

NASA Conference Publication 3224  
- Part 2

# 15th Copper Mountain Conference on Multigrid Methods



$$U^H = \frac{1}{2} U^H + \frac{1}{2} U^H + v^H$$

$$F_H = F_H + \Delta F_H$$

$$G^{-1} \quad G^0 \quad G^1 \quad G^2$$

Proceedings of a workshop held at  
 Copper Mountain, Colorado  
 April 7-9, 1993

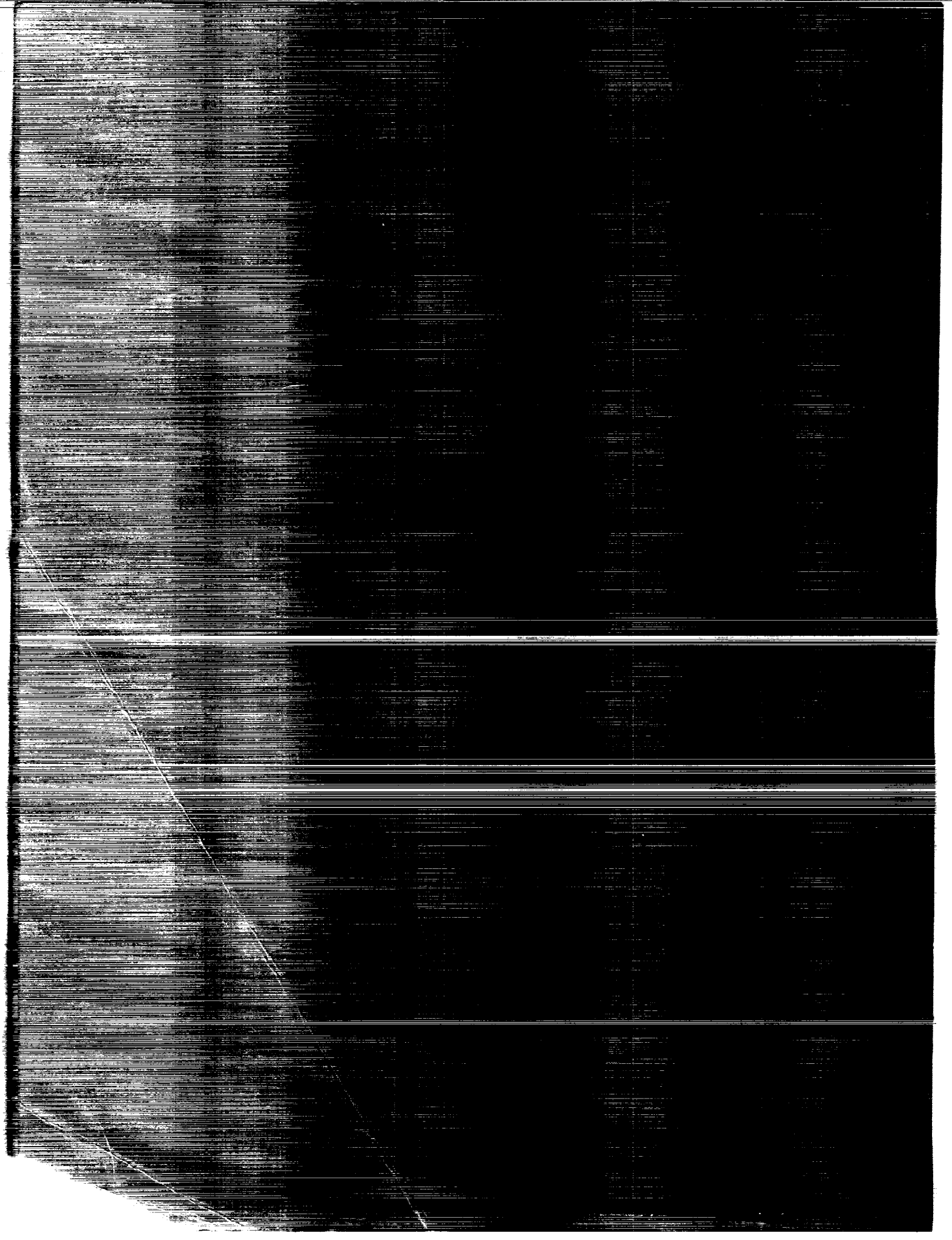
(NASA-CP-3224-Pt-2) THE SIXTH  
COPPER MOUNTAIN CONFERENCE ON  
MULTIGRID METHODS, PART 2 (NASA)  
342 p

N94-21464  
--THRU--  
N94-21487  
Unclass



The following text is a list of references:

H1/64 0197560



*NASA Conference Publication 3224*  
*Part 2*

# **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

*The use of trademarks or manufacturer's names in this publication does not constitute endorsement, either expressed or implied, by the National Aeronautics and Space Administration.*

PRECEDING PAGE BLANK NOT FILMED

## **ORGANIZING COMMITTEE**

**Joel Dendy**  
Los Alamos National Laboratory

**Craig Douglas**  
IBM and Yale University

**Paul Frederickson**  
RIACS

**Van Henson**  
Naval Postgraduate School

**Jan Mandel**  
The University of Colorado at Denver

**Duane Melson**  
NASA Langley Research Center

**Seymour Parter**  
University of Wisconsin - Madison

**Joseph Pasciak**  
Brookhaven National Lab

**Boris Rozovski**  
University of Southern California

**John Ruge**  
University of Colorado at Denver

**Klaus Stueben**  
Gesellschaft f. Math. u. Datenverarbeitung

**James Thomas**  
NASA Langley Research Center

**Pieter Wesseling**  
Delft University

**Olof Widlund**  
Courant Institute

# CONTENTS

<b>Preface</b> . . . . .	iii
<b>Organizing Committee</b> . . . . .	iv

## Part 1\*

<b>A Multigrid Solver for the Semiconductor Equations</b> . . . . .	1
Bernhard Bachmann and Asea Brown Boveri	
<b>FAS Multigrid Calculations of Three Dimensional Flow Using Non-Staggered Grids</b> . . . .	17
D. Matovic, A. Pollard, H. A. Becker, and E. W. Grandmaison	
<b>Multigrid and Cyclic Reduction Applied to the Helmholtz Equation</b> . . . . .	31
Kenneth Brackenridge	
<b>Uniform Convergence of Multigrid V-Cycle Iterations for Indefinite and Nonsymmetric Problems</b> . . . . .	43
James H. Bramble, Do Y. Kwak, and Joseph E. Pasciak	
<b>A Multilevel Cost-Space Approach to Solving the Balance Long Transportation Problem</b> . . . . .	61
Kevin J. Cavanaugh and Van Emden Henson	
<b>Vectorization and Parallelization of the Finite Strip Method for Dynamic Mindlin Plate Problems</b> . . . . .	77
Hsin-Chu Chen and Ai-Fang He	
<b>Domain Decomposition Methods for Nonconforming Finite Element Spaces of LaGrange-Type</b> . . . . .	93
Lawrence C. Cowsar	
<b>Nested Krylov Methods and Preserving the Orthogonality</b> . . . . .	111
Eric De Sturler and Diederik R. Fokkema	
<b>Implementing Abstract Multigrid or Multilevel Methods</b> . . . . .	127
Craig C. Douglas	
<b>Numerical Solution of Flame Sheet Problems with and without Multigrid Methods</b> . . . .	143
Craig C. Douglas and Alexandre Ern	
<b>A Mixed Method Poisson Solver for Three-Dimensional Self-Gravitating Astrophysical Fluid Dynamical Systems</b> . . . . .	159
Comer Duncan and Jim Jones	
<b>Multigrid Methods for Differential Equations with Highly Oscillatory Coefficients</b> . . . .	175
Bjorn Engquist and Erding Luo	

---

\*Part 1 is presented under separate cover.

<b>Application of Multigrid Methods to the Solution of Liquid Crystal Equations on a SIMD Computer</b>	191
Paul A. Farrell, Arden Ruttan, and Reinhardt R. Zeller	
<b>An Adaptive Multigrid Model for Hurricane Track Prediction</b>	207
Scott R. Fulton	
<b>Relaxation Schemes for Chebyshev Spectral Multigrid Methods</b>	215
Yimin Kang and Scott R. Fulton	
<b>Multigrid Methods for a Semilinear PDE in the Theory of Pseudoplastic Fluids</b>	231
Van Emden Henson and A. W. Shaker	
<b>A Multilevel Adaptive Projection Method for Unsteady Incompressible Flow</b>	243
Louis H. Howell	
<b>Wavelet Multiresolution Analyses Adapted for the Fast Solution of Boundary Value Ordinary Differential Equations</b>	259
Björn Jawerth and Wim Sweldens	
<b>Comparison of Locally Adaptive Multigrid Methods: L.D.C., F.A.C. and F.I.C.</b>	275
Khodor Khadra, Philippe Angot, and Jean-Paul Caltagirone	
<b>Multi-Grid Domain Decomposition Approach for Solution of Navier-Stokes Equations in Primitive Variable Form</b>	293
Hwar-Ching Ku and Bala Ramaswamy	
<b>Compressible Turbulent Flow Simulation with a Multigrid Multiblock Method</b>	305
Hans Kuerten and Bernard Geurts	
<b>A Nonconforming Multigrid Method Using Conforming Subspaces</b>	317
Chang Ock Lee	
<b>Multigrid Method for Integral Equations and Automatic Programs</b>	331
Hosae Lee	
<b>An Object-Oriented Approach for Parallel Self Adaptive Mesh Refinement on Block Structured Grids</b>	345
Max Lemke, Kristian Witsch, and Daniel Quinlan	

## **Part 2**

<b>Optimal Resolution in Maximum Entropy Image Reconstruction from Projections with Multigrid Acceleration</b>	361-1
Mark A. Limber, Tom A. Manteuffel, Stephen F. McCormick, and David S. Sholl	
<b>Flow Transition with 2-D Roughness Elements in a 3-D Channel</b>	377-2
Zhining Liu, Chaoqun Liu, and Stephen F. McCormick	
<b>Multilevel Methods for Transport Equations in Diffusive Regimes</b>	393-3
Thomas A. Manteuffel and Klaus Ressel	

<b>Analysis of Multigrid Methods on Massively Parallel Computers: Architectural Implications</b> . . . . .	405-4
Lesley R. Matheson and Robert E. Tarjan	
<b>Time-Accurate Navier-Stokes Calculations with Multigrid Acceleration</b> . . . . .	423-5
N. Duane Melson, Mark S. Sanetrik, and Harold L. Atkins	
<b>MGGHAT: Elliptic PDE Software with Adaptive Refinement, Multigrid and High Order Finite Elements</b> . . . . .	439-6
William F. Mitchell	
<b>Looking for <math>O(N)</math> Navier-Stokes Solutions on Non-Structured Meshes</b> . . . . .	449-7
Eric Morano and Alain Dervieux	
<b>The Block Adaptive Multigrid Method Applied to the Solution of the Euler Equations</b> . . . . .	465-8
Nikos Pantelelis	
<b>Multigrid Schemes for Viscous Hypersonic Flows</b> . . . . .	481-9
R. Radespiel and R. C. Swanson	
<b>Layout Optimization with Algebraic Multigrid Methods</b> . . . . .	497-10
Hans Regler and Ulrich Rude	
<b>Optimal Convolution SOR Acceleration of Waveform Relaxation with Application to Semiconductor Device Simulation</b> . . . . .	513-11
Mark Reichelt	
<b>A Multigrid Method for Steady Euler Equations on Unstructured Adaptive Grids</b> . . . . .	527-12
Kris Rienslagh and Erik Dick	
<b>Two-Level Schwarz Methods for Nonconforming Finite Elements and Discontinuous Coefficients</b> . . . . .	543-13
Marcus Sarkis	
<b>An Automatic Multigrid Method for the Solution of Sparse Linear Systems</b> . . . . .	567-14
Yair Shapira, Moshe Israeli, and Avram Sidi	
<b>A Multigrid Algorithm for the Cell-Centered Finite Difference Scheme</b> . . . . .	583-15
Richard E. Ewing and Jian Shen	
<b>A Semi-Lagrangian Approach to the Shallow Water Equations</b> . . . . .	593-16
J. R. Bates, Stephen F. McCormick, John Ruge, David S. Sholl, and Irad Yavneh	
<b>Multigrid Solution of the Navier-Stokes Equations on Highly Stretched Grids with Defect Correction</b> . . . . .	605-17
Peter M. Sockol	
<b>The Multigrid Preconditioned Conjugate Gradient Method</b> . . . . .	621-18
Osamu Tatebe	
<b>Mapping Robust Parallel Multigrid Algorithms to Scalable Memory Architectures</b> . . . . .	635-19
Andrea Overman and John Van Rosendale	

<b>Unstructured Multigrid Through Agglomeration</b> . . . . .	649	20
V. Venkatakrishnan, D. J. Mavriplis, and M. J. Berger		
<b>Multigrid Properties of Upwind-Biased Data Reconstruction</b> . . . . .	663	21
Gary P. Warren and Thomas W. Roberts		
<b>On the Prediction of Multigrid Efficiency Through Local Mode Analysis</b> . . . . .	679	22
R. V. Wilson		
<b>Numerical Study of a Multigrid Method with Four Smoothing Methods for the Incompressible Navier-Stokes Equations in General Coordinates</b> . . . . .	691	
S. Zeng and P. Wesseling		

Part 2

51-64

197561

N 94-21465

OPTIMAL RESOLUTION IN  
MAXIMUM ENTROPY IMAGE RECONSTRUCTION  
FROM PROJECTIONS WITH  
MULTIGRID ACCELERATION

Mark A. Limber  
Tom A. Manteuffel  
Stephen F. McCormick

*Computational Mathematics Group  
University of Colorado at Denver  
Denver, CO.*

David S. Sholl

*Program in Applied Mathematics  
University of Colorado at Boulder  
Boulder, CO.*

**Abstract**

We consider the problem of image reconstruction from a finite number of projections over the space  $L^1(\Omega)$ , where  $\Omega$  is a compact subset of  $\mathbb{R}^2$ . We prove that, given a discretization of the projection space, the function that generates the correct projection data and maximizes the Boltzmann-Shannon entropy is piecewise constant on a certain discretization of  $\Omega$ , which we call the "optimal grid". It is on this grid that one obtains the maximum resolution given the problem setup. The size of this grid grows very quickly as the number of projections and number of cells per projection grow, indicating fast computational methods are essential to make its use feasible.

We use a Fenchel duality formulation of the problem to keep the number of variables small while still using the optimal discretization, and propose a multilevel scheme to improve convergence of a simple cyclic maximization scheme applied to the dual problem.

# 1 Introduction

In computerized tomography (CT), one encounters the problem of reconstructing an image, or a density, defined by the function  $\hat{x}(s, t)$ , given only a finite number of projection data. General references include [1] and [2].

The projection data is typically of the form

$$b_k^m = \int_{\Omega_k^m} \hat{x}(s, t) ds dt \quad (1.1)$$

where the support of  $\hat{x}$  is assumed to lie in the bounded region  $\Omega \subset \mathbb{R}^2$  and  $\Omega_k^m$  is the  $k^{\text{th}}$  strip orthogonal to the  $m^{\text{th}}$  projection (see Figure 1). We assume that there are  $M$  projections and that the  $m^{\text{th}}$  projection has  $K_m$  cells. Let  $b$  be the vector of projection data,  $N$  the length of  $b$ , and  $\psi_k^m$  the characteristic function of  $\Omega_k^m$ . We then rewrite (1.1) as

$$b = A\hat{x}, \quad (1.2)$$

where  $A : L^1(\Omega) \rightarrow \mathbb{R}^N$  via

$$(Ax)_{k,m} = \int_{\Omega} x(s, t) \psi_k^m(s, t) ds dt. \quad (1.3)$$

The reconstruction problem we study is: given the projection data  $b$ , find a density function  $x$  such that  $Ax = b$ . Since  $A$  has an infinite-dimensional kernel, solutions, if they exist, are not unique. The problem then becomes: find the "best" function  $x_0$  such that  $Ax_0 = b$ . The concept of "best" is ambiguous to be sure, but some criteria have been gaining acceptance. In this paper we choose to study the solution with maximum entropy as defined by Shannon [3] in information theory. For an informal discussion of entropy and information theory, see for example [4]. For a discussion of maximum entropy in image reconstruction, see [5].

In our context, we wish to find the function  $x_0 \in L^1(\Omega)$  such that  $x_0$  attains

$$\sup \left\{ - \int_{\Omega} x(s, t) \ln[x(s, t)] ds dt : Ax = b \right\}. \quad (1.4)$$

This is the maximum entropy solution to  $Ax = b$ . Simply because we would rather minimize a convex function than maximize a concave function, we rewrite this as a convex minimization program via

$$p = \inf \left\{ \int_{\Omega} x(s, t) \ln[x(s, t)] ds dt : Ax = b \right\}. \quad (1.5)$$

This is called the *primal problem*. If we further define the function  $\phi : \mathbb{R} \rightarrow (-\infty, +\infty]$  by

$$\phi(u) = \begin{cases} u \ln u & u > 0 \\ 0 & u = 0 \\ +\infty & u < 0 \end{cases}, \quad (1.6)$$



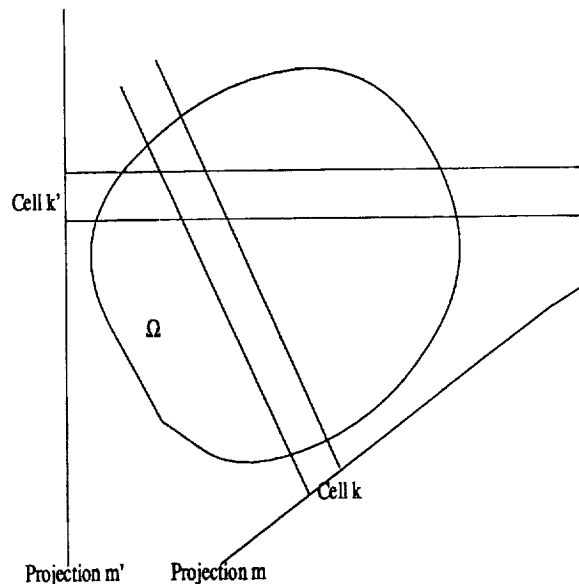


Figure 1: The geometry setup.

then we can rewrite (1.5) as

$$p = \inf \left\{ \int_{\Omega} \phi(x(s, t)) ds dt : Ax = b \right\}, \quad (1.7)$$

which is in a form that has been receiving much attention in the optimization community of late. In particular, see Borwein and Lewis [6]. One of the features of this functional is that it forces feasible functions to be nonnegative, as a density or image function should be. We shall see that it has other properties that are computationally and theoretically attractive, one of the most important being that solutions exist in  $L^1(\Omega)$  under rather mild conditions.

In this paper we shall characterize solutions to (1.5) given some reasonable conditions on the data,  $b$ , and show that the solution in  $L^1(\Omega)$  for the CT case is piecewise constant, but usually not on a rectangular grid. While this result seems to be known in the tomography community, it is rare that one finds a mathematically sound derivation of the solution. The first half of this paper discusses the difficulties in addressing this problem and references the literature to outline a correct proof of our characterization. Note that we do *not* initially impose a discretization of  $L^1(\Omega)$  or  $\Omega$ , only of the data. The discretization we shall use arises as a consequence of the form of the functions  $\psi_k^m$ .

The second half of this paper discusses implementation details. It will turn out that the appropriate grid for optimal resolution (the “optimal grid”) is very large compared to the amount of data,  $N$ , one has. We shall also see that finding the optimal function can be reduced to solving a problem in  $\mathbb{R}^N$ , but the intermediate calculations require use of the (large) optimal grid. We have found that a simple cyclic coordinate maximization

scheme applied to the Fenchel dual of (1.5), while convergent, tends to stall after a few iterations. We describe a multigrid approach to accelerate convergence.

## 2 Maximum Entropy Solutions

### 2.1 The Entropy Functional

Using the entropy functional

$$f : L^1(\Omega) \rightarrow (-\infty, +\infty] : x \mapsto - \int_{\Omega} \phi(x(s, t)) ds dt \quad (2.1)$$

to pick a “best” density function  $x$  has been popularized by Shannon in information theory, but also arises in the context of thermodynamics with Boltzmann. For this reason we call this the Boltzmann-Shannon entropy.

Entropy is, in short, the expected amount of information present in a probability density  $x$ . In our context, one can think of an image (appropriately scaled) as a probability density function, and computing the feasible density with maximum entropy yields the density carrying the most information. The standard reference is Shannon and Weaver [3], but many basic probability books contain some discussion of information and entropy (see [4, 7] for example). References that deal specifically with entropies in image reconstruction are [5, 8].

We remark that the theory and methods developed here apply directly to other objective functionals, in particular, minimum  $L^2$ -norm solutions. In fact, with the minimum  $L^2$ -norm functional, our iterative method is essentially only changed by replacing  $*$  by  $+$  and  $/$  by  $-$ .

### 2.2 Existence of Solutions

In this section we prove that solutions to (1.5) exist. Usually, this point is ignored, but is nevertheless an important issue.

Throughout, let  $X$  be a linear normed space with topology  $\tau$ . We begin with some definitions.

**Definition 2.1** *We say a set  $K \subseteq X$  is  $\tau$ -sequentially compact if every sequence from  $K$  has a  $\tau$ -convergent subsequence.*

**Definition 2.2** *Given a function  $f : X \rightarrow (-\infty, +\infty]$ , for  $\alpha \in \mathbb{R}$ , we define the lower level sets  $L_{\alpha}$  of  $f$  to be*

$$L_{\alpha} = \{x : f(x) \leq \alpha\}. \quad (2.2)$$

The following is a general existence theorem for solutions to constrained optimization problems.

**Theorem 2.3** Let  $f$  be  $\tau$ -lower semicontinuous (lsc) possessing  $\tau$ -sequentially compact lower level sets, and let  $C$  be a  $\tau$ -closed set. Then

$$p = \inf \{f(x) : x \in C\} \quad (2.3)$$

is attained for some  $x_0 \in C$ .

**Proof:** Let  $\{x_n\} \subset C$  be a sequence such that  $f(x_n) \rightarrow p$ . Letting  $\alpha = p + 1$ , eventually all  $x_n \in L_\alpha$ , for  $n \geq N$ . By  $\tau$ -sequential compactness of  $L_\alpha$  there is a subsequence  $x_{n_k}$  that converges to a point  $x_0 \in L_\alpha$ . Since  $C$  is  $\tau$ -closed, then  $x_0 \in C$ ;  $f$  is  $\tau$ -lsc, so

$$p = \lim_{k \rightarrow \infty} f(x_{n_k}) \geq f(x_0) \geq p \quad (2.4)$$

and, thus,  $p = f(x_0)$ . ■

We are interested in the case where the  $\tau$ -topology is the weak topology on  $L^1(\Omega)$ .

Let  $C = \{x : Ax = b\}$  for  $A : X \rightarrow \mathbb{R}^N$  linear and continuous. Then  $C = A^{-1}(\{b\})$  is closed (since  $A$  is continuous) and convex (since  $A$  is linear) and, hence, weakly closed. This is called Mazur's theorem; see [9, Corollary 4 Chapter 2], for example.

We now direct our attention to the functional

$$f(x) = \int_{\Omega} x(s, t) \ln[x(s, t)] ds dt. \quad (2.5)$$

From [10, Theorem 2.2], we see that  $f$  is weakly lsc with weakly compact lower level sets, provided  $\Omega$  is of finite measure. To apply Theorem 2.3, we need weakly sequentially compact lower level sets. The Eberlein-Šmulian theorem [9] states that a subset of a Banach space is weakly compact if and only if it is weakly sequentially compact, and thus we see that the lower level sets of  $f$  are weakly sequentially compact, so we can apply Theorem 2.3 to obtain the following:

**Theorem 2.4** With  $f$  defined as in (2.5) and  $C = \{x : Ax = b\}$  for  $A : L^1(\Omega) \rightarrow \mathbb{R}^N$  linear and continuous, the infimum

$$p = \inf \{f(x) : Ax = b\}, \quad (2.6)$$

if finite, is attained for some  $x_0 \in L^1(\Omega)$  such that  $Ax_0 = b$ .

Note that if  $X = L^\infty$ , then the level sets  $L_\alpha$  of  $f$  are not bounded in the norm topology. Hence  $L_\alpha$  is not compact for any of the norm, weak or weak-\* topologies, which is a consequence of the fact that a continuous function attains its maximum on a compact set, of the theory of dual pairs[11], and of the principle of uniform boundedness, respectively. Thus, although it is tempting to approach the image reconstruction problem in  $L^\infty$ , the initial problem of existence is much more difficult. However we will see that  $L^1$ -optimal solutions are actually  $L^\infty$ -optimal solutions as well. We will also see why one might want to pose the problem in  $L^\infty$ .

## 2.3 Uniqueness of Solutions

In this section we show that solutions to (1.5) are unique.

**Definition 2.5** A function  $g : X \rightarrow (-\infty, +\infty]$  is **convex** if, for all  $x, y \in X$  and all  $\lambda \in (0, 1)$ , we have

$$g(\lambda x + (1 - \lambda)y) \leq \lambda g(x) + (1 - \lambda)g(y). \quad (2.7)$$

Also,  $g$  is **strictly convex** if, whenever  $x \neq y$  and  $g(x), g(y) \in \mathbb{R}$ , this inequality is strict. A set  $C \subseteq X$  is **convex** if, whenever  $x, y \in C$  and  $\lambda \in (0, 1)$ , then

$$\lambda x + (1 - \lambda)y \in C. \quad (2.8)$$

**Lemma 2.6** If  $f(x) = \int \phi(x(u))du$ , then  $f$  is strictly convex if and only if  $\phi$  is strictly convex.

**Proof:** Assume that  $f$  is strictly convex and that there exists a  $u \neq v$  and a  $\lambda \in (0, 1)$  such that

$$\phi(\lambda u + (1 - \lambda)v) \geq \lambda \phi(u) + (1 - \lambda)\phi(v). \quad (2.9)$$

Then let  $x(t) = u$  and  $y(t) = v$ , so that

$$f(\lambda x + (1 - \lambda)y) \geq \lambda f(x) + (1 - \lambda)f(y), \quad (2.10)$$

contradicting the assumption that  $f$  is strictly convex.

