\gamma(0)}(\ell^*=m)} \right)^{1/m} \quad , \text{with} \quad m = \max\left\{ \ell^* \in \mathbb{N}^* \right\}$$

Here m=3 and $\gamma(0)=10$, see Tab.3, and Fig.2b.

The convergence rate of LDC, FAC and FIC have been also compared (Tab.4):

* for FAC : we study the variations of the Euclidean norm of the composite residual $|\underset{\bullet}{r_0^k}(u)|_2$ for k = 1 to $\gamma(0)$ (Fig.5a), and a convergence rate ρ is calculated by a geometric mean :

$$\rho = \left(\frac{| \underset{\bullet}{r_0^{\gamma(0)}}(u)|_2}{| \underset{\bullet}{r_0^1}(u)|_2} \right)^{1/(\gamma(0)-1)}$$

* for LDC or FIC: we study the variations of quantities $\delta_0^k = |u_0^k - u_0^{k-1}|_{L^2}$ for k = 1 to $\gamma(0)$ (Fig.5b), and a convergence rate ρ is then estimated by :

$$\rho = \left(\frac{\delta_0^{\gamma(0)}}{\delta_0^1} \right)^{1/(\gamma(0)-1)}$$

Comparative Numerical Results

1) - By comparing a no-zoom method and a ZOOM one, we notice that the error globally decreases ; between two increments of p or ℓ^*, it is divided by an elevated average τ-factor of between 1.5 and 3.5 (Tab.1, Tab.2, Tab.3). For problem ($\mathcal{P}2$), the decrease is monotonic and there seems to be good analogy between the variation of the error as a function of p (for $\ell^*=1$) or of ℓ^* (for p=1), (see Fig.2a and Fig.2b). The influence of the position and dimensions of the local grids (ZG) becomes negligible as h_0 decreases [7]. Due to the choice

of discretizing on a Cartesian mesh independently of the geometry of the heterogeneity, the error for problem ($\mathcal{P}1$) does not decrease monotonically as already noticed in [7,8].

2) - In many cases, the error obtained with zoom is less than computed without zoom on a single basic grid of mesh size $h_0 \leq h_{\ell^*}$. In particular, Fig.4 shows that the local discrete error $|e_\ell^k|$ calculated point by point on the diagonal of the domain ($\mathcal{P}2$), by a two-grid FIC method (h_0=1/16, h_1=h_0/2, k=2) is globally better than the error obtained with No Zoom at the corresponding nodes of BG (ℓ=0, h_0=1/32). The former results are more accurate inside the refinement region and get closer to the latter case far from the interface. Such remarks can also be made for the discrete error norms in the other Tables or Figures. However, the error is not reduced beyond a threshold value consistent with the order of precision of the discretization schemes on the different grids (cf, the multigrid defect correction method using Richardson extrapolation [1]).

3) - The two-grid FAC and FIC methods yield error results of the same order of magnitude for both problems. These results are far better than for LDC for problem ($\mathcal{P}1$), where the flux conservation plays an important role. On the contrary, LDC yields as good results as the others for problem ($\mathcal{P}2$), and sometimes better. However, as LDC does not deal with the interface fluxes, but only works on the solution inside the open refinement region, it can become inefficient (τ = 1) if the refinement region does not contain enough coarse nodes on which the local defect correction is performed (Tab.1, Tab.2, Tab.3).

4) - The results with the version FIC(ω) for 0,1 $\leq \omega \leq$ 0.5 are nearly similar to those obtained with $\omega^*= \omega(\ell,x)(u)$ calculated by (6) (Fig.3). That could justify the interest of the approximate version FIC(ω), and particularly as a preconditioner of the discrete problem, as suggested in [4].

5) - Because of its exact character, the FAC method yields the far best convergence rate, a mean value of 0.16, nearly independant of both h_0 and h_1 (Fig.5a and Tab.4). We obtain a mean convergence rate of 0.42 for FIC(ω=0.2), just a little better than LDC with 0.50 . These convergence rates remain not very sensitive to the variations of h_0 and ℓ^* (Fig.5b and Tab.4). However, those of FIC have a noticeable tendency to become better as the number of grid levels (or ℓ^*) increases (see Tab.4).

CONCLUSION

Despite its non-exact character, FIC provides as good results as FAC, concerning the analysis of discrete errors for both the two tested problems. In particular, FAC and FIC proved to be better than LDC for problems where the flux conservation locally plays a main role.

FAC yields very good convergence rates ($\rho \simeq 0.16$), better than LDC ($\rho \simeq 0.50$) or FIC ($\rho \simeq 0.42$), but its multilevel implementation remains more difficult. However, the use of FIC as a preconditioning technique of the discrete problem is likely to be very interesting, especially for the approximate version FIC(ω) where the factor ω becomes a relaxation parameter. We are currently testing such a procedure for Navier-Stokes problems.

REFERENCES

1. **Hackbush, W.**: *Multigrid Methods and Applications*, Series in Computational Mathematics, Springer-Verlag, Berlin, 1985.

2. **Berger, M.J.; and Oliger, J.**: An Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations, J. Comp. Phys., 53, pp. 484-512, 1984.

3. **Bai, D.; and Brandt, A.**: Local Mesh Refinement Multilevel Techniques, SIAM J. Sci. Stat. Comput., Vol.8, No. 2, pp. 109-134, 1987.

4. **Bramble, J.H.; Ewing, R.E.; Pasciak, J.E.; and Schatz, A.H.**: A Preconditioning Technique for the Efficient Solution of Problems with Local Grid Refinement, Comp. Meth. Appl. Mech. Eng., 67, pp. 149-159, 1988.

5. **McCormick, S.F.**: *Multilevel Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.

6. **Angot, Ph.; and Caltagirone, J.P.**: New Graphical and Computational Architecture Concept for Numerical Simulation on Supercomputers, Proc. 2nd World Congress on Computational Mechanics, pp. 973-976, Stuttgart, 1990.

7. **Angot, Ph.; Caltagirone, J.P.; and Khadra, K.**: Une Méthode Adaptative de Raffinement Local : la Correction du Flux à l'Interface, C. R. Acad. Sci. Paris, t. 315, Série I, pp. 739-745, 1992.

8. **Angot, Ph.; and Caltagirone, J.P.**: Homogénéisation Numérique en Thermique des Structures Hétérogènes Périodiques, Proc. 4th EUROTHERM Conf., pp. 122-126 Nancy, 1988.

Tab. 1. Problem ($\mathcal{P}1$) - Two Grid Zoom $\gamma(0) = 2$ - Discrete L^2 norm of the error

h_0	NO ZOOM	$h_1 = \frac{h_0}{2^p}$	ZOOM x_1=0.375 et x_2=0.625			ZOOM x_1=0.25 et x_2=0.75		
			LDC	FAC	FIC ω = 0.55	LDC	FAC	FIC ω = 0.40
1/8	0.434E-1	p=1	0.434E-1	0.493E-1	0.210E-1	0.434E-1	0.732E-2	0.639E-2
		p=2	0.434E-1	0.790E-1	0.122E-1	0.434E-1	0.133E-1	0.106E-1
		p=3	0.434E-1	0.330E-1	0.589E-2	0.434E-1	0.289E-2	0.251E-2
		τ	**1.00**	**1.10**	**1.95**	**1.00**	**2.47**	**2.59**
1/16	0.689E-2	p=1	0.689E-2	0.167E-1	0.119E-1	0.739E-2	0.986E-2	0.106E-1
		p=2	0.689E-2	0.374E-2	0.394E-2	0.297E-2	0.123E-2	0.161E-2
		p=3	0.689E-2	0.255E-2	0.339E-2	0.214E-2	0.673E-3	0.592E-3
		τ	**1.00**	**1.39**	**1.27**	**1.48**	**2.17**	**2.27**
1/32	0.982E-2	p=1	0.136E-1	0.342E-2	0.538E-2	0.244E-2	0.160E-2	0.201E-2
		p=2	0.140E-1	0.295E-2	0.412E-2	0.174E-2	0.428E-3	0.392E-3
		p=3	0.134E-1	0.326E-2	0.331E-2	0.153E-2	0.233E-3	0.262E-3
		τ	**0.90**	**1.44**	**1.44**	**1.86**	**3.48**	**3.35**
1/64	0.192E-2							
1/128	0.648E-3							
1/256	0.193E-3							

Tab. 2. Problem ($\mathcal{P}2$) - Two Grid Zoom $\gamma(0) = 2$ - Discrete L-Energy norm of the error

h_0	NO ZOOM	$h_1 = \frac{h_0}{2^p}$	ZOOM x_1=0 et x_2=0.25			ZOOM x_1=0 et x_2=0.5		
			LDC	FAC	FIC ω = 0.20	LDC	FAC	FIC ω = 0.20
1/8	0.342E-1	p=1	0.342E-1	0.152E-1	0.168E-1	0.120E-1	0.138E-1	0.139E-1
		p=2	0.342E-1	0.817E-2	0.110E-1	0.441E-2	0.511E-2	0.553E-2
		p=3	0.342E-1	0.622E-2	0.990E-2	0.221E-2	0.250E-2	0.346E-2
		τ	**1.00**	**1.77**	**1.51**	**2.49**	**2.39**	**2.15**
1/16	0.206E-1	p=1	0.723E-2	0.829E-2	0.837E-2	0.704E-2	0.818E-2	0.819E-2
		p=2	0.270E-2	0.308E-2	0.337E-2	0.232E-2	0.282E-2	0.285E-2
		p=3	0.140E-2	0.152E-2	0.215E-2	0.627E-3	0.975E-3	0.107E-2
		τ	**2.45**	**2.38**	**2.12**	**3.20**	**2.76**	**2.68**
1/32	0.133E-1	p=1	0.454E-2	0.527E-2	0.527E-2	0.453E-2	0.526E-2	0.526E-2
		p=2	0.150E-2	0.182E-2	0.184E-2	0.149E-2	0.180E-2	0.180E-2
		p=3	0.407E-3	0.628E-3	0.699E-3	0.387E-3	0.570E-3	0.577E-3
		τ	**3.20**	**2.77**	**2.67**	**3.25**	**2.86**	**2.85**
1/64	0.888E-2							
1/128	0.607E-2							
1/256	0.420E-2							
1/512	0.294E-2							

Tab. 3. Problem ($\mathcal{P}2$) - Multilevel Zoom LDC/FIC
$\gamma(0) = 10$, $h_{\ell+1} = h_\ell/2$ $\quad 0 \le \ell \le \ell^*-1$, $x_1 = 0$ and $x_2 = 0.5$ -
Discrete L^2 norm of the error

h_0	NO ZOOM $\ell^* = 0$	number of grids	ZOOM LDC	ZOOM FIC	
				$\omega = 0.2$	$\omega = 0.35$
1/8	0.209E-1	$\ell^* = 1$	0.806E-2	0.869E-2	0.787E-2
		$\ell^* = 2$	0.440E-2	0.625E-2	0.465E-2
		$\ell^* = 3$	0.368E-2	0.606E-2	0.428E-2
		τ	**1.78**	**1.51**	**1.70**
1/16	0.105E-1	$\ell^* = 1$	0.399E-2	0.404E-2	0.394E-2
		$\ell^* = 2$	0.157E-2	0.199E-2	0.166E-2
		$\ell^* = 3$	0.102E-2	0.165E-2	0.115E-2
		τ	**2.18**	**1.85**	**2.09**
1/32	0.529E-2	$\ell^* = 1$	0.186E-2	0.202E-2	0.249E-2
		$\ell^* = 2$	0.969E-3	0.768E-3	0.809E-3
		$\ell^* = 3$	0.711E-3	0.416E-3	0.514E-3
		τ	**1.95**	**2.33**	**2.18**
1/64	0.265E-2				
1/128	0.132E-2				
1/256	0.662E-3				
1/512	0.331E-3				

Tab. 4. Problem ($\mathcal{P}2$) - Multilevel Zoom LDC/FIC - Two Grid Zoom FAC
$\gamma(0) = 10$, $h_{\ell+1} = h_\ell/2^p$ $\quad 0 \le \ell \le \ell^*-1$, $x_1 = 0$ and $x_2 = 0.5$ -
Convergence rate ρ

h_0	number of grids	ZOOM LDC	ZOOM FIC $\omega = 0.2$	ZOOM FAC		
		p = 1	p = 1	p=1	p=2	p=3
1/8	$\ell^* = 1$	0.53	0.46	0.14	0.15	0.15
	$\ell^* = 2$	0.45	0.42			
	$\ell^* = 3$	0.43	0.34			
1/16	$\ell^* = 1$	0.56	0.44	0.14	0.16	0.16
	$\ell^* = 2$	0.49	0.42			
	$\ell^* = 3$	0.48	0.39			
1/32	$\ell^* = 1$	0.50	0.47	0.15	0.17	0.18
	$\ell^* = 2$	0.51	0.44			
	$\ell^* = 3$	0.50	0.40			

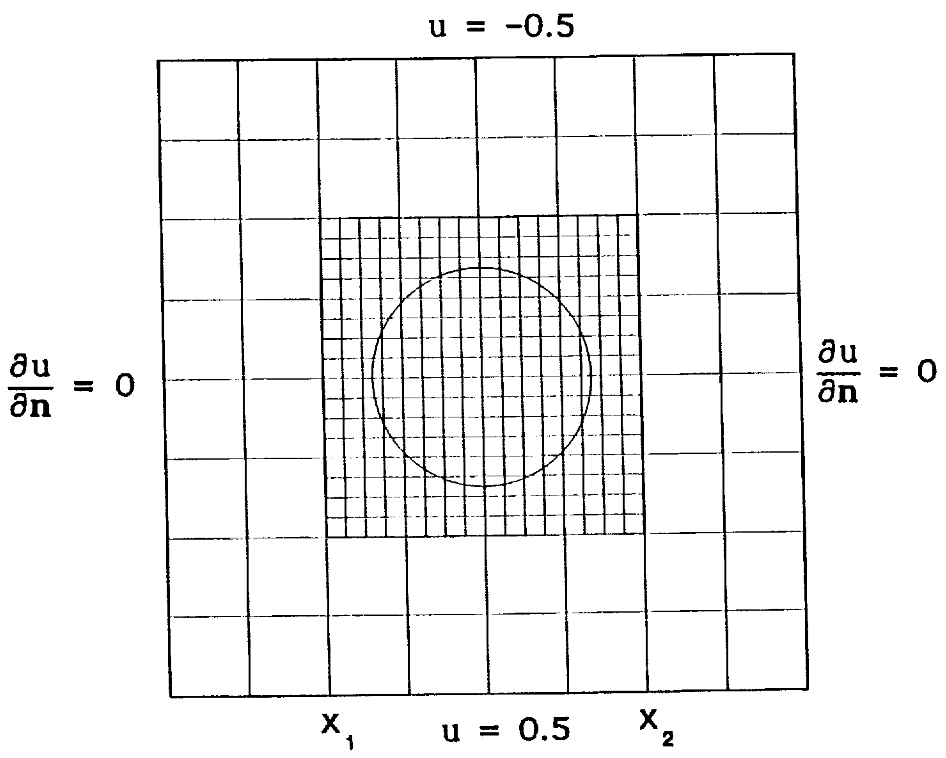

u = -0.5
$\frac{\partial u}{\partial \mathbf{n}} = 0$
$\frac{\partial u}{\partial \mathbf{n}} = 0$
X_1
u = 0.5
X_2

Fig. 1a. Problem ($\mathcal{P}1$)

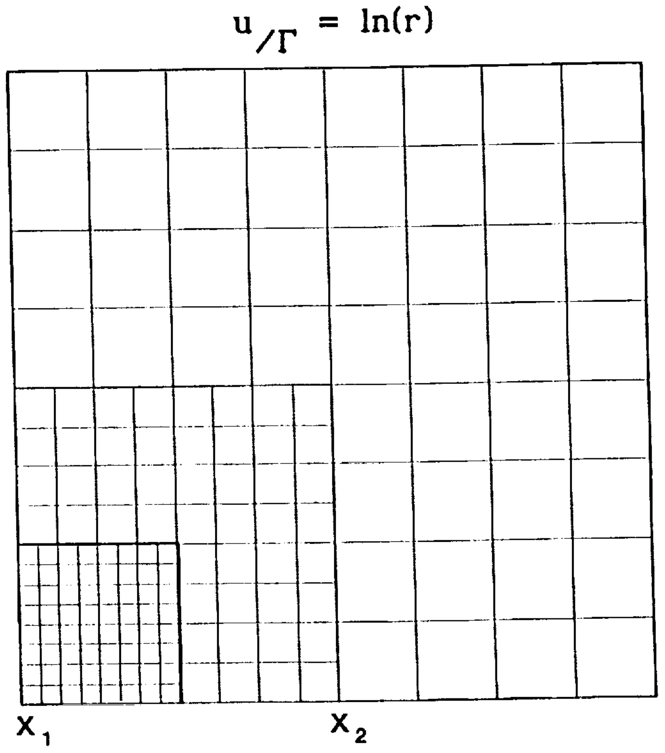


Fig. 1b. Problem ($\mathcal{P}2$)

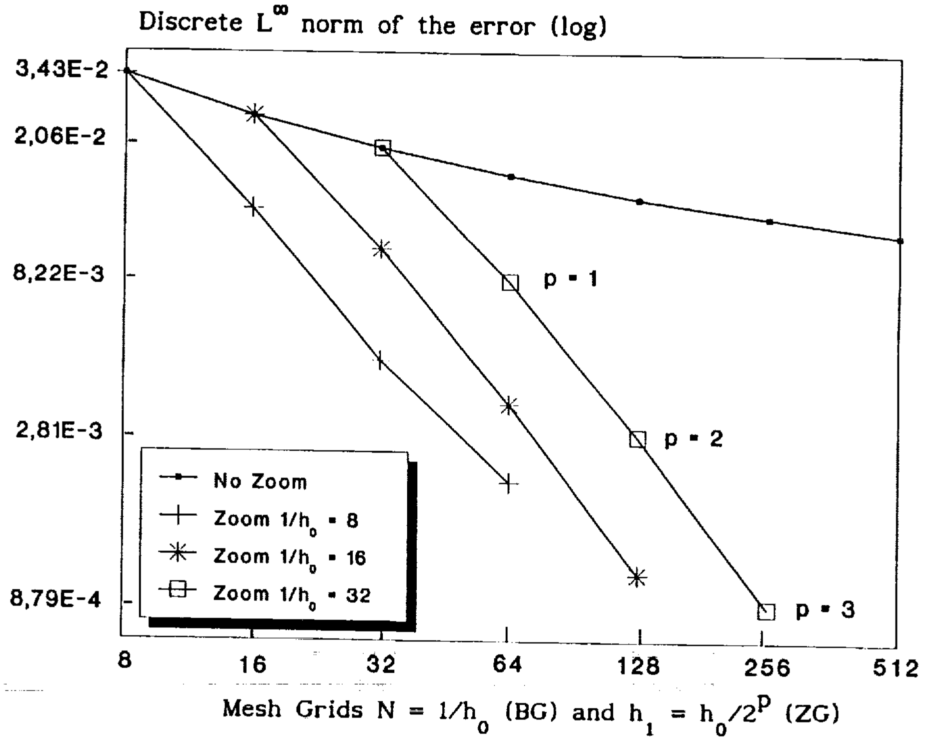


Fig. 2a. Problem ($\mathcal{P}2$) - Two Grid Zoom with FIC (ω = 0.35)
$\gamma(0) = 2 - x_1 = 0$ and $x_2 = 0.5$

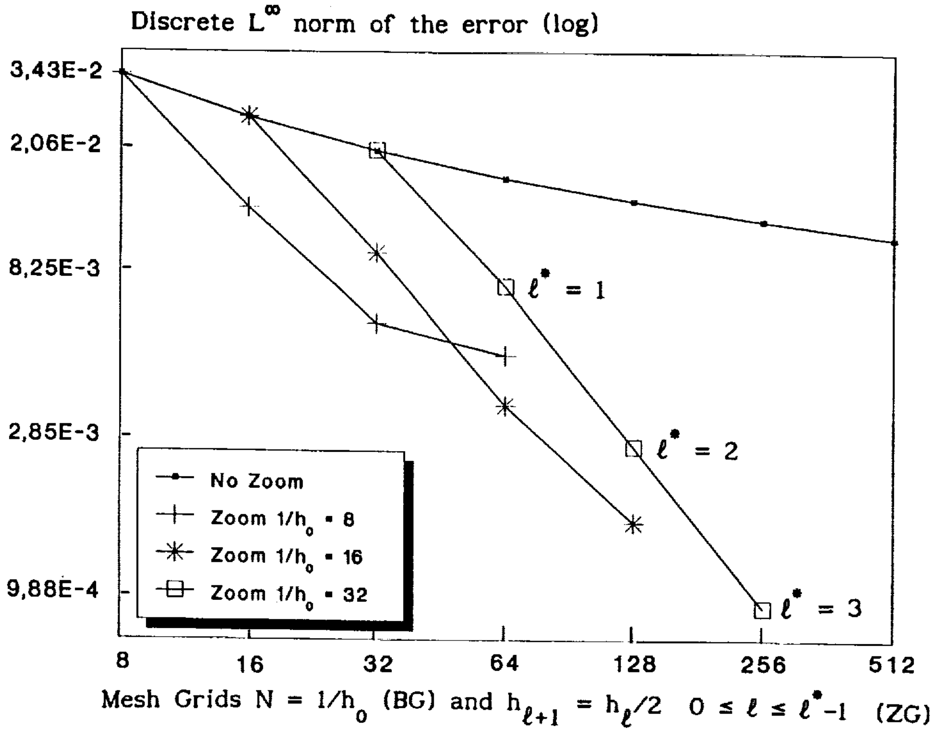


Fig. 2b. Problem ($\mathcal{P}2$) - Multilevel Zoom with FIC (ω = 0.35)
$\gamma(0) = 10 - x_1 = 0$ and $x_2 = 0.5$

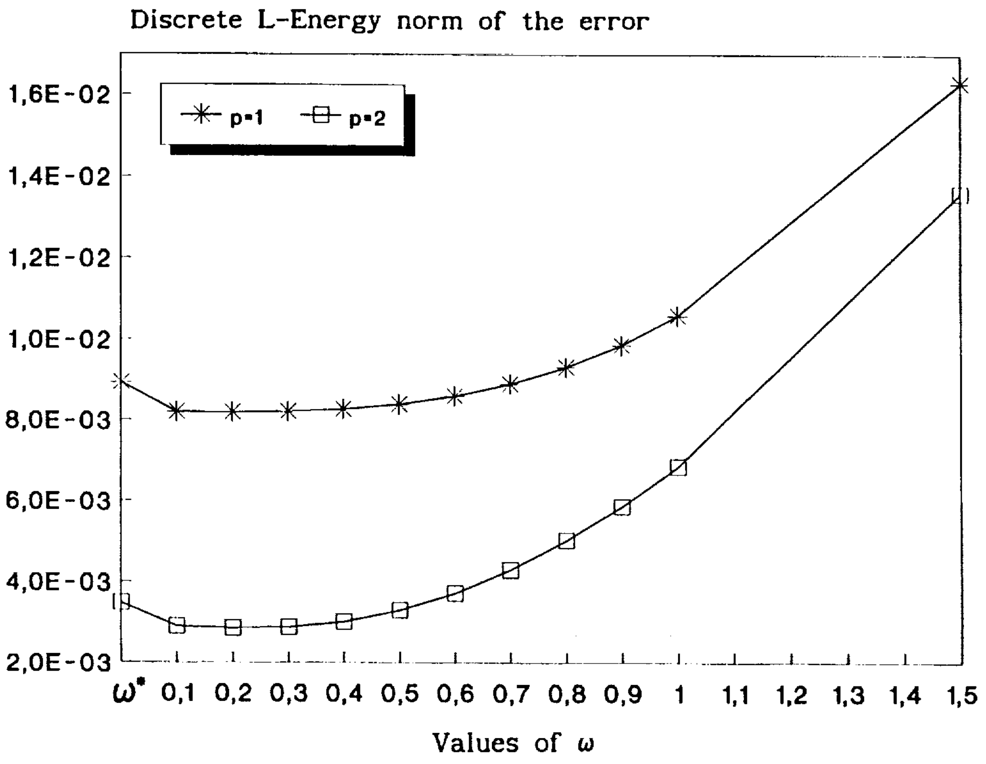


Fig. 3. Problem ($\mathcal{P}2$) - Two Grid Zoom FIC - Variations with ω
$h_0 = 1/16$, $h_1 = h_0/2^p$, $\gamma(0) = 2$, $x_1 = 0$ and $x_2 = 0.5$

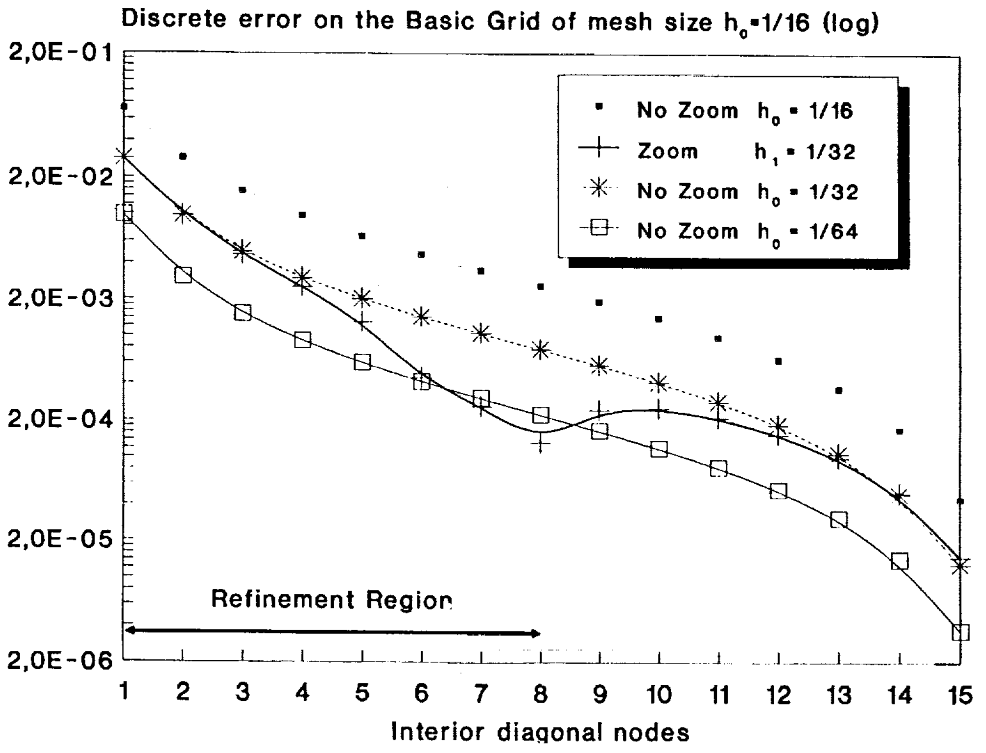


Fig. 4. Problem ($\mathcal{P}2$) - Two Grid Zoom with FIC (ω = 0.31)
$\gamma(0) = 2$, $x_1 = 0$ and $x_2 = 0.5$

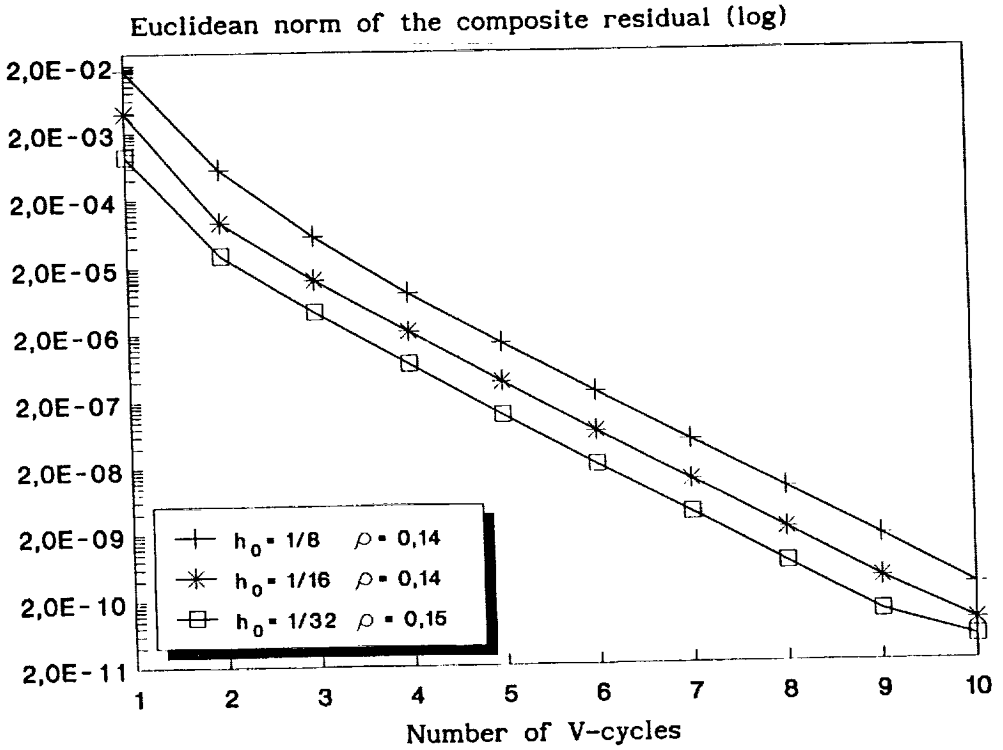


Fig. 5a. Problem ($\mathcal{P}2$) - Two Grid Zoom FAC - Convergence rate ρ
$h_1 = h_0/2$, $\gamma(0) = 10$, $x_1 = 0$ and $x_2 = 0.5$

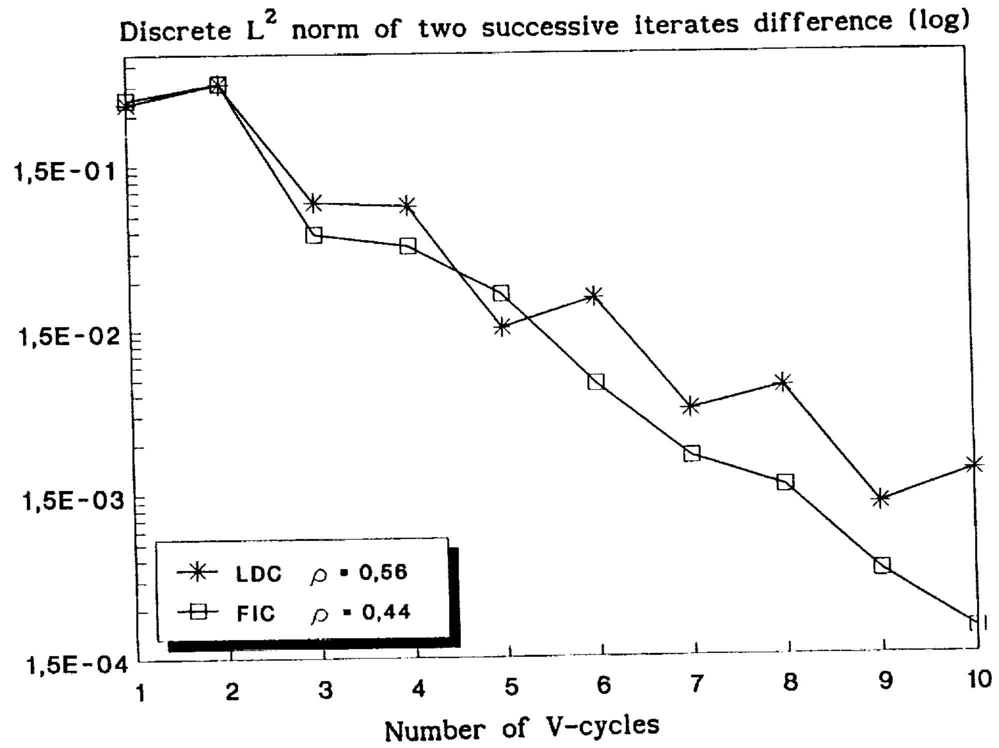


Fig. 5b. Problem ($\mathcal{P}2$) - Two Grid Zoom with LDC and FIC ($\omega = 0.2$) - Convergence rate ρ
$h_0 = 1/16$, $h_1 = h_0/2$, $\gamma(0) = 10$, $x_1 = 0$ and $x_2 = 0.5$

S20-34
197152
N94-29693

Multi-Grid Domain Decomposition Approach for Solution of Navier-Stokes Equations in Primitive Variable Form

Hwar-Ching Ku *

Johns Hopkins University Applied Physics Laboratory

Johns Hopkins Road, Laurel, MD 20723

Bala Ramaswamy

Department of Mechanical Engineering & Material Science

Rice University

Houston, Texas 77251

Summary The new multi-grid (or adaptive) pseudospectral element method has been carried out for the solution of incompressible flow in terms of primitive variable formulation. The desired features of the proposed method include (1) the ability to treat complex geometry; (2) the high resolution adapted in the interesting areas; (3) the minimal working space; and (4) effective under the multiple processors working environment.

The approach for flow problems, complex geometry or not, is to first divide the computational domain into a number of fine-grid and coarse-grid subdomains with the inter-overlapping area. Next, implement the Schwarz alternating procedure (SAP) to exchange the data among subdomains, where the coarse-grid correction is used to remove the high frequency error that occurs when the data interpolation from the fine-grid subdomain to the coarse-grid subdomain is conducted. The strategy behind the coarse-grid correction is to adopt the operator of the divergence of the velocity field, which intrinsically links the pressure equation, into this process. The solution of each subdomain can be efficiently solved by the direct (or iterative) eigenfunction expansion technique with the least storage requirement, i.e., $O(N^3)$ in 3-D and $O(N^2)$ in 2-D.

Numerical results of both driven cavity and jet flow will be presented in the paper to account for the versatility of the proposed method.

*Partially supported by the SPAWAR under the Contract Number N00039 - 91 - C - 0001

1 Introduction

Due to the advance of numerical techniques, numerous CFD algorithms have been developed to pursue the hard-to-approach flow problems. Nevertheless, numerical algorithms should have desired features of (1) the ability to deal with the variety of geometrical shapes; (2) arbitrary layout of dense grid points in the interesting areas; (3) the minimal working space; and (4) the low computational time to achieve such a goal. The development of a pseudospectral element method in these areas is our major concern.

One of the improvements in the area of feature (2) is the multi-grid technique, which has long been advocated by the finite-difference method [1, 2]. On the same computational domain, a sequence of uniform grids are employed to accelerate the convergence of iterative methods. The work rests on the "standard coarsening," i.e., doubling the mesh in each direction from one grid to the next coarsest grid and also smoothing the residual to the next coarse grid (restriction). Solve the problem on the coarse grid (low frequency domain) and the coarse-grid correction transfers back (prolongation) to the fine grid (high frequency domain) to gain rapid convergence. The technique developed so far, even with the inclusion of an adaptive scheme, is still limited to the simple complex geometry with uniform grids in the Cartesian coordinates, but is less for the non-uniform grids in the curvilinear coordinates.

The SAP iterative scheme has been successfully applied by the pseudospectral element method to those (simple complex) configurations where the overlapped grids are located at the same places [3, 4]. Here we refer to such cases as a single-grid SAP method because no error is involved during the data interpolation process. But under some circumstances, due to the complexity of the geometrical configuration such as possible layout of mixed types of grids (Cartesian, "O" or "C") or the necessity of applying adaptive fine grids for high resolution in one area and coarse grids for less resolution in others, the overlapped grids cannot be collapsed at the same position. Careful treatment on the overlapped grids by the SAP iterative scheme to eliminate the high frequency error due to the data interpolation will be the main objective in this paper. On the other hand, the question arises of how the continuity equation is satisfied in the overlapping area (including the interfaces between the fine-grid and coarse-grid subdomains) when solving the incompressible Navier-Stokes equations in primitive variable form. It reflects the fact that the boundary conditions for the pressure should link the incompressibility constraint in some respects. Extension of single-grid SAP to the multi-grid SAP domain decomposition method to overcome the above-mentioned difficulties will also be addressed.

The paper consists of five additional sections. Section 2 derives a primitive variable form of the Navier-Stokes equations. Section 3 discusses the multi-grid SAP domain decomposition method. Section 4 presents numerical results of proposed 2-D problems, and the final section provides the conclusions.

2 Primitive Variable Formulation

In tensor notation, the time-dependent Navier-Stokes equations in dimensionless form can be described as

$$\frac{\partial u_i}{\partial t} + u_j \frac{\partial u_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{\mathrm{Re}} \frac{\partial^2 u_i}{\partial x_j^2} \tag{1a}$$

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{1b}$$

Here u_i is the velocity component and Re is the Reynolds number.

The method applied to solve the Navier-Stokes equations is Chorin's [5] splitting technique. According to this scheme, the equations of motion read

$$\frac{\partial u_i}{\partial t} + \frac{\partial p}{\partial x_i} = F_i \tag{2}$$

where $F_i = -u_j \partial u_i/\partial x_j + 1/\mathrm{Re}\, \partial^2 u_i/\partial x_j^2$.

The first step is to split the velocity into a sum of predicted and corrected values. The predicted velocity is determined by time integration of the momentum equations without the pressure term

$$\bar{u}_i^{n+1} = u_i^n + \Delta t F_i^n \tag{3}$$

The second step is to develop the pressure and corrected velocity fields that satisfy the continuity equation by using the relationships

$$u_i^{n+1} = \bar{u}_i^{n+1} - \Delta t \frac{\partial p}{\partial x_i} \tag{4a}$$

$$\frac{\partial u_i^{n+1}}{\partial x_i} = 0 \tag{4b}$$

Here the superscript n denotes the n-th time step. Note that the size of a stable time-step Δt can be increased by using an adaptation of Runge-Kutta techniques [6] for the high Reynolds number and the Stokes solution for the low Reynolds number [3], respectively.

An equation for the pressure can be obtained by taking the divergence of Eq. (4a). In view of Eq. (4b), it governs

$$\frac{\partial^2 p}{\partial x_i^2} = \frac{1}{\Delta t}\frac{\partial \bar{u}_i}{\partial x_i} \tag{5}$$

Note that whenever solving Eq. (5) the identity of Eq. (4a) should be utilized to absorb the given boundary conditions of the velocity components [7].

If p satisfies Eq. (5), then $\mathbf{u}^{n+1}$ does indeed satisfy Eq. (4b). The solution of the pressure equation, Eq. (5), is the most computationally expensive step, while in Cartesian coordinates it can be directly solved numerically by the separation of variables [7]. Eq. (5) is of the general form,

$$Lp = S \tag{6}$$

for some linear operator L on some finite dimensional vector space. The properties of the operator L depend on the methods chosen to represent the fields and their derivatives.

Let the pressure p and source term S in Eq. (6) be expanded in a series of eigenfunctions such that

$$\mathbf{p} = \mathbf{EX}\,\hat{\mathbf{p}}\,\mathbf{EY}^T\,\mathbf{EZ}^T \tag{7a}$$

$$\mathbf{S} = \mathbf{EX}\,\hat{\mathbf{S}}\,\mathbf{EY}^T\,\mathbf{EZ}^T. \tag{7b}$$

then the solution of three-dimensional pressure Eq. (5) can be reduced to the simplest algebraic form

$$(\alpha_i + \beta_j + \gamma_k)\hat{p}_{i,j,k} = \hat{S}_{i,j,k} \tag{8}$$

where α_i, β_j and γ_k are the eigenvalues with respect to the discrete derivative matrices of the linear operator, L, and **EX**, **EY**, **EZ** are the corresponding eigenvectors associated with each eigenvalue. However, eigenvalues may not be real due to the complexity of an operator L. Without putting any restriction on eigenvalues, complex eigenvalues and their associated eigenvectors are permitted if the pressure gradient at the imaginary part vanishes. This is true because only the pressure gradient drives flow instead of the pressure itself. However, the effort of the matrix multiplication will be increased by a factor of four if all the calculations of Eqs. (7) are performed by the purely complex variables. Fortunately, not exceeding a factor of two will be reached if one takes advantage of (1) the purely real part of eigenfunctions for matrix multiplication; (2) the source term S being real; and (3) choosing only the real part of pressure as the pressure solution. The way for (1) includes reordering the eigenfunction into two parts: real versus complex, and similarly for the eigenvalues.

The iterative preconditioned method for the solution of pressure in the curvilinear coordinate system can be found in [8]. Note that if there are N degrees of freedom in each direction the overall memory required for finding the solution to the pressure equation in three dimensions is $O(N^3)$. This is the same type of maximal storage efficient scaling that we have for the velocity field.

Viewing the solution of the Navier-Stokes equations by the splitting method, two steps account for most of the run time, predicted velocity and the pressure solution. The bulk of these two steps can be concisely described in terms of dot products and matrix multiplication between subsets of array. Importantly, no data dependency occurs when running programs on parallel machines.

3 Domain Decomposition with Multi-Grid SAP

The solution of the Navier-Stokes equations via the domain decomposition approach consists of first dividing the computational domain into a number of subdomains with inter-overlapping areas, where the grids inside the overlapping area may or may not be located at the same places. Next implement the SAP for exchanging data among subdomains, i.e., solving the problem on each subdomain separately and then updating the velocity field on the overlapped interfaces. The advantages of this approach include (i) less memory access, local rather than global, and (ii) easy treatment of complex geometry.

In addition to the Lagrangian constraint between the pressure and velocity fields, the noncoincident overlapped grids in the inter-overlapping areas among subdomains even enhance the difficulty of applying the multi-grid technique. However, the idea of "coarse-grid correction" is still effective to reduce the high frequency error from the interpolated residual of the fine-grid subdomain. The strategy behind the coarse-grid correction process is to adopt the idea proposed by Thompson and Ferziger [9] and is modified as

$$\nabla_c \cdot \mathbf{u}_c - \nabla_c \cdot (I_c^f \mathbf{u}_f) = I_c^f (r_f - \nabla_f \cdot \mathbf{u}_f) \tag{9}$$

Here $\nabla_c \cdot$ represents the operator of divergence on the coarse-grid subdomain. I_c^f is an interpolation operator from the fine-grid subdomain f to coarse grid subdomain c, and $\mathbf{u}$ is the velocity component. r_f is simply the result of the divergence of the velocity field which should be set to zero. The left hand side of Eq. (9) is the difference between the coarse-grid operator acting on the coarse-grid subdomain and the coarse-grid operator acting on the interpolated fine-grid subdomain (which is held fixed). The right hand side of Eq. (9) is the interpolated residual of the fine-grid subdomain. It is obvious that once the solution of the fine-grid subdomain has been found the residual will be zero (exactly satisfying the pressure equation), and it also implies

$$\mathbf{u}_c = I_c^f \mathbf{u}_f \tag{10}$$

When the residual is non-zero, Eq. (9) acts as a forcing term for the coarse-grid correction to transfer the correction of the velocity field back to the fine-grid subdomain, i.e.,

$$\mathbf{u}_f^{new} = \mathbf{u}_f^{old} + I_f^c(\mathbf{u}_c - I_c^f \mathbf{u}_f^{old}) \tag{11}$$

This is vital for the success of the scheme. Changes in the velocity field are transferred back to the fine-grid subdomain rather than the velocity field itself. Notice that when the overlapped grids in the overlapping areas are collapsed at the same places the interpolation operator I_c^f automatically becomes a unitary matrix.

The multi-grid SAP iterative solution of the incompressible Navier-Stokes equation in primitive variable form for a driven cavity flow sketched in Fig. 1 is summarized by the following algorithm:

1. First assume $\mathbf{u}^{n+1}$ on $\overline{AB}$. Usually $\mathbf{u}^n$ will be a good initial guess.
2. Solve fine-grid domain II employing the boundary conditions derived from the divergence of the velocity field, including on $\overline{AB}$, where the pressure solution is directly obtained by the eigenfunction expansion method.
3. With the interpolated solution of $\mathbf{u}^{n+1}$ from step (2) on domain III$\subset$ I, solve coarse-grid domain I employing the same type boundary conditions including on $\overline{CD}$ to update $\mathbf{u}^{n+1}$ on domain III $\subset$ II by the coarse-grid correction process.
4. Repeat steps (2) & (3) until the velocity $\mathbf{u}^{n+1}$ on $\overline{AB}$, $\overline{CD}$ does not change.

In order to guarantee that consistent values of velocity (or pressure gradient) be generated in the overlapping domains III, satisfying Eq. (10), the divergence of the velocity field $\nabla \cdot \mathbf{u}$ needs to be actually computed in whichever domain I or II is counted [4]. Since $\mathbf{u}$ on domains III is not known a priori, the divergence of the velocity field is only set to zero at the first SAP iteration for step (2). According to this approach, the continuity equation is satisfied on domains II (including III $\subset$ II) and I (excluding III $\subset$ I), which is revealed from Eq. (9) that the continuity equation is only satisfied on the fine-grid domain II. More specifically, the issue of how to satisfy the continuity equation along the interfaces of fine-coarse grid domain can be easily resolved by the proposed approach, namely, $\nabla \cdot \mathbf{u}$ on $\overline{AB}$ satisfied on the coarse-grid domain I, and $\nabla \cdot \mathbf{u}$ on $\overline{CD}$ satisfied on the fine-grid domain II. However, the error index of the continuity equation on domain III $\subset$ I will indicate how good the interpolation is (affected by the layout of overlapped grids) and whether any steep change of flow field exists.

Three main issues occurring in the overlapping area between the fine-grid and coarse-grid subdomains one might often encounter are how to (1) efficiently implement the interpolation; (2) adequately represent the predicted velocity; and (3) explicitly impose the global mass conservation. Each will be addressed separately.

3.1 Data interpolation

Finding the image (ξ,η), $-1 \leq \xi \leq 1, -1 \leq \eta \leq 1$, of a collocation point (x,y) from the fine-grid subdomain II mapped onto the coarse-grid subdomain I (or vice versa) is first determined by using the two-dimensional Lagrange interpolation to seek its corresponding position falling into an element on the coarse-grid subdomain that contains $(M+1)$ x $(N+1)$ collocation points, $\xi_i = cos\pi i/M(i = 0,...,M), \eta_j = cos\pi j/N(j = 0,...,N)$, such that

$$x = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{mn} T_m(\xi) T_n(\eta) \tag{12a}$$

$$y = \sum_{m=0}^{M} \sum_{n=0}^{N} b_{mn} T_m(\xi) T_n(\eta) \tag{12b}$$

where T_m denotes the *mth* order Chebyshev polynomials. Unknown expansion coefficients, a_{mn}, b_{mn}, can be easily obtained by the prescribed points (x,y) on the coarse-grid subdomain I through

$$x(\xi_i,\eta_j) = \sum_{m=0}^{M} \sum_{n=0}^{N} a_{mn} T_m(\xi_i) T_n(\eta_j) \tag{13a}$$

$$y(\xi_i,\eta_j) = \sum_{m=0}^{M} \sum_{n=0}^{N} b_{mn} T_m(\xi_i) T_n(\eta_j) \tag{13b}$$

With a given point (x,y) in the physical space of fine-grid subdomain II, its image (ξ,η) on the coarse-grid subdomain I can be iteratively solved by the Newton-Raphson method. Once the one-to-one correspondence between the fine-grid and coarse-grid subdomains has been established, the equation required to generate a function $\phi^f(x,y)$ on the fine-grid subdomain interpolated from the coarse-grid subdomain, now becomes

$$\phi^f(x,y) = \sum_{i=0}^{M} \sum_{j=0}^{N} N_i(\xi) N_j(\eta) \phi^c(\xi_i,\eta_j) \tag{14}$$

where $\phi^c(\xi_i,\eta_j)$ denotes the function value at the collocation point (ξ_i,η_j) on the coarse-grid subdomain, and $N_i(\xi), N_j(\eta)$ are the shape functions defining the geometry of the element on the coarse-grid subdomain, whose expressions are

$$N_i(\xi) = \sum_{m=0}^{M} T_m(\xi) \hat{T}_m(\xi_i) \tag{15a}$$

$$N_j(\eta) = \sum_{n=0}^{N} T_n(\eta) \hat{T}_n(\eta_j) \tag{15b}$$

where the matrices $T_m(\xi)$ and $\hat{T}_m(\xi)$ are the Fourier cosine series and their inverse [7]. Note that the shape functions $N_i(\xi), N_j(\eta)$ satisfy the Kronecker-delta property, i.e., $N_i(\xi_m) = \delta_{im}, N_j(\eta_n) = \delta_{jn}$.

Be aware that it requires much less effort to perform the data interpolation if the one-to-one correspondence for the shape functions between subdomains can be stored (once and for all). Also the cost for such additional memory is negligible compared to that declared by a single variable.

3.2 Predicted velocity

Since the predicted velocity in the overlapping area generated from Eq. (3) by the fine-grid subdomain is slightly different from that obtained by the coarse-grid subdomain, how to control the predicted velocity in order to keep the error index, ℓ_2 norm of $\omega = \|\mathbf{u}_c - I_c^f \mathbf{u}_f\|$ minimal, is of great importance. Numerical experiments suggest that the following dynamic relationship

$$\bar{\mathbf{u}}_f = (1 - \omega^\alpha)\,\bar{\mathbf{u}}_f + \omega^\alpha\, I_f^c \bar{\mathbf{u}}_c \qquad (16)$$

gives the best fit. Here the exponent α is chosen as 0.4 for various tested problems.

3.3 Global flow rate

For the inflow-outflow problems the coarse-grid velocity field interpolated from the fine-grid subdomain may not exactly satisfy the global mass conservation, and a slight adjustment to the velocity field is imperative. A common-used formula will meet such a requirement, i.e.,

$$u_i^{new} = u_i^{old} \frac{\int \mathbf{u}^{exact} \cdot dA}{\int \mathbf{u}^{old} \cdot dA} \qquad (17)$$

4 Results and Discussion

For the numerical test of the driven cavity flow problem, layout of elements (6 points per element) in the fine-grid and coarse-grid subdomains at the Reynolds number of 400 and 100 are displayed in Figs. 2a and 2b, respectively. The overlapping area is not explicitly shown in the figures, but just imagine the extension of one more element from the coarse-grid subdomain into the fine-grid subdomain. The layout of elements is in accordance with the requirement to resolve the steep changes inside the boundary layers. When exchanging the data through the interpolation in the inter-overlapping area, the high frequency error introduced by the fine-grid subdomain will pollute the results throughout the whole computational domain. It can be simply proved by checking the error index, ℓ_2 norm of $\omega = \|\mathbf{u}_c - I_c^f \mathbf{u}_f\|$, in the overlapping area. ω will increase with marching in time domain, and eventually become an unreasonably big number under which the solution does not exist. With the multi-grid SAP approach, both results produce $O(10^{-4})$ for ω, instead. When comparing the streamline plots, ψ_{min} = -0.1055 for Re = 100 and ψ_{min} = -0.1163 for Re = 400, with the most accurate results of Ghia [10], good agreements can be observed in Fig. 3.

For the inflow-outflow jet problem, a nozzle is designed to gain a high speed fluid with a smooth change of the convergent channel. The configuration of jet flow is plotted in Fig. 4. A jet emanating from the nozzle with an aspect ratio $H/D = 144$ (the width of tank to nozzle) is used to understand the turbulent characters through the direct numerical simulation. With a strong stratification imposed in the vertical direction the two-dimensional turbulent flow calculation will be a good approximation to the three-dimensional case. The calculation is carried out up to the Kolmogoroff length scale where the energy transferred from the large scales is in equilibrium with the energy dissipated in the smallest scale by the molecular viscosity. Certainly, for the purpose of direct numerical simulation the Reynolds number should not be large so that the machine can still handle the huge number of points required for the resolution of different length scales. The computational domain is decomposed into three subdomains: the upstream nozzle where the inflow is developed to gain a high speed, the immediate downstream from the exit of the nozzle where the high speed jet is discharged into the tank, and the far downstream (fine grids) where a well-developed turbulent flow can be traced.

Let us first check the error index ω for the inflow-outflow jet problem without using the multi-grid SAP technique. The ω around $O(10^{-3})$ seems all-right at Re = 100 initially, but the onset of noise starts to destabilize the downstream flow field at the Reynolds number of 250 and ω increases up to $O(10^{-2})$. That clearly demonstrates the high frequency polluting that results on the fine-grid subdomain, but the noise can be totally removed by using the multi-grid SAP technique. Fig. 5 depicts the streamline plot of jet flow at Re = 250. During the time evolution of the jet flow, the symmetry of the jet front will not be distorted at the early stage (Fig. 5a) until the phase speed of vortex shedding (due to flow instability) travels faster than that of the jet front. As seen in Fig. (5b), a pair of vortices adjacent to the jet front persisting throughout the course represent the extrusion of the jet into the ambient fluid. Once the jet front is caught up by the incoming travelling waves, the energy transferred by the vortex shedding, in terms of the cascade process from the highest at the nozzle exit (high shedding frequency) to the lowest at the jet front (low shedding frequency), splits into two parts, one for the jet front to push against the ambient viscous resistance, another for the vertical motion. The intensity of vertical motion behind the jet front is gradually enhanced as visualized by the splitting streamlines, and their patterns move backward toward the exit of the nozzle where a distinct pair of vortices exist. The appearance of similar pairs of vortices can also be confirmed by the experiment at the high Reynolds number [11].