Conversely, let  $E = \{t : x(t) \neq y(t)\}$  and assume that  $m(E) > 0$ . Then

$$f(\lambda x + (1 - \lambda)y) = \int_E \phi(\lambda x(t) + (1 - \lambda)y(t))dt + \int_E \phi(x(t))dt \quad (2.11)$$

$$< \int_E \lambda \phi(x(t)) + (1 - \lambda)\phi(y(t))dt + \int_E \phi(x(t))dt \quad (2.12)$$

$$= \lambda f(x) + (1 - \lambda)f(y) \quad (2.13)$$

where the strict inequality is due to the strict convexity of  $\phi$ . ■

It is easy to check that  $\phi(u)$  as defined in (1.6) is strictly convex, thus  $f$  from (2.5) is as well.

**Theorem 2.7** If  $f$  is strictly convex and  $C$  is a convex set, then solutions to

$$p = \inf\{f(x) : x \in C\} \quad (2.14)$$

are unique, provided they exist.

**Proof:** Suppose  $p = f(x) = f(y)$ , where  $x \neq y \in C$ . Since  $C$  is convex, then  $\frac{1}{2}x + \frac{1}{2}y \in C$  and

$$f(x/2 + y/2) < \frac{1}{2}f(x) + \frac{1}{2}f(y) = p \quad (2.15)$$

contradicting the definition of  $p$ . ■

Putting together these results we have the following.

**Theorem 2.8** *If  $\Omega$  is a set of finite measure, then the solution to (1.5) exists and is unique.*

## 2.4 Characterizing the Solution

In this section we characterize solutions to

$$p = \inf \left\{ \int \phi(x(s, t)) ds dt : \int x \psi_k^m = b_k^m, \quad m = 1, \dots, M, \quad k = 1, \dots, K_m \right\}, \quad (2.16)$$

where

$$\phi(u) = \begin{cases} u \ln u - u & u > 0 \\ 0 & u = 0 \\ +\infty & u < 0. \end{cases} \quad (2.17)$$

This is the image reconstruction problem we introduced in Section 1. We have included a linear factor in the objective functional; however, if we assume that the projections cover the image, then this factor does not change the solution; it only simplifies certain formulæ. A typical approach is to attach a Lagrange multiplier  $\lambda$  to the constraints and differentiate the Lagrangian at the optimal  $x_0$  (which we now know exists) to obtain

$$x_0(s, t) = (\phi')^{-1} (A^T \lambda(s, t)) = \exp (A^T \lambda(s, t)), \quad (2.18)$$

where

$$A^T : L^\infty(\Omega) \rightarrow \mathbb{R}^N \text{ via } (A^T \lambda)(s, t) = \sum_{m=1}^M \sum_{k=1}^{K_m} \lambda_k^m \psi_k^m(s, t). \quad (2.19)$$

However, the functional  $f(x) = \int \phi(x(s, t)) ds dt$  is *not* differentiable. Indeed,  $f = +\infty$  on a dense subset of  $L^1(\Omega)$  and is therefore not even continuous. It is for this reason that some people have chosen to work in  $C(\Omega)$  or  $L^\infty(\Omega)$  [12, 13] where  $f$  is differentiable, but the question of existence and attainment is much more difficult there.

A correct approach is to use Fenchel duality with a constraint qualification (CQ). While the classical CQ fails to apply in our example, in [6] a CQ is developed that does

apply to CT. Heuristically, we assume a multiplier  $\bar{\lambda}$  exists so that

$$p = \inf_x \{f(x) + \langle b - Ax, \bar{\lambda} \rangle\} \quad (2.20)$$

$$= \langle b, \bar{\lambda} \rangle + \inf_x \{f(x) - \langle x, A^T \bar{\lambda} \rangle\} \quad (2.21)$$

$$= \langle b, \bar{\lambda} \rangle - \sup_x \{\langle x, A^T \bar{\lambda} \rangle - f(x)\}. \quad (2.22)$$

For a convex function  $g : X \rightarrow (-\infty, +\infty]$  we define  $g^* : X^* \rightarrow (-\infty, +\infty]$  by

$$g^*(y) = \sup_x \{\langle x, y \rangle - g(x)\}. \quad (2.23)$$

This is called the Fenchel conjugate of  $g$  at  $y$ . Using this definition, we see from (2.22)

$$p = \langle b, \bar{\lambda} \rangle - f^*(A^T \bar{\lambda}). \quad (2.24)$$

Now, for any  $\lambda$ ,

$$\inf_x \{f(x) + \langle b - Ax, \lambda \rangle\} = \langle b, \lambda \rangle - f^*(A^T \lambda) \leq \inf \{f(x) : Ax = b\} = p. \quad (2.25)$$

Therefore, if  $\bar{\lambda}$  exists, then

$$p = \max_{\lambda \in \mathbb{R}^N} \{\langle b, \lambda \rangle - f^*(A^T \lambda)\}. \quad (2.26)$$

This is the Fenchel dual of (1.5). In our problem, it can be shown [14] that for  $y \in L^\infty(\Omega)$

$$f^*(y) = \int \phi^*(y(s, t)) ds dt, \quad (2.27)$$

that is, the conjugate of the integral functional  $f$  is given as an integral function of  $\phi^*$ . From [6], if  $\exists \bar{x} \in L^1(\Omega)$  where  $\bar{x} > 0$  a.e.,  $f(\bar{x}) \in \mathbb{R}$  and  $A\bar{x} = b$  (the constraint qualification), then a Lagrange multiplier  $\bar{\lambda}$  does exist. Also, if  $\bar{\lambda}$  solves (2.26), then the optimal  $x_0(s, t)$  for (1.5) is

$$x_0(s, t) = (\phi^*)'(A^T \bar{\lambda}(s, t)). \quad (2.28)$$

In the case  $\phi$  is of the form (2.17), it is easy to show that  $\phi^*(v) = e^v$ , and we get

$$x_0(s, t) = \exp(A^T \bar{\lambda}(s, t)), \quad (2.29)$$

where  $\bar{\lambda}$  solves (2.26). This matches the heuristic derivation, but this is no accident since in fair generality,  $(\phi')^{-1} = (\phi^*)'$ . Thus, solving the image reconstruction problem, (1.5), is equivalent to solving the dual problem (2.26), which is an unconstrained finite-dimensional differentiable concave maximization problem.

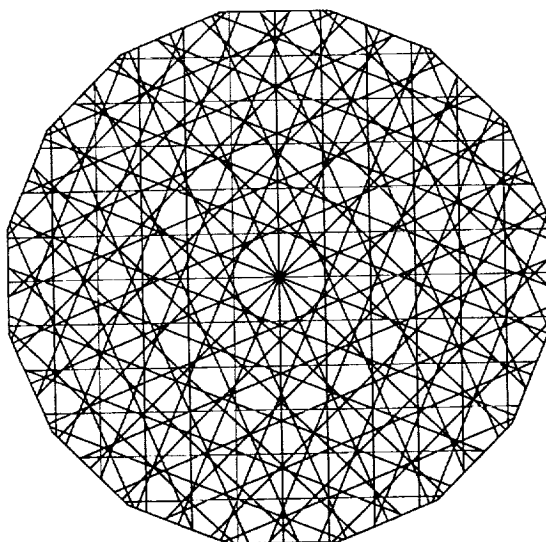


Figure 2: The optimal grid.

## 2.5 The Optimal Grid

Rewriting the solution to the image reconstruction problem (2.29) in a more explicit form, we have

$$x_0(s, t) = \exp \left( \sum_{m=1}^M \sum_{k=1}^{K_m} \bar{\lambda}_k^m \psi_k^m(s, t) \right), \quad (2.30)$$

see (2.19). It can be seen from the concavity of (2.26) that any function  $x$  of the form (2.30) that satisfies  $Ax = b$  must be optimal.

Recall that  $\psi_k^m$  is the characteristic function of the  $k^{\text{th}}$  strip emanating from the  $m^{\text{th}}$  projection. Thus, the solution in (2.30) implies that the optimal function in  $L^1(\Omega)$  is piecewise constant on the grid obtained by intersecting all of the strips. This grid has been observed for physical reasons, [8], but here we have shown that the best (from maximum entropy considerations) function from  $L^1(\Omega)$  is this piecewise constant function. For this reason, we call this discretization of  $\Omega$  the *optimal grid*.

A typical grid is shown in Figure 2. Here we have used 8 projections each divided into 12 cells. The central point of the theory presented above is that the exact solution of the image reconstruction problem is piecewise constant on this grid.

### 3 Implementation

#### 3.1 Relaxation

Now we develop a simple iterative procedure to solve the image problem by solving the dual problem (2.26) for the optimal  $\lambda$ . Recall that to solve the dual problem we seek to maximize

$$G(\lambda) = \langle b, \lambda \rangle - \int \exp(A^T \lambda(s, t)) ds dt. \quad (3.1)$$

This is a simple matter once one observes that  $G$  is concave and differentiable, so maximizing  $G$  is just finding critical points. The derivative with respect to  $\lambda$  is

$$\nabla G(\lambda) = b - A \int \exp(A^T \lambda(s, t)) ds dt. \quad (3.2)$$

This gives  $N$  equations for the  $N$  unknown  $\lambda$ 's.

An obvious iterative scheme is to cycle through the components,  $\lambda_k$ , of  $\nabla G$ , correcting each  $\lambda_k$  in turn so that the  $k^{\text{th}}$  component of  $\nabla G$  is zero, that is, so that  $G$  is maximal with respect to each individual variable. We choose  $\lambda_k^m$  so that

$$b_k^m = (A \exp(A^T \lambda))_{k,m} = \int_{\Omega_k^m} \exp(A^T \lambda(s, t)) ds dt. \quad (3.3)$$

After some simplifications, a single step of this scheme can be written as

$$\exp(\lambda_{k'}^{m'}) \leftarrow \frac{b_{k'}^{m'}}{\int_{\Omega_{k'}^{m'}} \exp(\sum' \lambda_k^m \psi_k^m)} = \frac{b_{k'}^{m'}}{\int_{\Omega_{k'}^{m'}} \prod' (\mu_k^m)^{\psi_k^m}}, \quad (3.4)$$

where  $\sum'$  is the sum over all projections  $m$  and cells  $k$  except  $m'$  and  $k'$ ,  $\prod'$  is similarly defined and  $\mu_k^m = \exp(\lambda_k^m)$ .

Now, because  $A^T \lambda$  is piecewise constant on the optimal grid, the integral of  $\exp(A^T \lambda)$  along any strip is just a sum over each polygon in that strip of the area of the polygon times the product of the  $\mu$ 's that correspond to the particular cell from each projection that makes up the polygon:

$$\int_{\Omega_{k'}^{m'}} \prod' (\mu_k^m)^{\psi_k^m} = \sum_{p \in \Omega_{k'}^{m'}} \left[ \text{area}(p) \prod_{m \neq m'} \mu_{k_m}^m \right], \quad (3.5)$$

where the product is only over the cells  $k_m$  in projection  $m$  for polygon  $p$ .

The point of this discussion is twofold. First, we can see from (3.4) that we never need to exponentiate since we only need the  $\mu$ 's. Second, the areas of the polygons can be precomputed, meaning that these integrals can be calculated exactly; no numerical integration is needed.



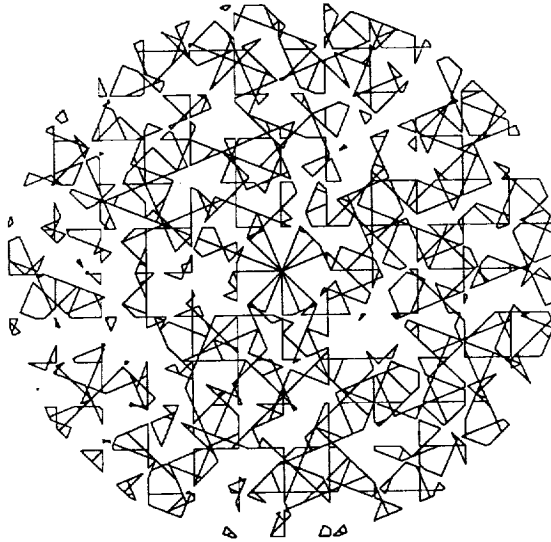


Figure 3: A subset of the optimal grid.

An important feature of this development is that there are no approximations or discretizations being made in the whole program, except for the initial discretization of the data into the vector  $b$ . We have characterized the  $L^1(\Omega)$  solution as a piecewise constant function on the optimal grid, and the necessary integrals can be performed exactly on this grid. For this reason, in our implementation we calculate the areas of each of the polygons in the optimal grid; in fact, to obtain Figure 2, we in fact plotted the polygons themselves, not just intersecting lines. To demonstrate this fact, in Figure 3 we plotted 1000 of the 2096 polygons from Figure 2.

As can be observed in Figure 2, the number of polygons in the optimal grid can be very large compared to the number of projections and cells per projection. This number grows very quickly as a function of these two variables; for example, with 12 projections placed uniformly around the circle and 10 cells per projection, we generate 2904 polygons; with 20 cells per projection we generate 12,561 polygons. While the optimal grid generation is a time and storage intensive procedure, given a fixed geometry we need only run this part of the program once. To reconstruct images using such large data sets, fast methods are essential.

The iterative method described above tends to stall after a few iterations. This effect can be observed in the reported data from [8]. In Figure 4 the top two curves represent the rates of convergence using this scheme on a 3 projection, 4 cell per projection problem. Here we have graphed both the rate associated with the residual  $\|Ax - b\|_\infty$  and the rate of convergence of the entropies computed as  $G(\lambda^{\text{new}})$ . Since we give as initial data a known function of given  $\lambda$ 's, we can compute the true entropy to which the iterates should be converging; this is a useful debugging tool since we can monitor both the residual and

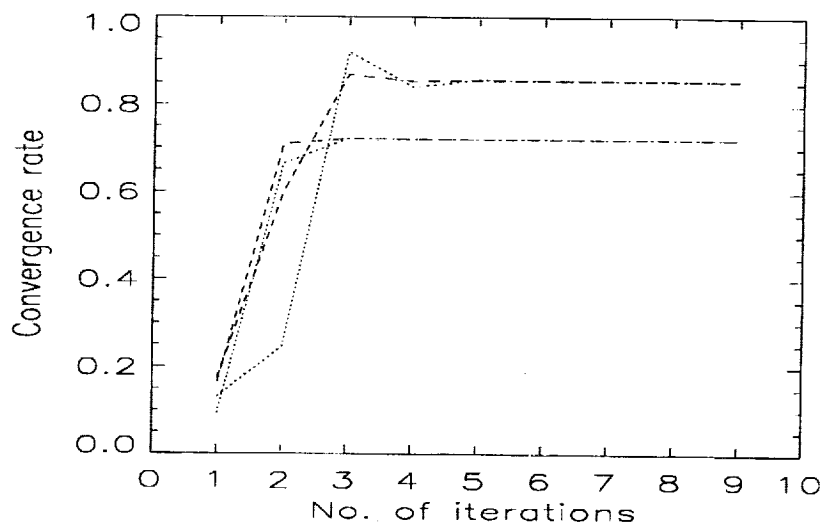


Figure 4: Residual and entropy rates, with and without MG.

the entropy. While convergent, note that the convergence is initially good, but the rate degrades rapidly to steady off at about 0.85.

### 3.2 Multilevel Methods

To improve the convergence rates observed, we have implemented a two-grid multilevel scheme with unigrid [15] corrections. In this section we describe our method and give some preliminary results.

Coarsening is achieved by pairing adjacent cells in the projections and updating the associated  $\mu$ 's with a single correction that makes their average residual zero. On the coarse grid, we iterate this relaxation process until the norm of the vector of average residuals is below a user supplied  $\epsilon$ , typically 0.05. All of the calculations are done in a unigrid fashion on the fine grid. This makes the process more expensive than necessary, but its performance is equivalent to the more efficient V-cycle multigrid scheme and it is much easier to implement and manipulate.

Using the same geometry and data as before, but with relaxation accelerated by coarse grid corrections, we obtain the rates given in the lower two curves in Figure 4. Note that with only a two-grid scheme, we have reduced the convergence rate from about 0.85 to about 0.72.

As a demonstration of the reconstructions we can obtain, we present Figures 5 and 6. Recall that the reconstruction is computed on the optimal grid, but for plotting purposes we essentially use a square grid. While there are several ways of translating from the optimal grid to a square grid, we have chosen simply to evaluate the optimal image

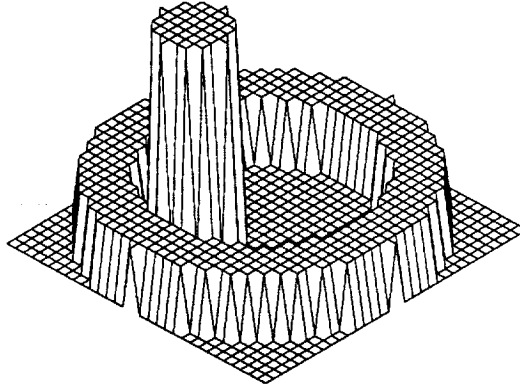


Figure 5: Original image.

function as reconstructed from the optimal  $\lambda$  via (2.30) at the lattice points of the square grid without performing interpolation. Improving this process could be a direction of future investigation.

Figure 5 is the original image. Note that this is *not* a piecewise constant function on the optimal grid, so our reconstructions cannot be exact. In fact, the function of maximum entropy with the data generated by this function is not the original image; as shown previously, it is piecewise constant on the optimal grid, as are our reconstructions.

To obtain Figure 6, we use a simple two-dimensional integrator, based on Simpson's rule, on the original function to make  $b$  using 5 projections each with 8 cells. We then iterated our multilevel scheme to convergence (so that the  $\ell_2$  norm of the average residuals was less than  $10^{-4}$ ) and plotted the result in Figure 6 as discussed above. This process produces a set of  $\lambda$ 's that we then used as our initial data in the routine to obtain a reconstruction of the first reconstruction. This next reconstruction is virtually identical to the first, as the theory predicts.

As a final note, an extra benefit of this scheme is data compression. Given a data collection geometry, when we collect the  $N$  pieces of data, we need only solve the reconstruction problem once to get the  $N$   $\mu$ 's. From these we can reconstruct the image to any level of resolution desired; indeed, we have shown that the piecewise constant function on the optimal grid is the most information one can extract from the data.

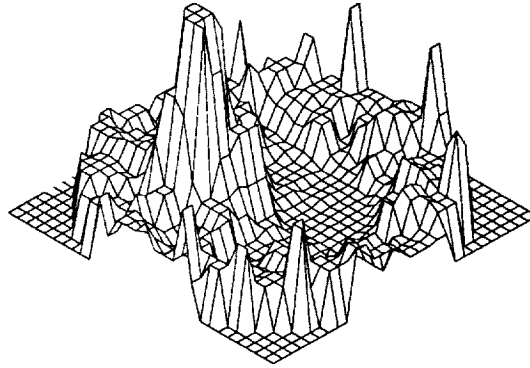


Figure 6: Reconstruction.

## 4 Conclusions

We have seen that the solution to the maximum entropy image reconstruction problem posed in  $L^1(\Omega)$  is a piecewise constant function on the optimal grid. We have also seen that each iterate of the simple iterative scheme for solving the associated dual problem can be computed exactly; no numerical integration or approximations are needed. Finally, we observed that a unigrid scheme to accelerate convergence shows potential, though more testing and analysis is needed.

As a final comment, we note that the mathematics used to derive the optimal grid and the iterative scheme can be applied to other objective functionals  $f$  and other geometries, for example, minimum  $L^2$ -norm, fan beam projections and non-symmetric placement of the projections. These issues and a more complete discussion of the mathematics in optimization techniques for image reconstruction from projections will be covered in a future paper.

## References

- [1] G. T. Herman. *Image Reconstruction From Projections, The Fundamentals of Computerized Tomography*. Academic Press, New York, New York, 1980.
- [2] F. Natterer. *The Mathematics of Computerized Tomography*. John Wiley & Sons, New York, New York, 1986.

- [3] C. E. Shannon and W. Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, Illinois, 1963.
- [4] A. E. Scheerer. *Probability on Discrete Sample Spaces with Applications*. International Textbook Company, Scranton, Pennsylvania, 1969.
- [5] J. Skilling and S. F. Gull. The entropy of an image. *SIAM-AMS Proceedings*, 14:167–189, 1984.
- [6] J. M. Borwein and A. S. Lewis. Partially-finite convex programming, parts I and II. *Mathematical Programming*, pages 15–48, 49–83, 1992.
- [7] S. M. Ross. *A First Course in Probability*. Macmillan, New York, New York, 1976.
- [8] M. L. Reis and N. C. Roberty. Maximum entropy algorithms for image reconstruction from projections. *Inverse Problems*, pages 623–644, 1992.
- [9] J. Diestel. *Sequences and Series in Banach Spaces*. Springer-Verlag, New York, New York, 1984.
- [10] A. Decarreau, D. Hilhorst, C. Lemaréchal, and J. Navaza. Dual methods in entropy maximization. Application to some problems in crystallography. *SIAM J. Optimization*, 2(2):173–197, May 1992.
- [11] A. P. Robertson and W. Robertson. *Topological Vector Spaces*. Cambridge University Press, Cambridge, England, 1973.
- [12] C. Auyeung and R. M. Mersereau. A dual approach to signal restoration. *IEEE International Conference on Acoustics, Speech and Signal Processing, Glasgow*, pages 1326–1329, May 1989.
- [13] J. H. McClellan and S. W. Lang. Duality for multidimensional MEM spectral analysis. *IEE Proceedings Part F*, pages 230–235, April 1983.
- [14] R. T. Rockafellar. Integrals which are convex functionals. *Pacific J. of Math.*, 24(3):525–539, 1968.
- [15] S. F. McCormick. *Multilevel Projection Methods for Partial Differential Equations*. SIAM, Philadelphia, Pennsylvania, 1992.



## FLOW TRANSITION WITH 2-D ROUGHNESS ELEMENTS IN A 3-D CHANNEL

Zhining Liu, Chaoqun Liu, and Stephen F. McCormick  
Computational Mathematics Group, University of Colorado at Denver  
Denver, CO

## SUMMARY

S2-34  
197562  
P. 15

We develop a new numerical approach to study the spatially evolving instability of the streamwise dominant flow in the presence of roughness elements. The difficulty in handling the flow over the boundary surface with general geometry is removed by using a new conservative form of the governing equations and an analytical mapping. The numerical scheme uses second-order backward Euler in time, fourth-order central differences in all three spatial directions, and boundary-fitted staggered grids. A three-dimensional channel with multiple two-dimensional-type roughness elements is employed as the test case. Fourier analysis is used to decompose different Fourier modes of the disturbance. The results show that surface roughness leads to transition at lower Reynolds number than for smooth channels.

## INTRODUCTION

Transition from laminar to turbulent flow is a phenomenon of great importance and practical interest. Major experimental work in aerodynamics has been pursued to study boundary layer transition, and this in turn has led to a critical need for further understanding of this fundamental process. Unfortunately, to date, no reliable methods exist for predicting transition in the presence of surface roughness, either experimentally or numerically. In fact, there are many factors that affect transition, such as solid wall temperature, solid wall curvature, pressure gradients, free-stream disturbance, and surface roughness. There has been some experimental activity dealing with the effect of both 2-D and 3-D roughness elements on transition from laminar to turbulent flow. For example, Klebanoff et al showed that surface roughness induces early transition [1]. Also, others have obtained limited numerical results with 2-D flow [2].

The purpose of the present work is to develop new efficient and easy-to-use methods for numerical simulation of the effect of surface roughness on flow transition. A new conservative form of the governing equations is derived. Because of the high sensitivity of transitional flows, a high order scheme based on our earlier work (cf. [3] - [6]) is developed. For the grid generation scheme, an analytical map is used, so the Jacobian coefficients are computed exactly. Moreover, we develop the governing equation in a form that enables a much simpler numerical process.

We impose single and multiple 2-D-type roughness elements on the lower solid wall to test the effect of surface roughness on flow transition. A Fourier transformation is employed to analyze different modes of the resulting disturbance. The computational results show that the induced mean flow distortion and other high frequency waves make the flow more unstable.

\* This work was supported by NASA under grant number NAS1-19312.

## GOVERNING EQUATIONS

In this study, the three-dimensional, time-dependent, incompressible Navier-Stokes equations, which are nondimensionalized by the channel half-height  $h$  and the centerline velocity  $U_\infty$  are considered as the governing equations for 3-D channel flow (Figure 1):

$$\frac{\partial u}{\partial t} + \frac{\partial uu}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} - \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right) + \frac{\partial P}{\partial x} = 0, \quad (1)$$

$$\frac{\partial v}{\partial t} + \frac{\partial vu}{\partial x} + \frac{\partial vv}{\partial y} + \frac{\partial vw}{\partial z} - \frac{1}{Re} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2} \right) + \frac{\partial P}{\partial y} = 0, \quad (2)$$

$$\frac{\partial w}{\partial t} + \frac{\partial wu}{\partial x} + \frac{\partial wv}{\partial y} + \frac{\partial ww}{\partial z} - \frac{1}{Re} \left( \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) + \frac{\partial P}{\partial z} = 0, \quad (3)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0, \quad (4)$$

where  $u$ ,  $v$ , and  $w$  are velocity components in the  $x$ -,  $y$ -, and  $z$ - directions, respectively,  $P$  is the pressure, and  $Re$  is the Reynolds number based on the centerline velocity  $U_\infty$  of mean flow, the channel half-height  $h$ , and the viscosity parameter  $\nu$ :

$$Re = \frac{U_\infty h}{\nu}. \quad (5)$$

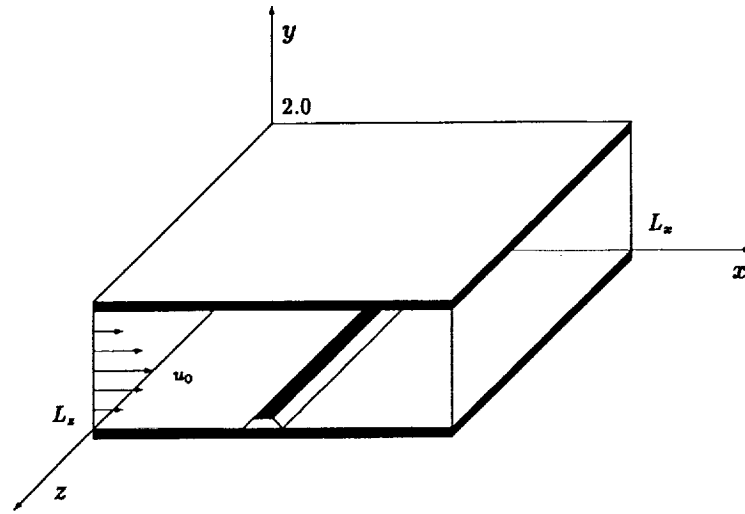


Figure 1. 3-D channel with a single 2-D-type roughness element.

For the current work, we consider a special mapping

$$\begin{aligned} x &= \xi & \xi &= x \\ y &= y(\xi, \eta, \zeta) & \text{or} & \eta = \eta(x, y, z) \\ z &= \zeta & \zeta &= z, \end{aligned}$$

which implies that

$$\begin{aligned} J &= \eta_y, \\ \xi_x &= \zeta_z = 1, \\ \xi_z &= \zeta_x = 0, \end{aligned}$$



and the final forms of the momentum and continuity equations are:

$$\frac{\partial u}{\partial t} + \eta_y \left( \frac{\partial u U}{\partial \xi} + \frac{\partial u V}{\partial \eta} + \frac{\partial u W}{\partial \zeta} \right) + \left( \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \right) P - \frac{1}{Re} \Delta_1 u = 0, \quad (6)$$

$$\frac{\partial v}{\partial t} + \eta_y \left( \frac{\partial v U}{\partial \xi} + \frac{\partial v V}{\partial \eta} + \frac{\partial v W}{\partial \zeta} \right) + \eta_y \frac{\partial P}{\partial \eta} - \frac{1}{Re} \Delta_1 v = 0, \quad (7)$$

$$\frac{\partial w}{\partial t} + \eta_y \left( \frac{\partial w U}{\partial \xi} + \frac{\partial w V}{\partial \eta} + \frac{\partial w W}{\partial \zeta} \right) + \left( \eta_z \frac{\partial}{\partial \eta} + \frac{\partial}{\partial \zeta} \right) P - \frac{1}{Re} \Delta_1 w = 0, \quad (8)$$

$$\frac{\partial U}{\partial \xi} + \frac{\partial V}{\partial \eta} + \frac{\partial W}{\partial \zeta} = 0, \quad (9)$$

for the total flow, or

$$\begin{aligned} \frac{\partial u}{\partial t} + \eta_y \left( \frac{\partial [u(U + U_0) + u_0 U]}{\partial \xi} + \frac{\partial [u(V + V_0) + u_0 V]}{\partial \eta} + \frac{\partial [u(W + W_0) + u_0 W]}{\partial \zeta} \right) \\ + \left( \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta} \right) P - \frac{1}{Re} \Delta_1 u = 0, \end{aligned} \quad (10)$$

$$\begin{aligned} \frac{\partial v}{\partial t} + \eta_y \left( \frac{\partial [v(U + U_0) + v_0 U]}{\partial \xi} + \frac{\partial [v(V + V_0) + v_0 V]}{\partial \eta} + \frac{\partial [v(W + W_0) + v_0 W]}{\partial \zeta} \right) \\ + \eta_y \frac{\partial P}{\partial \eta} - \frac{1}{Re} \Delta_1 v = 0, \end{aligned} \quad (11)$$

$$\begin{aligned} \frac{\partial w}{\partial t} + \eta_y \left( \frac{\partial [w(U + U_0) + w_0 U]}{\partial \xi} + \frac{\partial [w(V + V_0) + w_0 V]}{\partial \eta} + \frac{\partial [w(W + W_0) + w_0 W]}{\partial \zeta} \right) \\ + \left( \eta_z \frac{\partial}{\partial \eta} + \frac{\partial}{\partial \zeta} \right) P - \frac{1}{Re} \Delta_1 w = 0, \end{aligned} \quad (12)$$

$$\frac{\partial U}{\partial \xi} + \frac{\partial V}{\partial \eta} + \frac{\partial W}{\partial \zeta} = 0, \quad (13)$$

for the perturbation flow, where,

$$\Delta_1 = \frac{\partial^2}{\partial \xi^2} + (\eta_x^2 + \eta_y^2 + \eta_z^2) \frac{\partial^2}{\partial \eta^2} + \frac{\partial^2}{\partial \zeta^2} + 2\eta_x \frac{\partial^2}{\partial \xi \partial \eta} + 2\eta_z \frac{\partial^2}{\partial \eta \partial \zeta} + (\eta_{xx} + \eta_{yy} + \eta_{zz}) \frac{\partial}{\partial \eta}. \quad (14)$$

The inverse transformation of the variables under this special mapping becomes

$$u = U \eta_y, \quad (15)$$

$$w = W \eta_y, \quad (16)$$

$$v = V - U \eta_x - W \eta_z. \quad (17)$$

Here we have seven unknowns ( $u, v, w, P, U, V, W$ ), seven equations ((6) – (9), (15) – (17)) for the base flow or total flow, and seven equations ((10)–(13), (15) – (17)) for the perturbation.

Our solution process is outlined as follows:

1. Perform the surface and grid generation process to obtain the required Jacobian coefficients.

2. Solve system (6) – (9) and (15) – (17) to obtain the base flow solution.
3. Solve system (10)–(13) and (15) – (17) to obtain the perturbation solution based on the above base flow.

For the channel flow, though the boundary conditions are quite simple, there are still some difficulties we need to overcome. The so-called buffer domain [7] technique is used here for both the base flow and perturbation. For details, see [6]. The boundary condition for the solid wall is the no-slip boundary condition:

$$u_{wall} = v_{wall} = w_{wall} = U_{wall} = V_{wall} = W_{wall} = 0. \quad (18)$$

No boundary condition is needed for the pressure at the solid wall since we use a staggered grid. For the inflow, Poiseuille flow is imposed at the inlet for the base flow solution, and the eigenfunctions obtained from the linear stability theory with specified Reynolds number are employed at the flow inlet for the perturbation. The final outflow boundary conditions are:

- for the base flow,

$$\begin{aligned} \frac{\partial^2 U}{\partial \xi^2} &= 0 & \text{for } U, \\ \frac{\partial^2 V}{\partial \xi^2} &= 0 & \text{for } V, \\ \frac{\partial^2 W}{\partial \xi^2} &= 0 & \text{for } W, \end{aligned} \quad (19)$$

- for the perturbation flow,

$$\begin{aligned} \frac{\partial U}{\partial \xi} + \frac{\partial V}{\partial \eta} + \frac{\partial W}{\partial \zeta} &= 0 & \text{for } U, \\ \frac{\partial^2 V}{\partial \xi^2} &= 0 & \text{for } V, \\ \frac{\partial^2 W}{\partial \xi^2} &= 0 & \text{for } W, \end{aligned} \quad (20)$$

and the associated  $u$ ,  $v$ ,  $w$  can similarly be obtained by the inverse transformation (15) – (17).

Periodicity is assumed in the spanwise  $\zeta$ -direction.

## NUMERICAL PROCEDURE

### Surface and Grid Generation

Assume that no stagnation points exist in the computational domain. Solitary type roughness elements are overlapped on the lower solid wall of the channel to simulate surface roughness. For the 2-D roughness elements (because the grid is uniform and the domain is periodic in the spanwise  $z$  direction, we need only discuss the 2-D case here), the surface can be expressed as

$$f(x) = \sum_{l=1}^m \kappa_l \text{sech}^2(b_l(x - x_l)), \quad (21)$$

where  $\kappa_l$  is the height of the roughness element,  $b_l$  is a parameter for adjusting the curvature rate of the roughness element, and  $x_l$  is the peak point coordinate.

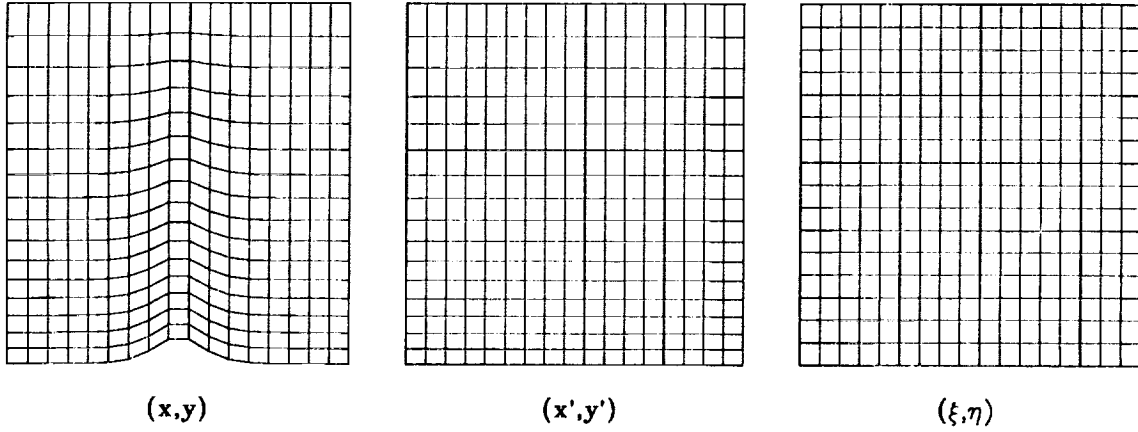


Figure 2. Physical, stretched, and uniform grids.

Our grid generation approach consists of two mappings(see Figure 2):

- from the physical grid that conforms to the rough boundary to the stretched intermediate grid that is uniform in the  $x$  direction but nonuniform in  $y$ ,
- from the stretched grid to the uniform computational grid.

The resulting mapping from the physical to the uniform computational grid is

$$\eta = \frac{y_{max}(\sigma + y_{max})(y - f(x))}{y_{max}(\sigma + y) - f(x)(\sigma + y_{max})}, \quad (22)$$

and its inverse map is

$$y = \frac{\eta\sigma(y_{max} - f(x)) + y_{max}f(x)(\sigma + y_{max} - \eta)}{y_{max}(\sigma + y_{max} - \eta)}, \quad (23)$$

where  $y_{max}$  is the maximum height of the computational domain and  $\sigma$  is the parameter for adjusting the density of grids near the lower solid wall. The required Jacobian coefficients are

$$\eta_x = \frac{y_{max}f_x\sigma(\sigma + y_{max})(y - y_{max})}{[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})]^2}, \quad (24)$$

$$\eta_y = \frac{y_{max}\sigma(\sigma + y_{max})(y_{max} - f(x))}{[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})]^2}, \quad (25)$$

$$\eta_{xx} = \frac{y_{max}\sigma(\sigma + y_{max})(y - y_{max})}{[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})]^3} \cdot \frac{f_{xx}[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})] + 2f_x^2(\sigma + y_{max})}{[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})]^3}, \quad (26)$$

$$\eta_{yy} = \frac{-2y_{max}^2\sigma(\sigma + y_{max})(y_{max} - f(x))}{[y_{max}(\sigma + y) - f(x)(\sigma + y_{max})]^3}, \quad (27)$$

$$\eta_x = \eta_{zz} = 0. \quad (28)$$

Here,  $f_x = \frac{\partial f}{\partial x}$  and  $f_{xx} = \frac{\partial^2 f}{\partial x^2}$ .

## Discretization

In the computational  $(\xi, \eta, \zeta)$  space, the grids are uniform. Suppose  $u, v, w$  and  $U, V, W$  are defined in terms of a staggered grid in the computational space (see Figure 3). Here, the values of  $P$  are associated with its cell centers,  $u$  and  $U$  with centers of the cell surfaces parallel to the  $(\eta, \zeta)$  plane,  $v$  and  $V$  with centers of the cell surfaces parallel to the  $(\xi, \zeta)$  plane, and  $w$  and  $W$  with centers of the cell surfaces parallel to the  $(\xi, \eta)$  plane.

Second-order backward Euler differences are used in the time direction, and fourth-order central differences are used in space. Details of such an approach can be found in [5]. With those assumptions, we can write the discretized governing equations symbolically as follows:

$$A_{EE}u_{EE} + A_E u_E + A_W u_W + A_{WW}u_{WW} + A_{NN}u_{NN} + A_N u_N + A_S u_S + A_{SS}u_{SS} + A_{FF}u_{FF} + A_F u_F + A_B u_B + A_{BB}u_{BB} - A_C u_C + D_{WW}P_{WW} + D_W P_W + D_E P_E - D_C P_C = S_u, \quad (29)$$

$$B_{EE}v_{EE} + B_E v_E + B_W v_W + B_{WW}v_{WW} + B_{NN}v_{NN} + B_N v_N + B_S v_S + B_{SS}v_{SS} + B_{FF}v_{FF} + B_F v_F + B_B v_B + B_{BB}v_{BB} - B_C v_C + E_{SS}P_{SS} + E_S P_S + E_N P_N - E_C P_C = S_v, \quad (30)$$

$$C_{EE}w_{EE} + C_E w_E + C_W w_W + C_{WW}w_{WW} + C_{NN}w_{NN} + C_N w_N + C_S w_S + C_{SS}w_{SS} + C_{FF}w_{FF} + C_F w_F + C_B w_B + C_{BB}w_{BB} - C_C w_C + F_{BB}P_{BB} + F_B P_B + F_F P_F - F_C P_C = S_w, \quad (31)$$

$$D_{U_{EE}}U_{EE} + D_{U_E}U_E + D_{U_W}U_W - D_{U_C}U_C + D_{V_{NN}}V_{NN} + D_{V_N}V_N + D_{V_S}V_S - D_{V_C}V_C + D_{W_{FF}}W_{FF} + D_{W_F}W_F + D_{W_B}W_B - D_{W_C}W_C = S_m, \quad (32)$$

$$u_C = \eta_y^{uc} U_C, \quad (33)$$

$$w_C = \eta_y^{wc} W_C, \quad (34)$$

$$v_C = \eta_x^{vc} U_C + V_C + \eta_z^{vc} W_C. \quad (35)$$

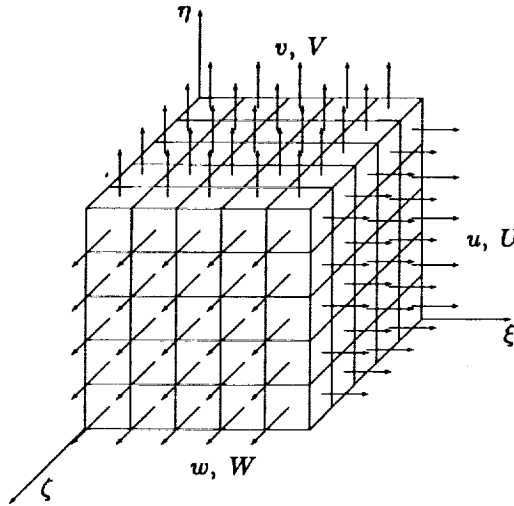


Figure 3. Staggered grid structure in the computational  $(\xi, \eta, \zeta)$  space.

The coefficients and source term for the interior points of the discrete  $\xi$ -momentum equation (29) associated with  $u_C$  are given as follows:

$$A_{EE} = -\frac{1}{12Re\Delta\xi^2} + \frac{\eta_y C}{12\Delta\xi}(U_{EE} + 2U_{0_{EE}}),$$

$$\begin{aligned}
A_E &= \frac{4}{3Re\Delta\xi^2} - \frac{2\eta_y C}{3\Delta\xi}(U_E + 2U_{0_E}), \\
A_W &= \frac{4}{3Re\Delta\xi^2} + \frac{2\eta_y C}{3\Delta\xi}(U_W + 2U_{0_W}), \\
A_{WW} &= \frac{1}{12Re\Delta\xi^2} - \frac{2\eta_y C}{12\Delta\xi}(U_{WW} + 2U_{0_{WW}}), \\
A_{NN} &= -\frac{\alpha_C}{12Re\Delta\eta^2} + \frac{\eta_y C}{12\Delta\eta}(V_{nn} + V_{0_{nn}}) - \frac{\gamma_C}{12Re\Delta\eta}, \\
A_N &= \frac{4\alpha_C}{3Re\Delta\eta^2} - \frac{2\eta_y C}{3\Delta\eta}(V_n + V_{0_n}) + \frac{2\gamma_C}{3Re\Delta\eta}, \\
A_S &= \frac{4\alpha_C}{3Re\Delta\eta^2} + \frac{2\eta_y C}{3\Delta\eta}(V_s + V_{0_s}) - \frac{2\gamma_C}{3Re\Delta\eta}, \\
A_{SS} &= -\frac{\alpha_C}{12Re\Delta\eta^2} - \frac{\eta_y C}{12\Delta\eta}(V_{ss} + V_{0_{ss}}) + \frac{\gamma_C}{12Re\Delta\eta}, \\
A_{FF} &= \frac{1}{12Re\Delta\zeta^2} + \frac{\eta_y C}{12\Delta\zeta}(W_{ff} + W_{0_{ff}}), \\
A_F &= \frac{4}{3Re\Delta\zeta^2} - \frac{2\eta_y C}{3\Delta\zeta}(W_f + W_{0_f}), \\
A_B &= \frac{4}{3Re\Delta\zeta^2} + \frac{2\eta_y C}{3\Delta\zeta}(W_b + W_{0_b}), \\
A_{BB} &= \frac{1}{12Re\Delta\zeta^2} - \frac{\eta_y C}{12\Delta\zeta}(W_{bb} + W_{0_{bb}}), \\
A_C &= \frac{3}{2\Delta t} + \frac{5}{2Re}\left(\frac{1}{\Delta\xi^2} + \frac{\alpha_C}{\Delta\eta^2} + \frac{1}{\Delta\zeta^2}\right), \\
D_E &= -D_{WW} = \frac{1}{24\Delta\xi}, \\
D_W &= D_C = \frac{27}{24\Delta\xi}, \\
S_u &= \frac{-4u_C^n + u_C^{n-1}}{2\Delta t} + \eta_{xC} \frac{-P_{nn} + 8P_n - 8P_s + P_{ss}}{12\Delta\eta} + \\
&\quad \eta_{yC} \left( \frac{-u_{0NN}V_{nn} + 8u_{0N}V_n - 8u_{0S}V_s + u_{0SS}V_{ss}}{12\Delta\eta} + \right. \\
&\quad \left. \frac{-u_{0FF}W_{ff} + 8u_{0F}W_f - 8u_{0B}W_b + u_{0BB}W_{bb}}{12\Delta\zeta} \right) - \\
&\quad \frac{1}{6Re} \left( \eta_{xC} \frac{-u_{\xi nn} + 8u_{\xi n} - 8u_{\xi s} + u_{\xi ss}}{\Delta\eta} + \eta_{zC} \frac{-u_{\zeta nn} + 8u_{\zeta n} - 8u_{\zeta s} + u_{\zeta ss}}{\Delta\eta} \right). \quad (36)
\end{aligned}$$

Here, superscripts  $n$  and  $n - 1$  are used to indicate values at previous time steps, and the superscript  $n + 1$ , which indicates the current time step, is dropped for convenience. Lower case subscripts denote the approximate values of the  $v$  and  $w$  at points where the associated values of  $u$  are located (Figure 4). Other symbols used in the above formulas are as follows:

$$\begin{aligned}
\alpha &= \eta_x^2 + \eta_y^2 + \eta_z^2, & \gamma &= \eta_{xx} + \eta_{yy} + \eta_{zz}, \\
u_\xi &= \frac{\partial u}{\partial \xi}, & u_\zeta &= \frac{\partial u}{\partial \zeta}.
\end{aligned}$$



We use here the same basic line distributive relaxation method developed in our previous work (cf. [5]), with some modifications. Figure 6 shows the distribution of corrections for the group of variables that are located in the  $(\xi, \eta)$  plane.

The process that we use for solving the discrete system (29)–(35) can be described as follows:

- Freezing  $P$ ,  $U$ ,  $V$ ,  $W$ ,  $v$ , and  $w$ , perform point Gauss-Seidel relaxation on (29) over the entire computational domain to obtain a new  $u$ .
- Freezing  $P$ ,  $U$ ,  $V$ ,  $W$ ,  $u$ , and  $w$ , perform point Gauss-Seidel relaxation on (30) over the entire computational domain to obtain a new  $v$ .
- Freezing  $P$ ,  $U$ ,  $V$ ,  $W$ ,  $u$ , and  $v$ , perform point Gauss-Seidel relaxation on (31) over the entire computational domain to obtain a new  $w$ .
- Use transformation (33)–(35) to obtain new  $U$ ,  $V$ ,  $W$ .
- For all  $j = 2, 3, \dots, n_j - 1$  at once: change  $U_{ijk}, U_{i+1jk}, V_{ijk}, W_{ijk}, W_{ij\ k+1}$  to satisfy the continuity equations, then update  $P_{ijk}$  so that the new  $U$ ,  $V$ ,  $W$  and  $P$  as well as the associated transferred  $u$ ,  $v$ ,  $w$  satisfy the three momentum equations.

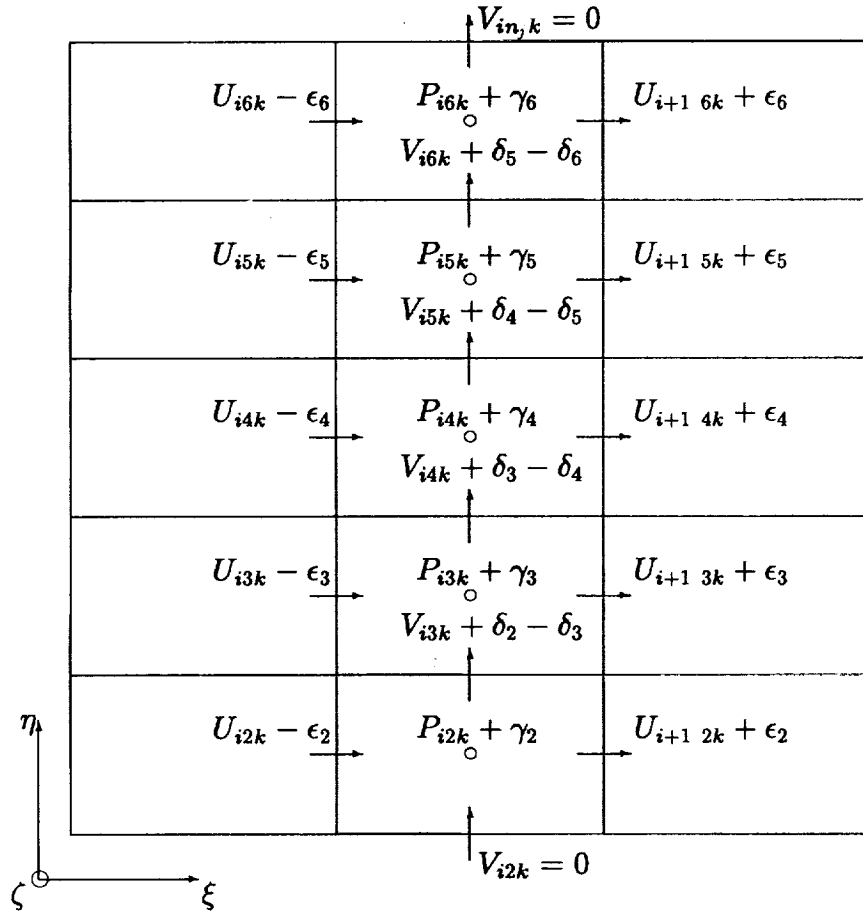


Figure 6. Distribution of corrections in the  $(\xi, \eta)$  plane.