5 Conclusions

The solution of the Navier-Stokes equations in a primitive variable form has been solved by the pseudospectral element method via the multi-grid domain decomposition technique. The computational domain is divided into a number of simple subdomains with the inter-overlapping zone, of which the fine grids (or fine-grid subdomain) are used in the areas with the steep change of flow field while the coarse grids (or coarse-grid subdomain) are used in the others. During the data exchange among subdomains, the coarse-grid correction technique is used to eliminate the high frequency error caused by the data interpolation from the fine-grid subdomain to the coarse-grid subdomain.

Both driven cavity and jet flow demonstrate the versatility of the proposed multi-grid method.

References

[1] W. Hackbusch, *Multi-Grid Methods and Applications*, Springer-Verlag (Berlin), 1985.

[2] A. Brandt, *Math. Comput.* **31**, 333 (1977).

[3] H. C. Ku, R. S. Hirsh, T. D. Taylor and A. P. Rosenberg, *J. Comput. Phys.* **83**, 260 (1989).

[4] H. C. Ku, T. D. Taylor and R. S. Hirsh, *Comput. Methods in Applied Mech. and Eng.*, **75**, 141 (1989).

[5] A. J. Chorin, *Math. Comp.* **22**, 745 (1968).

[6] H. C. Ku, A. P. Rosenberg and T. D. Taylor, in *Proceeding of the Twelth Conference on Numerical Methods in Fluid Dynamics*, K. W. Morton ed., Oxford, England (1990).

[7] H. C. Ku, R. S. Hirsh and T. D. Taylor, *J. Comput. Phys.*, **70**, 439 (1987).

[8] H. C. Ku, *Solution of Flow in Complex Geometries by the Pseudospectral Element Method*, submitted for publication, (1993).

[9] M. C. Thompson and J. H. Ferziger, *J. Comput. Phys.* **82**, 94 (1989).

[10] U. Ghia, K. Ghia and C. A. Shin, *J. Comput. Phys* **48**, 387 (1982).

[11] *Visualized Flow*, Japan Society of Mechanical Engineers, Pergamon Press (1988).

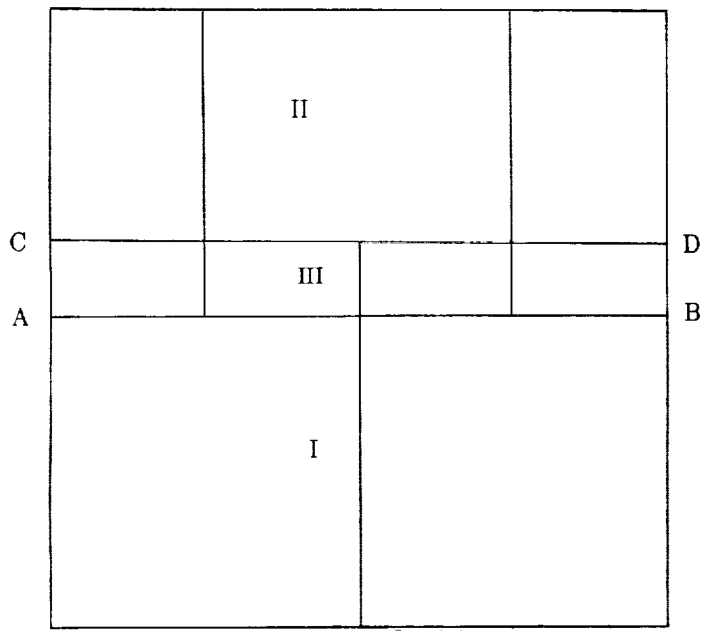


Fig. 1 Configuration of domain decomposition

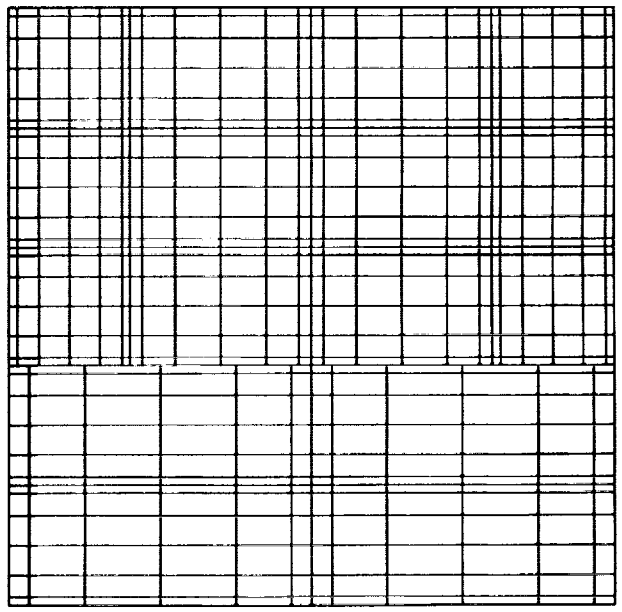


Fig. 2a Element layout of driven cavity flow for Re = 100

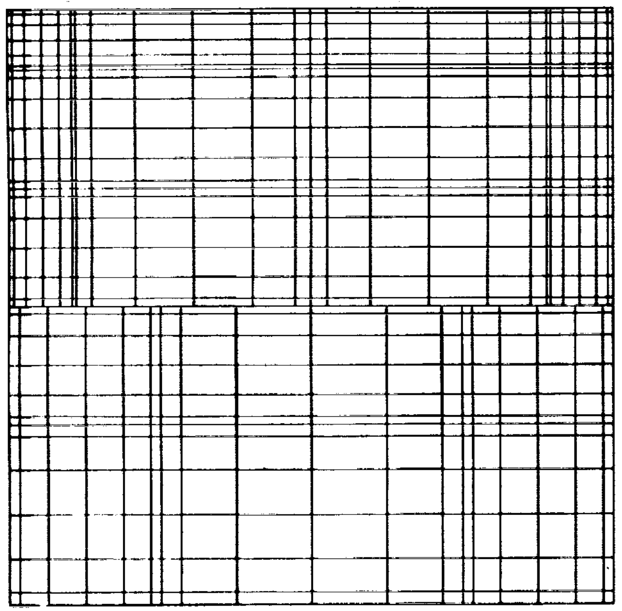


Fig. 2b Element layout of driven cavity flow for Re = 400

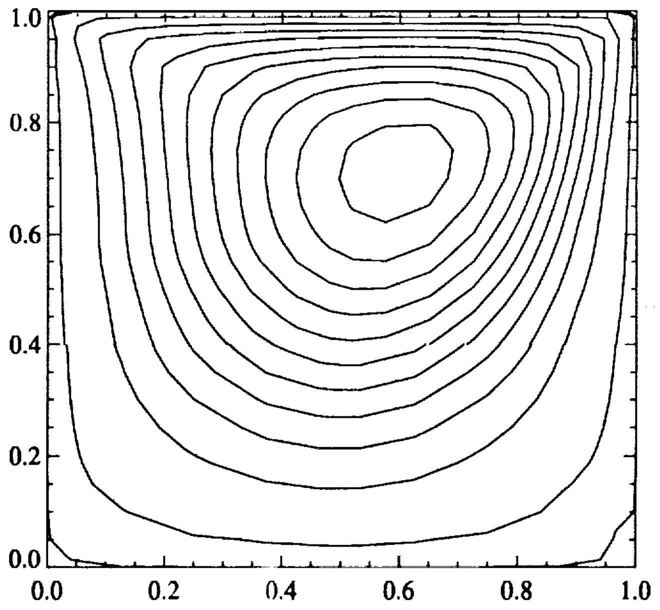


Fig. 3a Streamline plot for Re = 100

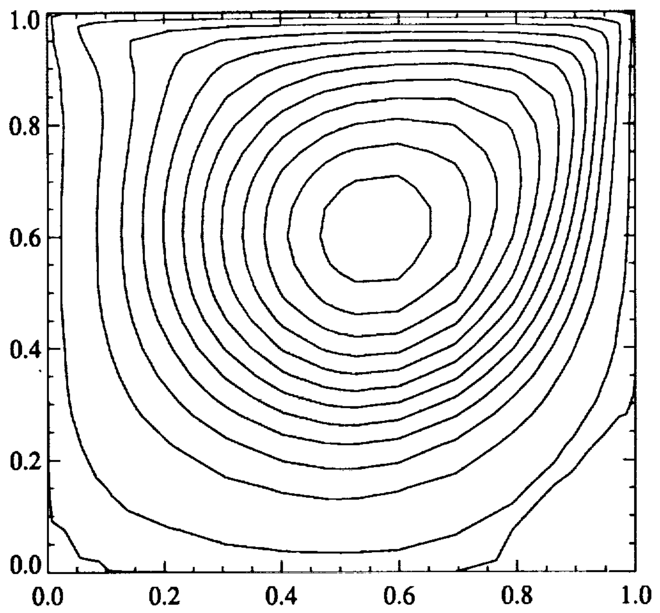


Fig. 3b Streamline plot for Re = 400

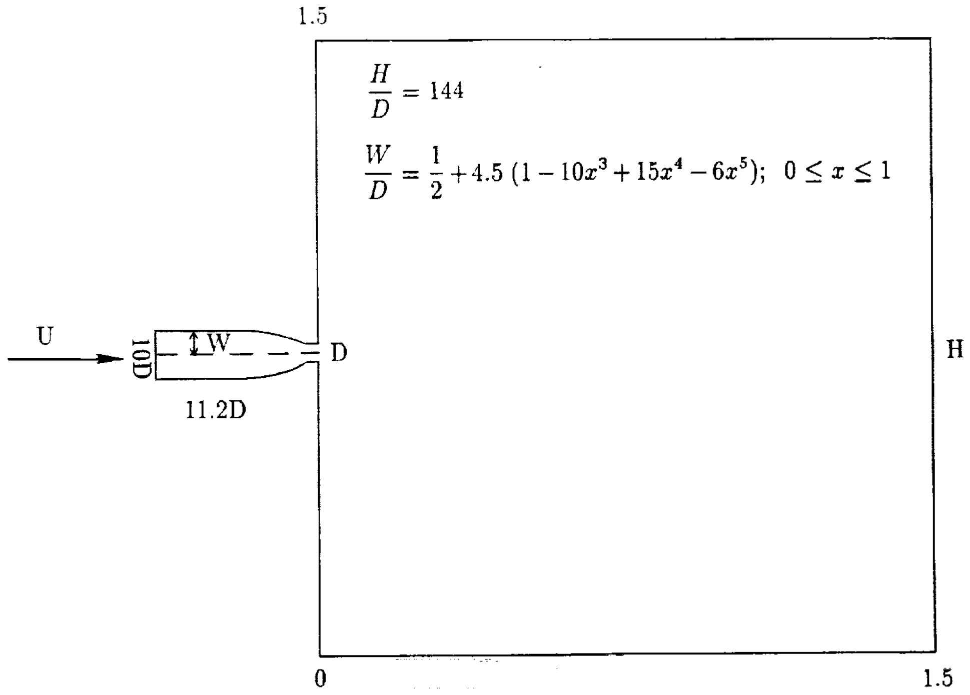
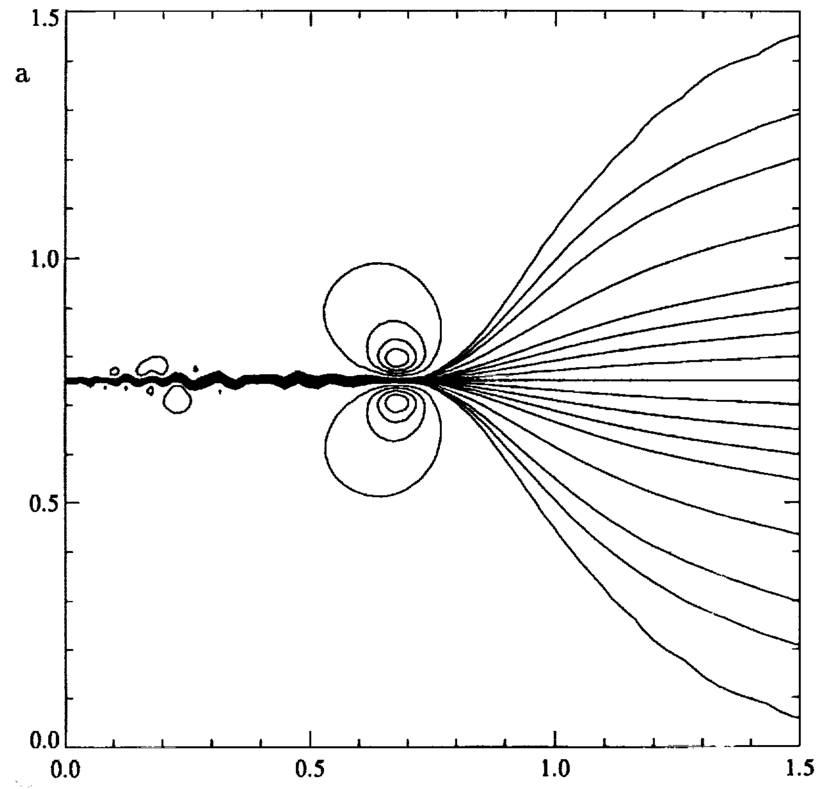

1.5
$\frac{H}{D} = 144$
$\frac{W}{D} = \frac{1}{2} + 4.5\,(1 - 10x^3 + 15x^4 - 6x^5);\quad 0 \le x \le 1$
U
10D
W
D
H
11.2D
0
1.5

Fig. 4 Configuration of jet flow



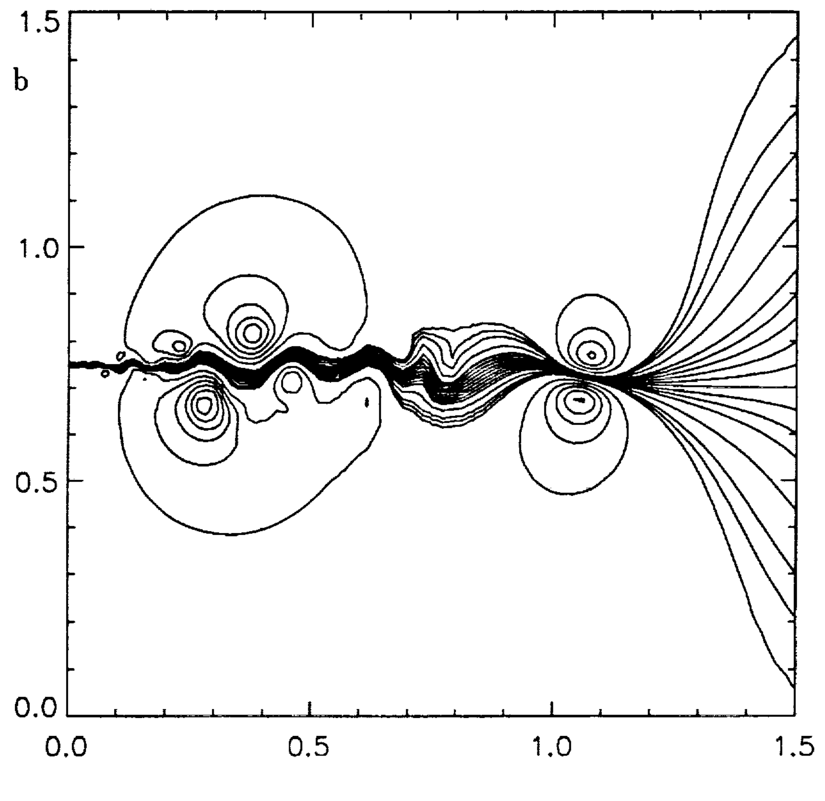


Fig. 5 Streamline plots of jet flow for Re = 250 at time (a) t = 180, (b) t = 315

N94-29694

COMPRESSIBLE TURBULENT FLOW SIMULATION WITH A MULTIGRID MULTIBLOCK METHOD

S21-34
197153
P, 11

Hans Kuerten and Bernard Geurts
Department of Applied Mathematics
University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands

SUMMARY

We describe a multigrid multiblock method for compressible turbulent flow simulations and present results obtained from calculations on a two-element airfoil. A vertex-based spatial discretization method and explicit multistage Runge-Kutta time-stepping are used. The slow convergence of a single grid method makes the multigrid method, which yields a speed up with a factor of about 20, indispensable. The numerical predictions are in good agreement with experimental results. It is shown that the convergence of the multigrid process depends considerably on the ordering of the various loops. If the block loop is put inside the stage loop the process converges more rapidly than if the block loop is situated outside the stage loop in case a three-stage Runge-Kutta method is used. If a five-stage scheme is used the process does not converge in the latter block ordering. Finally, the process based on the five-stage method is about 60% more efficient than with the three-stage method, if the block loop is inside the stage loop.

INTRODUCTION

Numerical simulations of turbulent flow in aerodynamic applications are frequently based on the Reynolds-averaged Navier-Stokes equations. One of the relevant problems in aeronautics is the prediction of flow quantities in complicated geometries, such as the multi-element airfoil (see figure 1). The simulation of turbulent flow around such a multi-element airfoil configuration was one of the

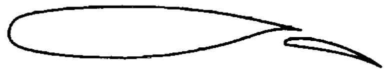


Figure 1: Geometry of a two-element airfoil.

applications selected for the compressible flow solver which was developed by our group and NLR as a part of the Dutch ISNaS project [1]. For this application the use of a single-block, boundary-conforming, structured grid is impossible and one may select either an unstructured grid approach or a block-structured grid approach. Although the former technique has been successfully applied by others [2], we selected the block-structured approach in view of the transparent data structure in the coding, ease of implementation of the turbulence model and a high flexibility with respect to the use of different physical models in different parts of the computational domain.

In a previous paper [3] it has been shown that for laminar and turbulent flow around a single airfoil the introduction of the multiblock structure has no influence on the results, with respect to both the steady-state solution and the convergence rate. Furthermore, invoking the Euler equations instead of the Navier-Stokes equations in blocks outside the boundary layer appeared to have no significant influence on the results. In this paper we describe the application of the multiblock concept to the multi-element airfoil. If the Euler equations are used throughout the computational domain, a converged steady-state solution is obtained within a reasonable calculation time. However, if the Reynolds-averaged Navier-Stokes equations are solved in the boundary layers, the rate of convergence is unacceptably low. Therefore, a multigrid technique was implemented in order to accelerate the convergence. The resulting gain in calculation time is close to a factor of 20, and the converged solution is in good agreement with wind-tunnel measurements.

In section 2 the numerical technique, which is based on a combination of a finite volume method with central spatial differencing and a Runge-Kutta explicit time-stepping method, is described. The results, both for inviscid and for viscous simulations, are presented in section 3. Finally, in section 4 some conclusions are summarized.

NUMERICAL METHOD

In this section we describe the numerical method used in the flow solver. The two-dimensional, compressible Navier-Stokes equations can be written in integral form as

$$\frac{\partial}{\partial t}\left[\int\!\!\int_{\Omega} U dx dy\right] + \int_{\partial\Omega}(F dy - G dx) = 0, \tag{1}$$

where U represents the vector of dependent variables,

$$U = [\rho, \rho u, \rho v, E]^T, \tag{2}$$

with ρ the density, u and v the Cartesian velocity components, and E the total energy density. Further, Ω is an arbitrary part of the two-dimensional space with boundary $\partial\Omega$ and F and G are the Cartesian components of the total flux vector. This flux vector consists of two parts: the non-dissipative or 'convective' part and the dissipative or 'viscous' part, which describes the effects of viscosity and heat conduction, and involves first order spatial derivatives. The Navier-Stokes equations (1) are averaged over a sufficiently large time interval. Due to the nonlinear terms in the convective fluxes, the resulting 'Reynolds-averaged Navier-Stokes' equations involve averages of products of two velocity components. These terms are modeled by a suitable turbulence model. In the present paper the algebraic Baldwin-Lomax turbulence model, in which the unknown terms are modeled by eddy viscosity terms, is adopted [4].

The discretization of the Navier-Stokes equations follows the method of lines, i.e. the spatial discretization is performed first, and subsequently the resulting set of ordinary differential equations is integrated in time, until the steady state solution is approximated. First the computational domain is divided into blocks and each block is partitioned in quadrilateral cells with the help of a structured, boundary-conforming grid. The variables are stored in the grid points. A finite volume method is used in which the integral form of the Navier-Stokes equations is applied to a control volume Ω, bounded by the dashed lines in figure 2. The convective flux through a boundary of this control volume is

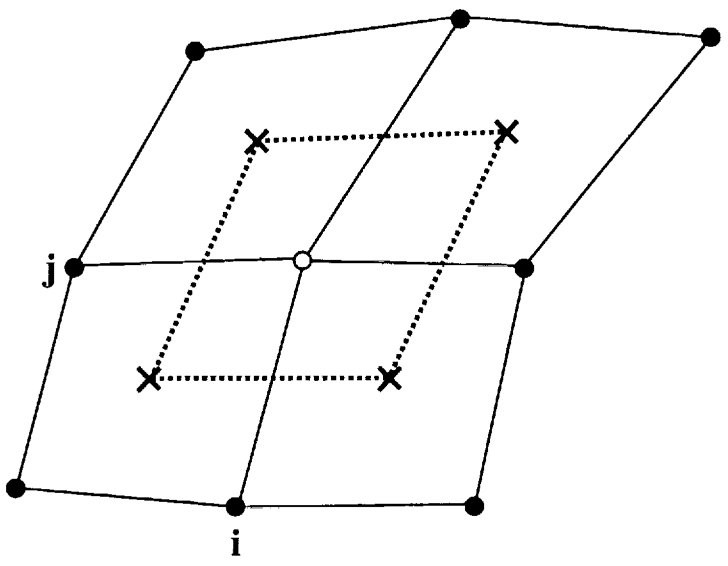


Figure 2: Control volume in the vertex-based method.

approximated using the value of the convective flux vector in the midpoint of the boundary. The latter is calculated by averaging over the two neighboring grid points. The viscous flux vector involves spatial derivatives of the state vector U and is approximated in the corner points of the control volume with the use of Gauss' theorem on a grid cell. The viscous flux is subsequently calculated using the trapezoidal rule. This method is called the vertex-based method.

The method of central differencing leads to a decoupling of odd and even grid points and to oscillations near shock waves. Even in viscous flow calculations the presence of the viscous dissipation is insufficient to damp these instabilities outside shear layers. Therefore, nonlinear artificial dissipation is added to the basic numerical scheme. This artificial dissipation consists of two contributions: fourth order difference terms which prevent odd-even decoupling, and second order difference terms to resolve shock waves. The second order terms are controlled by a shock sensor, which detects discontinuities in the pressure. In the present flow solver the artificial dissipation in the boundary layers, where the viscous dissipation should be dominant, may be reduced by multiplication with the ratio of the local and free-stream Mach number. The role of the artificial dissipation in relation to the viscous dissipation is discussed in more detail in reference [5].

At the solid wall boundaries the no-slip condition is used. The density and energy density in the grid points on a solid wall are calculated by solving the corresponding discrete conservation laws, using the two adjacent cells within the computational domain and their mirror images inside the wall as the control volume. The values of the density and energy density in the grid points inside the walls are adjusted such that the adiabatic wall condition is approximated. The boundary conditions at a (subsonic) far-field boundary are based on characteristic theory. The extent of the computational domain can be reduced without affecting the accuracy if a vortex is superimposed on the incoming free stream outside the computational domain [6].

Due to the topology of the two-element airfoil geometry, special points in the computational grid are unavoidable. The computational grids used contain two special points at block boundaries, where five cells meet (see figure 4). These points can be treated in an elegant way within the same numerical scheme, if the dummy vertices outside the 'current' block are defined appropriately. The multi-valuedness of the variables at the special point, caused by this asymmetric treatment, is eliminated by taking the average of the five different values after all blocks have been treated. This is sketched in figure 3.

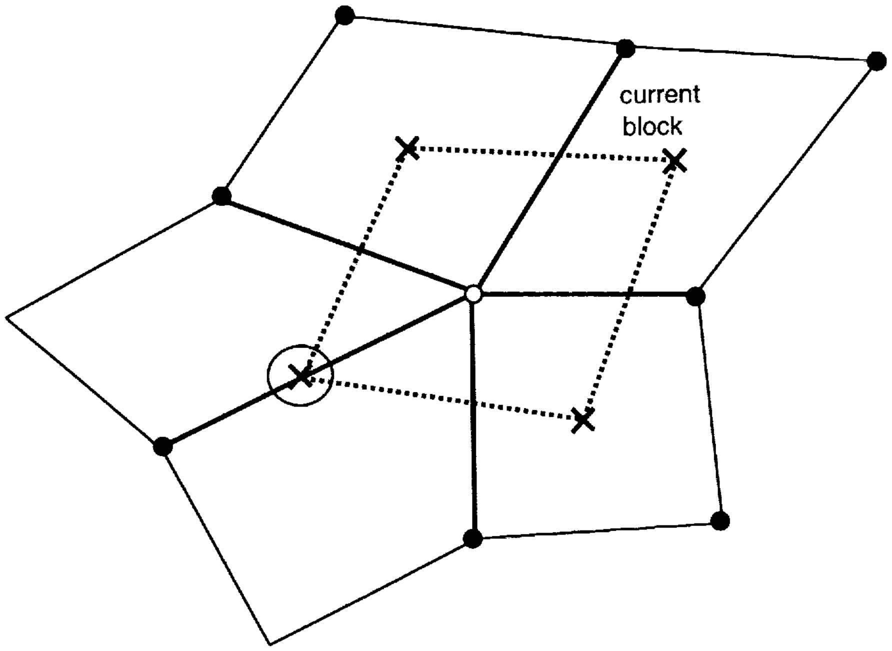


Figure 3: Control volume for a special point.

The system of ordinary differential equations, which results after spatial discretization, is integrated in time using a time-explicit multistage Runge-Kutta method. In the present flow solver a three-stage scheme in which the dissipative fluxes (both viscous and artificial) are calculated once per time-step, and a five-stage scheme in which the dissipative terms are calculated only at the odd stages, are implemented. With this treatment both calculation time is saved and the stability region of the method is increased. Extra calculation time is saved by advancing each grid point at the maximum local time-step according to its own stability limit. In this way the evolution from the initial solution to the steady state is no longer time accurate, but the steady state solution obtained is unaffected.