**Remark.** Since all of the  $u$ ,  $v$ ,  $w$  have been previously relaxed, and the  $U$ ,  $V$ ,  $W$  are updated to perform the latter corrections, we assume that equations (29)–(31) hold exactly. Let  $\epsilon$ ,  $\delta$ ,  $\sigma$  and  $\gamma$  represent the corrections for  $U$ ,  $V$ ,  $W$  and  $P$ , respectively. Thus, for cube  $ijk$ , when we distribute the corrections according to Figure 6, the correction equations corresponding to (29)–(31) are

$$(A_E^{ijk} \eta_y^{u_{i+1}jk} + A_C^{ijk} \eta_y^{u_{ijk}}) \epsilon_j - D_C^{ijk} \gamma_j = 0, \quad (39)$$

$$B_N^{ijk} (\delta_j - \delta_{j+1}) - B_C^{ijk} (\delta_{j-1} - \delta_j) - E_C^{ijk} \gamma_j = 0, \quad (40)$$

$$(C_F^{ijk} \eta_y^{w_{ijk+1}} + C_C^{ijk} \eta_y^{w_{ijk}}) \sigma_j - F_C^{ijk} \gamma_j = 0, \quad (41)$$

$$(DU_E^{ijk} + DU_C^{ijk}) \epsilon_j + (DW_F^{ijk} + DW_C^{ijk}) \sigma_j + DV_N^{ijk} (\delta_j - \delta_{j+1}) - DV_C^{ijk} (\delta_{j-1} - \delta_j) = S_m^{ijk}, \quad (42)$$

$j = 2, 3, \dots, n_j - 1.$

This system has  $4(n_j - 2)$  equations and  $4(n_j - 2)$  variables. Unfortunately, coupling between the correction variables makes the problem somewhat complicated. To develop a simpler approximate system, define

$$\omega_{xj} = \frac{\epsilon_j}{\delta_j},$$

$$\omega_{zj} = \frac{\sigma_j}{\delta_j},$$

with fixed  $i$  and  $k$ . Then

$$\omega_{xj} = \frac{D_C^{ijk} (B_N^{ijk} + B_C^{ijk}) \delta_j - B_N^{ijk} \delta_{j+1} - B_C^{ijk} \delta_{j-1}}{E_C^{ijk} (A_E^{ijk} \eta_y^{u_{i+1}jk} + A_C^{ijk} \eta_y^{u_{ijk}}) \delta_j}, \quad (43)$$

$$\omega_{zj} = \frac{F_C^{ijk} (B_N^{ijk} + B_C^{ijk}) \delta_j - B_N^{ijk} \delta_{j+1} - B_C^{ijk} \delta_{j-1}}{E_C^{ijk} (C_F^{ijk} \eta_y^{w_{ijk+1}} + C_C^{ijk} \eta_y^{w_{ijk}}) \delta_j}. \quad (44)$$

From (36), we see that  $A_C^{ijk} \sim \frac{3}{2\Delta t}$  for high  $Re$  and small  $\Delta t$ , which is much larger than  $A_E^{ijk}$ . Similarly,  $B_C^{ijk} \sim \frac{3}{2\Delta t}$  and  $C_C^{ijk} \sim \frac{3}{2\Delta t}$ . These yield

$$\omega_{xj} \sim \frac{D_C^j}{E_C^j \eta_y^{u_{ijk}}} = \frac{\Delta \eta}{\Delta \xi \eta_y^{u_{ijk}} \eta_y^{v_{ijk}}}, \quad (45)$$

$$\omega_{zj} \sim \frac{F_C^j}{E_C^j \eta_y^{w_{ijk}}} = \frac{\Delta \eta}{\Delta \zeta \eta_y^{w_{ijk}} \eta_y^{v_{ijk}}}. \quad (46)$$

With the above approximations,  $\omega_{xj}$  and  $\omega_{zj}$  can be treated as known parameters, so equation (44) can be written in terms of the unknowns  $\delta_j$  only:

$$[(DU_E^{ijk} + DU_C^{ijk}) \omega_{xj} + (DV_N^{ijk} + DV_C^{ijk}) + (DW_F^{ijk} + DW_C^{ijk}) \omega_{zj}] \delta_j - DV_C^{ijk} \delta_{j-1} - DV_N^{ijk} \delta_{j+1} = S_m^{ijk}. \quad (47)$$

Let

$$a_j = (DU_E^{ijk} + DU_C^{ijk}) \omega_{xj} + (DV_N^{ijk} + DV_C^{ijk}) + (DW_F^{ijk} + DW_C^{ijk}) \omega_{zj}, \quad (48)$$

$$b_j = -DV_N^{ijk}, \quad (49)$$

$$c_j = -DV_C^{ijk}, \quad (50)$$

$$j = 2, 3, \dots, n_j - 1.$$



Then we obtain the tridiagonal system

$$\begin{bmatrix} a_2 & b_2 & & & \\ c_3 & a_3 & b_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \\ & & & c_{n_j-2} & a_{n_j-2} & b_{n_j-2} \\ & & & & c_{n_j-1} & a_{n_j-1} \end{bmatrix} \begin{bmatrix} \delta_2 \\ \delta_3 \\ \vdots \\ \vdots \\ \delta_{n_j-2} \\ \delta_{n_j-1} \end{bmatrix} = \begin{bmatrix} S_m^{i2k} \\ S_m^{i2k} \\ \vdots \\ \vdots \\ S_m^{in_j-2k} \\ S_m^{in_j-1k} \end{bmatrix}. \quad (51)$$

Thus,  $\delta_j$ ,  $j = 2, 3, \dots, n_j - 1$  can be determined very efficiently. The other velocity corrections are given by

$$\begin{aligned} \epsilon_j &= \omega_{xj} \delta_j, & \sigma_j &= \omega_{zj} \delta_j, \\ j &= 2, 3, \dots, n_j - 1. \end{aligned}$$

The  $u$ ,  $v$ , and  $w$  are then updated on all cells in the  $i, k$   $y$ -line as follows:

$$\begin{aligned} U^{i+1jk} &\leftarrow U^{i+1jk} + \epsilon_j, \\ U^{ijk} &\leftarrow U^{ijk} - \epsilon_j, \\ W^{ijk+1} &\leftarrow W^{ijk+1} + \sigma_j, \\ W^{ijk} &\leftarrow W^{ijk} - \sigma_j, \\ j &= 2, 3, \dots, n_j - 1, \end{aligned} \quad (52)$$

$$\begin{aligned} V^{ijk} &\leftarrow V^{ijk} + \delta_{j-1} - \delta_j, \\ j &= 3, 4, \dots, n_j - 1. \end{aligned} \quad (53)$$

Finally, the pressure corrections  $\gamma_j$  are determined as follows:

$$\gamma_2 = \frac{\frac{\Delta\eta}{\Delta\xi} A_C^{i2k} + B_C^{i3k} + \frac{\Delta\eta}{\Delta\zeta} C_C^{i2k}}{(\frac{1}{\Delta\xi} + \frac{1}{\Delta\eta} + \frac{1}{\Delta\zeta}) \eta_y^{v_{i3k}}} \delta_j, \quad (54)$$

$$\begin{aligned} \gamma_j &= \frac{\frac{\Delta\eta}{\Delta\xi} A_C^{ijk} + B_C^{ijk} + \frac{\Delta\eta}{\Delta\zeta} C_C^{ijk}}{(\frac{1}{\Delta\xi} + \frac{1}{\Delta\eta} + \frac{1}{\Delta\zeta}) \eta_y^{v_{ijk}}} \delta_j, \\ j &= 3, \dots, n_j - 1. \end{aligned} \quad (55)$$

$P$  is then updated via

$$\begin{aligned} P_{ijk} &\leftarrow P_{ijk} + \gamma_j, \\ j &= 2, 3, \dots, n_j - 1. \end{aligned} \quad (56)$$

## COMPUTATIONAL RESULTS

We first tested the code by applying it to single and double roughness elements using a moderate sized grid. A  $170 \times 50 \times 4$  grid, including a five T-S wavelength physical domain and a one wavelength buffer domain, was used. Considering that  $Re = 5000$  corresponds to a decreasing mode according to the linear stability theory, we use this Reynolds number in our code to investigate the effect of roughness elements. Fourier analysis is used to decompose different Fourier modes of the disturbance. For details about the Fourier transformation approach, see [8].

Let  $\kappa_l \equiv 0.15$  and  $b_l \equiv \sqrt{2}$ . For the single roughness case, the peak point is at  $i = 30$ ; for the double roughness case, the peak points are at  $i = 42$  and  $i = 70$ . The stretch parameter  $\sigma$  is set to 4, and the amplitude of the disturbance  $\epsilon$  is set to  $0.0025\sqrt{2}$ . Figure 7 displays contours of the perturbation streamfunctions, showing that the roughness elements make the disturbance increase for a certain distance downstream. The results of Fourier transformation given in Figure 8 show that the mean flow distortion and first and second harmonic waves are amplified over this distance.

To test the effect of multiple elements, a  $402 \times 66 \times 4$  grid (including a nine wavelength physical domain and a one wavelength buffer domain) is used. Here we set  $\kappa_l \equiv 0.12$ ,  $b_l \equiv 2.0$ , and  $\sigma = 4.5$ . The first roughness is at  $i = 82$ , which is two wavelengths from the inflow boundary. We placed seven roughness elements in the computational domain, starting from  $i = 82$  and spaced 40 grid points apart. Figures 9 and 10 depict the contour plots of streamfunctions and vorticities, showing very clearly that the disturbance is amplified after it passes each element.

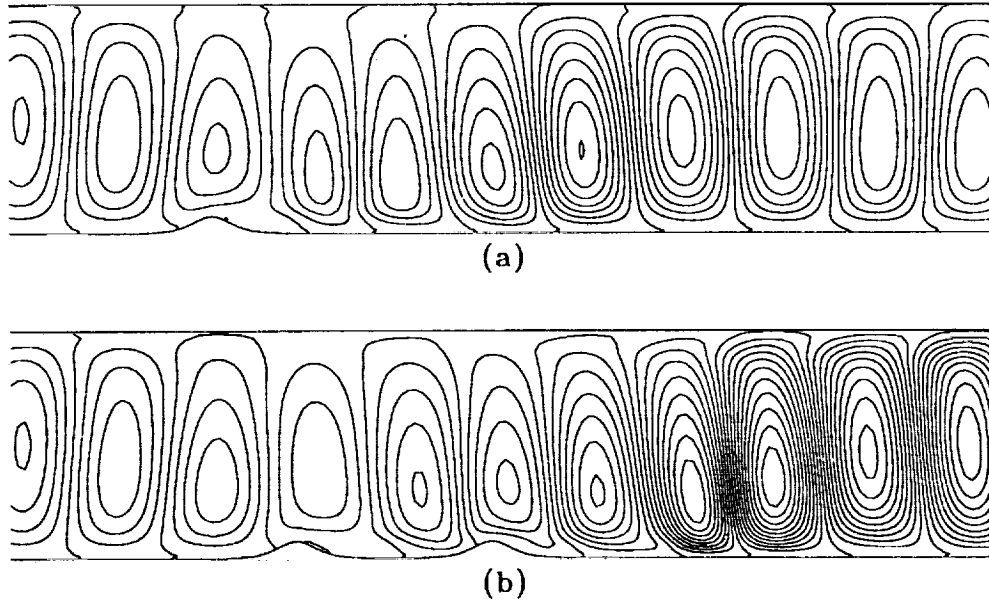


Figure 7. Contours of perturbation streamfunctions with  $Re = 5000$ ,  $\kappa_l = 1.5$ ,  $\epsilon = 0.0025\sqrt{2}$  and  $170 \times 50 \times 4$  grid. Flow direction is from left to right.

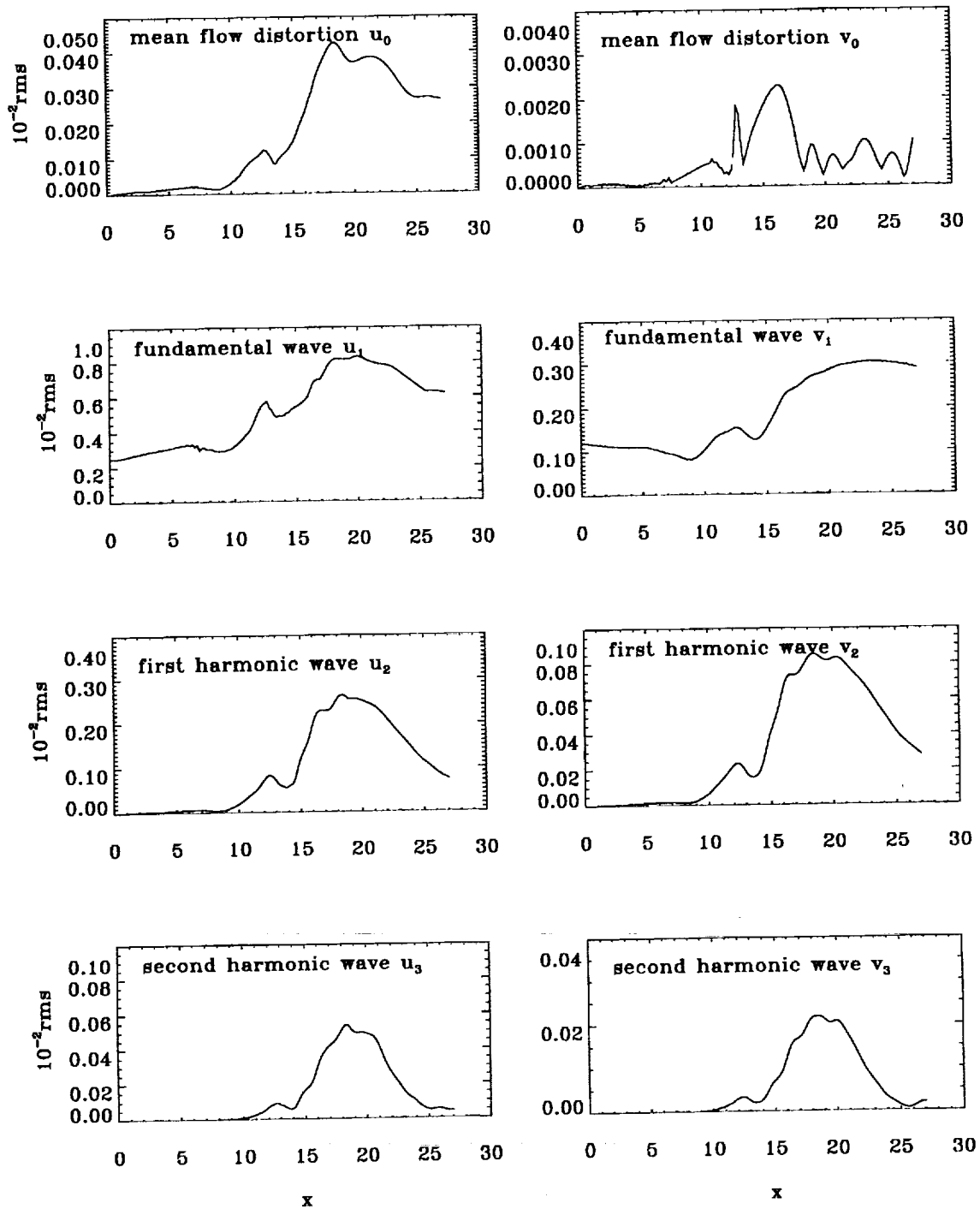


Figure 8. Maximum amplitudes of fundamental wave  $u_1$ ,  $v_1$ , mean-flow distortion  $u_0$ ,  $v_0$ , first harmonic wave  $u_2$ ,  $v_2$ , and second harmonic wave  $u_3$ ,  $v_3$  for  $Re = 5000$ ,  $\kappa_t = 0.15$ , and  $\epsilon = 0.0025\sqrt{2}$  with two roughness elements (grid:  $170 \times 50 \times 4$ ).

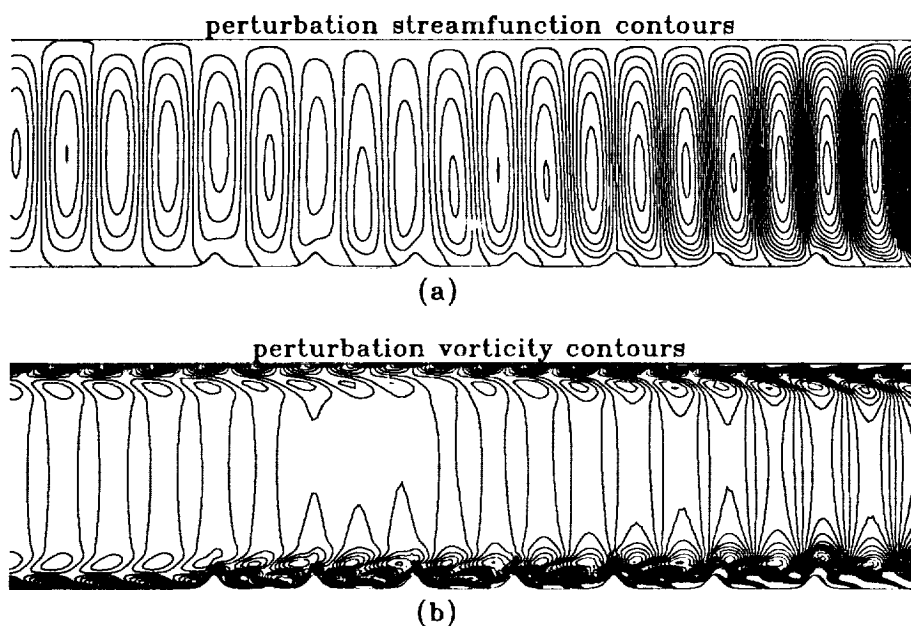


Figure 9. Contour plots of perturbation streamfunctions and vorticities for  $Re = 5000$ ,  $\kappa_l = 0.12$ , and  $\epsilon = 0.0025\sqrt{2}$  with seven roughness elements (grid:  $402 \times 66 \times 4$ , flow direction is from left to right).

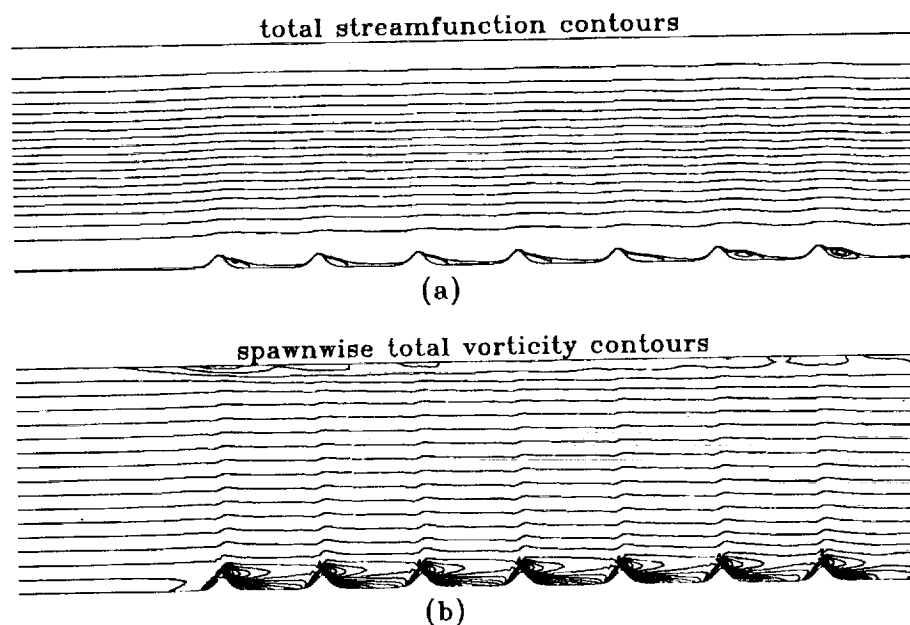


Figure 10. Contour plots of total streamfunctions and vorticities for  $Re = 5000$ ,  $\kappa_l = 0.12$ , and  $\epsilon = 0.0025\sqrt{2}$  with seven roughness elements (grid:  $402 \times 66 \times 4$ , flow direction is from left to right).

## CONCLUDING REMARKS

As expected on physical grounds, we find that the spatial growth rates of the disturbance increase when surface roughness is present. Though our work is limited to roughness without stagnation points in the computational domain, such a scope includes a rather large variety of real roughness elements. Moreover, the code is very efficient, requiring about 2.68 seconds per time step for the  $402 \times 66 \times 4$  grid case (equivalent to about  $26 \mu s$  per time step per grid point).

## ACKNOWLEDGEMENT

The authors thank Dr. R. Joslin at NASA Langley Research Center for providing inflow data and general guidance, and Dr. Helen Reed at Arizona State University for sharing her experience.

## References

- [1] P.S. Klebanoff and K.D. Tidstrom, Mechanism by which a two-dimensional roughness element induces boundary-layer transition, *Phys. Fluids* **15** No.7, 1173–1188 (1972).
- [2] G. Danabasoglu, S. Biringen, and C.L. Streett, Spatial simulation of boundary layer instability: effect of surface roughness, *AIAA 93-0075* (1993).
- [3] C.Liu, Z.Liu, and S.McCormick, Multigrid methods for flow transition in planar channel, *Comput. Phys. Commun.* **65**, 188-200 (1991).
- [4] Z.Liu and C.Liu, Fourth order finite difference and multigrid methods for modeling instabilities in flat plate boundary layers, *J. Comput. Wind. Eng.* **52** 412–417 (1992).
- [5] C. Liu and Z. Liu, Multigrid methods for modeling the secondary instabilities in a 3-D channel – temporal and spatial approaches, *Science Report to NASA Langley Research Center*, July 16, 1992.
- [6] C.Liu, Z.Liu, and S. McCormick, High order finite difference and multigrid methods for spatially-evolving instability in a planar channel, *J. Comput. Phys.* (in press).
- [7] C.L. Streett and M.G. Macaraeg, Spectral multi-domain for large-scale fluid dynamics simulations, *Appl. Num. Math.* **6**, 123–139 (1989).
- [8] Z. Liu, C. Liu, and S. McCormick, A comparison with spectral methods and parabolic stability equation theories for transition over a flat plate, *Science Report to NASA Langley Research Center*, October 10, 1992.



MULTILEVEL METHODS FOR TRANSPORT EQUATIONS  
IN DIFFUSIVE REGIMES

Thomas A. Manteuffel and Klaus Ressel  
Computational Mathematics Group  
University of Colorado at Denver  
Denver, Co 80217-3364

## SUMMARY

We consider the numerical solution of the single-group, steady state, isotropic transport equation. An analysis by means of the moment equations shows that a discrete ordinates  $S_N$  discretization in direction (angle) with a least squares finite element discretization in space does not behave properly in the diffusion limit. A scaling of the  $S_N$  equations is introduced so that the least squares discretization has the correct diffusion limit. For the resulting discrete system a full multigrid algorithm was developed.

## 1. INTRODUCTION

The single-group, steady state, isotropic form of the Boltzmann transport equation for one dimensional slab geometry is given by ( Lewis and Miller [6] )

$$\left\{ \begin{array}{l} \mu \frac{\partial \psi(x, \mu)}{\partial x} + \sigma_t \psi(x, \mu) - \sigma_s \frac{1}{2} \int_{-1}^1 \psi(x, \mu') d\mu' = q(x, \mu) \\ \psi(a, \mu) = g_1(\mu) \quad \text{for } \mu > 0 \\ \psi(b, \mu) = g_2(\mu) \quad \text{for } \mu < 0 \end{array} \right\} \quad (1.1)$$

where  $x \in [a, b]$  and  $\mu \in [-1, 1]$ . When  $\sigma_t \rightarrow \infty$  and  $\frac{\sigma_s}{\sigma_t} \rightarrow 1$ , which is the so called *diffusion limit*, this equation becomes singular. The limit operator  $(I - P)$ , where  $P$  denotes the operator  $P\psi(x, \mu) \equiv \frac{1}{2} \int_{-1}^1 \psi(x, \mu') d\mu'$ , has in its nullspace all functions that are independent of angle  $\mu$ .

Moreover, in this limit transport theory transitions into diffusion theory in the following way. Let  $\varepsilon$  be a small parameter. Substituting  $\sigma_t$  by  $\frac{1}{\varepsilon}$ ,  $\sigma_s$  by  $(\frac{1}{\varepsilon} - \varepsilon \sigma_a)$ , where  $\sigma_a$  is  $O(1)$ , and scaling the right hand side by  $\varepsilon$ , equation (1.1) becomes

$$\left[ \mu \frac{\partial}{\partial x} + \frac{1}{\varepsilon} I - \left( \frac{1}{\varepsilon} - \varepsilon \sigma_a \right) P \right] \psi(x, \mu) = \varepsilon q(x). \quad (1.2)$$

In addition, it is assumed that the external source  $q$  is independent of  $\mu$ . As a consequence of this parameterization the diffusion limit is now equivalent to the limit  $\varepsilon \rightarrow 0$ . By expanding the solution

of (1.2) as

$$\psi(x, \mu) = \psi^0(x, \mu) + \sum_{j=1}^{\infty} \epsilon^j \psi^j(x, \mu)$$

it can be shown (Larsen [2] [3], Larsen, Morel and Miller [4]) that some mean-free-paths away from the boundary the zeroth order term,  $\psi^0(x, \mu)$ , is independent of  $\mu$  and is a solution of the following diffusion equation

$$-\frac{1}{3} \frac{d^2 \phi(x)}{dx^2} + \sigma_a \phi(x) = q(x). \quad (1.3)$$

Thus, in the diffusion limit the solution of the transport equation will converge to the solution of a diffusion equation.

For the numerical solution of (1.1) it is important to find a discretization that has the same property; i.e., for diffusive regimes ( $\sigma_t$  large,  $\frac{\sigma_t}{\sigma_a} \approx 1$ ) the difference scheme for the transport equation must approximate a diffusion operator.

In the last two decades a large amount of work was dedicated to developing special discretizations for the transport equation that have the correct behavior in the diffusion limit. Among them are the Diamond Difference scheme [6], the Linear Discontinuous scheme [1], and the Modified Linear Discontinuous scheme [5]. In the one-dimensional case their implementation is straightforward, but their extension to higher dimensions is difficult.

In this paper we try to develop a general framework for finding discretizations for the transport equation that have the correct behavior in the diffusion limit. In Section 2 we describe the discrete ordinates  $S_N$  discretization in angle and a least squares finite element discretization in space and discuss why this simple approach does not behave properly in the diffusion limit. A scaling technique for the transport equation is introduced in Section 3 that yields a least squares discretization with the proper diffusion limit. In Section 4 we present numerical results based on a full multigrid solver for the resulting discrete system. In Section 5 we draw conclusions and suggest further applications of the scaling technique.

## 2. DISCRETIZATION

For the discretization in angle we use the standard discrete ordinates  $S_N$  method. In the case of one-dimensional Slab geometry, this is a Galerkin discretization with normalized Legendre polynomials as basis. That means we are looking for a flux solution that has an expansion in the first  $N$  normalized Legendre polynomials,

$$\psi(x, \mu) = \sum_{l=0}^{N-1} \phi_l(x) P_l(\mu). \quad (2.1)$$

Since the normalized Legendre polynomials form an orthonormal basis for  $L_2([-1, 1])$ , the moment coefficients  $\phi_l$  are given by the following integral, which can be



written exactly as a sum by using a Gauss quadrature rule. We have

$$\begin{aligned}\phi_l(x) &= \frac{1}{2} \int_{-1}^1 \psi(x, \mu) P_l(\mu) d\mu \\ &= \sum_{j=1}^N \omega_j \psi(x, \mu_j) P_l(\mu_j).\end{aligned}\tag{2.2}$$

Here  $\mu_j$  denotes the Gauss quadrature points and  $\omega_j$  denotes the Gauss quadrature weights.

By introducing the vector notation

$$\underline{\Psi} \equiv (\psi(x, \mu_1), \dots, \psi(x, \mu_N))^T; \quad \underline{\Phi} \equiv (\phi_0(x), \dots, \phi_{N-1}(x))^T$$

and defining matrices  $T$  and  $\Omega$  as

$$[T]_{i,j} \equiv P_{i-1}(\mu_j); \quad \Omega \equiv \text{diag}(\omega_1, \dots, \omega_N)$$

the relationship (2.1) and (2.2) between the *flux*  $\underline{\Psi}$  and the *moments*  $\underline{\Phi}$  can be written as

$$\underline{\Phi} = T\Omega\underline{\Psi}\tag{2.3}$$

$$\underline{\Psi} = T^T\underline{\Phi}.\tag{2.4}$$

As a result of the Galerkin discretization of (1.1) with the ansatz (2.1) we obtain the  $S_N$  equations

$$L\underline{\Psi} \equiv \varepsilon \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_N \end{pmatrix} \frac{\partial \underline{\Psi}}{\partial x} + I\underline{\Psi} - (1 - \varepsilon^2 \sigma_a) R\underline{\Psi} = \varepsilon^2 \underline{q},\tag{2.5}$$

where

$$R \equiv (1, \dots, 1)^T (\omega_1, \dots, \omega_N).$$

When we insert (2.4) into (2.5) and multiply by  $T\Omega$  from the left, we get the moment equations

$$M\underline{\Phi} \equiv \left[ \begin{array}{c|ccc} \varepsilon^2 \sigma_a & \varepsilon b_0 \frac{\partial}{\partial x} & 0 & 0 & \dots \\ \varepsilon b_0 \frac{\partial}{\partial x} & 1 & \varepsilon b_1 \frac{\partial}{\partial x} & 0 & \\ 0 & \varepsilon b_1 \frac{\partial}{\partial x} & 1 & \varepsilon b_2 \frac{\partial}{\partial x} & \\ \vdots & & \ddots & \ddots & \ddots \end{array} \right] \underline{\Phi} = \varepsilon^2 \underline{\hat{q}}\tag{2.6}$$

with

$$b_j \equiv \frac{j+1}{\sqrt{4(j+1)^2 - 1}}.$$

Normally, the computations are done in the flux representation (2.5) since in this representation the boundary conditions are equal to simple Dirichlet boundary conditions. However, as we will see later, the moment equations are very useful for theoretical insight. In the following the flux operator is denoted by  $L$  and the moment operator by  $M$ .

For the spatial discretization of the  $S_N$  equations we use a least squares finite-element method based on the functional

$$F(\underline{\Psi}) \equiv \int_a^b \langle \Omega (L\underline{\Psi} - \underline{q}), L\underline{\Psi} - \underline{q} \rangle_{\mathbb{R}^N} dx, \quad (2.7)$$

and piecewise linear continuous elements  $\eta_k$  as basis functions for each component of  $\underline{\Psi}$ . In the following  $S^h$  denotes the finite dimensional space  $S^h \equiv \text{span} \{\eta_k\}$  and  $(S^h)^N$  denotes the space of  $N$ -tuples whose elements are in  $S^h$ .

The advantage of this approach is that a least squares discretization converts the  $S_N$  equations, which are a coupled system of first order equations, into a self-adjoint variational formulation. Based on this variational formulation Multi-Level Projection Methods [7] can be applied in order to guide the development of a multigrid solver for the resulting discrete system.

Unfortunately, this discretization does not behave correctly in the diffusion limit. In order to explain this fact, we use the moment equations, since  $\phi_0 = \psi$  in the diffusion limit. Further, it is easy to see by using the relationship (2.3), (2.4) and the identity  $T^T T \Omega = I$  that

$$\begin{aligned} \min_{\underline{\Psi} \in (S^h)^N} \int_a^b \langle \Omega (L\underline{\Psi} - \underline{q}), L\underline{\Psi} - \underline{q} \rangle_{\mathbb{R}^N} dx &\iff \\ \min_{\underline{\Phi} \in (S^h)^N} \int_a^b \langle M\underline{\Phi} - \hat{\underline{q}}, M\underline{\Phi} - \hat{\underline{q}} \rangle_{\mathbb{R}^N} dx, \end{aligned}$$

which justifies why it is also possible to look at the least squares discretization of the moment equations.

In the  $S_2$  case with  $\sigma_a = 0$ , for example, a least squares discretization of the moment equations results in the following discrete system

$$\left( \begin{array}{c|c} \frac{\varepsilon^2}{3}(\eta', \eta') & \frac{\varepsilon}{\sqrt{3}}(\eta', \eta) \\ \hline \frac{\varepsilon}{\sqrt{3}}(\eta, \eta') & \frac{\varepsilon^2}{3}(\eta', \eta') + (\eta, \eta) \end{array} \right) \begin{pmatrix} \phi_0^h \\ \phi_1^h \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{\varepsilon^3}{\sqrt{3}}(\eta', q_0) \end{pmatrix},$$

where, for example,  $(\eta, \eta)$  is a mass matrix and  $(\eta', \eta)$  a stiffness matrix with elements

$$[(\eta, \eta)]_{k,l} \equiv \int_a^b \eta_k(x) \eta_l(x) dx, \quad [(\eta', \eta)]_{k,l} \equiv \int_a^b \frac{d\eta_k(x)}{dx} \eta_l(x) dx$$

and

$$(\eta', q_0) \equiv \left( \dots, \int_a^b \frac{d\eta_k(x)}{dx} q_0(x) dx, \dots \right)^T.$$

Forming the Schur complement we get the following equation for  $\phi_0^h$

$$\begin{aligned} &\left\{ \frac{\varepsilon^2}{3}(\eta', \eta') - \frac{\varepsilon^2}{3}(\eta', \eta) \left[ (\eta, \eta) + \frac{\varepsilon^2}{3}(\eta', \eta') \right]^{-1} (\eta, \eta') \right\} \phi_0^h \\ &= -\frac{\varepsilon^4}{3}(\eta', \eta) \left[ (\eta, \eta) + \frac{\varepsilon^2}{3}(\eta', \eta') \right]^{-1} (\eta', q_0). \end{aligned} \quad (2.8)$$

For sufficient small  $\varepsilon$  we have

$$\left[ (\eta, \eta) + \frac{\varepsilon^2}{3} (\eta', \eta') \right]^{-1} = (\eta, \eta)^{-1} - \frac{\varepsilon^2}{3} (\eta, \eta)^{-1} (\eta', \eta') (\eta, \eta)^{-1} + O(\varepsilon^4). \quad (2.9)$$

Plugging (2.9) into (2.8) and dividing by  $\varepsilon^2$  leads to

$$\begin{aligned} & \left\{ \frac{1}{3} \underbrace{[(\eta', \eta') - (\eta', \eta)(\eta, \eta)^{-1}(\eta, \eta')]}_{(*)} \right. \\ & \quad \left. + \frac{\varepsilon^2}{9} (\eta', \eta)(\eta, \eta)^{-1}(\eta', \eta')(\eta, \eta)^{-1}(\eta, \eta') + O(\varepsilon^4) \right\} \phi_0^h \\ & = -\frac{\varepsilon^2}{3} (\eta', \eta)(\eta, \eta)^{-1}(\eta', q_0) + O(\varepsilon^4). \end{aligned} \quad (2.10)$$

In the limit  $\varepsilon \rightarrow 0$ , the solution approaches a valid discretization for the diffusion equation (1.3) only if the term (\*) vanishes identically. For piecewise linear, continuous basis elements this term does not cancel out and becomes the leading term in the equation. Consequently, in the diffusion limit (2.10) is an approximation for  $\phi_0'' = 0$ , which results in a linear solution, connecting the boundary conditions. In general, the term (\*) does not have the proper behavior unless the mass matrix,  $(\eta, \eta)$ , is lumped, that is, replaced by a diagonal matrix.

### 3. SCALING

A closer look at the moment equations (2.6) shows that this system is unbalanced. There are  $O(\varepsilon^2)$ ,  $O(\varepsilon)$  entries as well as  $O(1)$  entries. The idea is to scale this system before the discretization.

First, let us consider the case  $\sigma_a \neq 0$ . In our inner product the adjoint moment operator, when homogeneous boundary conditions are assumed, is given by

$$M^* = \left[ \begin{array}{c|ccc} \varepsilon^2 \sigma_a & -\varepsilon b_0 \frac{\partial}{\partial x} & 0 & 0 & \dots \\ -\varepsilon b_0 \frac{\partial}{\partial x} & 1 & -\varepsilon b_1 \frac{\partial}{\partial x} & 0 & \\ 0 & -\varepsilon b_1 \frac{\partial}{\partial x} & 1 & -\varepsilon b_2 \frac{\partial}{\partial x} & \\ \vdots & & \ddots & \ddots & \ddots \end{array} \right].$$

Scaling the moment equations by

$$S \equiv \begin{pmatrix} \frac{1}{\varepsilon \sigma_a} & & & \\ & \varepsilon & & \\ & & \varepsilon & \\ & & & \ddots \end{pmatrix} \quad (3.1)$$

and forming the normal equations results in

$$M^* S M \Phi = \varepsilon^2 M^* S \hat{q} \iff$$

$$\begin{pmatrix} \varepsilon^3 \sigma_a - \frac{\varepsilon^3}{3} \frac{\partial^2}{\partial x^2} & 0 & -\varepsilon^3 b_0 b_1 \frac{\partial^2}{\partial x^2} & 0 & \dots \\ 0 & \varepsilon - \frac{\varepsilon b_0^2}{\sigma_a} \frac{\partial^2}{\partial x^2} - \varepsilon^3 b_1^2 \frac{\partial^2}{\partial x^2} & 0 & -\varepsilon^3 b_1 b_2 \frac{\partial^2}{\partial x^2} & \ddots \\ -\varepsilon^3 b_0 b_1 \frac{\partial^2}{\partial x^2} & 0 & \varepsilon - \varepsilon^3 (b_1^2 + b_2^2) \frac{\partial^2}{\partial x^2} & 0 & \ddots \\ 0 & -\varepsilon^3 b_1 b_2 \frac{\partial^2}{\partial x^2} & 0 & \varepsilon - \varepsilon^3 (b_2^2 + b_3^2) \frac{\partial^2}{\partial x^2} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix} \underline{\Phi} = \begin{pmatrix} \varepsilon^3 \hat{q}_0 \\ \frac{-\varepsilon^2}{\sqrt{3}\sigma_a} \frac{\partial \hat{q}}{\partial x} \\ 0 \\ 0 \\ \vdots \end{pmatrix}. \quad (3.2)$$

Note that (3.2) already has the correct limit equations for the moments on its diagonal and all first order derivatives are eliminated.

Applying a Galerkin discretization to (3.2) and forming the Schur complement leads after division by  $\varepsilon^3$  to the following discrete equation for  $\phi_0^h$

$$\left\{ \sigma_a(\eta, \eta) + \frac{1}{3}(\eta', \eta') + O(\varepsilon^2) \right\} \phi_0^h = (\eta, \hat{q}_0) + O(\varepsilon^2),$$

which is a valid discretization of the corresponding diffusion equation (1.3) in the limit  $\varepsilon \rightarrow 0$ .

When we define  $\underline{b}_{k,j} \equiv \underline{e}_j \eta_k(x)$ , where  $\underline{e}_j$  denotes the  $j$ -th canonical unit vector of  $\mathbb{R}^N$ , we can write the Galerkin discretization of (3.2) as follows

$$\forall \underline{b}_{k,j} \int_a^b \langle M^* S (M \underline{\Phi} - \underline{\hat{q}}), \underline{b}_{k,j} \rangle_{\mathbb{R}^N} dx = 0. \quad (3.3)$$

Assuming homogeneous boundary conditions and splitting  $S = \sqrt{S} \sqrt{S}$ , (3.3) is equivalent to

$$\begin{aligned} & \forall \underline{b}_{k,j} \int_a^b \langle \sqrt{S} (M \underline{\Phi} - \underline{\hat{q}}), \sqrt{S} M \underline{b}_{k,j} \rangle_{\mathbb{R}^N} dx = 0 \\ & \iff \min_{\underline{\Phi} \in (S^h)^N} \int_a^b \langle \sqrt{S} (M \underline{\Phi} - \underline{\hat{q}}), \sqrt{S} (M \underline{\Phi} - \underline{\hat{q}}) \rangle_{\mathbb{R}^N} dx, \end{aligned}$$

which is a least squares discretization of the moment equations, scaled by  $\sqrt{S}$ . Consequently, a least squares discretization of the moment equations, scaled by  $\sqrt{S}$ , also has the correct behavior in the diffusion limit.

Notice that (3.2) has the proper behavior for any  $\sigma_a \neq 0$ . The second equation contains  $\frac{1}{\sigma_a}$  on both sides and yields the proper solution for  $\phi_1$  as  $\sigma_a \rightarrow 0$ .

On the other hand, if  $\sigma_a = 0$  the scaling (3.1) cannot be applied. In this case, scaling the moment equations by

$$S_0 \equiv \begin{pmatrix} 1 & & & \\ & \alpha & & \\ & & \alpha & \\ & & & \ddots \end{pmatrix}, \quad (3.4)$$

with

$$\alpha \equiv \varepsilon^p, \quad \text{where } p > 2 \quad (3.5)$$

and forming the normal equations results in

$$\left( \begin{array}{c|cccc} -\varepsilon \alpha b_0^2 \frac{\partial^2}{\partial x^2} & -\varepsilon \alpha b_0 \frac{\partial}{\partial x} & -\varepsilon^2 \alpha b_0 b_1 \frac{\partial^2}{\partial x^2} & 0 & \dots \\ \hline \varepsilon b_0 \frac{\partial}{\partial x} & \alpha - \varepsilon^2 (b_0^2 + \alpha b_1^2) \frac{\partial^2}{\partial x^2} & 0 & -\varepsilon^2 \alpha b_1 b_2 \frac{\partial^2}{\partial x^2} & \ddots \\ -\varepsilon^2 \alpha b_0 b_1 \frac{\partial^2}{\partial x^2} & 0 & \alpha - \varepsilon^2 \alpha (b_1^2 + b_2^2) \frac{\partial^2}{\partial x^2} & 0 & \ddots \\ 0 & -\varepsilon^2 \alpha b_1 b_2 \frac{\partial^2}{\partial x^2} & 0 & \alpha - \varepsilon^2 \alpha (b_2^2 + b_3^2) \frac{\partial^2}{\partial x^2} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{array} \right) \underline{\Phi} = \begin{pmatrix} 0 \\ -\varepsilon^3 b_0 \frac{\partial \hat{q}_0}{\partial x} \\ 0 \\ \vdots \end{pmatrix}. \quad (3.6)$$

A Galerkin discretization of (3.6) leads to the following discrete system

$$\left( \begin{array}{c|c} \varepsilon^2 \alpha b_0^2 (\eta', \eta') & A_{01} \\ \hline A_{10} & A_{11} \end{array} \right) \underline{\Phi}^h = \begin{pmatrix} 0 \\ \varepsilon^3 b_0 (\eta', \hat{q}_0) \\ 0 \\ \vdots \end{pmatrix}, \quad (3.7)$$

where

$$A_{01} \equiv (-\varepsilon \alpha b_0 (\eta, \eta'), \varepsilon^2 \alpha b_0 b_1 (\eta', \eta'), 0, \dots)$$

$$A_{10} \equiv (\varepsilon \alpha b_0 (\eta, \eta'), \varepsilon^2 \alpha b_0 b_1 (\eta', \eta'), 0, \dots)^\top$$

and  $A_{11} \equiv$

$$\begin{pmatrix} \alpha(\eta, \eta) + \varepsilon^2 (b_0^2 + \alpha b_1^2) (\eta', \eta') & 0 & \varepsilon^2 \alpha b_1 b_2 (\eta', \eta') & 0 & \dots \\ 0 & \alpha(\eta, \eta) + \varepsilon^2 \alpha (b_1^2 + b_2^2) (\eta', \eta') & 0 & \varepsilon^2 \alpha b_2 b_3 (\eta', \eta') & \ddots \\ \varepsilon^2 \alpha b_1 b_2 (\eta', \eta') & 0 & \alpha(\eta, \eta) + \varepsilon^2 \alpha (b_2^2 + b_3^2) & 0 & \ddots \\ 0 & \varepsilon^2 \alpha b_1 b_2 (\eta', \eta') & 0 & \alpha(\eta, \eta) + \varepsilon^2 \alpha (b_3^2 + b_4^2) & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}.$$

For sufficient small  $\varepsilon$  we have

$$A_{11}^{-1} = \frac{1}{\varepsilon^2} \begin{pmatrix} \frac{1}{b_0^2} (\eta', \eta')^{-1} & & \\ & O(\frac{\varepsilon^2}{\alpha}) & \\ & & \ddots \end{pmatrix} + Q_1 + Q_2 + O(\frac{\alpha}{\varepsilon^2}),$$

with  $Q_1 = O(\frac{\alpha}{\varepsilon^4})$  and  $Q_2 = O(\frac{\varepsilon^2}{\alpha})$ . Using this expansion when forming the Schur complement in (3.7) we get

$$\left\{ \varepsilon^2 \alpha b_0^2(\eta', \eta') - A_{01} A_{11}^{-1} A_{10} \right\} \phi_0^h = -A_{01} A_{11}^{-1} \begin{pmatrix} \varepsilon^3 b_0(\eta', \hat{q}_0) \\ 0 \\ \vdots \end{pmatrix}$$

which after some algebra becomes

$$\begin{aligned} & \left\{ \varepsilon^2 \alpha b_0^2(\eta', \eta') - \alpha^2(\eta, \eta')(\eta', \eta')^{-1}(\eta, \eta') + O(\varepsilon^4 \alpha) \right\} \phi_0^h \\ & = -\varepsilon^2 \alpha(\eta, \eta')(\eta', \eta')^{-1}(\eta', \hat{q}_0) + O(\alpha^2) \end{aligned} \quad (3.8)$$

Because of (3.5) we have  $\frac{\alpha^2}{\varepsilon^2 \alpha} \rightarrow 0$ , so that (3.8) is a valid discretization of the corresponding diffusion equation in the diffusion limit. Using the same argument as above it follows that the least squares discretization of the moment equations, scaled by  $\sqrt{S_0}$ , also has the correct behavior in the diffusion limit.

As mentioned before, the computations are done in the flux representation. Therefore, we need to transfer the scaling of the moment equations to a scaling for the  $S_N$  equations. By means of the relationship (2.3) and (2.4) it follows that a least squares discretization of the moment equations, scaled by  $\sqrt{S}$  is equivalent to a least squares discretization of the  $S_N$  equations, scaled by

$$T^\top \sqrt{S} T \Omega = \frac{1}{\sqrt{\varepsilon \sigma_a}} R + \sqrt{\varepsilon} (I - R).$$

A further multiplication by  $\sqrt{\varepsilon \sigma_a}$  leads to the following scaling in the case  $\sigma_a \neq 0$

$$R + \varepsilon \sqrt{\sigma_a} (I - R). \quad (3.9)$$

Similarly, in the case  $\sigma_a = 0$  the least squares discretization of the moment equations, scaled by  $\sqrt{S_0}$  with  $p = 4$ , is equivalent to a least squares discretization of the  $S_N$  equations, scaled by

$$R + \varepsilon^2 (I - R). \quad (3.10)$$

In order to avoid an “if else” in the computations it is possible to combine the scalings (3.9) and (3.10) to

$$R + (\varepsilon \sqrt{\sigma_a} + \varepsilon^2) (I - R). \quad (3.11)$$

#### 4. NUMERICAL RESULTS

For the solution of the discrete system that results from a least squares discretization of the  $S_N$  equations scaled by (3.11), a full multigrid in space algorithm with

- standard coarsening in space by doubling the mesh width,
- $\mu$ -line red-black smoothing,

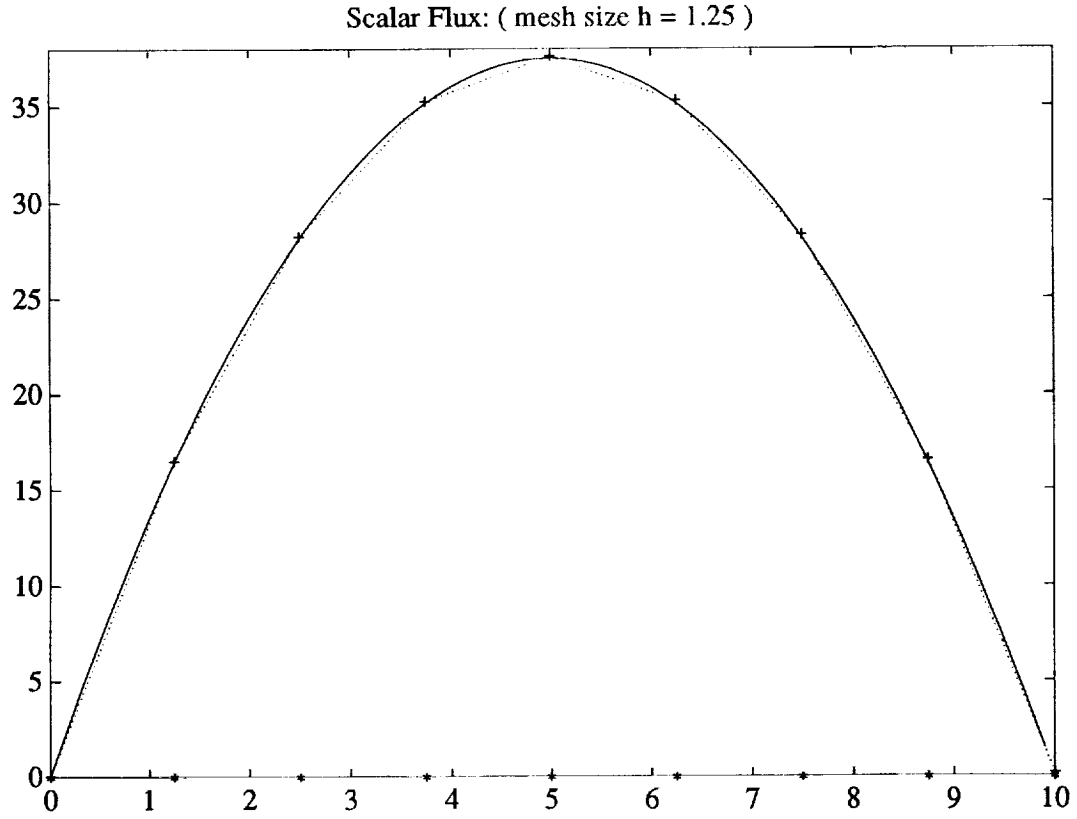


Figure 4.1: Scalar flux solution of problem (4.1)

- full weighting,
- linear interpolation,

was developed. The full multigrid process starts by solving the problem on the coarsest level and uses this solution as a starting guess for the next finer level, where a single V-cycle is performed. Recursively, the solution process proceeds from coarser levels to finer levels by halving each grid cell, using the coarse level solution as a starting guess and performing a single V-cycle on the next finer mesh. This algorithm yields V-cycle convergence rates that are below 0.09. Therefore, by performing one full multigrid V-cycle, a solution with an error on the order of the truncation error is obtained (cf. [7]).

As test problem we used the same problem that was used by Larsen, Morel and Miller in [4], which is shown below:

$$\left\{ \begin{array}{l} \mu_j \frac{\partial \psi_j}{\partial x} + 100 \psi_j - 100 \sum_{\nu=1}^N \omega_\nu \psi_\nu = 0.01 \\ \psi_j(0) = 0 \quad \text{for } \mu_j > 0 \\ \psi_j(10) = 0 \quad \text{for } \mu_j < 0 \end{array} \right\}. \quad (4.1)$$

In our parametrization (1.2) this implies  $\varepsilon = 0.01$ ,  $\sigma_a = 0$  and  $q = 1.0$ . The exact solution of the corresponding diffusion equation is

$$\phi(x) = -\frac{3}{2}x^2 + 15x,$$

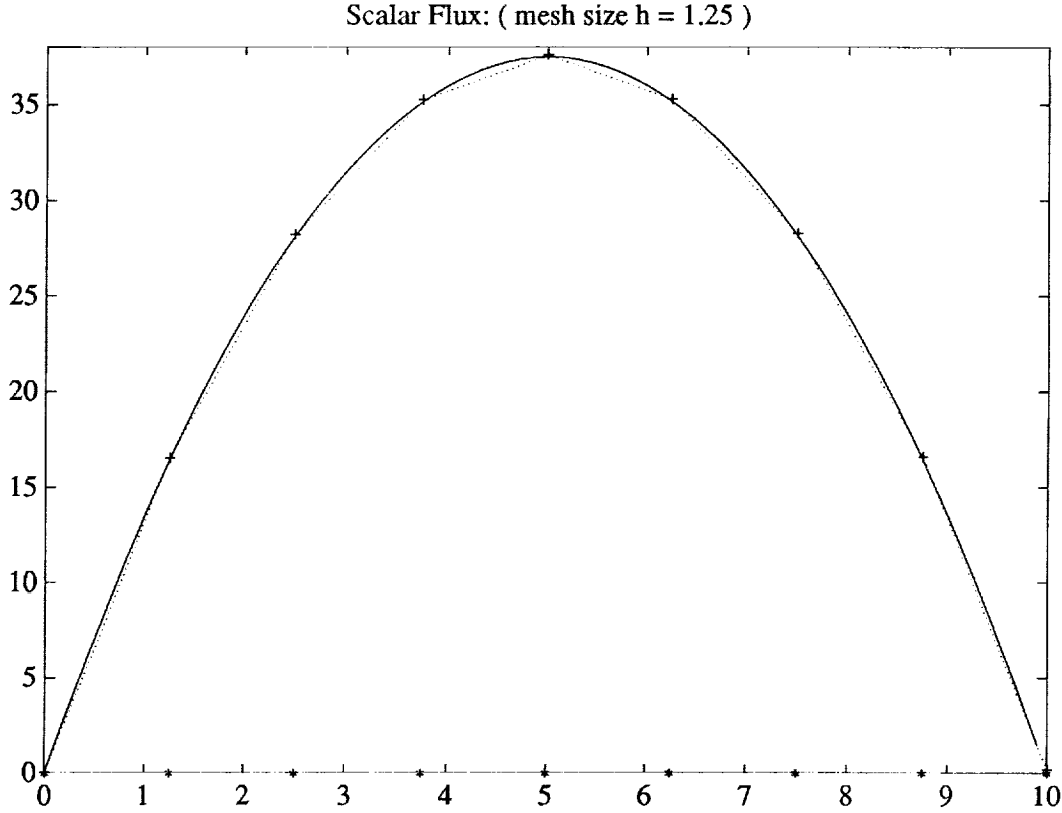


Figure 4.2: Scalar flux solution of problem (4.2)

which is plotted in Figure 4.1 in solid. The least squares solution of the scaled  $S_N$  equations, computed by one full multigrid V-cycle, is shown in Figure 4.1 by the crosses. We see that this is a very satisfactory result, especially when we take into consideration that we used a mesh size of 1.25, which is much larger than  $\varepsilon$ .