The above time-stepping method acts as the relaxation method and coarse grid operator in the multigrid solver (see reference [6]). In this solver an initial solution on the finest grid is obtained with a full multigrid method. This initial solution is corrected in the FAS-stage, where either V- or W-cycles can be chosen. A fixed number of pre- and post-relaxations is performed before turning to the next coarser or finer grid. The solution is transferred to a coarser grid by injection, the residuals by full weighting and the corrections to the solution are prolonged by bilinear interpolation. In order to increase the smoothing properties of the Runge-Kutta time-stepping technique an implicit averaging of the residuals is applied with frozen residuals at the block boundaries. For mono-block applications this method has given satisfactory results for both two-dimensional and three-dimensional flows [5].

In the multi-element airfoil application care has to be taken in the definition of the residual-vector in the special points. The proposed treatment of a special point implies that the control volume is different in each of the five blocks where such a point is found. In the required averaging the five residual-vectors in a special point are weighed with their corresponding time-steps. Without this weighing the multigrid process cannot converge to the single grid stationary state solution.

In this multigrid, multiblock solver with a multistage time-stepping method there are various possibilities for intertwining the different loops. In the present study the grid loop is chosen as the outer loop and the effect of interchanging the block and the stage loop will be studied. Several 'competing' requirements serve as possible guidance for selecting a specific ordering of these loops. On the one hand an anticipated parallel processing of the different blocks is more efficient, if the data transfer between the blocks is kept to a minimum, i.e. with the stage loop inside the block loop. On the other hand the good convergence of the multigrid mono-block solver may be reduced as the dummy variables near the block boundaries are kept frozen during more stages of the time-step. This would suggest to put the block loop inside the stage loop. In order to study this dilemma we implemented these two loop orders in a flexible way: a single parameter determines whether the block loop is situated inside or outside the stage loop.

RESULTS

Description of the test-case

We will present results for a two-component airfoil geometry consisting of the NLR7301 wing section, from which a flap has been cut out at a deflection angle of 20° and with a gap width of 2.6% chord length [7] (see figure 1). The combination of a Mach number of 0.185 and an angle of incidence of 6° or 13.1°, of which the latter is close to maximum lift conditions, yields subsonic flow. The Reynolds number based on the chord length of the airfoil is 2.51×10^6. In the viscous calculations the locations of the transition from laminar to turbulent flow are prescribed.

The C-type computational grids (either for inviscid or viscous flow) were constructed by J.J. Benton from British Aerospace, and are subdivided in 37 blocks (see figure 4). The grid lines are continuous over block boundaries. Two grids are used: one 'Euler' grid (inviscid) consisting of 16448 cells, and a 'Navier-Stokes' grid (viscous), which is refined in the boundary layers and wakes and consists of 28288 cells.

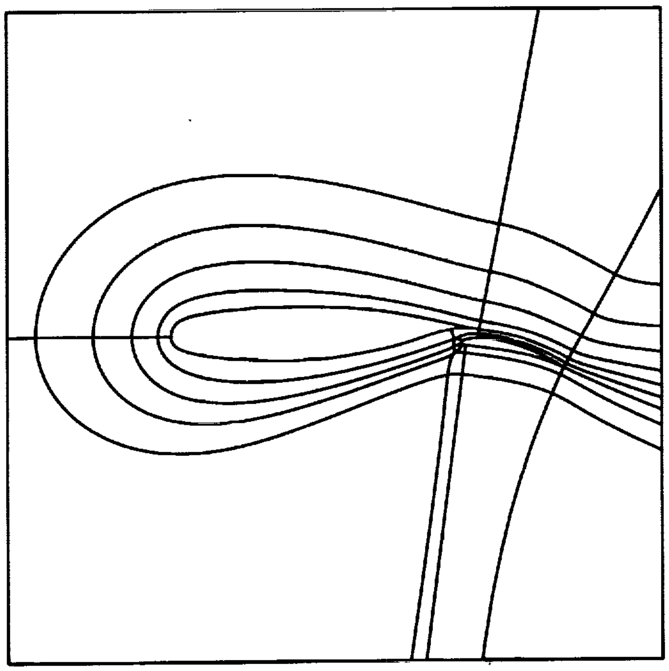


Figure 4: Block structure of the computational grid.

For both angles of incidence results from wind-tunnel measurement by Van den Berg [7] are available, including velocity profiles in the boundary layers and the pressure coefficient on the profile. Since the flow is attached apart from a small laminar separation bubble near the leading edge of the wing, the adopted turbulence model should be adequate and yield a useful comparison between experiment and calculation.

Inviscid Flow

In order to test the flow solver on the complicated block structure of the two-element airfoil geometry, we considered the relatively simple inviscid flow case, where in all blocks the Euler equations are solved. In this way problems related to the turbulence model are separated from possible algorithmic problems. The use of the Euler equations implies that the boundary conditions at the solid wall boundaries have to be changed. For inviscid flow there is only one physical boundary condition of zero mass flux through the wall. In the vertex based approach the density, the pressure and the tangential velocity at the wall are approximated by linear extrapolation.

In figure 5 the multigrid convergence behavior of the solver in the 13.1° case is shown. The discrete L_2-norm of the residual of the density is plotted as a function of the number of W-cycles. A converged solution is obtained within a much smaller calculation time when compared to the single grid approach even though only three different grid levels are available. Both for the single grid and the multigrid calculations machine accuracy was obtained. The specific block structure nor the treatment of the special points leads to any specific difficulties. For this inviscid test a comparison with experimental results is not meaningful and will not be made.

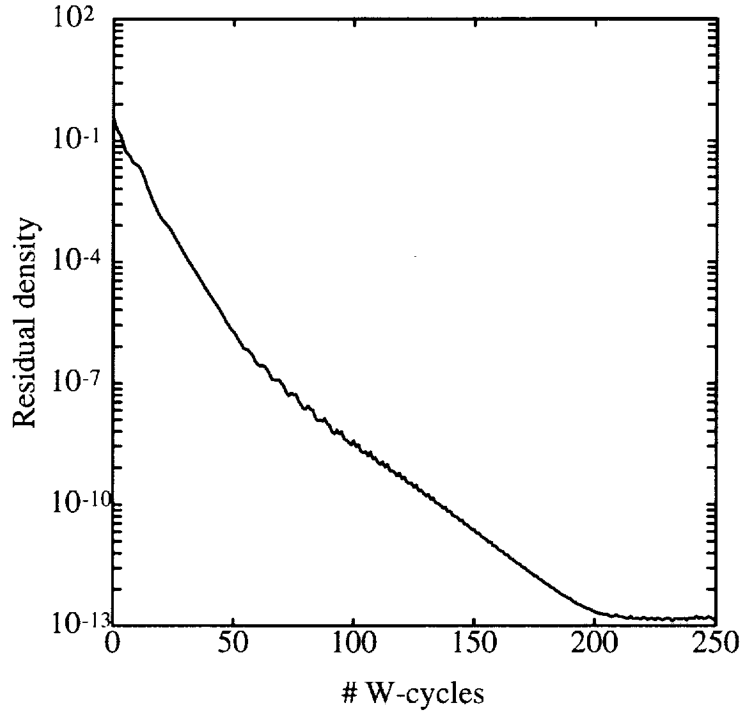


Figure 5: Convergence behavior for inviscid flow at an angle of incidence of 13.1°.

Viscous Flow

We consider the simulations of turbulent, viscous flow and present results for the 6° case only. Single-grid calculations in which only local time-stepping is applied as a convergence acceleration technique yield a steady-state solution which is in good agreement with the experimental results. However, in contrast with a fully inviscid simulation, the rate of convergence is very small, and renders this method unacceptable for practical applications. Therefore, as a method to increase the convergence rate further, the multigrid technique and implicit residual averaging as described in section 2 are indispensable.

In a simulation of turbulent flow at high Reynolds number it is important that the effects related to the physical dissipation are not outweighed by those of the numerical or artificial dissipation. This requirement could give rise to difficulties in the present multigrid method, since the time-stepping method used requires a certain minimum amount of dissipation for sufficient smoothing of the large wave-number components of the error (see reference [5]). If the artificial dissipation in the boundary layer is reduced by scaling with the ratio of the local and free-stream Mach number, i.e. decreasing the smoothing properties of the time-stepping method, a converged solution (engineering accuracy) could be obtained by increasing the number of pre- and post-relaxations. The convergence behavior of this calculation during the FAS stage is shown in figure 6, where the discrete L_2-norm of the residual of the density is plotted as a function of the number of W-cycles. In the blocks outside the boundary layers and wakes the Euler equations are solved instead of the Navier-Stokes equations. The good

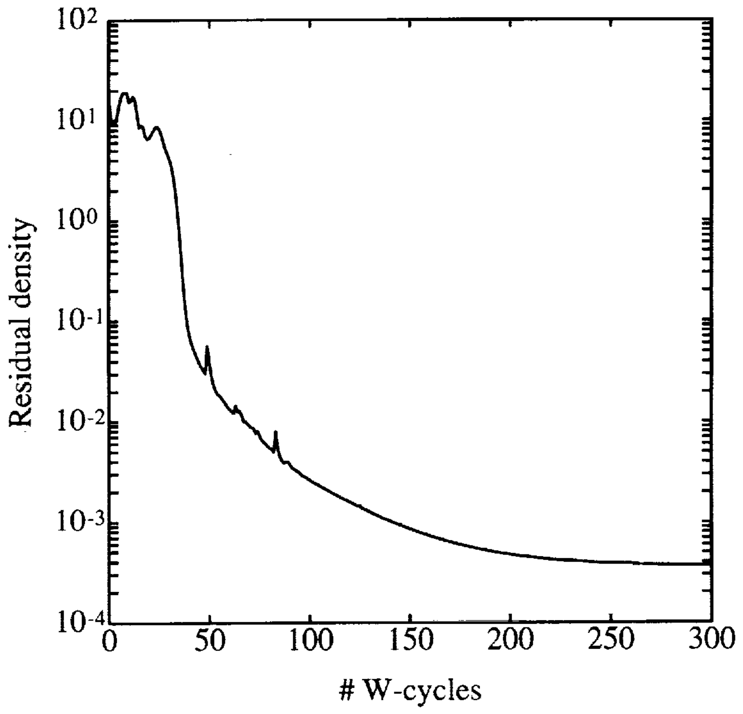


Figure 6: Viscous flow at an angle of incidence of 6.0°: convergence behavior

agreement with the wind-tunnel measurements can be inferred from figure 7, where the experimental and numerically predicted pressure coefficients on the airfoil and flap are shown.

This solution was obtained with the block loop inside the stage loop of the five-stage Runge-Kutta time-stepping method. Hence, the variables at the dummy vertices outside a block are updated after every stage, which implies that the effects of the multiblock structure on the convergence are kept to a minimum. The frequency of data transfer between the blocks makes this method less

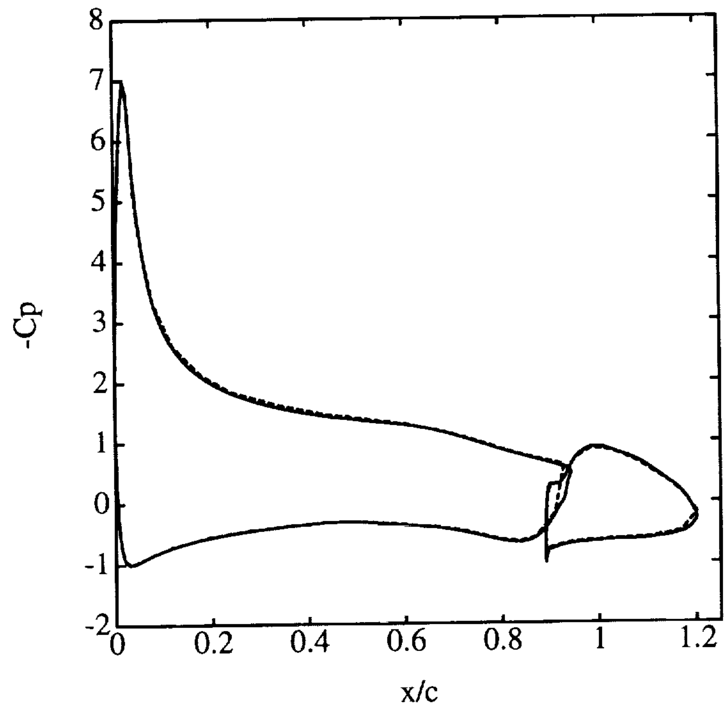


Figure 7: Viscous flow at an angle of incidence of 6.0°: comparison of the pressure coefficient on the airfoil between calculation (solid) and experiment (dashed).

efficient for parallel processing. However, with the block loop outside the stage loop, i.e. with an update of the dummy variables only after five flux evaluations, a converged solution could not be obtained. Apparently, the interval between two moments of data transfer between the blocks has to be sufficiently small in order to obtain a convergent multigrid method.

Further evidence for this statement is obtained from calculations with a three-stage instead of a five-stage Runge-Kutta time-stepping method. If the block loop is outside the stage loop, the dummy variables are updated after three flux evaluations. Although the rate of convergence is lower than in the case with the loops interchanged (see figure 8), the solution has converged within engineering accuracy after $\approx$ 200 W-cycles. A comparison of the three-stage and five-stage schemes with the block loop inside the stage loop shows that the five-stage scheme is more efficient: about 60 W-cycles suffice to get the residuals at the same level as with the three-stage scheme after 200 W-cycles. The five-stage scheme leads to a reduction in calculation time of approximately 60% in this instance.

DISCUSSION

We presented simulation results obtained with a multigrid multiblock method for a two-element airfoil. Both viscous and inviscid calculations were performed using the same multigrid process and the same vertex-based spatial discretization method. Moreover, either a three- or a five-stage

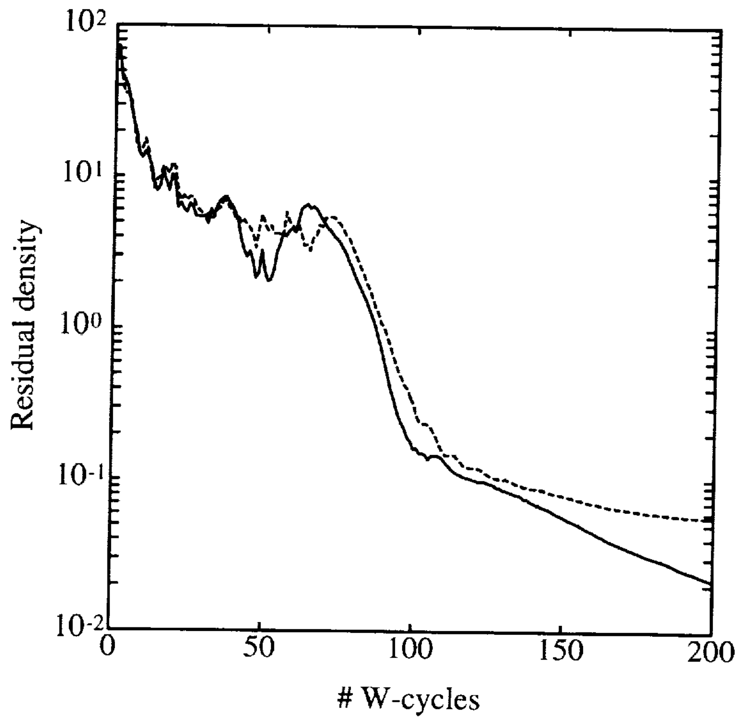


Figure 8: Convergence behavior of the three-stage Runge-Kutta scheme for turbulent flow; comparison between block loop inside (solid) and outside (dashed) stage loop.

Runge-Kutta scheme was considered for the integration in time and the smoothing properties of this relaxation method were further enhanced through the introduction of local time-stepping, implicit residual averaging in which the residuals at the block boundaries were kept fixed to their non-smoothed values.

The inviscid calculations have shown that a solution which is converged up to machine accuracy can be obtained with this multigrid method. A comparison with the single grid simulation method shows that a considerable reduction in calculation time was obtained with the multigrid method, although the convergence of the single grid method for inviscid calculations was already quite acceptable. We also investigated two different numerical boundary conditions at the solid walls. It appeared that linear extrapolation of the pressure not only leads to a better convergence than constant extrapolation, but also gives rise to a much smaller entropy layer around the airfoil. The resulting drag coefficient, which theoretically should equal zero in this subsonic flow, is reduced by almost 60%.

In the viscous calculations the single grid method was found to yield a well converged result in the 6°-case, however, the convergence towards the steady state solution was extremely slow and makes the use of a multigrid approach essential. A comparison of the calculation times required in both methods shows that a total speed-up with a factor of about 20 can be reached. The numerical predictions obtained for the lift- and pressure coefficients compare well with experimental results and give confidence in the use of the Baldwin-Lomax model for this application. The convergence of the multigrid process was studied in detail, showing that the ordering of the various loops in the process has a considerable effect. Interchanging the block and stage loops and keeping the grid loop as the outer loop, yields an optimal convergence when the block loop is put inside the stage loop. If the stage loop is put inside the block loop then convergence of the multigrid process was absent when using the five-stage Runge-Kutta method as the relaxation method. Apparently, the smoothing of the relaxation method becomes less effective as the number of stages between two 'updates' of the dummy-variables increases. This result has some less favorable consequences in view of a possible parallel processing of the multigrid method. On the one hand parallel processing seems more efficient if the frequency of data transfer between the blocks can be reduced. On the other hand the reduction of this frequency results in a reduction of the convergence rate of the multigrid process, and in some instances even to an absence of convergence. This suggests that in a possible parallel processing of this multigrid method, an optimal rate of data-exchange between the blocks should be determined.

Acknowledgement

The authors are greatly indebted to Frans Brandsma for several useful and stimulating discussions.

References

1. Brandsma, F.J.; Vogels, M.E.S.; Van der Vooren, J.; Dijkstra, D.; and Kuerten, J.G.M.: Predesign document of the ISNaS compressible flow simulator. ISNaS-88.04.027, April 1988.

2. Mavriplis, D.J.: Turbulent Flow Calculations using Unstructured and Adaptive Meshes. in *Proceedings of the 12th International Conference on Numerical Methods in Fluid Dynamics*, Oxford, July 9-13 1990, pp. 228-232.

3. Geurts, B.J.; and Kuerten, H.: Numerical Aspects of a Block Structured Compressible Flow Solver. *J. Engg. Math.*, vol. 27, 1993, pp. 195-214.

4. Baldwin, B.; and Lomax, H.: Thin layer approximation and algebraic model for separated turbulent flow. AIAA-78-257, 1978.

5. Kuerten, J.G.M.; Geurts, B.J.; Van der Burg, J.W.; Vreman, A.W.; and Zandbergen, P.J.: Development and applications of a 3-D compressible Navier-Stokes solver. in *Proceedings of the 13th International Conference on Numerical Methods in Fluid Dynamics*, Rome, July 6-10 1992.

6. Radespiel, R.: A cell-vertex multigrid method for the Navier-Stokes equations. NASA-TM-101557, 1989.

7. Van den Berg, B.: Boundary layer measurements on a two-dimensional wing with flap. NLR-TR-79009-U, 1979.

N94-23695

A NONCONFORMING MULTIGRID METHOD USING CONFORMING SUBSPACES*

S22-64
197154
P. 14

Chang Ock Lee
Department of Mathematics
University of Wisconsin-Madison

SUMMARY

For second-order elliptic boundary value problems, we develop a nonconforming multigrid method using the coarser-grid correction on the conforming finite element subspaces. The convergence proof with an arbitrary number of smoothing steps for $\mathcal{V}$-cycle is presented.

1. INTRODUCTION

Let Ω be a convex polygon in $\mathbf{R}^2$. Let $f \in L^2(\Omega)$, $\alpha \in C^1(\bar{\Omega})$ and $\beta \in C^0(\bar{\Omega})$. We assume there exists α_0 such that $\alpha \geq \alpha_0 > 0$ and $\beta \geq 0$. In this paper we discuss convergence properties of the multigrid method for solving the Dirichlet problem

$$-\nabla \cdot (\alpha \nabla u) + \beta u = f \quad \text{in} \quad \Omega, \tag{1}$$
$$u = 0 \quad \text{on} \quad \partial\Omega, \tag{2}$$

using $P1$ nonconforming finite elements(see [5, 6]).

The prototype of the multigrid convergence theory is that

> For some number of smoothing steps the multigrid process is a contraction for some norm. Moreover, the contraction number is independent of the mesh size h.

This was proved for conforming multigrid methods by Bank and Dupont[1]. Braess and Hackbusch[2] and Hackbusch[8] proved this for the $\mathcal{V}$ cycle with one smoothing step. For the nonconforming multigrid method, this was proved by Braess and Verfürth[3] and Brenner[4] for the $\mathcal{W}$-cycle under the condition that each iteration step contains many smoothing steps.

The method presented in this paper consists of a smoothing step on the nonconforming finite element space of the finest-grid and correction step which is obtained by the conforming multigrid

*This research was partially supported by the National Science Foundation under Grant No. CDA-9024618 and DMS-9203502.

method on the conforming finite element subspaces of coarser-grids. The *standard nonconforming multigrid* which was proved by Brenner in [4] is based on smoothings and correction on the nonconforming finite element spaces. The important difference is that $V_{k-1} \not\subseteq V_k$ and $W_{k-1} \subseteq V_k$, where V_k and W_k are the nonconforming and conforming finite element spaces on mesh level k, respectively. Hence we can simply use the natural injection for the intergrid transfer of grid functions and this intergrid transfer operator preserves the energy norm. Moreover, the error of the coarser-grid correction is orthogonal to W_{k-1}. Owing to these, the standard proof of convergence in [2] for the $\mathcal{V}$-cycle of one smoothing step of the conforming multigrid method carries over directly. In [3] Braess and Verfürth added the step length parameter in the correction step of the standard nonconforming multigrid algorithm to improve the convergence. They proved the convergence of two-level case of this *modified standard nonconforming multigrid* with one smoothing step. The rate of convergence of their algorithm should be better than or at least equal to that of the standard nonconforming multigrid method but it needs more cost for each iteration. While Brenner proved the convergence of the standard nonconforming multigrid algorithm only for the $\mathcal{W}$-cycle it is convergent for the $\mathcal{V}$ cycle with one smoothing step in real computation. Also the modified standard nonconforming multigrid algorithm converges for the $\mathcal{V}$ cycle with one smoothing step in real computation. Our multigrid method is easier to implement and more effective because it needs fewer computations and communications in a parallel sense. These computations were done in CM-5 Vector Units[†].

This paper is organized as follows. In Section 2 we discuss the fundamental estimates from the theory of finite elements and the intergrid transfer operator. The multigrid algorithm is discussed in Section 3. Section 4 contains the contracting properties of the k-level iteration. In the last section we compare the computational results of three algorithms.

2. THE FINITE ELEMENT SPACES

The variational formulation for (1) and (2) is defined as follows: Find $u \in H_0^1(\Omega)$ such that

$$a(u,v) = F(v) \quad \forall v \in H_0^1(\Omega),$$

where

$$a(u,v) = \int_\Omega (\alpha \nabla u \cdot \nabla v + \beta u v) \quad \text{and} \quad F(v) = \int_\Omega f v\,.$$

Here, $H_0^1(\Omega)$ denotes the usual Sobolev space (see [5]).

Let $\{\mathcal{T}^k\}$, $k \geq 1$, be a family of triangulations of Ω, where $\mathcal{T}^{k+1}$ is obtained by connecting the midpoints of the edges of the triangles in $\mathcal{T}^k$. Let $h_k := \max_{T\in\mathcal{T}^k} \operatorname{diam} T$, then $h_k = 2h_{k+1}$. Throughout this paper, C denotes the positive constant independent of k which may vary from occurence to occurence even in the proof of the same theorem.

[†]These results are based upon a test version of the software where the emphasis was on providing functionality and the tools necessary to begin testing the CM5 with vector units. This software release has not had the benefit of optimization or performance tuning and, consequently, is not necessarily representative of the performance of the full version of this software.

It is worth pointing out the motivation of the nonconforming finite elements. In the stationary Stokes problem for an incompressible viscous fluid, it is realized that a major difficulty exists in the numerical treatment of the incompressibility condition. Crouzeix and Raviart in [6] advocated the method that the incompressibility condition is approximated. They have found it very convenient to use nonconforming finite elements for this purpose. By Uzawa's method the Stokes equation is reduced to a sequence of Dirichlet problems for the operator $-\Delta$. Thus we shall develop a nonconforming multigrid method for solving (1) and (2).

Now let's define the nonconforming finite element space

$$V_k := \{v : v|_T \text{ is linear for all } T \in \mathcal{T}^k,\ v \text{ is continuous at the midpoints of the edges and } v = 0 \text{ at the mid points on } \partial\Omega\}.$$

Note that functions in V_k are not continuous.

We also use a conforming finite element space for our multigrid method *NC-CMG*. Define

$$W_k := \{w : w|_T \text{ is linear for all } T \in \mathcal{T}^k,\ w \text{ is continuous on } \Omega \text{ and } w|_{\partial\Omega} = 0\}.$$

The space V_k will be used in the finest-grid space and W_k in the coarser-grid spaces to obtain *NC-CMG*. Observe that $W_k = V_k \cap H_0^1(\Omega) = V_k \cap V_{k+1}$.

For each k, define (on $V_k + H_0^1(\Omega)$)

$$a_k(u,v) := \sum_{T\in\mathcal{T}^k} \int_T (\alpha \nabla u \cdot \nabla v + \beta u v)$$

and the energy norm induced by a_k

$$\|u\|_k := \sqrt{a_k(u,u)}.$$

The bilinear form $a_k(\cdot,\cdot)$ is symmetric and positive definite on V_k. Moreover, we have the inverse estimate[4]

$$\|u\|_k \le C h_k^{-1} \|u\|_{L^2} \quad \forall u \in V_k. \tag{3}$$

We also note that if $u, v \in H_0^1(\Omega)$, then $a_k(u,v) = a(u,v)$.

We now recall some fundamental estimates from the theory of finite elements.