Finally, we mention that the least squares discretization of the  $S_N$  equations without scaling will give the zero solution for problem (4.1), indicated by the stars in Figure 4.1.

For the sake of completeness we present in Figure 4.2 the results for the test problem

$$\left\{ \begin{array}{l} \mu_j \frac{\partial \psi_j}{\partial x} + 100 \psi_j - 99.99 \sum_{\nu=1}^N \omega_\nu \psi_\nu = 0.01 \left( 1 - \frac{3}{2} x^2 + 15x \right) \\ \psi_j(0) = 0 \quad \text{for } \mu_j > 0 \\ \psi_j(10) = 0 \quad \text{for } \mu_j < 0 \end{array} \right\}, \quad (4.2)$$

where  $\sigma_a = 1.0$ ,  $\varepsilon = 0.01$ ,  $q = 1 - \frac{3}{2}x^2 + 15x$ . The exact solution of the corresponding diffusion equation is the same as for problem (4.1) and is again plotted in Figure 4.2 in solid. The least squares solution, computed by 1 full multigrid V-Cycle is given in Figure 4.2 by the crosses and the



solution for the least squares discretization of the  $S_N$  equations without scaling is given by the stars.

## 5. CONCLUSIONS

From the analysis in Section 3 and the numerical results presented in Section 4, we conclude that the least squares discretization of the scaled  $S_N$  equations has the proper behavior in the diffusion limit.

Further, we point out that the scaling can be used in the case of nonhomogeneous material, where  $\sigma_t$  or, equivalently,  $\varepsilon$  are discontinuous, because the equations are only scaled from the left side so that no derivatives are applied to the scaling operator.

Adaptive refinement can be combined with the full multigrid solver in a natural way. Areas of new refinement can be identified by examining the difference in the solution for two consecutive grids. This is especially important for nonhomogeneous material, where interior layers may exist.

Numerical results show that with a slightly different scaling both a Galerkin finite element formulation with piecewise linear elements and an Upwind Difference discretization of the  $S_N$  equations also have the correct diffusion limit. We believe that this scaling approach will result in a general framework for the development of discretizations that possess the correct diffusion limit.

Finally, we hope to apply the scaling techniques developed here to higher dimensional problems.

## REFERENCES

- [1] R.E. ALCOUFFE, E.W. LARSEN, W.F. MILLER AND B.R. WIENKE, *Computational Efficiency of Numerical Methods for the Multigroup, Discrete Ordinates Neutron Transport Equations: The Slab Geometry Case*, Nuclear Science and Engineering, 71 pp. 111-127, (1979).
- [2] E.W. LARSEN, *Diffusion Theory as an asymptotic limit of Transport Theory for nearly critical systems with small mean free path*, Annals of Nuclear Energy, Vol 7, pp. 249-255.
- [3] E.W. LARSEN, *On Numerical Solutions of Transport Problems in the Diffusion Limit*, Nuclear Science and Engineering 83, (1983), pp. 90-99.
- [4] E.W. LARSEN, J.E. MOREL AND W.F. MILLER, *Asymptotic Solutions of Numerical Transport Problems in Optically Thick, Diffusive Regimes*, Journal of Computational Physics, Vol. 69, No. 2, (1987), pp. 283-324.
- [5] E.W. LARSEN AND J.E. MOREL, *Asymptotic Solutions of Numerical Transport Problems in Optically Thick Diffusive Regimes II*, J. Comp. Phys. 83, (1989), p. 212.
- [6] E.E. LEWIS AND W.F. MILLER, *Computational Methods of Neutron Transport*, John Wiley & Sons, (1984).
- [7] S.F. MCCORMICK, *Multilevel Projection Methods for Partial Differential Equations*, SIAM, (1992).



54-62  
197564  
N94-21468  
p. 17

# **Analysis of Multigrid Methods on Massively Parallel Computers: Architectural Implications**

**Lesley R. Matheson  
Robert E. Tarjan<sup>1</sup>**

**Department of Computer Science  
Princeton University  
Princeton, New Jersey 08544**

**NEC Research Institute  
Princeton, New Jersey, 08540**

## **Abstract**

We study the potential performance of multigrid algorithms running on massively parallel computers with the intent of discovering whether presently envisioned machines will provide an efficient platform for such algorithms. We consider the domain parallel version of the standard V-cycle algorithm on model problems, discretized using finite difference techniques in two and three dimensions on block structured grids of size  $10^6$  and  $10^9$ , respectively. Our models of parallel computation were developed to reflect the computing characteristics of the current generation of massively parallel multicomputers. These models are based on an interconnection network of 256 to 16,384 message passing, "workstation size" processors executing in an SPMD mode. The first model accomplishes interprocessor communications through a multistage permutation network. The communication cost is a logarithmic function which is similar to the costs in a variety of different topologies. The second model allows single stage communication costs only. Both models were designed with information provided by machine developers and utilize implementation derived parameters. With the medium grain parallelism of the current generation and the high fixed cost of an interprocessor communication, our analysis suggests an efficient implementation requires the machine to support the efficient transmission of long messages, (up to 1000 words) or the high initiation cost of a communication must be significantly reduced through an alternative optimization technique. Furthermore, with variable length message capability, our analysis suggests the low diameter multistage networks provide little or no advantage over a simple single stage communications network.

<sup>1</sup> Research at Princeton University partially supported by the National Science Foundation, Grant No. CCR-8920505, the Office of Naval Research, Contract No. N0014-91-J-1463, and by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), a National Science and Technology Center, Grant No. NSF-STC88-09648

## 0. Introduction

In the current generation of massively parallel (MP) computers there is a convergence towards a common set of architectural characteristics. From the standpoint of a computational scientist, this convergence presents the opportunity to study the class of machines as a whole, in order to determine whether or not they can be efficient platforms for the solution of various computationally intensive tasks.

We studied the potential use of these machines for the solution of multigrid algorithms. Our study included a wide range of multigrid algorithms and encompassed several different architectural characteristics.

In this paper we present the architectural ideas suggested by this study which would enable the current generation of MP machines to become efficient platforms for various multigrid applications.

Our approach was to develop a set of models of parallel computation based on the common characteristics of the current generation of MP machines. We implemented a representative set of structured multigrid algorithms on these models. We then looked at the performance predictions and tried to understand their implications.

The remainder of the paper is organized as follows. First, the models of computation are developed, followed by a brief description of the multigrid algorithms and their implementations. Next, the performance predictions are presented and finally their implications are summarized.

## 1. The Current Generation of Massively Parallel Computers

The power and availability of RISC microprocessor chips have increased dramatically over the past several years. The proliferation and decreased cost of these "workstation-size" processors have spawned the current generation of multicomputers. Some of the major architectural similarities of this generation are summarized below.

**Multicomputers** These multicomputers are interconnection networks of physically distributed processors and memory, linked in a variety of different topological configurations.

**Powerful Microprocessors** The processors are generally "off the shelf" single chip RISC microprocessors. They can perform integer and floating point computation significantly faster than the bit-serial processors which characterized many machines of the previous generation.

*Medium Grain Size* The increased size, cost and speed of the individual processing elements has delineated a medium grain size for the current generation. Most of the machines are targeted for the range of 1K processors, with larger machines possibly ranging up to 16K processors.

*Slow Network Communication* The current machines generally exhibit slow interprocessor communication speeds relative to on-chip events. This is frequently a result of handling the network communications processing in the software layer.

*Single Program Multiple Data Mode of Execution* Unlike the more rigid SIMD and asynchronous MIMD patterns of the previous generation most of the newer machines execute the same program on each processing element with different data, enforcing synchronization only as required by interprocessor communication.

The current generation includes the CM5 by Thinking Machines, a network of Sun SPARC processor nodes, potentially with vector accelerators, connected in a fat tree topology; the Touchstone Delta, developed by Intel and Caltech, a three dimensional mesh of two Intel i860s per node; the Paragon by Intel, a 3D mesh topology with one to four i860 processors per node; the Kendall Square Research machines, a hierarchy of concentric rings with shared virtual memory, with two custom designed chips per node. Cray Research is building a machine with DEC Alpha processors connected by a yet unrevealed topology.

## 2. Models of Parallel Computation

The models of parallel computation presented in this paper were designed to capture the salient characteristics of the current generation of massively parallel computers. The guiding philosophy behind the development of these models was to strike a reasonable balance between machine independence and practicality, simplicity and accuracy. The goal is to find a set of models which facilitates efficient algorithm design, and ideally, provides feedback into the machine design process itself.

The models of computation reflect the paradigm of the multicomputer: processors and memory are physically distributed throughout an interconnection network. Motivated by the large disparity between the speeds of on-chip and network events, the models reflect the costs of a two level memory hierarchy. The cost of a local memory access is included in the cost of an arithmetic operation while the cost of a remote memory access is treated separately. The models were parameterized to facilitate analysis under different ratios of problem to machine

size. In addition, this parameterization allows the incorporation of changes in technology, such as increases in on-chip computation speed or a decrease in network communication latency. The models assume the processors operate in a Single Program Multiple Data mode of execution.

The analysis of this paper utilizes three of the models developed. The characteristics of these models are similar, differing only in their treatment of communication costs. The different treatment of communications costs ranges from assigning a simple topologically-blind cost for a network communication to a more complex function which potentially provides more accuracy. The cost of a floating point computation, treated similarly in each of the models, is separated from the cost of a remote memory access.

### 3. Communication Costs

Accurately and simply accounting for inter-processor communication is the toughest challenge in the development of a useful model of parallel computation. The three alternative treatments presented here are based on the common components of network communication costs exhibited by the current generation of multicomputers.

**1. Fixed Start-Up Costs** There is a large fixed start-up cost associated with any message passing, packet-based communication. To execute a network communication often requires a processor interrupt, complete with a full context switch. The message must be packaged and tagged with destination information and injected into the network.

**2. Variable Cost Per Node** This component of communications cost is the time to route the message through the network to its destination. Cut-through, circuit switched routing, a common general technique, for example, imposes a per node path formation cost. These routing and path formation costs are actually a complex function of the routing algorithm, the communications pattern and network topology. Taken in sum, these comprise the different aspects of contention. In these models, this complex distance-related component is simplified. It is approximated as the product of a machine-dependent constant and the number of processor nodes along the required communications path. Sensitivity analysis is used to potentially understand the impact of different degrees of contention.

**3. Spooling Costs Per Node** A third component of network communications costs is the cost to physically spool the message through the network. Experimental results suggest the spooling cost can be approximated by a linear function of the message length, up to a message size of 1000 words. In these models the spooling cost of a

message is treated as the product of a per-word cost and the length of the message in four-byte words.

**4. Fixed Costs of Receipt** Finally, receiving a message generates a set of costs on the receiving processor, analogous to those required by the originating processor, namely, interrupts, context switches and message unpacking.

## 4. Three Models of Computation

The three models of computation used in this analysis were based on the common architectural characteristics discussed above and differ only in their approximation of the components of network communications costs. The following descriptions assume the models use only fixed constant size messages to accomplish all network communication.

**The GAP Model** The first model, the GAP model, is a simple, topologically blind model which grew out of a set of discussions with a group of researchers at Berkeley. So named because of the "gap" in processor utilization caused by the initiation of a network communication, the GAP model charges one fixed cost for every communication regardless of its source and destination.

**The LOG Model** The second model, the LOG model, introduces a variable topologically-based cost to the fixed cost component. The LOG model assumes the processors are physically connected in a 2D mesh with an overarching multi-stage permutation network. To approximate the distance a message must travel, the LOG model uses the logarithm of the Manhattan distance (or  $L_1$  norm) between the sending and receiving processors on the 2D mesh. The motivation for the use of this function is twofold. First, it realizes the lower bound on path length between any two nodes in a network with a bounded branching factor. Second, it generally approximates the behavior of a variety of networks which realize logarithmic communication distances, such as butterfly and shuffle-exchange networks. Thus, communications cost in the LOG model, with fixed length messages, is approximated by the following function.

$$\text{Communications Cost} = \text{Fixed} + \text{Variable} * \text{Distance}$$

$$\text{where Distance} = \text{Log}(\text{Manhattan Distance})$$

**The Single Stage Model** The Single Stage model also treats the cost of a communication as the sum of fixed costs and a per-node distance dependent cost. This model, like the LOG model, also assumes the processors are physically connected in a two dimensional mesh. In the single stage model, however, there is no overarching multi-stage network. All communication is accomplished by single

or multiple hops along the physical connections of the 2D mesh. The motivation for this model was the possibility of quantifying the impact of a multi-stage network on performance for various applications. The cost of a communication with fixed length messages, therefore, is approximated by the following function.

$$\text{Communications Cost} = \text{Fixed} + \text{Variable} * \text{Distance}$$

where Distance = Manhattan Distance

### ***Model Parameters***

The three models, with fixed length messages, are parameterized by a fixed component and a per-node variable component of communication, the cost of a floating point operation, and the machine size (number of processors). In this analysis eight different pairs of values for fixed and variable cost per node are used. Five pairs are used to represent different possible conditions in the LOG and Single Stage models, while the last three, where the variable costs are zero, represent the similar conditions in the GAP model. The values in the table below were based on timings of random end to end communication patterns on an early release of the CM5 performed by both an internal Thinking Machines applications group and more independent sources.

**Table 1**

**Model Parameters**

Fixed	Variable	Machine State
2500	200	Current
1000	200	Current-Low
500	200	Potential
500	100	Potential
100	50	Ideal
5000	0	Current
3600	0	Current Low
1000	0	Potential

The first two pairs of values approximate the current fixed and variable cost on working machines running "off the shelf" software. The first pair (2500, 200) is an averaged approximation while the second pair is more idealized. With a 33Mhz clock, such as the current clock speed of the SPARC chip used in the CM5, for example, a 2500 cycle fixed cost and a 200 cycle per-node variable cost translates to approximately 75 and 6-7 microsecond costs, respectively. The next two pairs represent reductions in cost which may be possible within this generation. The fifth pair represents an ideal. The last three pairs attempt to replicate the three different states within the GAP model. The cost of a 32-bit floating point operation, in



machine cycles, is estimated at 6 cycles. While on-chip computing speeds are rapidly increasing, this value attempts to approximate the current state without accelerators which have a "peak" rate of two operations per cycle.

When the message size is allowed to vary up to approximately 1000 words, a fourth parameter, the per-word spooling rate, is introduced. Experimental data suggested a 4 cycle per-word cost would be a reasonable value, with a sensitivity analysis up to approximately 12 cycles per-word.

### *Parallel Machine Size*

The current generation of massively parallel machines is characterized by "medium-grain" machines typically consisting of approximately 256 to 16K processors. This analysis considers machines with  $2^8$ ,  $2^{10}$ ,  $2^{12}$ , and  $2^{14}$  processors.

## **5. Multigrid Algorithms and Implementations**

The analysis presented here considers the standard V and F-cycle in two and three dimensions. This analysis considers only the simplest problems and solution schemes: model problems are considered on structured meshes spanning square and cubic domains. Explicit weighted Jacobi schemes are used to solve problems discretized using second order finite difference techniques. The hierarchy of structured meshes is constructed using a coarsening ratio of two in each dimension. The cycling schemes execute two relaxation sweeps on the downstroke and one on the upstroke.

The problems were implemented on the parallel models using simple, practical domain partitioning strategies. In two dimensions the finest mesh was simply partitioned into load-balanced square subdomains and mapped to the analogous processor in the 2D mesh of processors. In three dimensions, the domain was analogously partitioned and the processor mapping was only slightly more complicated and was within a factor of two of optimal.

## **6. Analysis Overview**

The remainder of this paper presents the results and implications of the implementation of the standard multigrid algorithms on the three models of parallel computation. The following two sections present the performance predictions for the two and three dimensional V-cycle when fixed length messages are used to execute all of the required network communication. Next, the results of the same analysis are repeated with variable length messages where the message size is allowed to vary up to 1000 words. The results of an implementation of the 3D

V-cycle on the Single Stage model are then presented. Finally, the implications of the set of predictions are summarized.

## 7. The Standard V-cycle in Two Dimensions

The performance predictions for the two dimensional V-cycle on both the LOG and GAP models were not encouraging. On moderate sized machines, those with 1K to 4K processors, with a 2500 fixed communications cost (approx. 75 microseconds), the models predicted speed-ups of only 55 times over the serial implementation. For larger machines, the speed-ups do not even reach 200 times. The table below shows the speed-ups of the V-cycle for different machine sizes under different assumptions of fixed and variable communications costs. The problem size is 1,000,000 points or 1000 points per dimension.

Table 2

### Speed-Up Two Dimensional V-cycle with Fixed Length Messages

$$N^2 = 1,000,000$$

#### GAP and LOG Model Predictions

Processors	256	1024	4096	16,384
Fixed, Variable				
2500, 200	27.1	55.1	103.2	172.6
1000, 200	58.3	125.5	238.6	387.1
500, 200	94.4	218.8	424.0	660.9
500, 100	94.9	223.5	450.5	755.0
100, 50	190.4	585.7	1462.1	2881.7
5000, 0	19.5	39.3	74.4	128.6
3600, 0	19.8	40.4	79.3	147.0
1000, 0	58.7	128.6	255.4	453.2

Because the information provided by these models attempts to bridge the gap between abstract models of computation and machine-dependent benchmarks, interpreting the data is not straightforward. From a theoretical perspective these speed-ups are far from linear. On the other hand computing the wall clock time associated with these predictions, then scaling these model problem times to reflect the increased complexity of actual applications, produces running times which are unacceptably slow.

If the fixed cost of a communication can be reduced to 500 cycles or 15 microseconds with a 33MHz clock, the models predict speed-ups in the range of 200

times. Only in the ideal case where the fixed cost of a communication is 100 cycles or approximately 3.3 microseconds, do the speed-ups become somewhat attractive.

These discouraging predictions are a result of very high communications latencies. With a fixed problem size of 1,000,000 points, in this range of processors, the fine grid communications costs dominate both the cost of the computation and the cost of coarse grid communications.

Very fast processors and relatively slow network communication create very poor processor utilization in this range of processors and problem sizes. The increased cost of the microprocessors in these machines makes efficiency an important performance criterion. We define efficiency here as the ratio of the time spent on computation to the total time on both computation and network communication. The table below shows the efficiency predicted by the models for the two dimensional V-cycle using the same eight pairs of values for the fixed and variable cost of a network communication.

**Table 3**

### **Efficiency**

Two Dimensional V-cycle  
with Fixed Length Messages

$$N^2 = 1,000,000$$

#### **GAP and LOG Model Predictions**

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	10.6%	5.39%	2.55%	1.13%
1000, 200	22.77%	12.29%	5.91%	2.53%
500, 200	36.88%	21.44%	10.51%	4.32%
500, 100	37.10%	21.90%	11.17%	4.95%
100, 50	74.41%	57.38%	36.36%	17.54%
5000, 0	5.62%	2.80%	1.33%	.60%
3600, 0	7.63%	3.85%	1.84%	.85%
1000, 0	22.94%	12.60%	6.33%	2.97%

Both the LOG and the GAP models predict very low efficiency levels when the fixed cost of a communication is high. With a fixed cost of 2500 cycles, small to modest sized machines, consisting of 256-1024 processors, reach only 5%-10% efficiency. With a fixed cost of 1000 cycles (approximately 30.3 microseconds using a 33Mh clock), efficiency is still only 10%-20%. Driving the fixed cost down to 500 cycles (15 microseconds) produces more reasonable levels of 20%-30% for modestly sized machines. To reach 40%-60% efficiency where the machine begins to leverage

the power of these new microprocessors, the fixed cost needs to be reduced all the way down to the 100 cycle range (3.3 microseconds).

## 8. The Standard V-cycle in Three Dimensions

The implementation and analysis of three dimensional problems differs from the two dimensional analysis in several ways. First, the additional dimension increases the computational burden by a factor of  $O(N)$ , to  $O(N^3)$ , while increasing the required communication by a factor of  $N/P^{1/6}$ . Second, mapping the three dimensional problem domain to a two dimensional machine model tends to increase not only the complexity, but the distance of interprocessor communications. Third, the problem size in the analysis is increased by a factor of 1000, while still considering the same range of machine sizes.

The LOG and GAP models predict only slightly improved levels of performance for the three dimensional V-cycle. Table 4 below lists the speed-ups predicted by the models for three dimensional problems with one billion points.

Table 4

### Speed-Up

#### Three Dimensional V-cycle with Fixed Length Messages

$$N^3 = 1,000,000,000$$

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	49.1	130.6	338.1	859.3
1000, 200	86.7	240.3	636.8	1632.5
500, 200	116.2	333.7	902.7	2332.2
500, 100	129.5	389.2	1102.2	2969.2
100, 50	202.7	702.6	2288.7	6954.7
5000, 0	30.1	79.2	205.3	526.7
3600, 0	39.9	106.8	279.7	722.5
1000, 0	102.3	302.4	855.2	2333.5

Generally, the predictions are not encouraging. The slight increase in performance is due to the increased amount of computation relative to both the amount of communication and the number of processors. For a 1024 processor machine with a 2500 cycle fixed communication cost, the LOG model predicts a speed-up of only 130 times. If the fixed cost of a communication drops to 500 cycles, this improves by a factor of 2-3. Only in the ideal case of a 100 cycle fixed

communication cost do the results approach acceptable levels for moderately-sized machines and design tool levels, with thousand-fold speed-ups, for very large machines.

As with the two dimensional predictions, the sluggish predictions are due mainly to the overwhelming costs of the network communication. Table 5 below shows the efficiency levels which coincide with these speed-up predictions.

**Table 5**

### **Efficiency**

#### **Three Dimensional V-cycle with Fixed Length Messages**

$$N^3 = 1,000,000,000$$

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	19.19%	12.75%	8.25%	5.26%
1000, 200	33.85%%	23.46%	15.54%	9.96%
500, 200	45.40%	32.59%	22.03%	14.23%
500, 100	50.58%	38.01%	26.92%	18.12%
100, 50	79.17%	68.61%	55.87%	42.45%
5000, 0	11.7%	7.7%	5.01%	3.22%
3600, 0	15.59%	10.42%	6.83%	4.40%
1000, 0	39.95%	29.53%	20.87%	12.42%

The predictions of these models are in contrast to the asymptotic predictions of more abstract models of computation. Asymptotic analysis suggests the fine grid communications costs become negligible as the problem size gets large for a fixed range of machine sizes. These results suggest the huge imbalance between the cost of communication per word and the cost of a floating point computation causes communication time to dominate the time spent on computation, even with one billion points.

The standard V-cycle algorithm alternates between computation and communication systolically, placing a heavy communications burden on a multi-stage interconnection network. On medium-grain multiprocessors, those with 256 to 16K processors, for realistic problem sizes, local, fine-grid communication is predominant. By the time the grids have coarsened beyond one point per processor, only a small fraction of the computation remains. This magnifies the importance of a small fixed cost per word and de-emphasizes the importance of low variable per-node communications costs. Unfortunately, the models in the previous section show the demand for inexpensive local communication is answered in the current

generation of massively parallel machines by a high fixed communications cost producing discouraging levels of performance for both two and three dimensional problems.

## 9. The Standard V-cycle with Variable Length Messages

The previous analysis assumed all communication was accomplished through fixed length messages consisting of only a small constant number of words. Sensitivity analysis suggested that acceptable levels of performance required lower fixed costs per word. The low spooling rate per word exhibited by these machines motivates potentially lowering the average communication cost per word by transmitting large blocks of words per message. With large messages, the fixed cost of initiating a network communication can be amortized over a larger number of words, lowering the effective fixed cost per word.

In the analysis of this section, spooling costs are added to the communication cost functions of the previous section. The cost of a message is a function of the distance and the length in words, and is the sum of fixed start-up and receipt costs, variable per-node costs and spooling costs.

Experimental data suggest that approximating the total spooling costs as a linear function of message size is reasonable up to approximately 5000 words. The analysis here assumes a maximum message size of 1000 words and uses a per word spooling cost of 4 clock cycles. Approximating the spooling rate was accomplished with the help of timings provided by Pablo Tomayo of Thinking Machines, Inc. The rate was determined by a regression analysis on three node ping pong rates of message sizes ranging from 1 to 5000 words. Sensitivity analysis with rates up to 12 cycles per word showed the results of this section are relatively insensitive to small changes in the per-word spooling rate.

The predictions for the standard V-cycle algorithm in two dimensions with large message transmission were generally far more encouraging than the fixed message length predictions. The table below lists the speed-up and efficiency predictions for the same eight pairs of fixed and variable communications costs.

Table 6

**Speed-Up****Efficiency**

**Two Dimensional V-cycle  
with Variable Length Messages**

$$N^2 = 1,000,000$$

Processors Fixed, Variable	256	1024	4096	16,384
2500, 200	170.1 66.49%	314.0 30.76%	368.0 9.12%	354.5 2.32%
1000, 200	208.3 81.41%	501.8 49.17%	709.4 17.59%	715.5 4.69%
500, 200	225.1 87.99%	626.9 61.42%	1027.0 25.47%	1083.1 7.10%
500, 100	338.1 89.23%	667.1 65.36%	1197.3 29.69%	1360.8 8.92%
100, 50	446.1 96.21%	882.6 86.47%	2396.4 59.44%	3839.6 25.1%
5000, 0	132.5 51.77%	200.8 19.67%	216.5 5.36%	207.7 1.36%
3600, 0	152.8 59.72%	258.6 25.33%	294.2 7.29%	286.7 1.88%
1000, 0	213.8 83.55%	555.4 54.42%	882.9 21.9%	979.7 6.42%

These predictions show at least a factor of 6 speed-up on moderate-size machines and a factor of two speed-up on large machines over the fixed length predictions. For example, on a 1024 processor machine, with a fixed communications cost of 2500 cycles, with variable length messages, the speed-up predicted is 314 as compared to 55 on the models with constant message size. There is a corresponding improvement in the efficiency of 30% versus 5%. If fixed costs can be driven down to 500 cycles, the variable message length still provides approximately a factor of two improvement over the fixed length predictions.