Since $f \in L^2(\Omega)$, elliptic regularity implies that $u \in H^2(\Omega)$(see [7]). For the same f, let $u_k \in V_k$ satisfy

$$a_k(u_k, v) = \int_\Omega f v \quad \forall v \in V_k$$

and let $\tilde{u}_k \in W_k$ satisfy

$$a_k(\tilde{u}_k, v) = \int_\Omega f v \quad \forall v \in W_k.$$

Since V_k satisfies the patch test(see [11]), we have the following estimate for the discretization error:

$$\|u - u_k\|_{L^2} + h_k \|u - u_k\|_k \le C h_k^2 \|u\|_{H^2} \tag{4}$$

(see [6]). The estimate for the conforming descretization error is, of course, well known(see [5]):

$$\|u - \tilde{u}_k\|_{L^2} + h_k\|u - \tilde{u}_k\|_k \leq Ch_k^2\|u\|_{H^2}. \tag{5}$$

From the spectral theory, there exist eigenvalues $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{n_k}$ and eigenfunctions $\psi_1, \psi_2, \ldots, \psi_{n_k} \in V_k$, $(\psi_i, \psi_j)_{L^2} = \delta_{ij}$ (= the Kronecker delta), such that $a_k(\psi_i, v) = \lambda_i(\psi_i, v)_{L^2}$ for all $v \in V_k$. From the inverse estimate (3), there exists $C > 0$ such that

$$\lambda_i \leq Ch_k^{-2}. \tag{6}$$

The same results hold for the conforming finite element spaces. The norm $|||v|||_{s,k}$ is defined (see [1]) as follows:

$$|||v|||_{s,k} := \left(\sum_{i=1}^{n_k} \lambda_i^s \nu_i^2\right)^{1/2} \quad \text{where} \quad v = \sum_{i=1}^{n_k} \nu_i \psi_i \in V_k. \tag{7}$$

Moreover,

$$|||v|||_{0,k} = \|v\|_{L^2} \quad \text{and} \quad |||v|||_{1,k} = \|v\|_k. \tag{8}$$

And, the Cauchy-Schwarz inequality implies

$$|a_k(v, w)| \leq |||v|||_{1+t,k} |||w|||_{1-t,k}$$

for any $t \in \mathbf{R}$ and $v, w \in V_k$.

For $v \in V_{k-1}$ the intergrid transfer operator $I_{k-1}^k : V_{k-1} \to V_k$ is defined as follows. Let p be a midpoint of a side of a triangle in $\mathcal{T}^k$. If p lies in the interior of a triangle in $\mathcal{T}^{k-1}$, then we define

$$(I_{k-1}^k v)(p) := v(p).$$

Otherwise, if p lies on the common edge of two adjacent triangles T_1 and T_2 in $\mathcal{T}^{k-1}$, then we define

$$(I_{k-1}^k v)(p) := \frac{1}{2}[v|_{T_1}(p) + v|_{T_2}(p)].$$

From the definition of I_{k-1}^k, it is clear that

$$I_{k-1}^k v = v \quad \forall v \in W_{k-1} = V_k \cap V_{k-1} \subseteq H_0^1(\Omega).$$

In other words, $I_{k-1}^k|_{W_{k-1}}$ is just the natural injection.

Now we are ready to state an approximation property.

Lemma 1 *Given $u \in V_k$ let $u^\star \in W_{k-1}$ be the solution of*

$$a_k(u - u^\star, v) = 0 \quad \forall v \in W_{k-1}.$$

Then

$$|||u - u^\star|||_{1,k} \leq Ch_k|||u|||_{2,k}.$$

Proof. Let $g \in V_k$ satisfy

$$(g, v) = a_k(u, v) \quad \forall v \in V_k .$$

Then

$$\forall v \in W_{k-1}, \quad a_k(u^\star, v) = a_k(u, v) = (g, v).$$

Now let $w \in H_0^1(\Omega)$ be the solution of the Dirichlet problem

$$\begin{aligned} -\nabla \cdot (\alpha \nabla w) + \beta w &= g \quad \text{in } \Omega \\ w &= 0 \quad \text{on } \partial\Omega . \end{aligned}$$

Then by elliptic regularity $\|w\|_{H^2} \leq C\|g\|_{L^2}$. It follows from the discretization error estimates (4) and (5) that

$$\begin{aligned} \|u - u^\star\|_{L^2} &\leq \|u - w\|_{L^2} + \|w - u^\star\|_{L^2} & (9) \\ &\leq Ch^2\|w\|_{H^2} & (10) \\ &\leq Ch^2\|g\|_{L^2} . & (11) \end{aligned}$$

But

$$\|g\|_{L^2}^2 = (g, g) = a_k(u, g) \leq |\!|\!| u |\!|\!|_{2,k}\|g\|_{L^2} .$$

Therefore,

$$\|g\|_{L^2} \leq |\!|\!| u |\!|\!|_{2,k} .$$

Combining inverse estimate (3) and (11), we obtain

$$|\!|\!| u - u^\star |\!|\!|_{1,k} \leq \frac{C}{h}\|u - u^\star\|_{L^2} \leq Ch|\!|\!| u |\!|\!|_{2,k} . \quad \square$$

3. THE MULTIGRID ALGORITHM

Now, we consider a decreasing sequence of mesh size h_k:

$$h_0 > h_1 > \cdots > h_k > \cdots > h_{k_{\max}} .$$

We first describe the k-level iteration scheme of the conforming multigrid algorithm. The k-level iteration with initial guess z_0 yields $CMG(k, z_0, G)$ as a conforming approximate solution to the following problem.

Find $z \in W_k$ such that $a_k(z, v) = G(v) \quad \forall v \in W_k$, where $G \in W_k'$.

Here, W_k' is the dual space of W_k. For $k = 1$, $CMG(1, z_0, G)$ is the solution obtained from a direct method. For $k > 1$, $CMG(k, z_0, G) = z_m + I_{k-1}^k q$, where the approximation $z_m \in W_k$ is constructed recursively from the initial guess z_0 and the equations

$$z_i = z_{i-1} + \frac{1}{\Lambda_k}(G - A_k z_{i-1}), \quad 1 \leq i \leq m.$$

Here, Λ_k is greater than or equal to the largest eigenvalue of A_k which is the stiffness matrix of a_k in the conforming finite element space W_k, and m is an integer to be determined later. The coarser-grid correction $q \in W_{k-1}$ is obtained by applying the $(k-1)$-level iteration 1 time. In other words, it is the $\mathcal{V}$-cycle multigrid method. More precisely,

$$q = CMG(k-1, 0, \bar{G})$$

where $\bar{G} \in W'_{k-1}$ is defined by $\bar{G}(v) := G(I^k_{k-1}v) - a_k(z_m, I^k_{k-1}v)$ for all $v \in W_{k-1}$.

The nonconforming multigrid algorithm of this paper is as follows: The $k_{\max}$-level iteration with initial guess z_0 yields $NC\text{-}CMG(k_{\max}, z_0, F)$ as a nonconforming approximate solution to the following problem.

Find $z \in V_{k_{\max}}$ such that

$$a_{k_{\max}}(z, v) = F(v) = \int_\Omega fv \quad \forall v \in V_{k_{\max}}. \tag{12}$$

For $k_{\max} = 1$, $NC\text{-}CMG(1, z_0, F)$ is the solution obtained from a direct method.

For $k_{\max} > 1$,

Smoothing Step: the approximation $z_m \in V_k$ is constructed recursively from the initial guess z_0 and the equations

$$z_i = z_{i-1} + \frac{1}{\Lambda_{k_{\max}}}(F - A_{k_{\max}} z_{i-1}), \quad 1 \le i \le m\,. \tag{13}$$

Here, $\Lambda_{k_{\max}}$ is greater than or equal to the largest eigenvalue of $A_{k_{\max}}$ which is the stiffness matrix of $a_{k_{\max}}$ in the nonconforming finite element space $V_{k_{\max}}$.

Correction Step: The coarser-grid correction $q \in W_{k-1}$ is obtained by applying the $(k_{\max}-1)$-level conforming iteration 1 time. More precisely,

$$q = CMG(k_{\max}-1, 0, \bar{F})$$

where $\bar{F} \in W'_{k_{\max}-1}$ is defined by $\bar{F}(v) := F(I^k_{k-1}v) - a_k(z_m, I^k_{k-1}v)$ for all $v \in W_{k-1}$.

Put

$$NC\text{-}CMG(k_{\max}, z_0, F) = z_m + I^{k_{\max}}_{k_{\max}-1} q\,.$$

4. ESTIMATE OF CONVERGENCE RATE

Now, we can proceed with the well-known analysis of the conforming multigrid method in [2].

Define the linear mapping $J : V_k \to V_k$ by

$$Jw = \sum_i \nu_i \left(1 - \frac{\lambda_i}{\lambda_{\max}}\right) \psi_i \quad \text{for } w = \sum_i \nu_i \psi_i .$$

Here λ_i's are the eigenvalues of a_k. The smoothing step (13) amplifies the error $e_i = z - z_i$ by J, i.e., $e_i = Je_{i-1}$. Note that J is a self adjoint and semidefinite operator with respect to the energy norm.

Define the weaker seminorm

$$|w|^2 := \sum_i \lambda_i \left(1 - \frac{\lambda_i}{\lambda_{\max}}\right) \nu_i^2 \quad \text{for } w = \sum_i \nu_i \psi_i .$$

From (7) and (8) we know $\|w\|_k^2 = \sum \lambda_i \nu_i^2$ and $|w| \leq \|w\|_k$. Define the ratio

$$\rho(w) := \begin{cases} |w|^2/\|w\|_k^2 & \text{if } w \neq 0\,, \\ 0 & \text{if } w = 0 \end{cases} .$$

It can be regarded as a measure for the smoothness of $w \in V_k$ because for a smooth function the coefficient ν_i for small λ_i's dominate and $|w| \approx \|w\|_k$.

Lemma 2 *Given $w \in V_k$ put $\rho = \rho(J^m w)$. Then*

$$\|J^m w\|_k \leq \rho^m \|w\|_k .$$

Proof. Similar to the proof of Lemma 4.3. in [2]. □

Let $\bar{q}\,(\in W_{k-1})$ be the exact coarser-grid correction i.e.

$$a_{k-1}(\bar{q}, v) = F(v) - a_k(z_m, v) \quad \forall v \in W_{k-1} .$$

Define

$$Qe_m := e_m - \bar{q}$$

Then Q is the a_k-orthogonal projector from V_k into $W_{k-1}^{\perp}$. Note that $\bar{q}$ is a_k-orthogonal projection of e_m into W_{k-1}.

Lemma 3 *Given $w \in V_k$ we have*

$$|||Qw|||_{1,k} \leq \min\left\{1, C\sqrt{1 - \rho(w)}\right\} |||w|||_{1,k} .$$

Proof. For $w = \sum \nu_i \psi_i$, we have

$$\begin{aligned}
|||w|||_{1,k}^2 - |w|^2 &= \sum_i \lambda_i \nu_i^2 - \sum_i \lambda_i \left(1 - \frac{\lambda_i}{\lambda_{\max}}\right) \nu_i^2 \\
&= \frac{1}{\lambda_{\max}} \sum_i \lambda_i^2 \nu_i^2 = \frac{1}{\lambda_{\max}} |||w|||_{2,k}^2 .
\end{aligned}$$

It follows from Lemma 1 that $|||Qw|||_{1,k} \leq Ch|||w|||_{2,k}$. This and the estimate (6) for $\lambda_{\max}$ imply

$$\begin{aligned}
|||Qw|||_{1,k}^2 &\leq Ch^2 \lambda_{\max} (|||w|||_{1,k}^2 - |w|^2) \\
&\leq C(|||w|||_{1,k}^2 - |w|^2) \\
&= C(1 - \rho(w)) |||w|||_{1,k}^2
\end{aligned}$$

Moreover, since Q is an orthogonal projector, we have

$$|||Qw|||_{1,k}^2 \leq \min\left\{1, C\sqrt{1 - \rho(w)}\right\} |||w|||_{1,k}^2 . \quad \square$$

We are now (as in [10]) in a position to define three multigrid iterative schemes for the solution of (12).

1. the symmetric scheme *NC-CMGV*$_k$: symmetric smoothing *NC-CMG* scheme
2. the coarse-to-fine cycle *NC-CMG*$/_k$: postsmoothing *NC-CMG* scheme
3. the fine-to-coarse cycle: *NC-CMG*$\backslash_k$: our *NC-CMG* scheme.

In particular, we have[10]

$$\begin{aligned}
\|NC\text{-}CMG/_k\|_k &= \|NC\text{-}CMG\backslash_k\|_k , \\
\|NC\text{-}CMGV_k\|_k &= \|NC\text{-}CMG\backslash_k\|_k^2 .
\end{aligned}$$

The symmetrical method *NC-CMGV* enables us to use estimates with respect to the energy norm and to apply a duality argument.

Lemma 4 *The multigrid algorithm NC-CMGV$_k$ has a convergence factor*

$$\|NC\text{-}CMGV_k\|_k \leq \max_{0 \leq \rho \leq 1} \rho^{2m} \{\epsilon + (1 - \epsilon) \min(1, C[1 - \rho])\} , \tag{14}$$

with respect to the energy norm. ϵ is the error in $(k-1)$-level CMGV$_{k-1}$ and the constant C is independent of k and m.

We note that the right-hand side of (14) is a monotone function of ϵ due to the cut-off induced by the min-operation which is contained in the expression.

Proof.

$$z_{m+1} = z_m + \bar{q} + \epsilon w' \quad (\text{i.e. } \|q - \bar{q}\|_k \leq \epsilon\|\bar{q}\|_k)$$

with some $w' \in W_{k-1}$. Hence the error is

$$e_{m+1} = e_m - \bar{q} - \epsilon w' = Qe_m - \epsilon w'.$$

Since Qe_m is orthogonal to W_{k-1} and $w' \in W_{k-1}$, we get

$$\begin{aligned} \|Qe_m - w'\|_k^2 &= \|Qe_m\|_k^2 + \|w'\|_k^2 \leq \|Qe_m\|_k^2 + \|\bar{q}\|_k & (15)\\ &\leq \|Qe_m\|_k^2 + \|(I-Q)e_m\|_k^2 = \|e_m\|_k^2. & (16) \end{aligned}$$

In order to estimate the final error $e_{2m+1} = J^m e_{m+1}$, we use a duality argument:
$\|e_{2m+1}\|_k = \sup_{\hat{w}} a(\hat{w}, e_{2m+1})/\|w\|_k$. Note that (16) , $Q^2 = Q$ and Cauchy-Schwarz's inequality imply

$$\begin{aligned} a_k(\hat{w}, e_{2m+1}) &= a_k(\hat{w}, J^m(Qe_m - \epsilon w')) \\ &= a_k(J^m\hat{w}, (1-\epsilon)Q^2 e_m + \epsilon(Qe_m - w')) \\ &\leq (1-\epsilon)a_k(J^m\hat{w}, Q^2 e_m) + \epsilon\|J^m\hat{w}\|_k\|e_m\|_k \\ &\leq (1-\epsilon)\|QJ^m\hat{w}\|_k\|QJ^m e_0\|_k + \epsilon\|J^m\hat{w}\|_k\|J^m e_0\|_k \\ &\leq [(1-\epsilon)\|QJ^m\hat{w}\|_k^2 + \epsilon\|J^m\hat{w}\|_k^2]^{1/2}[(1-\epsilon)\|QJ^m e_0\|_k^2 + \epsilon\|J^m e_0\|_k^2]^{1/2}. \end{aligned}$$

Given $w \in V_k$ by the Lemmas 2 and 3 it follows that

$$(1-\epsilon)\|QJ^m w\|_k^2 + \epsilon\|J^m w\|_k^2 \leq \rho^{2m}\{\epsilon + (1-\epsilon)\min(1, C[1-\rho])\}\|w\|_k^2,$$

where $\rho = \rho(J^m w)$. Hence

$$\|e_{2m+1}\|_k \leq \max_{0\leq\rho\leq 1} \rho^{2m}\{\epsilon + (1-\epsilon)\min(1, C[1-\rho])\}\|e_0\|_k. \quad \square$$

Theorem 5 *If* $\|CMG\backslash_{k-1}\|_{k-1} \leq \delta^{1/2}$ *where* $\frac{C}{C+2m} \leq \delta < 1$, *then*

$$\|NC\text{-}CMG\backslash_k\|_k \leq \delta^{1/2}.$$

Proof. We conclude from Lemma 4,

$$\|NC\text{-}CMGV_k\|_k = \max_{0\leq\rho\leq 1} \rho^{2m}\{\delta + (1-\delta)\min(1, C[1-\rho])\},$$

because $\|CMGV_{k-1}\|_{k-1} = \|CMG\backslash_{k-1}\|_{k-1}^2 \leq \delta$. Maximum δ is attained at $\rho = 1$ when $\delta \geq \frac{C}{C+2m}$.

$$\|NC\text{-}CMG\backslash_k\|_k = \|NC\text{-}CMGV_k\|_k^{1/2} \leq \delta^{1/2}. \quad \square$$

Table I: Number of Grid = 8 i.e. h = 1/8

	S-NCMG		*M-NCMG*		*NC-CMG*	
smoothing	iter	time(sec)	iter	time(sec)	iter	time(sec)
1	4	.909	3	.788	3	.233
2	3	.689	2	.523	2	.156
3	2	.471	2	.540	2	.170
4	2	.483	2	.549	2	.177

Since the conforming multigrid method with the $\mathcal{V}$-cycle and arbitrary smoothing step is convergent we can choose δ such that $1 \geq \delta \geq \frac{C}{C+2m}$ and $\|CMG\backslash_{k-1}\|_{k-1} \leq \delta^{1/2}$.

5. EXPERIMENTAL RESULTS

We implement the standard nonconforming multigrid algorithm *S-NCMG* in [4], the modified standard nonconforming multigrid algorithm *M-NCMG* in [3] and *NC-CMG* with the $\mathcal{V}$-cycle for the Laplace's equation

$$\begin{aligned} -\Delta u &= -1 \quad \text{in} \quad \Omega = \text{unit square}, \\ u &= 0 \quad \text{on} \quad \partial\Omega. \end{aligned}$$

Let $\{\phi_1^k, \ldots, \phi_{n_k}^k\}$ be the basis of V_k such that each ϕ_j^k equals 1 at exactly one midpoint and equals 0 at all other midpoints. The stiffness matrix representing $a_k(\cdot,\cdot)$ with respect to this basis of nonconforming space has at most five entries per row. In the conforming case, the stiffness matrix has again at most five entries per row. Therefore z_m can be obtained from z_0 by iterating a sparse band matrix. We use the Gershgorin theorem in order to get the bounds of the maximum eigenvalues. These are the rough bounds so that the convergence rate is not optimal, but there is a trade-off because finding the exact maximum eigenvalue costs more. Note that the matrix for I_{k-1}^k has again at most five entries per row.

We take an initial guess $z_0 = 0$. The programs execute the multigrid iterations until the discrete energy norm of the real error is below the tolerance 1/(number of basis) for various mesh size and the number of smoothing. The real solution comes from the SSOR preconditioning conjugate gradient method for the five point finite difference scheme in which the difference of two consecutive solutions is less than the tolerance 10^{-9} in the descrete l_2 sense. The experiments reported here were run in double-precision arithmetic on CM-5 Vector Units which has 32K processors.

There are many ways to measure the performance of a parallel algorithm running on a parallel processor(see [9]). The most important and commonly used metric is the elapsed cpu time to run a job on a given machine even though it depends on how to optimize the program. We used the power method to get the rate of convergence. In the Table V–VIII the rate of convergence of *S-NCMG* and *M-NCMG* is slightly smaller or larger than the rate of convergence of *NC-CMG*.

Table II: Number of Grid = 16 i.e. h = 1/16

	S-NCMG		*M-NCMG*		*NC-CMG*	
smoothing	iter	time(sec)	iter	time(sec)	iter	time(sec)
1	7	2.604	5	2.089	5	.766
2	4	1.526	3	1.187	3	.481
3	3	1.183	3	1.247	3	.512
4	3	1.212	3	1.240	2	.360

Table III: Number of Grid = 32 i.e. h = 1/32

	S-NCMG		*M-NCMG*		*NC-CMG*	
smoothing	iter	time(sec)	iter	time(sec)	iter	time(sec)
1	10	6.037	7	4.294	7	1.625
2	6	3.723	5	3.163	4	.970
3	5	3.196	4	2.573	4	1.034
4	4	2.641	3	1.975	3	.832

Table IV: Number of Grid = 64 i.e. h = 1/64

	S-NCMG		*M-NCMG*		*NC-CMG*	
smoothing	iter	time(sec)	iter	time(sec)	iter	time(sec)
1	14	16.668	10	11.879	9	2.874
2	8	9.560	7	8.396	5	1.692
3	6	7.196	5	6.059	4	1.447
4	5	6.200	4	4.987	4	1.544

Table V: Number of Grid = 8 i.e. h = 1/8

	S-NCMG	*M-NCMG*	*NC-CMG*
smoothing	rate of conv.	rate of conv.	rate of conv.
1	.903	.903	.906
2	.815	.815	.820
3	.736	.736	.742
4	.665	.665	.672

Table VI: Number of Grid = 16 i.e. h = 1/16

	S-NCMG	*M-NCMG*	*NC-CMG*
smoothing	rate of conv.	rate of conv.	rate of conv.
1	.904	.904	.910
2	.817	.818	.829
3	.739	.739	.754
4	.668	.669	.687

Table VII: Number of Grid = 32 i.e. h = 1/32

	S-NCMG	*M-NCMG*	*NC-CMG*
smoothing	rate of conv.	rate of conv.	rate of conv.
1	.904	.904	.911
2	.818	.818	.830
3	.740	.740	.757
4	.669	.669	.689

Table VIII: Number of Grid = 64 i.e. h = 1/64

	S-NCMG	*M-NCMG*	*NC-CMG*
smoothing	rate of conv.	rate of conv.	rate of conv.
1	.904	.904	.911
2	.939	.818	.830
3	.888	.740	.757
4	.773	.669	.690

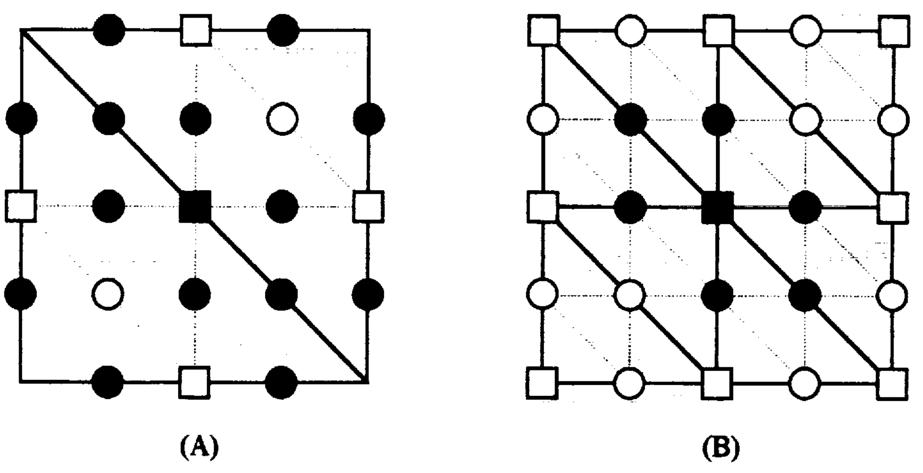


Figure 1: Nonconforming vs. conforming.

In Figure 1, (A) and (B) represent the location of the nodal basis of nonconforming finite elements and conforming finite elements, respectively. Squares represent the basis in V_{k-1} or W_{k-1} and circles represent the basis in V_k or W_k. In the correction step the centered black square is communicating with the black circles around it. Therefore *S-NCMG* and *M-NCMG* need further communications. Since the performance is determined mainly by the communication time in a massively parallel machine like CM-5, *S-NCMG* and *M-NCMG* require more cpu time than *NC-CMG*. It is shown in tables I–IV. Moreover *NC-CMG* does less computation and is easier to implement because the number of the basis of V_k is approximately three times of that of W_k and $W_{k-1} \subseteq W_k$.

Acknowledgements. I would like to thank Professor S. V. Parter for his advice.

REFERENCES

1. Bank, R. E.; and Dupont, T.: An optimal order process for solving finite element equations. *Math. Comp.*, vol. 36, 1981, pp. 35-51.
2. Braess, D.; and Hackbusch, W.: A new convergence proof for the multigrid including the V-cycle. *SIAM J. Numer. Anal.*, vol. 20, 1983, pp. 967-975.
3. Braess, D.; and Verfürth, R.: Multigrid methods for nonconforming finite element methods. *SIAM J. Numer. Anal.*, vol. 27, 1990, pp. 979-986.
4. Brenner, S.: An optimal-order multigrid method for P1 nonconforming finite elements. *Math. Comp.*, vol. 52, 1989, pp. 1-15.
5. Ciarlet, P.: *The finite element method for elliptic problems*, North-Holland, Amsterdam, 1978.
6. Crouzeix, M.; and Raviart, P.-A.: Conforming and nonconforming finite element methods for solving the stationary Stokes equations I. *RAIRO Anal. Numér. Sér. Rouge*, vol. 7, No. R-3, 1973, pp. 33-75.

7. Grisvard, P.: Behavior of solutions of an elliptic boundary value problem in polygonal or polyhedral domains. *Numerical solution of partial differential equations*-III (Synspade 1975) (B. Hubbard ed.), Academic Press, New York, 1976, pp. 207-274.

8. Hackbusch, W.: Multi-grid convergence theory. *Multigrid Methods*, Lecture Notes in Math., vol. 960 (W. Hackbusch and U. Trottenberg, eds.), Springer-Verlag, Berlin and New York, 1982, pp. 1-176.

9. Karp, H.; and Flatt, H.: Measuring parallel processor performance. *Comm. ACM*, vol. 33, 1990, pp. 539-543.

10. McCormick, S.: Multigrid methods for variational problems: Further results. *SIAM J. Numer. Anal.*, vol. 21, 1984, pp. 255-263.

11. Strang, G.; and Fix, G.: *An analysis of the finite element method*, Prentice-Hall, Englewood Cliffs, N.J., 1973.

N94-29696

MULTIGRID METHOD FOR INTEGRAL EQUATIONS AND AUTOMATIC PROGRAMS

S23-61
197155
P. 13

Hosae Lee
Department of Mathematics
Murray State University
Murray, KY 42071

SUMMARY

Several iterative algorithms based on multigrid methods are introduced for solving linear Fredholm integral equations of the second kind. Automatic programs based on these algorithms are introduced using Simpson's rule and the piecewise Gaussian rule for the numerical integration.

INTRODUCTION

Several multigrid iterative methods based on the Nyström method are applied for the fast solution of the large dense systems of equations that arise from the discretization of Fredholm integral equations of the second kind. We will consider the linear Fredholm integral equation of the second kind,

$$\lambda x(s) - \int_D k(s,t)x(t)dt = y(s), \quad s \in D \tag{1}$$

with D a bounded close domain, and y$\in X$ where X is the underlying Banach space. Necessary assumptions are

(i) $k(s,t)$ is such that the associated integral operator K is compact from X into X
(ii) λ is not an eigenvalue of K and $\lambda \neq 0$

The Nyström method for solving (1) uses some type of numerical integration to obtain the approximating equation

$$\lambda x_l(s) - \sum_{j=1}^{n_l} \alpha_j(s)x_l(t_j) = y(s), \quad s \in D \tag{2}$$

the nodes $t_1, t_2, ..., t_{n_l}$ are in D, and $x_l(t) \doteq x(t)$. The weights $\alpha_j(s)$ can be defined in a variety of ways, depending on the smoothness and form of the kernel function. If $k(s,t)$ and $x(t)$ are reasonably smooth, usually $\alpha_j(s) = w_j k(s,t_j)$, where

$$\int_D f(t)dt \approx \sum_{j=1}^{n} w_j f(t_j) \tag{3}$$

is a numerical integration formula. Let the numerical integration operator K_l be defined by

$$K_l x(s) = \sum_{j=1}^{n_l} w_j k(s, t_j) x(t_j), \qquad s \in D \tag{4}$$

Using (2) and (4), (1) approximated by the linear system

$$\lambda x_l(t_i) - \sum_{j=1}^{n_l} w_j k(t_i, t_j) x_l(t_j) = y(t_i) \tag{5}$$

We will denote (1) and (5) symbolically as

$$(\lambda - K)x = y \tag{6}$$

and

$$(\lambda - K_l)x_l = y \tag{7}$$

respectively. Our discussion is based on the convergence of a sequence of approximations to the unique solution of (1).

In finding numerical solutions for equations (1), the system (5) is too large to be solved directly. The purpose of this paper is to consider some iterative variants of (4). The basic assumptions needed in our algorithms are given in section 2. In section 3, linear iterative algorithms are given based on Simpson's rule and piecewise Gaussian quadrature rule for the numerical integraion formulae. And in the section 4, we include numerical examples.

BASIC ASSUMPTIONS

The methods will be defined and discussed using the abstract formulation of Anselone [1] and Atkinson [3], [4] for families of collectively compact operators.

Let $X_l, l = 0, 1, 2...$,be finite-dimensional subspaces of the Banach space X and let $P_l, l = 0, 1, 2, ...$, be a bounded projection operator from X onto X_l. We need the following assumptions for $\{X_l\}$ and $\{P_l\}$

(A1) $X_0 \subset X_1 \subset .... \subset X_l ... \subset X$
(A2) $\lim_{l \to \infty} \|f - P_l f\| = 0$ for all $f \in X$

The sequence $\{X_l\}$ is thought as being associated with a sequence of decreasing meshsizes $\{h_l\}$ with $\lim_{l\to\infty} h_l = 0$. Corresponding with this sequence $\{h_l\}$,we approximate K by a sequence of operators $\{K_l\}, K_l : X \longrightarrow X$. In multigrid iteration, the subscript l is called "level".The hypotheses on $\{K_l : l \geq 1\}$ and K are as follows.

(A3) K and $K_l, l \geq 1$ are linear operators on the Banach space X into X.
(A4) $K_l x \to Kx$ as $n \to \infty$, for all $x \in X$.
(A4) $\{K_l\}$ is a collectively compact family of operators.

The following is a consequence of the assumptions (A3) - (A5):

Lemma 1 *Assume (A3) - (A5). Then with n defined as in (3)*
(i) K is compact
(ii) $\|(K - K_l)K\|$ *and* $\|(K - K_l)K_l\|$ *converge to zero as* $n \to \infty$
(iii) If $a_l = \sup_{m\geq l}\sup_{n\geq 1} \|(K - K_m)K_n\|$, *then* $\lim_{l\to\infty} a_l = 0$

Proof. See Atkinson [4].

Lemma 2 *If* $(\lambda - K)^{-1}$ *exists, then*
$(\lambda - K_l)^{-1}$ *exists for sufficiently large l, say* $N(\lambda)$, *and is uniformly bounded by* $c_2(\lambda)$ *and*

$$\|x - x_l\| \leq c_2(\lambda)\,\|Kx - K_l x\|, \qquad l \geq N(\lambda)$$

where $x_l \equiv (\lambda - K_l)^{-1}y$

Proof. See Atkinson [4].

This shows $x_l \to x$ and gives a rate of convergence.

LINEAR ITERATIVE METHODS

Multigrid Methods

Assume that $x_{l,0}$ denotes a approximate solution of (7) with residual

$$d_l = y_l - (\lambda - K_l)x_{l,0} \tag{8}$$

Then improve on the accuracy by writing

$$x_{l,1} = x_{l,0} + \delta_l \tag{9}$$

where the correction δ_l satisfies the residual correction equation

$$(\lambda - K_l)\delta_l = d_l \tag{10}$$

In general, the correction term δ_l will be small, and it is unnecessary to solve the residual correction equation (10) exactly. Thus we may write

$$\delta_l = B_l d_l \tag{11}$$

where B_l denotes a bounded linear operator approximating $(\lambda - K_l)^{-1}$. By (??) and (9) together with (11), we obtain

$$x_{l,1} = [\lambda - B_l(\lambda - K_l)]x_{l,0} + B_l y_l \tag{12}$$

as the new approximate solution to (7). The equation (11) can be represented well by means of coarser grid functions

$$(\lambda - K_{l-1})\delta_{l-1} = d_{l-1} \tag{13}$$

where $d_{l-1}$is chosen reasonably and depends linearly on d_l. If $r : X_l \longrightarrow X_{l-1}$ is the restriction mapping, then

$$d_{l-1} = r d_l \tag{14}$$

Having defined $d_{l-1}$by (14), δ_{l-1} is obtained using (11) at level $l-1$. Having obtained δ_{l-1}which is defined only on the coarse grid level, we need to interpolate this coarse-grid function by

$$\tilde{\delta}_l = p\delta_{l-1} \tag{15}$$

where p describes the prolongation of a coarse grid function to a fine grid function.

We note here that the choice of the prolongation p in (15) must be small enough to satisfy

$$\|I - pr\| < C\, h_l^{\tau} \tag{16}$$

where the consistency order τ depends on the discretization. (e.g. on the order of the quadrature formula). For the restriction operator r, we will consider both trivial injection and Nystrom type restriction.

Our automatic algorithm is based on the following multigrid iteration which is given as a recursive procedure.

Multigrid iteration for solving $(\lambda - K_l)x_l = y$

Procedure Multigrid (l, x_l, y) (17)

```
if $l = 0$ then
    solve $x_0 = (\lambda - K_l)^{-1} y$
otherwise
```

```
$\bar{x}_l = \frac{1}{\lambda}[K_l x_l + y]$
$d_l = (I - K_l)\bar{x}_l - y$
$d_{l-1} = r d_l$
repeat the Procedure Multigrid with $(l-1, \delta_{l-1}, d_{l-1})$
$x_l^{new} = \bar{x}_l - p\delta_{l-1}$
```

We now give some basic results of the multigrid algorithm (17) that are used in our automatic algorithm.

Let ζ_k be the contraction number of the multigrid iteration employed at level k

$$\left\|x_k^{j+1} - x_k\right\| \leq \zeta_k \left\|x_k^j - x_k\right\| \tag{18}$$

Then it is known that $\{\zeta_k\}$ are uniformly bounded by some $\zeta < 1$.

Let

$$\zeta := \max_{1 \leq k \leq l} \zeta_k \tag{19}$$

where l is the maximum level in (17). The relative discretization error, the difference between x_k and x_{k-1}, is often estimated by

$$\begin{gathered}\|\tilde{p}x_{k-1} - x_k\| \leq C_1 h_k^\tau \\ for \;\; 1 \leq k \leq l\end{gathered} \tag{20}$$

where $\tilde{p}$ is a prolongation operator and τ is the consistency order.

Theorem 3 *Assume (20) and*

$$C_2 \zeta^i < 1 \tag{21}$$

with

$$C_2 := \max_{1 \leq k \leq l} \left[\frac{h_{k-1}}{h_k}\right]^\tau$$

then the i th iteration of the multigrid procedure (17) at level k results in $\tilde{x}_k$ and satisfies the error estimate

$$\begin{gathered}\|\tilde{x}_k - x_k\| \leq C_3 C_1 h_k{}^\tau \\ for \;\; 0 \leq k \leq l\end{gathered} \tag{22}$$

where

$$C_3 = \frac{\zeta^i}{1 - C_2 \zeta^i} \tag{23}$$

Proof. See Hackbush [11].

Theorem 4 *Assume the validity of (22) and suppose $\frac{h_{k-1}}{h_k} \leq \frac{1}{2}$ then the i th iteration of the multigrid procedure (17) at level k results in $\tilde{x}_k$ satisfies the error estimate*

$$\|\tilde{x}_k - x_k\| \leq C_4 \|x_k - x\| \tag{24}$$

where

$$C_4 = \frac{(2^\tau - 1)\zeta^i}{1 - C_2\zeta^i} \tag{25}$$

Proof. See Hackbush [11].

Automatic Algorithms

The automatic algorithm ζ_k in (18) is used to estimate the iteration error. Then together with the discretization error the global error in the solution is estimated. Often ζ_k is estimated by

$$\zeta_k \doteq \frac{\left\|x_k^{j+1} - x_k^j\right\|}{\left\|x_k^j - x_k^{j-1}\right\|} \tag{26}$$

Then

$$\left\|x_k - x_k^{j+1}\right\| \doteq \frac{\zeta_k}{1-\zeta_k}\left\|x_k^{j+1} - x_k^j\right\| \tag{27}$$

is used to estimate the iteration error. Thus at any level, a minimum of two iteration is required to estimate the iteration error. However, (24) together with (25) can be used to estimate ζ using

$$C_4 \doteq \frac{\textit{iteration error}}{\textit{discretization error}} \tag{28}$$

and it will enable us to estimate (27) with only one iteration.

Our first algorithm is based on Simpson's rule with double the node points as the level increases, i.e. dimension of the linear system at a level l is $2^{l+1} + 1$. In this case we have $C_2 = 16$ in (21). Thus by the condition (21), if $\zeta < \frac{1}{16}$ the estimates in (22) holds with i=1, i.e. only one multigrid iteration per level. The result is computational savings. As the level increases the amount of computation increases, so that there is a significant time savings in performing only one iteration as the dimension of the linear system being solved becomes larger. Moreover ζ_k in (18) goes to zero as the level k increases, which means that after a certain level k, ζ_k becomes so small that the iteration error becomes much less significant than the discretization error, hence more accurate estimation of it is not needed. Thus one iteration is sufficient at this stage.

The second algorithm is based on the piecewise Gaussian quadrature rule for the numerical integration scheme. We adapt the iteration error estimation scheme discussed earlier.

For simplicity we use $h_l = \frac{b-a}{2^l}$ for $l = 1, 2, \ldots$ This means that we reduce the length of each subinterval by half as the level increases. Suppose at some level l, we have a partition

$$Q_l = \{a = q_0 < q_1 < \ldots. < q_{m_l} = b\} \tag{29}$$

with

$$q_i = a + i * h_l \quad for \;\; i = 0, 1, 2, \ldots, m_l$$

and $m_l = 2^l$:=number of subintervals, for $l = 0, 1, 2, \ldots.$

Then

$$\int_a^b f(t)dt \doteq \sum_{i=1}^{m_l} h_i \sum_{j=1}^{p} \hat{w}_j \, f(q_{i-1} + h_i \hat{t}_j) \tag{30}$$

where

$$\int_0^1 f(t)dt \doteq \sum_{j=1}^{p} \hat{w}_j \, f(\hat{t}_j) \tag{31}$$

is the Gaussian quadrature rule on [0,1] with p node points.

Unlike Simpson's rule, we do not have nested node points. In the following algorithm, both restriction and prolongation are done with Nyström type interpolation.

Procedure Multigrid with piecewise Gaussian (l, x_l, y) (32)

```
if $l = 0$ then
    solve $x_0 = (\lambda - K_0)^{-1} y$
otherwise
    $\bar{x}_l = \frac{1}{\lambda}[K_l x_l + y]$
    $d_l = (\lambda - K_l)\bar{x}_l - y = K_l x_l - K_l \bar{x}_l$
    $d_{l-1} = r(K_l x_l - K_l \bar{x}_l)$
    repeat the Procedure Multigrid with $(l - 1, \delta_{l-1}, d_{l-1})$
    $x_l^{new} = \bar{x}_l - p\delta_{l-1}$
```

Nyström type interpolations as in the procedure (32) are costly. Each interpolation involves $O(n_l^2)$ multiplications at each level. However this can be improved as suggested in our conclusion later.

The following theorem which is due to Atkinson-Potra [7] gives the theoretical iterative rate of convergence for piecewise Gaussian quadrature with Nyström type interpolation. We will assume that the kernel $k(s,t)$ belongs to the class $G(\alpha,\gamma)$. This means that the kernel $k(s,t)$ has the following properties:

(G1) Define

$$\Psi_1 = \{(s,t) \mid a \leq s \leq t \leq b\}$$
$$\Psi_2 = \{(s,t) \mid a \leq t \leq s \leq b\}$$

Then there are functions $k_i \in C^{\alpha}(\Psi_i), i = 1,2$ with

$$k(s,t) = k_1(s,t), \quad (s,t) \in \Psi_1, \quad t \neq s$$
$$k(s,t) = k_2(s,t), \quad (s,t) \in \Psi_2$$

(G2) If $\gamma \geq 0$, then $k(s,t) \in C^{\gamma}([a,b] \times [a,b])$. If $\gamma = -1$, then the kernel $k(s,t)$ may have a discontinuity of the first kind along the line $t = s$

Theorem 5 *Assume that $k(s,t) \in G(\alpha,\gamma)$. Then solve the Nyström equation*

$$x_l(s) = \sum_{j=1}^{N} w_j k(s,t_j) x_l(t_j) + y(s) \tag{33}$$

using piecewise Gaussian quadrature rule with p node points in subintervals by first obtainning $x_l(t_1), \ldots., x_l(t_N)$ as a solution of the linear system

$$x_l(t_i) = \sum_{j=1}^{N} w_j k(t_i,t_j) x_l(t_j) + y(t_i) \tag{34}$$

then using (33) as an iterpolation formula gives an error estimate

$$\|x = x_l\| = O(h_l^{\,w}) \tag{35}$$

where $w = \min\{\alpha, 2p, \gamma + 2\}$.

Proof. See Atkinson-Potra [7] for the case p=r+1.

Finally to determine i, the needed number of iteration at any level l, use (24) and (25) with $\tau = 2p$,hence $C_2 = 2^{2p}$.

Automatic Implementation

Our automatic implementation is divided into two stages based on the results from the iteration method. In stage 1, $(\lambda - K_m)x_m = y$ is solved directly, and then an attempt is made to solve $(\lambda - K_l)x_l = y \quad for\ l > m$, iteratively. If the rate of convergence is sufficiently rapid then the stage 2 is entered. Otherwise m is replaced by l and the stage 1 is repeated. In stage 2, the value of m will serve as the coarsest grid level in the multigrid procedure (17) and solve $(\lambda - K_l)x_l = y$ iteratively until termination of the algorithm. The iteration procedure attempts to use the minimum number of iterates such that once the iterative solutions satisfy a certain criteria we will try to estimate the rate of convergence asymptotically, which enables the estimation of the rate of convergence with only one iteration per level. As shown in our numerical examples, this scheme results in computational savings at finer grid levels.

The initial guess for an iteration of the higher level is the interpolation of the solution of the preceding level which may have been obtained either directly or iteratively. The error $\|x - x_m\|$ and $\|x - x_l\|$ in stages 1 and 2, respectively, are monitored continuously, regardless of whether the iteration method is being used or not. Thus the multigrid iteration may not have been invoked successfully before the attainment of an answer within the desired error tolerance.

In order to estimate the global error in the current solution, we need to monitor the discretization error and the iteration error. For the iteration error estimation, (27) is used with estimated ζ in place of ζ_k. In stage 1, a test is made to determine whether the speed of convergence is sufficient to enter stage 2. If

$$\zeta \leq [\text{Ratio}]^{1/2} \tag{36}$$

then the speed of convergence is adequate for stage 2. This requirment will usually insure that only two iterates are needed to be calculated in stage 2 at any given level. The number Ratio is the theoretical rate at which the error in x_l should decrease when l is increased to the next level. In our case, since we are doubling the node points as the level increases, Ratio $= \left(\frac{1}{2}\right)^{\tau}$ with $\tau = 4$ for Simpson's rule and $\tau = 2p$ for p points piecewise Gaussian quadrature in each subinterval.

For the discretization error estimation, we compute the rate at which the error is decreasing for the current level. For each computed level l,

$$\text{NumDE} := \|x_l - x_{l-1}\| \tag{37}$$

and let DenDE be the previous value of NumDE, if any. Then the rate is computed using

$$\text{DE} := \frac{\text{NumDE}}{\text{DenDE}} \tag{38}$$

Using this value of DE, we estimate the error $x - x_l$,

$$\text{Error} := \left[\frac{\text{DE}}{1 - \text{DE}}\right] \text{NumDE} \tag{39}$$

which is a standard error estimate for sequences which are converging geometrically with a rate DE. Having estimated Error as in (39), we use the final test

$$\text{Error} \leq \epsilon \tag{40}$$

with ϵ a desired error tolerance supplied by the user.

To ensure that only needed accuracy in x_l is computed, we want to test

$$\text{iteration error} \leq \text{quadrature error} \tag{41}$$

This is done by

$$\left\|x_l^{(2)} - x_l^{(1)}\right\| \le \left(\frac{1-\zeta}{\zeta}\right)\left(\frac{\mathrm{DE}}{1 - \mathrm{DE}}\right)\left\|x_l^{(2)} - x_l^{(0)}\right\| \tag{42}$$

The test (42) is obtained by using (41) and the approximations

$$\|x - x_l\| \doteq \left(\frac{\mathrm{DE}}{1 - \mathrm{DE}}\right)\left\|x_l^{(2)} - x_l^{(0)}\right\| \tag{43}$$

$$\left\|x - x_l^{(2)}\right\| \doteq \left(\frac{\zeta}{1-\zeta}\right)\left\|x_l^{(2)} - x_l^{(1)}\right\| \tag{44}$$

If the test (42) is not satisfied, then the new iterate is calculated, and (42) is tested again. Once an iterate is acceptable according to (42), we check for accuracy in the most recently computed iterate using (39) and (40).

NUMERICAL EXAMPLES

The integral equation

$$x(s) - \lambda \int_a^b k(s,t)x(t)dt = y(s), \qquad a \le s \le b \tag{45}$$

is solved with the kernel

$$k(s,t) = \cos(\pi st)$$

on [0,1]. A variety of parameters λ that are close to the dominant characteristic values (the reciprocals of eigenvalues) are considered, as the equation becomes more difficult to solve as λ approaches characteristic values. The dominant characteristic value that we use in our example is 1.4278. The right hand function y(s) is so chosen that

$$x(s) = e^x \cos(7s), \quad 0 \le s \le 1 \tag{46}$$

Table I. The First Algorithm

				Dimension (Level)	
λ	Desired	Estimated	Actual	Coarsest	Finest
1.00	1.0E-6	6.82E-7	6.76E-7	3 (0)	65 (5)
1.40	1.0E-4	1.62E-5	1.60E-5	5 (1)	65 (5)
1.43	1.0E-4	1.31E-5	1.31E-5	5 (1)	129 (6)

In Table I, the Estimated column is computed using (39). As λ approaches the characteristic value of 1.4278, both the coarsest grid level and the finest grid level were increased. In Table II, we give the iterative rate of convergence at each level, and the number of iterations performed at each level is also given in parentheses. As noted in section 3, only one iteration is needed as the level increases. Whenever only one iteration is performed at any given level, the iterative rate of convergence is the maximum contraction number ζ in (19) estimated using (24) and (25).

Table II. Iterative Rate of Convergence of The First Algorithm

λ	Desired	Level 1	Level 2	Level 3
1.00	1.0E-6	2.10E-2 (2)	5.14E-2 (1)	2.03E-3 (1)
1.40	1.0E-4	2.10E-1 (2)	5.31E-2 (2)	7.57E-3 (2)
1.43	1.0E-4	-	1.44E-1 (2)	1.44E-2 (2)
		4	5	6
1.00	1.0E-6	3.40E-3 (1)	3.80E-3 (1)	-
1.40	1.0E-4	5.93E-2 (1)	3.79E-3 (1)	-
1.43	1.0E-4	4.40E-2 (1)	3.75E-3 (1)	3.89E-3 (1)

For the second algorithm, the coarsest level corresponds to two subintervals. In order to give a reasonable comparison with the first algorithm, we first give the results with 2 node points in each subinterval. Thus the quadrature order coincides with that of the first algorithm.

Table III. The Second Algorithm with p=2

				Dimension (Level)	
λ	Desired	Estimated	Actual	Coarsest	Finest
1.00	1.0E-6	6.82E-7	6.76E-7	4 (0)	64 (5)
1.40	1.0E-4	1.62E-5	1.60E-5	4 (0)	64 (5)
1.43	1.0E-5	8.74E-6	8.72E-6	4 (0)	128 (6)

In the next table, we have results from the second algorithm with more node points on each subinterval. To show the superiority of the Gaussian quadrature rule, we give results for a smaller desired error for $\lambda = 1.40$ and $\lambda = 1.43$.

Table IV. The Second Algorithm with p=3,4

					Dimension (Level)	
λ	p	Desired	Estimated	Actual	Coarsest	Finest
1.40	3	1.0E-8	1.95E-9	1.93E-9	6 (0)	96 (4)
1.43	3	1.0E-8	3.97E-10	3.96E-10	6 (0)	192 (5)
1.43	4	1.0E-8	6.52E-10	6.28E-10	8 (0)	64 (3)

Table V. Iterative Rate of Convergence of The Second Algoritm with p=3, 4

			Level		
λ	p	Desired	1	2	3
1.40	3	1.0E-8	1.09E-4 (2)	1.43E-6 (1)	8.84E-4 (1)
1.43	3	1.0E-8	1.39E-3 (2)	1.04E-2 (1)	2.10E-4 (1)
1.43	4	1.0E-8	9.35E-6 (2)	2.55E-3 (1)	1.30E-5 (1)
				4	5
1.40	3	1.0E-8		9.90E-4 (1)	-
1.43	3	1.0E-8		2.36E-4 (1)	2.42E-4 (1)
1.43	4	1.0E-8		-	-

CONCLUSION

The piecewise Gaussian rule is superior to Simpson's rule. However, as pointed out in section 3, restrictions and prolongations are done with Nyström type interpolation. And it involves $O(n_l^2)$ multiplications at each level l without counting kernel evaluations. It appears that these operations cause the bottleneck of our algorithms. We are in the process of applying the idea suggested by Achi Brandt in [9] to our current algorithms which will reduce the operation count by far. Our preliminary results appear to be promising, and progress is being made in developing them further.

REFERENCES

1. Anselone, P.M.: *Collectively Compact Operator Approximation Theory*, Prentice-Hall, 1971.

2. Atkinson, K.E.: The Numerical Solution of Fredholm Integral Equations of the Second Kind. SIAM J. Numer. Anal., Vol. 4, 1967, pp. 337-348.

3. Atkinson, K.E.: Iterative Variants of the Nyström Method for the Numerical Solution of Integral Equations. Numer. Math., Vol. 22, 1973, pp. 17-31.

4. Atkinson, K.E.: *A Survey of Numerical Methods for the Solution of Fredholm Integral Equations of the Second Kind*, SIAM, 1976.

5. Atkinson, K.E.; and Potra, F.A.: Projection and Iterative Projection Methods for Nonlinear Integral Equations, SIAM J. Numer. Anal., Vol. 20, 1987, pp. 1352-1373.

6. Atkinson, K.E.; and Potra, F.A.: The Discrete Galerkin Method for Nonlinear Integral Equations. J. Intgeral Eqns. and Appl., Vol. 1, no. 1, 1988, pp. 17-54.

7. Atkinson, K.E.; and Potra, F.A.: On the Discrete Galerkin Method for Fredholm Integral Equations of the Second Kind, IMA J. Numer. Anal., Vol. 9, 1989, pp. 385-403.

8. Brandt, A.: Multi-level Adaptive Solutions to Boundary-value Problems. Math. Comp., Vol. 31, 1977, pp. 333-390.

9. Brandt, A.: Multilevel Computations of Integral Transforms and Particle Interaction with Oscillatory Kernels. Comp. Phy. Comm., Vol. 65, 1991, pp. 24-38.

10. Chatelin, F.; and Lebbar, R.: Superconvergence Results for the Iterated Projection Method Applied to Fredholm Integral Equations of the Second Kind and the Corresponding Eigenvalue Problems, J. Int. Eqns. Vol. 6, 1984, pp. 71-91.

11. Hackbush, W.: *Multigrid Methods and Applications*, Springer-Verlag, 1985.

12. Hemker, P.W.; and Schippers, H.: Multiple Grid Method for the Solution of Fredholm Integral Equations of the Second Kind. Math. Comp., Vol. 36, 1981, pp. 215-232.

13. Hashimoto, M.: A Method of Solving Large Matrix Equations Reduced From Fredholm Integral Equations of the Second Kind. J. Assoc. Comp. Mach., Vol. 17, 1970, pp. 629-636.

14. Kantorovich, L.; and Akilov, G.: *Functional Analysis in Normed Spaces*, Pergamon Press, 1964.

15. Kress, R.: *Linear Integral Equations*, Springer-Verlag, Applied Math Sciences 82, 1989.

16. Schippers, H.: The Automatic Solution of Fredholm Equations of the Second Kind. Report NW 99/80, Mathematisch Centrum, Amsterdam 1980.

17. Schippers, H.: Application of Multigrid Methods for Integral Equations to Two Problems from Fluid Dynamics. J. of Comp. Physics, Vol 48, 1982, pp. 441-461.