With large messages reducing the fine grid communication costs, the coarse grid communications costs, which are proportional to  $\log^2 P$ , grow to counterbalance the computational speed-ups provided by additional processors. The increase in speed-up as the number of processors gets large is less pronounced. In addition, the optimal number of processors implied by this trade-off occurs in a more reasonable range. For example, with fixed and variable costs of 2500 and 200 cycles respectively,

the models predict that the optimal number of processors for this computation is approximately 4900.

With the three dimensional V-cycle, the models suggest that the ability to send variable length messages, up to 1000 words, produces a marked increase in solution speed in this range of processors, on problems up to one billion points. The table below shows the speed-ups predicted for the three dimensional algorithm by the LOG and GAP models.

**Table 7**  
**Speed-Up**  
**Three Dimensional V-cycle**  
**with Variable Length Messages**

$$N^3 = 1,000,000,000$$

Processors ( $t_F, t_V$ )	256	1024	4096	16,384
2500, 200	253.3	1006.1	3965.4	15,192.1
1000, 200	253.9	1010.3	3999.2	15,555.7
500, 200	254.1	1011.7	4010.6	15,680.8
500, 100	254.2	1012.3	4016.9	15,773.9
100, 50	254.4	1013.7	4029.3	15,924.2
5000, 0	252.5	1000.3	3922.4	14,785.1
3600, 0	253.0	1004.2	3953.3	15,105.8
1000, 0	254.1	1011.5	4011.8	15,739.9

With variable length messages, the high fixed communications cost can be effectively amortized over a large number of words, driving down the average cost per word to a more ideal range. Computation costs dominate the total execution time, producing almost linear speed-ups in this range of problem to processor size. Almost all of the complementary efficiency levels are above 90% for each of the eight fixed, variable communications cost pairs throughout the entire range of machine sizes.

These results suggest the average communications cost per word can be driven down far enough through the efficient transmission of large messages to effectively leverage the increased computational speeds of the current generation of microprocessors. Thus, the ability to package messages into large blocks, up to a 1000 word maximum, can potentially bring these machines closer to the goal of design tool performance on these problems.



## 10. The F-Cycle

Performance predictions for the standard F-cycle were very similar to the V-cycle results. With both fixed, constant length and variable length message transmission, the F-cycle slightly outperformed the V-cycle. With fixed message lengths, this was due mainly to the reduced amount of fine grid communication of the F-cycle. With the ability to send large messages, the F-cycle outperformed the V-cycle in three dimensions because of the reduction in the amount of required computation.

## 11. Standard V-cycle on a Single Stage Machine

The two and three dimensional V-cycle algorithms were implemented on the Single Stage model in order to try to determine the impact of a multi-stage network on the performance of multigrid algorithms. The single stage model assumes the processors are connected by a 2D mesh and all communication takes place along these physical connections. There is no overarching multi-stage communication network. The model is parameterized by the same machine dependent costs, namely, fixed and variable communications costs, spooling rates and floating point computation rates. The only difference in communications costs is in the variable, distance related cost component. In this model the distance a message must travel is simply the Manhattan distance (the  $L_1$  norm) of the location of the sending and receiving processors on the mesh.

The results in both two and three dimensions suggest the impact of a multi-stage network on performance is very small, regardless of the maximum message length. The table below shows the increase in total time caused by sending messages through the mesh connections rather than through the logarithmic multi-stage network defined by the LOG model.

Table 8

**The Percentage Increase in Total Time for the 3D V-cycle  
Implemented on the Single Stage Model  
from the Time Required On the Multi Stage LOG Model**

Number of Processors	% Difference M=1	% Difference M=1000
256	5.88%	.05%
1024	8.17%	.19%
4096	11.54%	1.19%
16,384	16.47%	8.79%

The table shows a less than 10% increase on moderate sized machines with fixed message length communication, where the fixed and variable costs of a

communication are 2500 and 200 cycles respectively. For machines with variable length message capability, the increase in total time is less than 1% for moderate machines.

In three dimensions, the increase in communications costs alone with variable length messages is small, except on very large machines. The table below isolates the communications costs and shows the percentage increases.

**Table 9**

**The Percentage Increase in Communications Time for the 3D V-cycle Implemented on the Single Stage LPSS Model from the Time Required On the Multi Stage LOG Model**

Number of Processors	% Difference M=1000
256	4.38%
1024	10.87%
4096	37.30%
16,384	120.91%

The table shows the small increase in communications costs with only a single stage permutation network. For very large machines the increase is only slightly over a factor of two. These results suggest that even for very large machines with fixed length messages, the addition of multi-stage networks does not seem to enhance performance enough to justify the additional machine complexity.

## 12. Conclusions

The performance predictions presented here suggest the fixed cost of a communication on the current generation of massively parallel machines needs to be driven down into the range of 15 microseconds to produce acceptable levels of performance. Ideally, the cost should be in the range of 3 microseconds. The computational speeds of the next generation of microprocessors appear to be increasing rapidly. Though these and other hardware advances may produce enhanced performance, they will certainly exacerbate the huge disparity between the speeds of on-chip and network events. Driving the average cost of a local communication appears to be imperative if these machines are to become efficient platforms for the the solution of multigrid applications.

One way to accomplish this reduction in the average cost per word of a network communication may be through the efficient transmission of large messages. This capability would allow the fixed cost of a communication to be amortized over a large number of words.

Finally, expensive multi-stage networks appear to have little impact on the performance of standard multigrid algorithms. In this range of problem to machine sizes, with both fixed and variable length message transmission, performance

degrades only slightly when communication is forced to traverse the physical connections of a 2D mesh of processors.



55-64  
1-97565  
N 94 - 21469

TIME-ACCURATE NAVIER-STOKES CALCULATIONS  
WITH MULTIGRID ACCELERATION

N. Duane Melson  
NASA Langley Research Center  
Hampton, VA 23681-0001

Mark D. Sanetrik  
Analytical Services and Materials, Incorporated  
Hampton, VA 23681-0001

Harold L. Atkins  
NASA Langley Research Center  
Hampton, VA 23681-0001

SUMMARY

A numerical scheme to solve the unsteady Navier-Stokes equations is described. The scheme is implemented by modifying the multigrid-multiblock version of the steady Navier-Stokes equations solver, TLNS3D. The scheme is fully implicit in time and uses TLNS3D to iteratively invert the equations at each physical time step. The design objective of the scheme is unconditional stability (at least for first- and second-order discretizations of the physical time derivatives). With unconditional stability, the choice of the time step is based on the physical phenomena to be resolved rather than limited by numerical stability which is especially important for high Reynolds number viscous flows, where the spatial variation of grid cell size can be as much as six orders of magnitude.

An analysis of the iterative procedure and the implementation of this procedure in TLNS3D are discussed. Numerical results are presented to show both the capabilities of the scheme and its speedup relative to the use of global minimum time stepping. Reductions in computational times of an order of magnitude are demonstrated.

INTRODUCTION

Although significant progress has been made in the last twenty years to numerically model many physical situations, most numerical schemes are limited to the prediction of steady flows. This limitation is particularly true in the field of computational fluid dynamics (CFD), where solutions to the Navier-Stokes equations for steady flows are now calculated on a regular basis. (See, for example, references [1-3]). An important factor that has lead to the increased use of Navier-Stokes solvers is the recent success in

reducing the computer resources necessary to obtain converged solutions. Perhaps the most promising work has been in the use of multigrid acceleration techniques. Convergence to steady state has been shown in  $O[\log(n)]$  work, where  $n$  represents the number of unknowns to be solved. This reduction in computer requirements has made steady-state solutions affordable to the practicing engineer.

However, many physical phenomena (e.g., separated flows, wake flows, buffet) are intrinsically unsteady. The solution of unsteady problems in CFD has been limited to simplified subsets of the Navier-Stokes equations (panel methods, potential-flow solvers, and some limited use of Euler equation solvers). Unsteady Navier-Stokes calculations have been too expensive for routine use.

The present approach is to apply an iterative procedure for the solution of an implicit equation; thus, the approach is called an *iterative-implicit* method. The concept is not new; in fact, many of the methods developed in the field of linear algebra for inverting large matrices are iterative. Within the field of CFD, similar work is discussed by Jameson [4] for unsteady flows and by Taylor, Ng, and Walters [5] for steady-state flows. The present approach is similar to that of Jameson in that a Runge-Kutta based multigrid method is used to solve the implicit unsteady flow equations. The Navier-Stokes equations have been treated in the present work, and Jameson's implementation has been modified so that the robustness of the scheme is dramatically increased.

A detailed description of the implementation will be followed by an analysis of the method and the numerical results from one- and two-dimensional test problems.

## TIME-DEPENDENT METHOD

In the present work, a modified version of the thin-layer Navier-Stokes (TLNS) equations is used to model the flow. The acronym "TLNS" used here describes an equation set obtained from the complete Reynolds-averaged Navier-Stokes equations by retaining only the viscous diffusion terms normal to the solid surfaces. The effects of turbulence are modeled through an eddy-viscosity hypothesis. The Baldwin-Lomax turbulence model [6] is used for turbulence closure. For a body-fitted coordinate system  $(\xi, \eta, \zeta)$  fixed in time, these equations can be written in the conservation-law form as

$$-\frac{\partial}{\partial T}(J^{-1}U) = \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} + \frac{\partial H}{\partial \zeta} - \frac{\partial F_v}{\partial \xi} - \frac{\partial G_v}{\partial \eta} - \frac{\partial H_v}{\partial \zeta} \quad (1)$$

where  $U$  represents the conserved variable vector and  $F$ ,  $G$ , and  $H$  represent the convective flux vectors. In the above equation set  $F_v$ ,  $G_v$  and  $H_v$  represent the viscous flux vectors in the three coordinate directions  $(\xi, \eta, \zeta)$ , and  $J$  is the Jacobian of the transformation. These equations represent a more general form of the classical thin-layer equations introduced in reference [6] because the diffusion terms in all three coordinate directions are included in this form. The Euler equations can easily be recovered from equation (1) by simply dropping the last three terms on the right-hand side.

The temporal derivatives are cast as a fully implicit operator in physical time. For first- or second-order discretizations in time, this produces an unconditionally stable scheme, which allows the time-step size to be chosen based on the temporal resolution needed in the solution rather than limited by the numerical

stability requirements. The fully implicit terms are iteratively solved with multigrid acceleration rather than direct inversion, which would be too costly for the nonlinear three-dimensional Navier-Stokes equations.

## IMPLEMENTATION OF TIME-DEPENDENT METHOD

### Original TLNS3D Method

In the original TLNS3D program, a semidiscrete cell-centered finite-volume algorithm, based on a Runge-Kutta time-stepping scheme [1][7][8], is used to obtain the steady-state solutions to the TLNS equations. A linear fourth-difference-based and nonlinear second-difference-based artificial dissipation is added to suppress both the odd-even decoupling and the oscillations in the vicinity of shock waves and stagnation points, respectively. Both the scalar and matrix forms of the artificial dissipation models [9] are incorporated.

In the steady-state implementation, the physical time  $T$  is replaced by a pseudo time  $\tau$ , which gives

$$-\frac{\partial}{\partial \tau}(J^{-1}U) = \frac{\partial F}{\partial \xi} + \frac{\partial G}{\partial \eta} + \frac{\partial H}{\partial \zeta} - \frac{\partial F_v}{\partial \xi} - \frac{\partial G_v}{\partial \eta} - \frac{\partial H_v}{\partial \zeta}. \quad (2)$$

At steady state, the left-hand side of equation (2) disappears, and the right-hand side (the residual) goes to zero, so that any stable scheme may be used to advance the solution in pseudo time.

In the original TLNS3D program, the solution is advanced with a five-stage Runge-Kutta time-stepping scheme. Three evaluations of the artificial dissipation terms (computed at the odd stages) are used to obtain a larger parabolic stability bound, which allows a higher CFL number in the presence of physical viscous diffusion terms. Such a scheme is computationally efficient for solving both the steady Navier-Stokes and the steady Euler equations. The stability range of the numerical scheme is further increased with the use of the implicit residual smoothing technique that employs grid aspect-ratio-dependent coefficients [1][10][11].

Equation (2) can be rewritten to group the convective and diffusive terms from the right-hand side as

$$Vol \frac{\partial(U)}{\partial \tau} + C(U) - D_p(U) - D_a(U) = 0 \quad (3)$$

where the equation has been multiplied through by the volume  $Vol$  and  $C(U)$ ,  $D_p(U)$ , and  $D_a(U)$  are the convection, physical diffusion, and artificial diffusion terms, respectively. The implementation of the Runge-Kutta time stepping is shown by rewriting equation (3) as

$$Vol \frac{U^k - U^0}{\alpha_k \Delta \tau} + C^{k-1}(U) - D_p^0(U) - \widetilde{D_a^{k-1}}(U) = 0 \quad (4)$$

where the superscript  $k$  indicates that the given term should be evaluated at the  $k$ th Runge-Kutta stage. The  $k-1$  superscript indicates that the terms are evaluated with a linear combination of the values from previous Runge-Kutta stages.

The solution is advanced in pseudo time with the maximum allowable time step for each cell. Enthalpy damping, which has been used in several previous studies to accelerate the convergence of the numerical scheme, is not employed because the Navier-Stokes equations generally do not admit constant enthalpy as a solution. The efficiency of the steady numerical scheme is also significantly enhanced through the use of a multigrid acceleration technique as described in reference [1]. The original TLNS3D program was extensively modified to facilitate solution of the flow fields over a wide range of geometric configurations through domain decomposition. A consequence of this work is the generalization of the boundary conditions of the program to easily accommodate any arbitrary grid topology. A detailed description of this capability is given in reference [12].

### Time-dependent TLNS3D-MB

The physical time derivative of equation (1) is approximated by a discrete operator of the form

$$\frac{\partial U}{\partial T} \approx L_t U^{n+1} \equiv \frac{1}{\Delta T} \left( \sum_{m=0}^M a_m U^{n+1-m} \right) = \frac{1}{\Delta T} \left[ a_0 U^{n+1} + E(U^n, U^{n-1}, \dots, U^{n+1-M}) \right] \quad (5)$$

to give

$$L_t U^{n+1} = S(U^{n+1}) \quad (6)$$

where  $E$  denotes the portion of the discrete physical time derivative that involves values from the previous time steps and  $S(U^{n+1})$  denotes the discrete approximation to the right-hand side of equation (1). Equation (6) is an implicit time-accurate equation for the time advancement of the unsteady solution of the Navier-Stokes equations. The first task is to put equation (6) in a form that is amenable to time-asymptotic steady-state methods such as TLNS3D. This involves construction of an iteration procedure that can be interpreted as a pseudo time. The TLNS3D code employs a Runge-Kutta and multigrid methodology to advance the pseudo time, which introduces an additional level of iteration. To avoid the use of multiple indices and, hopefully, avoid confusion between the various iteration procedures within the overall algorithm, the following change of variables is introduced. Let  $W \equiv U^{n+1}$  and  $W^l$  denote the  $l$ th approximation to  $W$ . Thus,  $\lim_{l \rightarrow \infty} W^l$  is equal to  $U^{n+1}$  whenever the iterative method used for the solution of equation (6) is convergent. In describing the Runge-Kutta scheme, let  $V^k$  denote the solution obtained in the  $k$ th stage of the Runge-Kutta scheme.

Equation (6) is rewritten in this notation to obtain

$$\frac{a_o}{\Delta T} W + \frac{E}{\Delta T} = S(W) \quad (7)$$

where  $E$  again involves the portion of the physical time derivative at previous time steps and is invariant during the iteration process which advances the solution from  $T^n$  to  $T^{n+1}$ . An iterative equation is constructed from equation (6) simply by adding a pseudo-time derivative term to the left-hand side. The only consideration is that the sign of the new term must be the same as that of the physical time derivative.

$$W_\tau + \frac{a_o}{\Delta T} W + \frac{E}{\Delta T} = S(W) \quad (8)$$



When  $\bar{\lambda} \equiv a_0 \Delta \tau / \Delta T$  is small, equation (8) differs from equation (2) by only a small perturbation. Thus, we can reasonably expect that any method that efficiently solves equation (2) would also solve equation (8).

In the previous work of Jameson [4], all contributions from the physical time term were carried to the right-hand side of the equation and treated as explicit terms within the Runge-Kutta stages. With efficiency as a consideration, Jameson suggested the use of this approach only when the CFL, based on the physical time step, is greater than 200. (Note that a large CFL corresponds to a small  $\bar{\lambda}$ ). A later section shows that this explicit approach is actually unstable for large values of  $\bar{\lambda}$ .

In many flows, especially high Reynolds number viscous flows,  $\bar{\lambda}$  may easily become large. In the present work, the Runge-Kutta scheme is made stable for all values of  $\bar{\lambda}$  by treating the contribution of the physical time derivative implicitly within the Runge-Kutta scheme. Because the time derivative appears only as a diagonal term, the modification is easily implemented as follows.

$$\begin{aligned} V^0 &= W^l \\ (1 + \alpha_k \bar{\lambda}) V^k &= V^0 + \alpha_k \Delta \tau \hat{R}^{\overline{k-1}}(V) \quad k = 1, 2, 3, \dots, K \\ W^{l+1} &= V^K \end{aligned} \quad (9)$$

where  $\hat{R}(V) = S(V) - E/\Delta T$  denotes the modified residual, and the superscript  $\overline{k-1}$  denotes that the residual may be a combination of  $\hat{R}(V)$  at all the previous Runge-Kutta stages.

This formulation is not yet appropriate for steady-state flow solvers such as TLNS3D. The reason is that these codes use several acceleration techniques, such as implicit-residual smoothing and multigrid, both of which are designed to operate on a residual term that goes to 0 as the solution converges. Because  $\hat{R}$  contains only the portion of the physical time derivative at previous physical time steps, it converges to  $\bar{\lambda}W$  as  $W^l$  goes to  $W$ . To accommodate the above acceleration techniques,  $\hat{R}$  is rewritten as

$$\Delta \tau \hat{R}(V) = \bar{\lambda}V - \bar{\lambda}V + \Delta \tau \hat{R}(V) = \bar{\lambda}V + \Delta \tau R(V) \quad (10)$$

where

$$R(V) = S(V) - (a_0 V + E)/\Delta T \quad (11)$$

The residual  $R$  contains all the physical terms and goes to 0 as  $W^l$  goes to  $W$ . The Runge-Kutta method, with implicit-residual smoothing and multigrid, becomes

$$V^0 = W^l \quad (12a)$$

$$(1 + \alpha_k \bar{\lambda}) V^k = V^0 + \alpha_k \bar{\lambda} V^{\overline{k-1}} + \alpha_k \Delta \tau L_{irs}^{-1} \bullet \left[ R^{\overline{k-1}}(V) + f \right] \quad k = 1, 2, 3, \dots, K \quad (12b)$$

$$W^{l+1} = V^K \quad (12c)$$

where  $L_{irs}$  denotes the implicit-residual smoothing operator, and  $f$  denotes the multigrid forcing function (which is zero on the finest grid).

The usual coarse-grid equation that would result from applying multigrid to equation (8) is

$$W_{\tau}^{(2h)} = R^{(2h)}(W) + \left[ I_{(h)}^{(2h)} R^{(h)}(W^{(h)}) - R^{(2h)}(I_{(h)}^{(2h)} W^{(h)}) \right] \quad (13)$$

where the superscript in parentheses denotes the multigrid grid level at which the operator or variable is defined ( $h$  on the fine grid,  $2h$  on the next coarse grid, etc.), and the operator  $I_{(h)}^{(2h)}$  denotes the restriction

process from the fine grid ( $h$ ) to the coarse grid ( $2h$ ). Because  $E$  is invariant during the multigrid cycle, its influence in the operator  $R^{(2h)}$  cancels on the coarse grid. Consequently, the coarse-grid residual can be rewritten to exclude  $E$ , which eliminates the need to restrict and store this term. The actual coarse-grid equation is now

$$W_\tau^{(2h)} = \tilde{R}^{(2h)}(W) + \left[ I_{(h)}^{(2h)} \tilde{R}^{(h)}(W^{(h)}) - \tilde{R}^{(2h)}(I_{(h)}^{(2h)} W^{(h)}) \right] \equiv \tilde{R}^{(2h)}(W) + f^{(2h)} \quad (14)$$

where

$$\tilde{R}^{(m)} = \begin{cases} R(W) & (m) = h \\ S(W) - \bar{\lambda}W & (m) = 2h, 4h, \dots \end{cases} \quad (15)$$

### STABILITY ANALYSIS

Two stability issues are associated with the iterative-implicit method. The first issue is the stability of the implicit equation that contains all the physics (equation (6)). The second issue is the stability and convergence of the iterative algorithm that is used to invert the implicit equation. The second issue can easily be studied independently of the first. The first issue can be studied independently of the second by assuming that the implicit equation can be solved exactly (i.e., the iterative procedure is convergent). The following analysis concentrates on the stability of the Runge-Kutta/multigrid method that is used to solve the implicit equation; however, the section is concluded with a few comments that pertain to the stability of equation (6).

Fourier stability analysis is used to illustrate the effect of treating the physical time derivative implicitly instead of explicitly, as well as to illustrate other algorithmic choices. The analysis is performed for the Runge-Kutta method given by equation (12) with a scalar model equation of the form

$$\frac{\partial W}{\partial \tau} + \bar{\lambda}W = a \left( \frac{\partial W}{\partial x} - \frac{\varepsilon_4}{32} (\Delta x)^3 \frac{\partial^4 W}{\partial x^4} \right) \equiv S_c + S_d \quad (16)$$

The fourth derivative and its scaling closely model the numerical dissipation common to codes such as TLNS3D. Note that because the terms  $E$  and  $f$  are constant during the Runge-Kutta integration, they have no influence on the stability and have been dropped from the analysis. The particular version of Runge-Kutta used by TLNS3D and for this analysis is a five-stage method defined by  $\{\alpha_k\} = \{1/4, 1/6, 3/8, 1/2, 1\}$ . The convection terms are commonly treated differently from the dissipation with regard to the definition of the  $\bar{k} - 1$  index. In addition, the  $V^{\bar{k}-1}$  terms, which appear twice on the right-hand side of equation (12b), may be treated differently in each instance. In this stability analysis, the convective and dissipative terms are treated exactly as TLNS3D treats them in a steady-state case. These algorithmic choices are denoted by rewriting (12b) as

$$\begin{aligned} (1 + \gamma\alpha_k\bar{\lambda})V^k &= V^0 + \gamma\alpha_k\bar{\lambda}\widehat{V^{k-1}} + \alpha_k\Delta\tau L_{irs}^{-1} \bullet \left( S_c^{k-1} + \widetilde{S_d^{k-1}} - \bar{\lambda}\widehat{V^{k-1}} \right) \\ \widetilde{S_d^{k-1}} &= \beta_k S_d^{k-1} - (1 - \beta_k) S_d^{k-2} \\ \widetilde{S_d^0} &= 0 \end{aligned} \quad (17)$$

where  $\{\beta_k\} = \{1.0, 0.0, 0.56, 0.0, 0.44\}$ . The new parameter  $\gamma$  allows both implicit ( $\gamma = 1$ ) and explicit ( $\gamma = 0$ ) treatments of the physical time derivative to be studied. The choice of  $V^{k-1}$  and  $\widehat{V^{k-1}}$  is the subject of this analysis; however, the analysis will focus on forms that are easily implemented within the current version of TLNS3D. To this end, both  $V^{k-1}$  and  $\widehat{V^{k-1}}$  are defined in a manner similar to the dissipative terms with the use of  $\alpha_k$  and  $\sigma_k$ , respectively.

An evaluation of the spatial derivative with central-difference operators and the transformation to Fourier space gives

$$G^k = \frac{1 + \gamma\alpha_k\bar{\lambda}G_\alpha^{k-1} + \alpha_k \frac{[i\lambda \sin(\theta)G_\beta^{k-1} - \lambda\frac{1}{2}\varepsilon_4 \sin^4(\theta/2)G_\beta^{k-1} - \bar{\lambda}G_\sigma^{k-1}]}{1 + \varepsilon_2 \sin^2(\theta/2)}}{(1 + \gamma\alpha_k\bar{\lambda})} \quad (k = 1, 2, 3, 4, 5) \quad (18)$$

where

$$\begin{aligned} G_\beta^{k-1} &= \beta_k G^{k-1} - (1 - \beta_k)G_\beta^{k-2} \\ G_\alpha^{k-1} &= \alpha_k G^{k-1} - (1 - \alpha_k)G_\alpha^{k-2} \\ G_\sigma^{k-1} &= \sigma_k G^{k-1} - (1 - \sigma_k)G_\sigma^{k-2} \\ G^0 &= G_\beta^0 = G_\alpha^0 = G_\sigma^0 = 1 \\ G_\beta^{-1} &= G_\alpha^{-1} = G_\sigma^{-1} = 0 \end{aligned} \quad (19)$$

$\varepsilon_2$  is the coefficient of implicit-residual smoothing, and  $\lambda \equiv a\Delta\tau/\Delta x$  is the CFL number based on the pseudo-time step.

The influence of implicit versus explicit treatment of the physical time derivative is best illustrated by considering a simplified case of equation (18). Consider the case in which  $\varepsilon_2 = 0$  and  $\alpha_k = \sigma_k = \beta_k$ , for which equation (18) becomes

$$G^k = \frac{1 + \alpha_k \left\{ i\lambda \sin(\theta)G^{k-1} - [\lambda\frac{1}{2}\varepsilon_4 \sin^4(\theta/2) + \bar{\lambda}(1 - \gamma)]G_\beta^{k-1} \right\}}{(1 + \gamma\alpha_k\bar{\lambda})} \quad (k = 1, 2, 3, 4, 5) \quad (20)$$

Equation (20) clearly shows that an explicit treatment of the physical time derivative ( $\gamma = 0$ ) simply translates the stability region to the right as  $\bar{\lambda}$  increases from 0; an implicit treatment reduces the amplification factor, which expands the stability region. This difference is illustrated in figures 1a–c, which show equally spaced contours of the amplification factor  $\|G^5\|$  as a function of the real (dissipative) and imaginary (convective) parts of the spatial operator  $Z(\theta) = \lambda[i\sin(\theta) - \frac{1}{2}\varepsilon_4 \sin^4(\theta/2)]$ . Values of the contour lines are indicated by line types as indicated in the figure legend. Figure 1a shows the steady-state case  $\bar{\lambda} = 0$  as a point of reference. Figures 1b and 1c show the explicit and implicit cases, respectively, for  $\bar{\lambda} = 1$ . Each figure also shows the locus of the spatial operator for  $\lambda = 3$ . Other choices for  $\varepsilon_2$ ,  $\alpha_k$ , and  $\sigma_k$  give qualitatively similar results. An explicit treatment of the physical time derivative will always become unstable for sufficiently large values of  $\bar{\lambda}$ ; the implicit approach is stable for all values of  $\bar{\lambda}$ .

By plotting the amplification along the locus (figure 2), an unusual property of equation (16) is revealed. For  $\bar{\lambda} = 0$ , the amplification goes to 1 as  $\theta$  goes to 0, which is often considered a consistency condition. However, for  $\bar{\lambda} \neq 0$ , the solution is damped across the entire spectrum. This property is a consequence of the source term  $\bar{\lambda}W$ , which appears in both equations (8) and (16) and is not caused by an inconsistency in the derivative operators. The acceleration technique known as enthalpy damping makes use of the same property to improve the convergence of inviscid flows. This analysis suggests

that an implicit treatment of enthalpy damping may lead to further improvements; however, verification of this possibility is beyond the scope of this work.

Alternative choices for  $\alpha_k$  and  $\sigma_k$  are considered with the implicit-residual smoothing reinstated with a coefficient of 0.25. The parameters are tuned to obtain good high-frequency damping, but also to obtain a scheme for which  $\|G^5\|(\theta) > \|G^5\|(2\theta)$  whenever possible. The latter criterion will reduce the influence that aliasing error may have on the coarse-grid equation. Other obvious choices for  $\alpha_k$  and  $\sigma_k$  (in addition to  $\alpha_k = \sigma_k = \beta_k$  used above) are  $\{1.0, 0.0, 0.0, 0.0, 0.0\} \equiv S0$  and  $\{1.0, 1.0, 1.0, 1.0, 1.0\} \equiv S1$ . The first choice corresponds to the case in which  $\widehat{V^{k-1}}$  or  $\widehat{V^{k-1}} = V^0$ ; the second corresponds to the case in which  $\widehat{V^{k-1}}$  or  $\widehat{V^{k-1}} = V^{k-1}$ . Figure 3 shows the amplification for several implicit schemes, where, for example,  $\{S1, \beta_k\}$  denotes that  $\{\alpha_k\} = S1$  and  $\{\sigma_k\} = \{\beta_k\}$ . Although the case with  $\{S1, \beta_k\}$  has the lowest overall amplification, the case with  $\{S1, S1\}$  is more monotone in  $\theta$  and, as such, is the preferred method. Figure 4 shows the amplification of the latter case for a range of  $\bar{\lambda}$ .

So far, the discussion on stability has been limited to the behavior of the Runge-Kutta used to solve equation (6), which does not imply that equation (6) is stable. Equation (6) falls into the class of multistep schemes for which the usual notion of absolute stability is not sufficient to ensure convergence. Instead, the scheme must satisfy the more stringent conditions of *relative stability* [13] to ensure convergence to the proper solution. Equation (6) can be shown to be unconditionally stable when the time operator is approximated to either first or second order. Similarly, time operators of the form given in equation (6) for which  $M$  is also the order of the operator are unconditionally unstable for  $M > 5$ . Although conditionally stable methods would not normally be considered appropriate for large  $\Delta T$  calculations, the nature of the instability is such that even the conditionally stable methods are useful in many situations. A detailed study is beyond the scope of this work; however, the interested reader is referred to the cited reference.

## NUMERICAL RESULTS

To demonstrate the capability of the present method, the results of several numerical experiments are given. The first case that is examined is the solution of an impulsively accelerated flat plate, which is also known as Stokes first problem [14]. An analytic solution for incompressible flow is available for this problem, which allows comparison with solutions obtained numerically with global minimum time stepping (GMTS) and the present method (with variations in time-step size and in the temporal order of the numerical discretization).

The analytic solution of Stokes first problem [14] shows that the time-dependent solution collapses to a single solution of nondimensional velocity versus the similarity parameter  $\eta$  defined as

$$\eta = \frac{y}{2\sqrt{\nu T}} \quad (21)$$

where  $y$  is the direction normal to the flat plate,  $\nu$  is the kinematic viscosity, and  $T$  is the physical time. Figure 5a shows the analytic solution plotted in the similarity parameters. A solution calculated with GMTS after 2000 time steps is also plotted. Calculations were also performed with the present method with 1, 5, and 10 time steps to reach the same physical time as the GMTS solution. Different orders

of accuracy of the discretization of the physical time derivative were used, and the solutions are plotted in figures 5b–f.

Figure 5b shows the comparison of a first-order solution for the various time steps with the GMTS solution. The single time step does not accurately match the GMTS. As more time steps are used (smaller physical time steps), the comparison improves, although all show some error from  $\eta = 1.5$  to  $\eta = 2.0$ .

In figure 5c, comparisons of the present method with second-order physical time discretizations for the three time-step sizes are made with the GMTS. As expected, the smaller the time step, the better the agreement.

Comparisons for third-, fourth-, and fifth-order discretizations are shown in figures 5d–f, respectively. For the third-order solution, good agreement occurs for the two smaller time steps; however, this agreement degrades for the fourth- and fifth-order solutions. This degradation can be attributed to starting errors caused by the unavailability of information at previous time steps for the first few steps of the higher order schemes. This study demonstrates that third-order discretization agrees with the GMTS solution better than second-order discretization.

The work units required to obtain the solution of Stokes first problem at the same physical time as the GMTS solution after 2000 steps are shown in table 1 for a first- through a fifth-order physical time discretization. The single step solutions with the present method were performed in the least number of work units; however, the accuracy of the solution is unacceptably poor. For first- and second-order time discretizations, the ten-step solutions were obtained in fewer work units than the five-step solutions because of better convergence. (All work units in the table are based on converging the viscous drag at each time step to six significant digits.) Third- through fifth-order solutions required fewer work units for the five-step calculations than for the ten-step calculations. These results suggest that a balance exists between convergence speed at a given time step and the number of time steps used to obtain a solution with the present method. If the time step is too large, then the accuracy is poor. If the time step is too small, then unnecessary work is expended to converge at unneeded time steps.

Table 1. – Work units required to calculate solution for Stokes first problem.

Order of Physical Time Discretization	Number of Physical Time Steps		
	1	5	10
1st	54.6	210.9	193.8
2nd	51.2	182.4	171.0
3rd	42.1	159.5	171.0
4th	15.8	165.3	193.8
5th	15.8	153.8	171.0
GMTS	2000.0		

The second test case used to demonstrate the present method is the unsteady flow over an impulsively started two-dimensional circular cylinder (with a Reynolds number of 1200 and a Mach number of 0.3). Detailed experimental and numerical investigations of the flow behind a cylinder have been performed previously by other authors. (See, for example, reference [15].) The initial flow is symmetric with zero

lift as the wake behind the cylinder begins to grow. As the wake continues to grow, it becomes unstable and begins to shed from alternate sides of the cylinder.

An examination is made of the first part of the solution where the symmetric wake begins to grow. In these calculations, 4600 GMTS steps were used to reach  $t^* = 2.4$ . Detailed comparisons of the first-through fourth-order solutions with both the present method and GMTS for  $t^* = 0$  to  $t^* = 2.4$  are shown in figure 6. The calculations were performed with the present scheme with 10, 20, and 40 steps to reach  $t^* = 2.4$ . As with Stokes first problem, the first-order discretization does not give satisfactory results (figure 6a). Second-order differencing (figure 6b) shows much better agreement with the GMTS solution. Third-order differencing shows good agreement for the two smaller time-step sizes (figure 6c). Fourth-order differencing gives bad overshoots for the 10-step calculation and instability for the 20- and 40-step calculations (figure 6d).

The present scheme was then used to calculate the flow around the cylinder out to times where the vortex shedding occurred. Time histories of the lift coefficient  $C_l$  and the drag coefficient based on integrated pressures  $C_{d_p}$  are shown in figure 7. From experimental data and the results of the GMTS calculations shown in reference [15], the period of the oscillation of  $C_{d_p}$  is known to be approximately 4 in terms of the nondimensional time  $t^*$ . To give 40 time steps per period, a time step of  $\Delta t^* = 0.1$  was used. This time-step size is roughly equal to the time step used in the 20-step calculations shown in figure 6. The first-order discretization predicted a Strouhal number of 0.21. The second- and third-order discretizations predicted a Strouhal number of 0.24 compared with the experimentally obtained value of 0.21. The fourth-order physical time discretization calculation diverged.

## CONCLUSIONS

A method to accurately calculate time-accurate solutions to the unsteady Navier-Stokes equations has been presented. Multigrid acceleration has been successfully employed to accelerate the calculations of the *iterative-implicit* method. Run times that are one order of magnitude smaller than the run times required for global minimum time stepping have been demonstrated.

# FIGURES

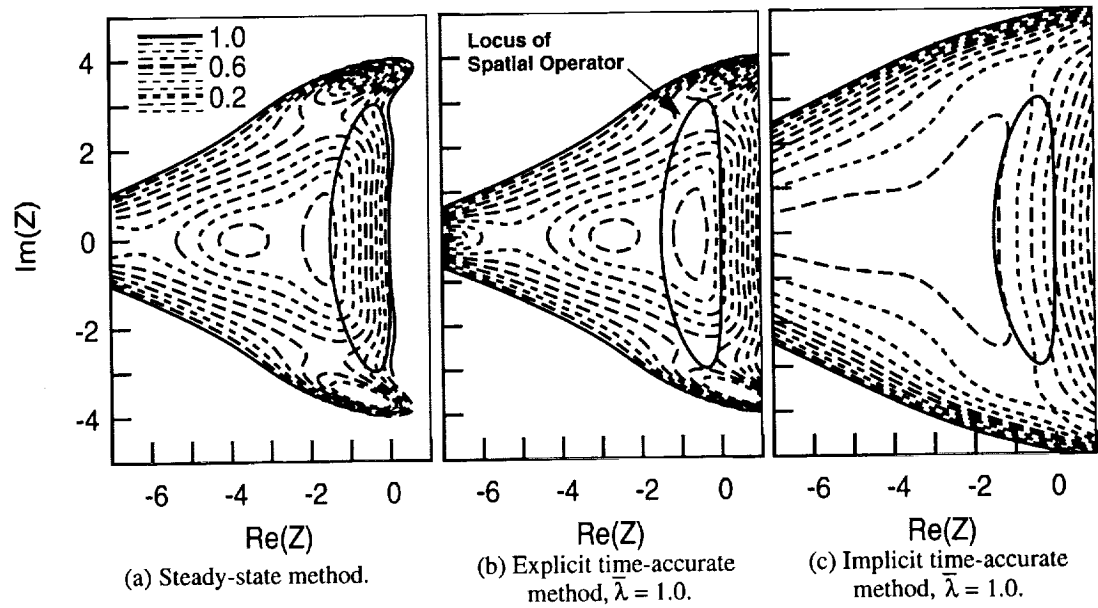


Figure 1. Contours of stability region.

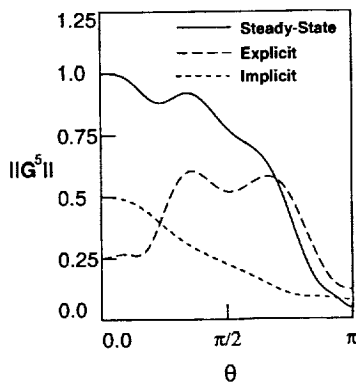


Figure 2. Amplification along the locus of the spatial operator.

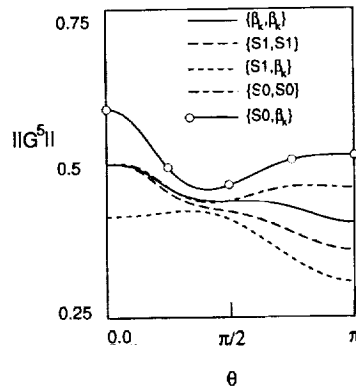


Figure 3. Amplification of implicit schemes with  $\bar{\lambda} = 1.0$ .

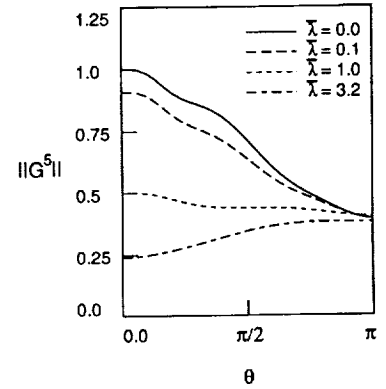


Figure 4. Amplification of  $\{S_1, S_1\}$  case for a range of  $\bar{\lambda}$ .

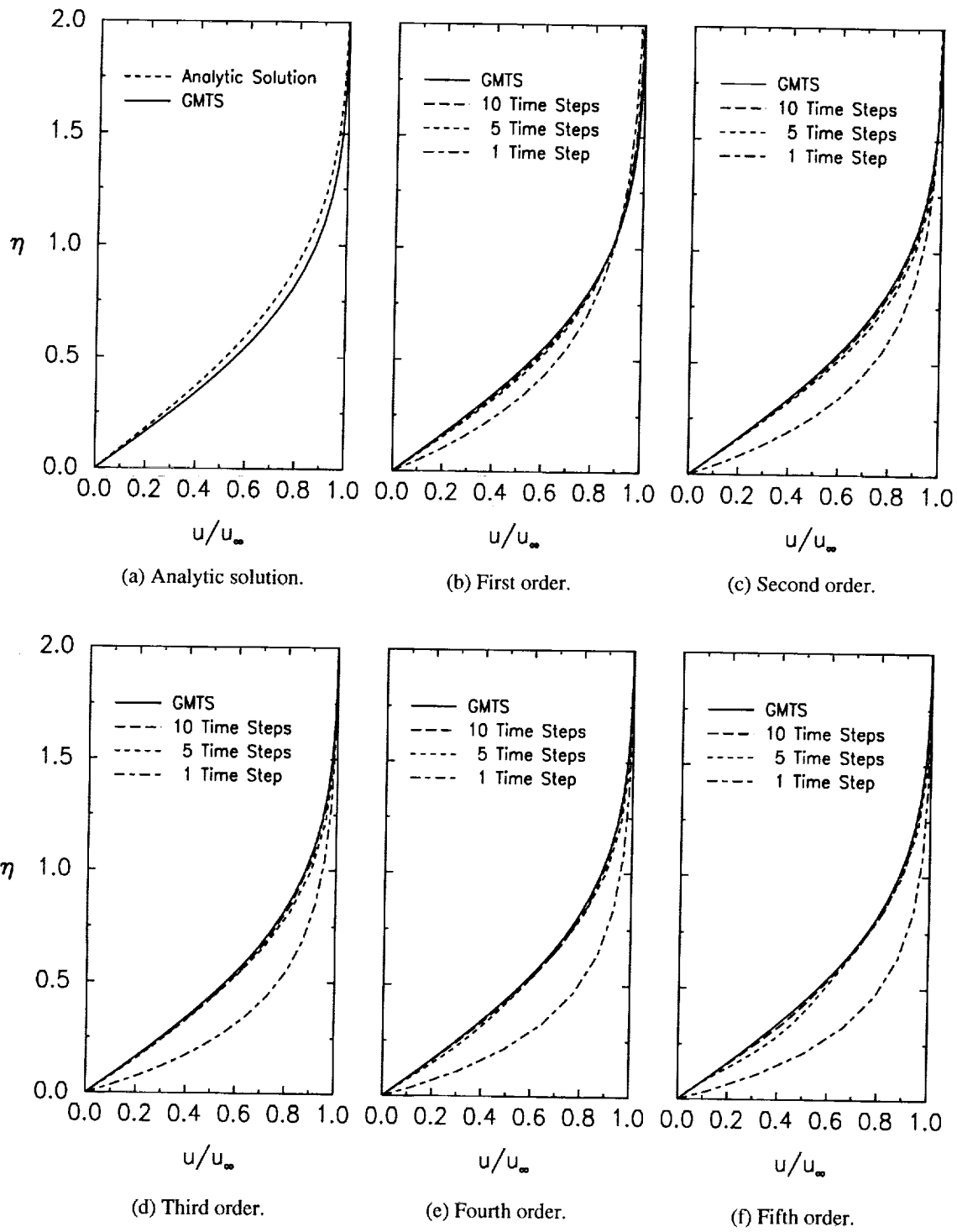


Figure 5. Velocity profiles for impulsively started flat plate.



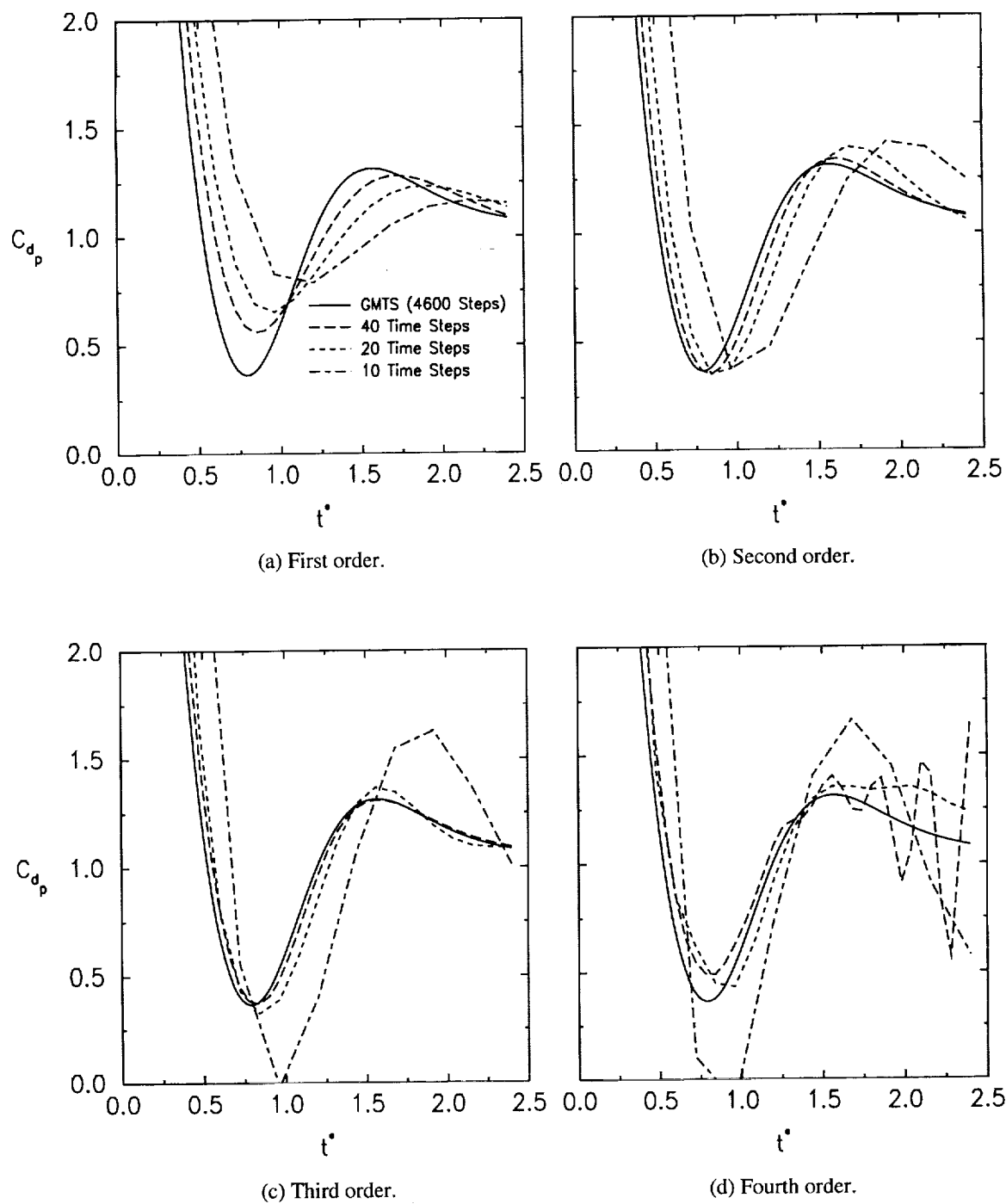
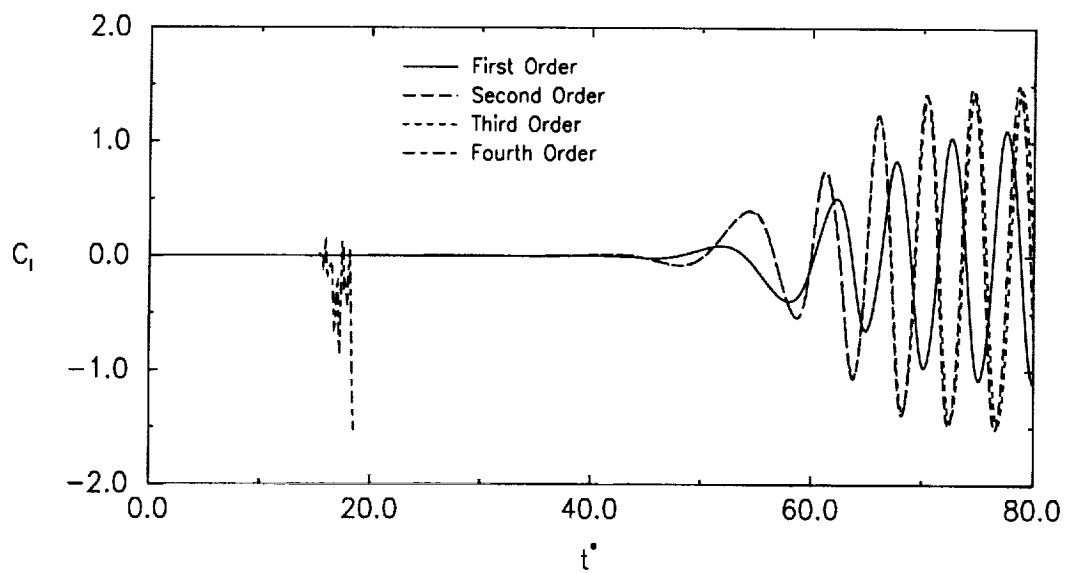
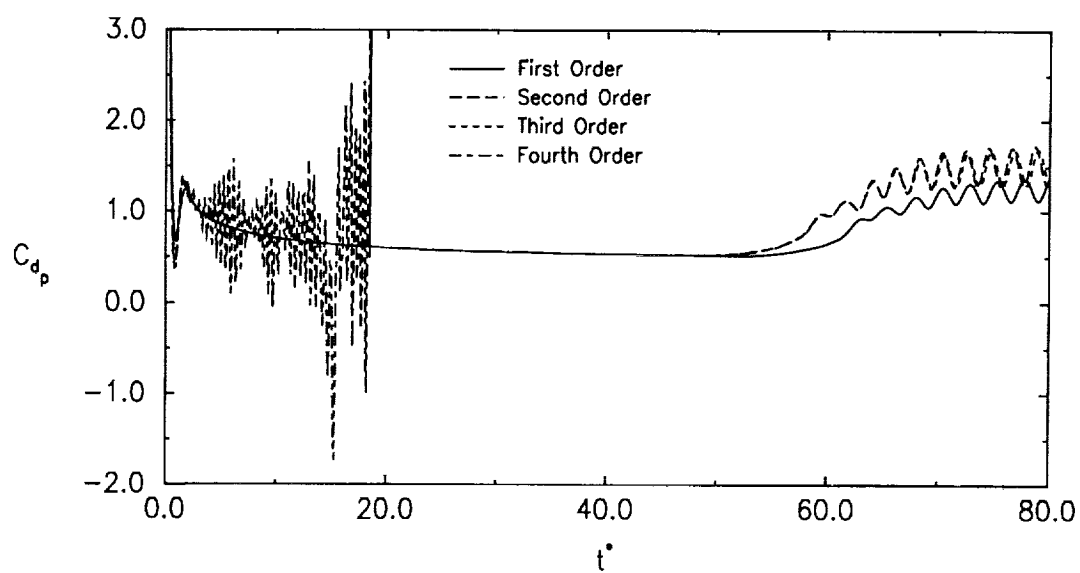


Figure 6. Initial pressure drag-coefficient history for impulsively started cylinder.



(a) Lift history.



(b) Pressure drag history.

Figure 7. Lift- and drag-coefficient histories for impulsively started cylinder.

## REFERENCES

- [1] V. N. Vatsa and B. W. Wedan. Development of an Efficient Multigrid Code for 3-D Navier-Stokes Equations and its Application to a Grid-Refinement Study. *Computers and Fluids*, 18(4):391-403, 1990.
- [2] F. Ghaffari, J. M. Luckring, J. L. Thomas, and B. L. Bates. Navier-Stokes Solutions About the F/A-18 Forebody-Leading-Edge Extension Configuration. *Journal of Aircraft*, 27:737-748, 1990.
- [3] H. L. Atkins. A Multi-Block Multigrid Method for the Solution of the Euler and Navier-Stokes Equations for the Three-Dimensional Flows. AIAA Paper 91-0101, 1991.
- [4] A. Jameson. Time Dependent Calculations Using Multigrid, with Applications to Unsteady Flows Past Airfoils and Wings. AIAA Paper 91-1596, 1991.
- [5] III Arthur C. Taylor, Wing-Fai Ng, and Robert W. Walters. Upwind Relaxation Methods for the Navier-Stokes Equations Using Inner Iterations. *Journal of Computational Physics*, 99:68-72, 1992.
- [6] B. S. Baldwin and H. Lomax. Thin Layer Approximation and Algebraic Model for Separated Turbulent Flows. AIAA Paper 78-257, 1978.
- [7] A. Jameson, W. Schmidt, and E. Turkel. Numerical Solutions of