18. Stetter, H.J.: The Defect Correction Principle and Discretization Methods, Numer. Math., Vol. 29, 1978, pp. 425-443.

N94-23697

AN OBJECT-ORIENTED APPROACH FOR PARALLEL SELF ADAPTIVE MESH REFINEMENT ON BLOCK STRUCTURED GRIDS[1]

S24-62
197156
P-15

Max Lemke[2] and Kristian Witsch
Mathematisches Institut der Universität Düsseldorf, Germany

Daniel Quinlan[2]
Computational Mathematics Group, University of Colorado, Denver

SUMMARY

Self-adaptive mesh refinement dynamically matches the computational demands of a solver for partial differential equations to the activity in the application's domain. In this paper we present two C++ class libraries, P++ and AMR++, which significantly simplify the development of sophisticated adaptive mesh refinement codes on (massively) parallel distributed memory architectures. The development is based on our previous research in this area. The C++ class libraries provide abstractions to separate the issues of developing parallel adaptive mesh refinement applications into those of parallelism, abstracted by P++, and adaptive mesh refinement, abstracted by AMR++. P++ is a parallel array class library to permit efficient development of architecture independent codes for structured grid applications, and AMR++ provides support for self-adaptive mesh refinement on block-structured grids of rectangular non overlapping blocks. Using these libraries the application programmers' work is greatly simplified to primarily specifying the serial single grid application, and obtaining the parallel and self-adaptive mesh refinement code with minimal effort.

Initial results for simple singular perturbation problems solved by self-adaptive multilevel techniques (FAC, AFAC), being implemented on the basis of prototypes of the P++/AMR++ environment, are presented. Singular perturbation problems frequently arise in large applications, e.g. in the area of computational fluid dynamics. They usually have solutions with layers which require adaptive mesh refinement and fast basic solvers in order to be resolved efficiently.

INTRODUCTION

The purpose of local mesh refinement during the solution of partial differential equations (PDEs) is to match computational demands to an application's activity: In a fluid flow problem this means that only regions of high local activity (shocks, boundary layers, etc.) can demand increased computational effort; regions of little flow activity (or interest) are more easily solved using only relatively little computational effort. In addition, the ability to adaptively tailor the computational mesh to the changing requirements of the application problem at runtime (e.g. moving fronts in time dependent problems) provides for much faster solution methods than static refinement or even uniform grid methods. Combined with increasingly powerful parallel computers that are becoming available, such methods allow for much larger and more comprehensive applications to be run. With local refinement methods, the greater disparity of scale introduced in larger applications can be addressed locally. Without local refinement, the resolution of smaller features in the applications domain can impose global limits either on the mesh size or the time step. The increased computational work associated with processing the global mesh cannot be readily offset even by the increased computational power of advanced parallel computers. Thus, local refinement is a natural part of the use of advanced massively parallel computers to process larger and more comprehensive applications.

[1]Revised and shortened version of [10]. This research has been supported by the National Aeronautics and Space Administration under grant number NASI-18606 and the German Federal Ministry of Research and Technology (BMFT) under PARANUSS, grant number ITR 900689.

[2]Part of this work belongs to the author's dissertation.

Our experiments with different local refinement algorithms for the solution of the simple potential flow equation on parallel distributed memory architectures (e.g. [8]) demonstrates that, with the correct choice of solvers, performance of local refinement codes shows no significant sign of degradation as more processors are used. In contrast to conventional wisdom, the fundamental techniques used in our adaptive mesh refinement methods do not oppose the requirements for efficient vectorization and parallelization. However, the best choice of the numerical algorithm is highly dependent on its parallelization capabilities, the specific application problem and its adaptive grid structure, and, last but not least, the target architectures' performance parameters. Algorithms that are expensive on serial and vector architectures, but are highly parallelizable, can be superior on one or several classes of parallel architectures.

Our previous work with parallel local refinement, which was done in the C language to better allow access to dynamic memory management, has permitted only simplified application problems on non block structured composite grids of rectangular patches. The work was complicated by the numerical properties of local refinement, including self adaptivity and their parallelization capabilities like, for example, static and dynamic load balancing. In particular, the explicit introduction of parallelism in the application code is very cumbersome. Software tools for simplifying this are not available, e.g., existing grid oriented communication libraries (as used in [6]) are far too restrictive to be efficiently applied to this kind of dynamic problem. Thus, extending this code for the solution of more general complex fluid flow problems on complicated block structured grids is limited by the software engineering problem of managing the large complexities of the application problem, the numerical treatment of self-adaptive mesh refinement, complicated grid structures, and explicit parallelization. The development of codes that are portable across different target architectures and that are applicable to not just one problem and algorithm, but to a larger class, is impossible under these conditions.

Our solution to this software difficulty presents abstractions as a means of handling the combined complexities of adaptivity, mesh refinement, the application specific algorithm, and parallelism. These abstractions greatly simplify the development of algorithms and codes for complex applications. As an example, the abstraction of parallelism permits the development of application codes (necessarily based on parallel algorithms as opposed to serial algorithms, whose data and computation structures do not allow parallelization) in the simplified serial environment, and the same code to be executed in a massively parallel distributed memory environment.

This paper introduces an innovative set of software tools to simplify the development of parallel adaptive mesh refinement codes for difficult algorithms. The tools are present in two parts, which form C++ class libraries and allow for the management of the great complexities described above. The first class library, P++ (short summary in Section 2, details in [10]), forms a data parallel superset of the C++ language with the commercial C++ array class library M++ (Dyad Software Corporation). A standard C++ compiler is used with no modifications of the compiler required. The second set of class libraries, AMR++ (Section 3), forms a superset of the C++/M++, or P++, environment and further specifies the combined environment for local refinement (or parallel local refinement). In Section 4 we introduce multilevel algorithms that allow for the introduction of self-adaptive mesh refinement (Asynchronous) Fast Adaptive Composite Methods (FAC and AFAC)). In Section 5, we present first results for a simple singular perturbation problem that has been solved using FAC and AFAC algorithms being implemented on the bases of AMR++ and P++ prototypes. This problem serves as a good model problem for complex fluid flow applications, because several of the properties that are related to self-adaptive mesh refinement are already present in it.

We are particularly grateful to Steve McCormick, without whose support this joint work would not have been possible, and to the people at the Federal German Research Center Jülich (KFA) for their generous support in letting us use their iPSC/860 environment. In addition we would like to thank everybody who discussed P++ or AMR++ with us or in any other way supported our work.

P++, A PARALLEL ARRAY CLASS LIBRARY FOR STRUCTURED GRIDS

P++ is an innovative, robust, and architecture-independent array class library that simplifies the development of efficient parallel programs for large scale scientific applications by abstracting parallelism. The target machines are current and evolving massively parallel distributed memory multiprocessor systems (e.g. Intel iPSC/860 and PARAGON, Connection Machine 5, Cray MPP, IBM RS 6000 networks) with different types of node architectures (scalar, vector, or superscalar). Through the use of portable communication and tool libraries (e.g. EXPRESS, ParaSoft Corporation), the requirements of shared memory computers are also addressed. The P++ parallel array class library is implemented in standard C++ using the serial M++ array class library, with absolutely no modification of the compiler. P++ allows for software development in the preferred serial environment, and such software to be efficiently run, unchanged, in all target environments. The runtime support for parallelism is both completely hidden and dynamic so that array partitions need not be fixed during execution. The added degree of freedom presented by parallel processing is exploited by use of an optimization module within the array class interface. For detail, please refer to [10].

Application class: The P++ application class is currently restricted to structured grid-oriented problems, which form a primary problem class currently represented in scientific supercomputing. This class is represented by dimensionally independent block structured grids (1D - 4D) with rectangular or logically rectangular grid blocks. The M++ array interface, which is also used as the P++ interface and whose functionality is similar to the array features of Fortran 90, is particularly well suited to express operations on grid blocks to the compiler and to the P++ environment at runtime.

Programming Model and Parallelism: P++ is based on a Single Program Multiple Data Stream (SPMD) programming model, which consists of executing one single program source on all nodes of the parallel system. Its combination with the Virtual Shared Grids (VSG) model of data parallelism (a restriction of virtual shared memory to structured grids, whose communication is controlled at runtime) is essential for the simplified representation of the parallel program using the serial program and hiding communication within the grid block classes. Besides different grid partitioning strategies, two communication update principles are provided and automatically selected at runtime: Overlap Update for very efficient nearest neighbor grid element access of aligned data and VSG Update for general grid (array) computations. By use of local partitioning tables, communication patterns are derived at runtime, and the appropriate send and receive messages of grid portions are automatically generated by P++ selecting the most efficient communication models for each operation. As opposed to general Virtual Shared Memory implementations, VSG allows for obtaining similar parallel performance as for codes based on the traditionally used explicit Message Passing programming model. Control flow oriented functional parallelism until now is not particularly supported in P++. However, a cooperation with the developers of CC++ ([4]) is planned.

Summary of P++ Features:

- Object oriented indexing of the array objects simplifies development of serial codes by removing error prone explicit indexing common to *for* or *do* loops.
- Algorithm and code development takes place in a serial environment. Serial codes are re-compilable to run in parallel without modification.
- P++ codes are portable between different architectures. Vectorization, parallelization and data partitioning are hidden from the user, except for optimization switches.
- P++ application codes exhibit communication as efficiently as codes with explicit message passing. With improved C++ compilers and an optimized implementation of M++, single node performance of C++ with array classes has the potential to approximate that of Fortran.

Current State, Performance Issues and Related Work: The P++ prototype is currently implemented on the bases of the AT&T C++ C-Front precompiler using the Intel NX-2 communication library (or, on an experimental basis, an EXPRESS-like portable communication library from Caltech). Current versions are running on the Intel iPSC/860 Hypercube, the Intel Simulator, SUN workstations, the Cray 2, and IBM PCs. The prototype contains all major concepts described above. At several points, without loss of generality, its functionality is restricted to the needs within our own set of test problems (3D multigrid codes and FAC/AFAC codes).

The feasibility of the approach has been proven by the successful implementation and use of our set of test problems on the basis of P++, in particular, the very complex AMR++ class library. The results that have been obtained with respect to parallel efficiency, whose optimization was one of the major goals of the P++ development, are also very satisfying: Comparisons for P++ and Fortran with message passing based test codes, respectively, have shown that the number of messages and the amount of communicated data is roughly the same. Thus, besides a negligible overhead, similar parallel efficiency can be achieved. With respect to single node performance, only little optimization has been done. The major reason is that the used system software components (AT&T C++ C-Front precompiler 2.1, M++) are not very well optimized for the target machines. However, our experiences with C++ array language class libraries on workstations and on the Cray Y-MP (in collaboration with Sandia National Laboratories: about 90% of the Fortran vector performance is achieved) are very promising: With new optimized system software versions, Fortran performance can be approximated. Therefore, altogether, we expect the parallel performance for P++ based codes to be similar to that obtained for optimized Fortran codes with explicit message passing.

AMR++, AN ADAPTIVE MESH REFINEMENT CLASS LIBRARY

AMR++ is a C++ class library that simplifies the details of building self-adaptive mesh refinement applications. The use of this class library significantly simplifies the construction of local refinement codes for both serial and parallel architectures. AMR++ has been developed in a serial environment using C++ and the M++ array class interface. It runs in a parallel environment, because M++ and P++ share the same array interface. The nested set of abstractions provided by AMR++ uses P++ at its lowest level to provide architecture independent support. Therefore, AMR++ inherits the machine targets of P++, and, thus, has a broad base of machines on which to run. The efficiency and performance of AMR++ is mostly dependent on the efficiency of M++ and P++, in the serial and parallel environments respectively. In this way, the P++ and AMR++ class libraries separate the abstractions of local refinement and parallelism to significantly ease the development of parallel adaptive mesh refinement applications in an architecture independent manner. The AMR++ class library represents work which combines complex numerical, computer science, and engineering application requirements. Therefore, the work naturally involves compromises in its initial development. In the following sections, the features and current restrictions of the AMR++ class library are summarized.

Block Structured Grids — Features and Restrictions: The target grid types of AMR++ are 2D and 3D block structured with rectangular or logically rectangular blocks. On the one hand, they allow for a very good representation of complex internal geometries introduced through local refinement in regions with increased local activity. This flexibility of local refinement block structured grids equally applies to global block structured grids that allow for matching complex external geometries. On the other hand, the restriction to structures of rectangular blocks, as opposed to fully unstructured grids, allows for the application of the VSG programming model of P++ and, therefore, is the foundation for good efficiency and performance in distributed environments, which is one of the major goals of the P++/AMR++ development. Thus, we believe that block structured grids are the best compromise between full generality of the grid structure and efficiency in a distributed parallel environment. The application class forms a broad cross section of important scientific applications.

In the following, the global grid is the finest uniformly discretized grid that covers the whole physical domain. Local refinement grids are formed from the global grid, or recursively from refinement grids, by standard refinement with $h_{fine} = \frac{1}{2}h_{coarse}$ in each coordinate direction. Thus, boundary lines of block structured refinement grids always match grid lines on the underlying discretization level. The construction of block structured grids in AMR++ has some practical limitations that simplify the design and use of the class libraries. Specifically, grid blocks at the same level of discretization cannot overlap. Block structures are formed by distinct or connected rectangular blocks that share their boundary points (block interfaces) at those places where they adjoin each other. Thus, a connected region of blocks forms a block structured refinement grid. It is possible that one refinement level consists of more than one disjunct block structured refinement grid. In the dynamic adaptive refinement procedure, refinement grids can be automatically merged, if they adjoin each other.

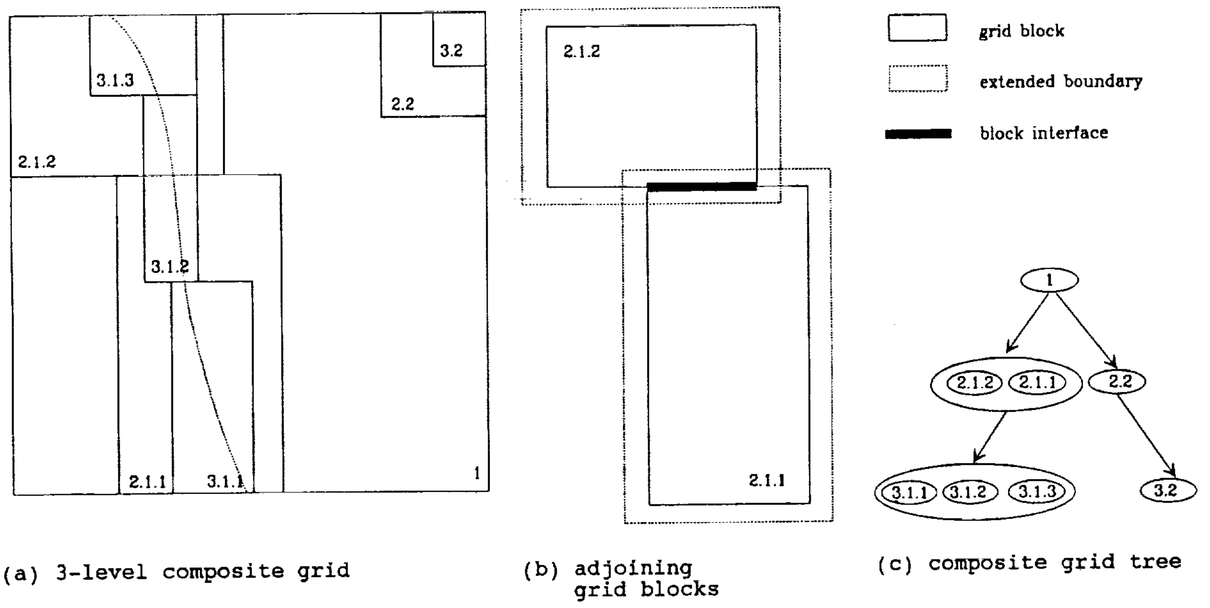


Figure 1: Example of a composite grid, its composite grid tree, and a cut out of 2 blocks with their extended boundaries and interface.

In Figure 1 (a), an example for a composite grid is illustrated: The composite grid shows a rectangular domain within which we center a curved front and a corner singularity. The grid blocks are ordered lexicographically: the first digit represents the level, the second digit the connected block structured refinement grid, and the third digit the grid block. Such problems could represent the structure of shock fronts or multi-fluid interfaces in fluid flow applications: In oil reservoir simulations, for example, the front could be an oil water front moving with time and the corner singularity could be a production well. In this specific example, the front is refined with two block structured refinement grids: the first grid on refinement level 2 is represented by grid blocks 2.1.1 and 2.1.2, and the second grid on level 2 by grid blocks 3.1.1, 3.1.2 and 3.1.3. In the corner on each of the levels, a single refinement block is introduced.

For ease of implementation, in the AMR++ prototype the global grid must be uniform. This simplification of the global geometry was necessary in order to be able to concentrate on the major issues of this work, namely, the implementation of local refinement and self adaptivity in an object-oriented environment. This restriction is not critical and can be eased in future versions of the prototype. Aside from implementation issues, some additional functionality must be made available:

- For implicit solvers, the resulting domain decomposition of the global grid may require special capabilities within the single grid solvers (e.g., multigrid solvers for block structured grids with adequate smoothers, such as inter-block line or plane relaxation methods).

- The block structures in the current AMR++ prototype are defined only by the needs of local refinement of a uniform global grid. This restriction allows them to be Cartesian. More complicated structures as they result from difficult non Cartesian external geometries (e.g., holes; see [11]) currently are not taken into consideration. An extension of AMR++, however, is principally possible. The wide experience for general 2D block structured grids that has been gained at GMD [11] can form a basis for these extensions. Whereas our work is comparably simple in 2D, because no explicit communication is required, extending the GMD work to 3D problems is very complex.

Some Implementation Issues: In the following, some implementation issues are detailed. They also demonstrate the complexity of a proper and efficient treatment of block structured grids and adaptive refinement. AMR++ takes care of all of these issues, which would otherwise have to be handled explicitly at the application level.

- Dimensional independence and multi-indexing: The implementation of most features of AMR++ and its user interface is dimensionally independent. Being derived from user requirements, on the lowest level, the AMR++ prototype is restricted to 2D and 3D applications. This, however, is a restriction that can easily be removed.

  One important means by which dimensional independence is reached, is multi-dimensional indices (multi-indices), which contain one index for each coordinate direction. On top of these multi-indices are index variants defined for each type of sub-block (interior, interior and boundary, boundary only, ...), which contain multiple multi-indices. For example, for addressing the boundary of a 3D block (non-convex), one multi-index is needed for each of the six planes. In order to avoid special treatment of physical boundaries, all index variants are defined twice, including and excluding the physical boundary, respectively. All index variants, several of them also including extended boundaries (see below), are precomputed at the time when a grid block is allocated. In the AMR++ user interface and in the top level classes, only index variants or indicators are used and, therefore, allow a dimensionally independent formulation, except for very low level implementations.

- Implementation of block structured grids: The AMR++ grid block objects consist of the interior, the boundary, an extended boundary of a grid block, and links that are formed between adjacent pairs of grid block objects. The links contain P++ array objects that do not consist of actual data, but serve as views (subarrays) of the overlapping parts of the extended boundary between adjacent grid block objects. The actual boundaries that are shared between different blocks (block interfaces) are very complex structures that are represented properly in the grid block objects. For example, in 3D, interfaces between blocks are 2D planes, those between plane-interfaces are 1D-line interfaces, and, further, those between line-interfaces are points (zero-dimensional).

  In Figure 1 (b), grid blocks 2.1.1 and 2.1.2 of the composite grid in Figure 1 (a) are depicted including their block interface and their extended boundary. The regular lines denote the outermost line of grid points of each block. Thus, with an extended boundary of two, there is one line of points between the block boundary line and the dashed line for the extended boundary. In its extended boundary, each grid block has views of the values of the original grid points of its adjoining neighboring block. This way it is possible to evaluate stencils on the interface and, with an extended boundary width of two, to also define a coarse level of the block structured refinement grid in multigrid sense.

- Data structures and iterators: In AMR++, the composite grid is stored as a tree of all refinement grids, with the global grid being the root. Block structured grids are stored as lists of blocks (for ease of implementation; collections of blocks would be sufficient in most cases).

In Figure 1 (c), the composite grid tree for the example composite grid in Figure 1 (a) is illustrated.

The user interface for doing operations on these data structures are so-called iterators. For example, for an operation on the composite grid (e.g., zeroing each level or interpolating a grid function to a finer level), an iterator is called that traverses the tree in the correct order (preorder, postorder, no order). This iterator as arguments takes the function to be executed and two indicators that specify the physical boundary treatment and the type of sub grid to be treated. The iteration starts at the root and recursively traverses the tree. For doing an operation (e.g. Jacobi relaxation) on a block structured grid, iterators are available, that process the list of blocks and all block interface lists. They take arguments similar to those for the composite grid tree iterators.

Object-Oriented Design and User Interface: The AMR++ class libraries are customizable by using the object oriented features of C++. For example, in order to obtain efficiency in the parallel environment, it may be necessary to introduce alternate iterators that traverse the composite grid tree or the blocks of a refinement region in a special order. This is implemented by alternate use of different base classes in the serial and parallel environment. The same is true for alternate composite grid cycling strategies as, for example, needed in AFAC, in contrast to FAC algorithms (Section 4). Application specific parts of AMR++, such as the single grid solvers or criteria for adaptivity, which have to be supplied by the user, are also simply specified through substitution of alternate base classes: A pre-existing application (e.g., problem setup and uniform grid solver) uses AMR++ to extend its functionality and to build an adaptive mesh refinement application. Thus, the user supplies a solver class and some additional required functionality (refinement criteria, ...) and uses the functionality of the highest level AMR++ ((Self_)Adaptive_)Composite_Grid class to formulate his special algorithm or to use one of the supplied PDE solvers. In the current prototype of AMR++, FAC and AFAC based solvers (Section 4) are supplied. If the single grid application is written using P++, then the resulting adaptive mesh refinement application is architecture independent, and so can be run efficiently in a parallel environment.

The design and interface of AMR++ is object-oriented and the implementation of our prototype extensively uses features like encapsulation and inheritance: The abstraction of self-adaptive local refinement, which involves the handling of many issues (including memory management, interface for application specific control, dynamic adaptivity, and efficiency), is reached through grouping these different functionalities in several interconnected classes. For example, memory management is greatly simplified by the object oriented organization of the AMR++ library: Issues such as lifetime of variables are handled automatically by the scoping rules for C++, so memory management is automatic and predictable. Also, the control over construction of the composite grid is intuitive and natural: The creation of composite grid objects is similar to the declaration of floating point or integer variables in procedural languages like Fortran and C. The user basically formulates a solver by allocating one of the predefined composite grid solver objects, or by formulating it on the basis of the composite grid objects and associated iterators and by supplying the single grid solver class.

Although not part of the current implementation of AMR++, C++ introduces a template mechanism in the latest standardization of the language, which is only just beginning to be part of commercial products. The general purpose of this template language feature is to permit class libraries to access user specified base types. For AMR++, for example, the template feature could be used to allow the specification of the base solver and adaptive criteria for the parallel adaptive local refinement implementation. In this way, the construction of an adaptive local refinement code from the single grid application on the basis of the AMR++ class library can become even simpler and cleaner. The object-oriented design of interconnected classes will not be further discussed. The reader is referred instead to [10] and [7].

Static and Dynamic Adaptivity, Grid Generation: In the current AMR++ prototype, static adaptivity is fully implemented. The user can specify a composite grid either interactively or by some input file: For each grid block, AMR++ needs its global coordinates and the parent grid block. Block structured local refinement regions are formed automatically by investigating neighboring relationships. In addition, the functionalities for adding and deleting grid blocks under user control are available within the Adaptive_Composite_Grid object of AMR++.

Recently, dynamic adaptivity has been a subject of intensive research. Initial results are very promising, and some basic functionality has been included in the AMR++ prototype: Given a global grid, a flagging criteria function, and some stopping criteria, the Self_Adaptive_Composite_Grid object contains the functionality for iteratively solving on the actual composite grid and generating a new discretization level on top of the respective finest level. Building a new composite grid level works as follows:

1. The flagging criteria delivers an unstructured collection of flagged points in each grid block. For representing grid block boundaries, all neighboring points of flagged points are also flagged.

2. The new set of grid blocks to contribute to the refinement level (gridding) is built by applying a smart recursive bisection algorithm similar to the one developed in [2]: If building a rectangle around all flagged points of the given grid block is too inefficient, it is bisected in the longer coordinate direction and new enclosing rectangles are computed. The efficiency of the respective fraction is measured by the ratio of flagged points to all points of the new grid block. In the following tests, 75% is used. This procedure is repeated recursively if any of the new rectangles is also inefficient. Having the goal of building the rectangles as large as possible within the given efficiency constraint, the choice of the bisection point (splitting in halves is too inefficient because it results in very many small rectangles) is done by a combination of signatures and edge detection. A detailed description of this method reaches beyond the scope of this paper, so the reader is referred to [2] or [7].

3. Finally, the new grid blocks are added to the composite grid to form the new refinement level. Grouping these blocks into connected block structured grids is done the same way as it is done in the static case.

This flagging and gridding algorithm has the potential for further optimization: The bisection method can be further improved, and a clustering and merging algorithm could be applied. This is especially true for refinement blocks of different parent blocks that could form one single block with more than one parent. Internal to AMR++, this kind of parent / child relationship is supported. The results in Section 5, however, show that the gridding already is quite good. The number of blocks that are constructed automatically is only slightly larger ($< 10\%$) than a manual construction would deliver. A next step in self-adaptive refinement would be to support time dependent problems whose composite grid structure changes dynamically with time (e.g., moving fronts). In this case, in addition to adding and deleting blocks, enlarging and diminishing blocks must be supported. Though some basic functionality and the implementation of the general concept is already available, this problem has not yet been further pursued.

Current State and Related Work: The AMR++ prototype is implemented using M++ and the AT&T Standard components class library to provide standardized classes (e.g., linked list classes). Through the shared interface of M++ and P++, AMR++ inherits all target architectures of P++. The prototype has been successfully tested on SUN workstations and on the Intel iPSC/860, where it has proved its full functionality with respect to parallelization. Taking into account the large application class of AMR++, there are still several insufficiencies and restrictions, as well as a large potential for optimization. For parallel environments, e. g., efficiently implementing self-adaptivity, including load (re)balancing, requires further research. In addition, the iterators that are currently available in AMR++, though working in a parallel environment, are best suited for serial environments. Special parallel iterators that, for example, support functional parallelism on the internal AMR++ level would have to be provided. Until now, AMR++ has been successfully used as a research tool for the algorithms and model problems described in the next two sections.

However, AMR++ provides the functionality to implement much more complicated application problems.

Concerning parallelization, running AMR++ under P++ on the Intel iPSC/860 has proven its full functionality. Intensive optimization, however, has only been done within P++. AMR++ itself offers a large potential for optimization.

To the authors' knowledge, the AMR++ approach is unique. There are several other developments in this area (e.g. [11]), but they either address a more restricted class of problems or are restricted to serial environments.

MULTILEVEL ALGORITHMS WITH ADAPTIVE MESH REFINEMENT

The fast adaptive composite grid method (FAC, [12]), which was originally developed from and is very similar to the Multi-Level Adaptive Technique (MLAT, [3]), is an algorithm that uses uniform grids, both global and local, to solve partial differential equations. This method is known to be highly efficient on scalar or single processor vector computers, due to its effective use of uniform grids and multiple levels of resolution of the solution. On distributed memory multiprocessors, methods like MLAT or FAC benefit from their tendency to create multiple isolated refinement regions, which may be effectively treated in parallel. However, for several problem classes, they suffer from the way in which the levels of refinement are treated sequentially in each region. Specifically, the finer levels must wait to be processed until the coarse-level approximations have been computed and passed to them; conversely, the coarser levels must wait until the finer level approximations have been computed and used to correct their equations. Thus, the parallelization potential of these "hierarchical" methods is restricted to intra-level parallelization.

The asynchronous fast adaptive composite method (AFAC) eliminates this bottleneck of parallelism. Through a simple mechanism used to reduce inter-level dependencies, individual refinement levels can be processed by AFAC in parallel. The result is that the convergence rate for AFAC is the square root of that for FAC. Therefore, since both AFAC and FAC have roughly the same number of floating point operations, AFAC requires twice the serial computational time as FAC, but AFAC allows for the introduction of inter-level parallelization.

As opposed to the original development of FAC and AFAC, in this paper, the modified algorithms known as FACx and AFACx are discussed and used. They differ in the treatment of the refinement levels. Whereas in FAC and AFAC, a rather accurate solution is computed (e.g., one MG V-cycle), FACx uses only a couple of relaxations. AFACx uses a two-grid procedure (of FMG-type) on the refinement level and its standard coarsening with several relaxations on each of these levels. Experiments and some theoretical observations show that all of the results that have been obtained for FAC and AFAC also hold for FACx and AFACx (see [14]). In the following, FAC and AFAC always denote the modified versions (FACx and AFACx).

Numerical algorithms: Both FAC (MLAT) and AFAC consist of two basic steps, which are described loosely as follows:

1. Smoothing phase: Given the solution approximation and composite grid residuals on each level, use relaxation or some restricted multigrid procedure to compute a correction local to that level (a better approximation is required on the global grid, the finest uniform discretization level).

2. Level transition phase: Combine the local corrections with the global solution approximation, compute the global composite grid residual, and transfer the local components of the approximation and residual to each level.

The difference between MLAT and FAC on the one hand and AFAC on the other hand is in the order in which the levels are processed and in the details of how they are combined:

- FAC and MLAT can roughly be viewed as standard multigrid methods with mesh refinement and a special treatment of the interfaces between the refinement levels and the underlying coarse level. In FAC and MLAT the treatment of the refinement levels is hierarchical. Theory on FAC is based on its interpretation as a multiplicative Schwarz Alternating Method or as a block relaxation method of Gauss-Seidel type.

  FAC and MLAT mainly differ by their motivation. Whereas it is the goal of FAC to compute a solution for the composite grid (grid points of the composite grid are all the interior points of the respective finest discretization level), the major goal of MLAT is to get the best possible solution on a given uniform grid (with using local refinement). Thus, in FAC, coarse levels of the composite grid serve for the computation of corrections. Therefore, FAC was originally formulated as a correction scheme (CS). The MLAT formulation requires a full approximation scheme (FAS), because coarse levels serve as correction levels for the points covered by finer levels. MLAT was first developed using finite difference discretization, whereas for FAC finite volume discretizations were used. However, they are closely related and in many problems lead to the same stencil representation. This is true except perhaps for the interface points, where finite volume discretizations generally lead to conservative discretizations (FAC), whereas finite difference discretizations do not (MLAT). Instead, in MLAT, usually a higher order interpolation is used on the interface. Other than this exception, because of the modification of the original FAC algorithm as discussed above, there is no difference in the treatment of the refinement levels between the original MLAT algorithm and the modified FAC algorithm that is discussed in this paper. It can be shown ([7]) that an FAS version of FAC with a special choice of the operators on the interface is equivalent to the originally developed Multilevel Adaptive Technique (MLAT).

- AFAC on the other hand consists of the same discretization and operators as FAC, but a decoupled and asynchronous treatment of the refinement levels in the solution phase, which dominates the arithmetic work in the algorithm. Theory on AFAC can be based on its interpretation as an additive Schwarz Alternating Method or as a block relaxation method of Jacobi type.

Theory in [12] shows that, under appropriate conditions, the convergence factors of FAC and AFAC have the relation $\rho_{AFAC} = \sqrt{\rho_{FAC}}$. This implies that two cycles of AFAC are roughly equivalent to one cycle of FAC. If the algorithmic components are chosen slightly different than for the convergence analysis or if applied to singular perturbation problems as discussed in the next section, experiences show that AFAC is usually better than as suggested by the above formula: In several cases, the convergence factor of AFAC shows only a slight degradation of the FAC rate (Section 5).

Parallelization - an Example for the Use of P++/AMR++: By example, we demonstrate some of the features of AMR++ and examples for the support of P++ for the design of parallel block structured local refinement applications on the basis of FAC and AFAC algorithms.

In a parallel environment, partitioning the composite grid levels becomes a central issue in the performance of composite grid solvers. In Figure 2, two different partitioning strategies that are supported within P++/AMR++ are illustrated for the composite grid in Figure 2. For ease of illustration, grid blocks 2.2 and 2.3 are not included. The so-called FAC partitioning in Figure 2 (b) is typical for implicit and explicit algorithms, where the local refinement levels have to be treated in a hierarchical manner (FAC, MLAT,...). The so-called AFAC partitioning in Figure 2 (a) can be optimal for implicit algorithms that allow an independent and asynchronous treatment of the refinement levels. In the case of AFAC, however, it must be taken into consideration that this partitioning is only optimal for the solution phase, which dominates the arithmetic work of the algorithm. The efficiency of the level transition phase, which is based on the same hierarchical structure as FAC and which can eventually dominate the aggregate communication work of the algorithm, highly depends on the architecture and the application (communication / computation ratio, single node (vector) performance, message latency, transfer rate, congestion, ...). For

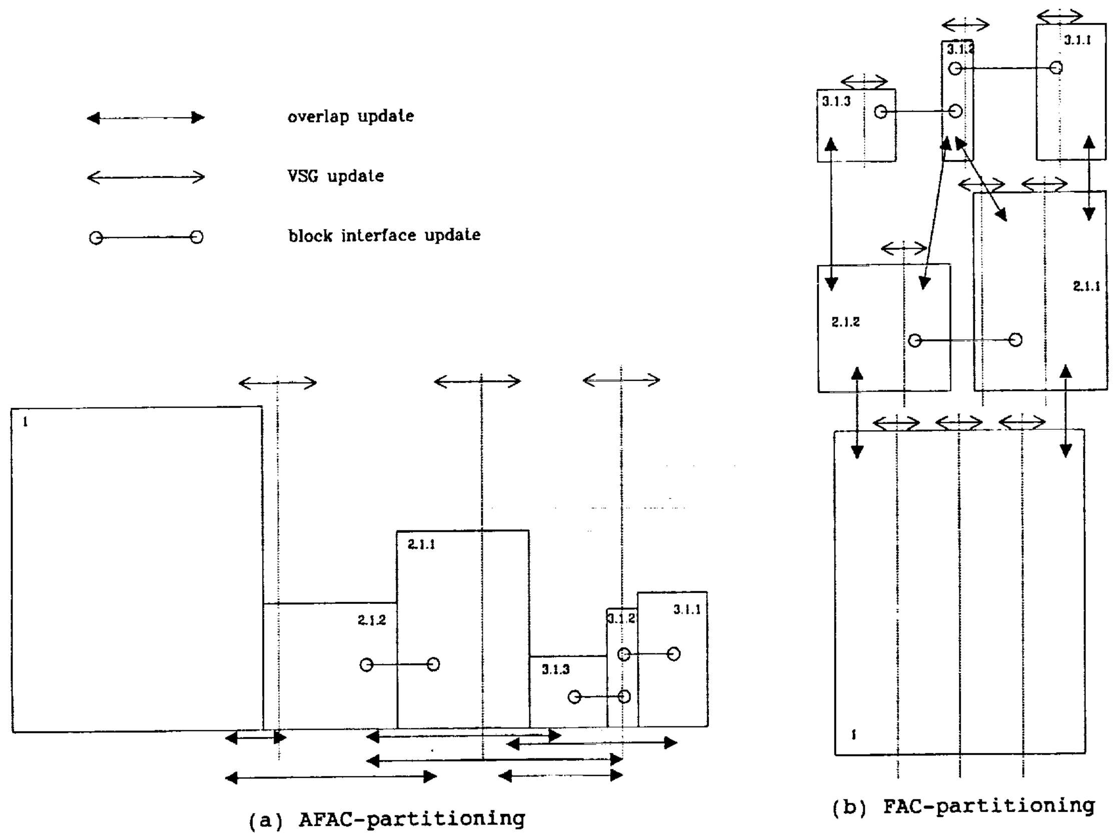


Figure 2: Parallel multilevel local refinement algorithms on block structured grids — an example for the use of AMR++ and the hidden interaction of the P++ communication models.

determining whether AFAC is better than FAC in a parallel environment, the aggregate efficiency and performance of both phases and the relation of the convergence rates must be properly evaluated. For more detail, see [10] and [7]. Both types of partitioning are supported in the P++/AMR++ environment.

Solvers used on the individually partitioned composite grid levels make use of overlap updates within P++ array expressions, which automatically provide communication as needed. The inter-grid transfers between local refinement levels, typically located on different processors, rely on VSG updates. The VSG updates are also provided automatically by the P++ environment. Thus, the underlying support of parallelism is isolated in P++ through either overlap update or VSG update, or a combination of both, and the details of parallelism are isolated away from the AMR++ application. The block structured interface update is handled in AMR++. However, communication is hidden in P++ (mostly the VSG update).

RESULTS FOR SINGULAR PERTURBATION PROBLEMS

Use of the tools described above is now demonstrated with initial examples. The adaptivity provided by AMR++ is necessary in case of large gradients or singularities in the solution of the PDE. They may be due to rapid changes in the right-hand side or coefficients of the PDE, corners in the domain, or singular perturbations. Here, the first and the last case will be examined on the basis of model problems.

Singularly perturbed PDEs represent the modelling of physical processes with relatively small diffusion (viscosity) and dominating convection. They may occur as a single equation or within systems of complex equations, e.g., as the momentum equations within the Navier-Stokes or, in addition, as supplementary transport equations in the Boussinesq system of equations. Here, we merely treat a single equation. However, we only use methods that generalize directly to more complex situations. Therefore, we do not rely on the direct solution methods provided by downstream or ILU relaxations for simple problems with pure upstream discretization. The latter are not direct solution methods for systems of equations. Further, these types of flow direction dependent relaxations are not efficiently parallelizable in the case of only a few relaxations as is usually used in multilevel methods. This in particular holds on massively parallel systems.

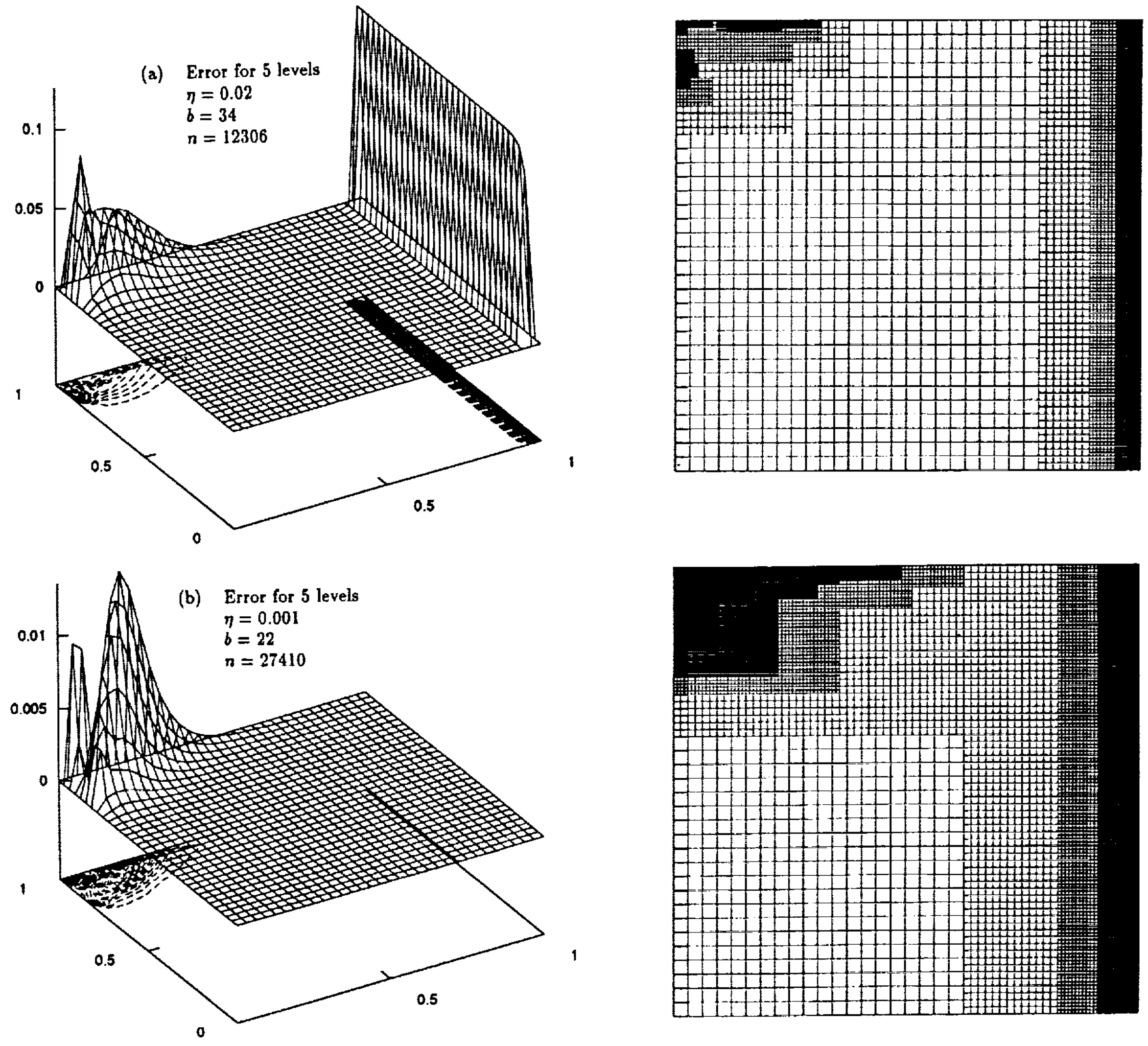


Figure 3: Results for a singular perturbation problem: Plots of the error and composite grid, with two different choices of the accuracy η in the self-adaptive refinement process.

Model Problem and Solvers: Numerical results have been obtained for the model problem

$$-\varepsilon\Delta u + au_x + bu_y = f \qquad \text{on } \Omega = (0,1)^2$$

with Dirichlet boundary conditions on $\partial\Omega$ and $\varepsilon = 0.00001$. This problem serves as a good model for complex fluid flow applications, because several of the properties that are related to self-adaptive mesh refinement are already present in this simple problem. The equation is discretized using isotropic artificial viscosity (diffusion):

$$\begin{aligned} L_h &:= -\varepsilon_h \Delta_h + aD_{2h,x}u + bD_{2h,y}u \text{ with } \Delta_h = D_{h,x}^2 + D_{h,y}^2 \\ \varepsilon_h &:= \max\{\varepsilon, \beta h \max\{|a|,|b|\}/2\} \end{aligned}$$

The discrete system is solved by multilevel methods – MG on the finest global grid and FAC or AFAC on composite grids with refinement. For the multigrid method, it is known that, with artificial viscosity, the two-grid convergence factor (spectral radius of the corresponding iteration matrix) is bounded below by 0.5 (for $h \to 0$). Therefore, multilevel convergence factors converge to 1.0 with an increasing number of levels. In [5], a multigrid variant which shows surprisingly good convergence behavior has been developed: MG convergence factors stay far below 0.5 (with three relaxations on each level). Here, essentially this method is used, which is described as follows:

- Discretization with additional isotropic artificial viscosity using $\beta = 3$ on the finest grid m and $\beta_{l-1} = 1/2\ (\beta_l + 1/\beta_l)$ for coarser grids $l = m-1, m-2, \ldots$,
- MG components: odd/even relaxation, non-symmetric transfer operators corresponding to linear finite elements. These components fulfil the Galerkin condition for the Laplacian.

Anisotropic artificial viscosity may also be used, but generally requires (parallel) zebra line relaxation, which has not yet been fully implemented.

For FAC and AFAC, the above MG method with V(2,1) cycling is used as a global grid solver. On the refinement levels, three relaxations are performed, and $\beta = 3$ is chosen on refinement grids.

Convergence Results: In Table 1, several convergence factors for FAC, AFAC, and, for comparison, for MG are shown. The finest grids have mesh sizes of $h = 1/64$ or $h = 1/512$, respectively. For FAC and AFAC, the global grid has the mesh size $h = 1/32$, the (predetermined) fine block always covers 1/2 of the parent coarse block along the boundary layer. The following conclusions can be drawn:

- For MG, the results are as expected. In the case of FAC and AFAC, the choice of β has to be further investigated.
- V cycles are used; W or F cycles would yield better convergence rates but worse parallel efficiency.
- If $\rho(FAC)$ is small, the expected result $\rho(AFAC) \approx \sqrt{\rho(FAC)}$ can be observed, otherwise $\rho(FAC) \approx \rho(AFAC) \ll \sqrt{\rho(FAC)}$.

	Poisson		SPP: $\beta = 3$		SPP: $\beta = 1$	
h	1/64	1/512	1/64	1/512	1/64	1/512
MG-V	0.14	0.14	0.17	0.30	0.18	0.50
FAC	0.17	0.18	0.30	0.65	0.30	0.80
AFAC	0.40	0.41	0.41	0.67	0.45	0.95

Table 1: Convergence factors for a singular perturbation problem (SPP: $a = b = 1, \varepsilon = 0.00001$) and, for comparison, for Poisson's equation.

Self-Adaptive Mesh Refinement Results: More interesting for the goal of this paper are applications of the self-adaptive process. As opposed to the convergence rates, they do not depend only on the PDE, but also on the particular solution. The results in this paper have been obtained for the exact solution

$$u(x) = \frac{e^{(x-1)/\varepsilon} - e^{-1/\varepsilon}}{1 - e^{-1/\varepsilon}} + \frac{1}{2}\, e^{-100(x^2+(y-1)^2)},$$

which has a boundary layer for $x = 1, 0 \leq y \leq 1$ and a steep hill around $x = 0, y = 1$. In order to measure the error of the approximate solution, a discrete approximation to the L_1 error norm is used. This is appropriate for this kind of problem: For solutions with discontinuities of the above type, one can observe 1st order convergence only with respect to this norm (no convergence in the L_∞ norm, order 0.5 in the L_2 norm).

The results have been obtained using the flagging criteria

$$h^f\ [\beta h \max\{|a|, |b|\}\ (|D^2_{h,x}u| + |D^2_{h,y}u|)]\ \geq\ \eta$$

with a given value of η. For $\varepsilon < \varepsilon_h$, the second factor is an approximation to the lowest order error term of the discretization. Based on experiments, $f = 1$ is a good choice. Starting with the global grid, the composite grid is self-adaptively built on the basis of the flagging and gridding algorithm described in Section 3.

	MG-V		FAC								
	uniform		$\eta = 0.02$			$\eta = 0.01$			$\eta = 0.001$		
h	e	n	e	n	b	e	n	b	e	n	b
1/32	0.0293	961	0.0293	961	1	0.0293	961	1	0.0293	961	1
1/64	0.0159	3969	0.0160	1806	4	0.0160	1967	4	0.0159	2757	3
1/128	0.0083	16129	0.0089	3430	10	0.0087	3971	10	0.0083	6212	7
1/256	0.0043	65025	0.0056	6378	19	0.0051	7943	16	0.0043	13473	12
1/512	0.0023	261121	0.0073	12306	34	0.0044	15909	30	0.0023	27410	22

Table 2: Accuracy (L1-norm e) vs. the number of grid points (n) and the number of blocks (b) for MG-V on a uniform grid and FAC on self-adaptively refined composite grids.

In Table 2, the results for MG and FAC are presented for three values of η. In Figure 3, two of the corresponding block structured grids are displayed. The corresponding error plots give an impression of the error distribution restricted from the composite grid to the global uniform grid. Thus, larger errors near the boundary layer are not visible. The results allow the following conclusions:

- In spite of the well known difficulties in error control of convection dominated problems, the grids that are constructed self-adaptively are reasonably well suited to the numerical problem.
- As long as the accuracy of the finest level is not reached, the error norm is approximatively proportional to η. As usual in error control by residuals, with the norm of the inverse operator being unknown, the constant factor is not known.
- If the refinement grid does not properly match the local activity, convergence rates significantly degrade and the error norm may even increase.
- Additional tests have shown that, if the boundary layer is fully resolved with an increased number of refinement levels, the discretization order, as expected, changes from one to two.
- The gridding algorithm is able to treat very complicated refinement structures efficiently: The number of blocks that are created is nearly minimal (compared to hand coding).

- Though this example needs relatively large refinement regions, the overall gain by using adaptive grids is more than 3.5 (taking into account the different number of points and the different convergence rates). For pure boundary layer problems, factors larger than 10 have been observed.
- These results have been obtained in a serial environment. AMR++, however, has been successfully tested in parallel. For performance and efficiency considerations, see Sect. 2 and 3.

References

[1] Balsara, D.; Lemke, M.; Quinlan, D.: AMR++, a parallel adaptive mesh refinement object class library for fluid flow problems; *Proceedings of the Symposium on Adaptive, Multilevel and Hierarchical Strategies*, ASME Winter Annual Meeting, Anaheim, CA, 1992.

[2] Bell, J; Berger, M.; Saltzman, J.; Welcome, M.: Three dimensional adaptive mesh refinement for hyperbolic conservation laws; *Internal Report*, Los Alamos National Laboratory.

[3] Brandt, A.: Multi-level adaptive solutions to boundary value problems; *Math. Comp.*, 31, 1977, pp. 333-390.

[4] Chandy, K.M.; Kesselman, C.: CC++: A Declarative Concurrent Object Oriented Programming Notation; California Institute of Technology, *Report*, Pasadena, 1992.

[5] Dörfer, J.: Mehrgitterverfahren bei singulären Störungen; *Dissertation*, Heinrich-Heine Universität Düsseldorf, 1990.

[6] Hempel, R.; Lemke, M.: Parallel black box multigrid; *Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods*, 1989, SIAM, Philadelphia.

[7] Lemke, M.: Multilevel Verfahren mit selbst-adaptiven Gitterverfeinerungen für Parallelrechner mit verteiltem Speicher; *Dissertation*, Universität Düsseldorf, to appear in 1993.

[8] Lemke, M.; Quinlan, D.: Fast adaptive composite grid methods on distributed parallel architectures; *Communications in Applied Numerical Methods*, Vol. 8, No. 9, Wiley, 1992.

[9] Lemke, M.; Quinlan, D.: P++, a C++ Virtual Shared Grids Based Programming Environment for Architecture-Independent Development of Structured Grid Applications; *Lecture Notes in Computer Science*, No. 634, Springer Verlag, September 1992.

[10] Lemke, M.; Quinlan, D.: An Object-Oriented Approach for Parallel Self-Adaptive Mesh Refinement on Block Structured Grids; Proceedings of the 9th GAMM-Seminar on Adaptive Methods, Kiel, Germany, 1993; *Notes of Numerical Fluid Mechanics*, Vieweg, to appear.

[11] Lonsdale, G; Schüller, A.: Multigrid efficiency for complex flow simulations on distributed memory machines; *Parallel Computing*, 19, 1993, pp23 - 32.

[12] McCormick, S.: Multilevel Adaptive Methods for Partial Differential Equations; *Frontiers in Applied Mathematics*, SIAM, Vol. 6, Philadelphia, 1989.

[13] McCormick, S.; Quinlan, D.: Asynchronous multilevel adaptive methods for solving partial differential equations on multiprocessors: Performance results; *Parallel Computing*, 12, 1989.

[14] McCormick, S.; Quinlan, D.: Idealized analysis of asynchronous multilevel methods; *Proceedings of the Symposium on Adaptive, Multilevel and Hierarchical Strategies*, ASME Winter Annual Meeting, Anaheim, CA, Nov. 8 - 13, 1992.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE November 1993	3. REPORT TYPE AND DATES COVERED Conference Publication

4. TITLE AND SUBTITLE Sixth Copper Mountain Conference on Multigrid Methods	5. FUNDING NUMBERS WU 505-59-53-01
6. AUTHOR(S) N. Duane Melson, T. A. Manteuffel, and S. F. McCormick, editors	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Langley Research Center Hampton, VA 23681-0001	8. PERFORMING ORGANIZATION REPORT NUMBER L-17275
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration, Washington, DC 20546-0001; Air Force Office of Scientific Research, Bolling AFB, Washington, DC 20338; the Department of Energy, Washington, DC 20585; and the National Science Foundation, Washington, DC 20550.	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CP-3224 Part 1

11. SUPPLEMENTARY NOTES

Organizing Institutions: University of Colorado at Denver, Front Range Scientific Computations, Inc., and the Society for Industrial and Applied Mathematics.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified–Unlimited Subject Category 64	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The Sixth Copper Mountain Conference on Multigrid Methods was held on April 4–9, 1993, at Copper Mountain, Colorado. This book is a collection of many of the papers presented at the conference and so represents the conference proceedings. NASA Langley graciously provided printing of this document so that all of the papers could be presented in a single forum. Each paper was reviewed by a member of the conference organizing committee under the coordination of the editors.

The multigrid discipline continues to expand and mature, as is evident from these proceedings. The vibrancy in this field is amply expressed in these important papers, and the collection clearly shows its rapid trend to further diversity and depth.

14. SUBJECT TERMS Multigrid; Algorithms; CFD	15. NUMBER OF PAGES 368
	16. PRICE CODE A16

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT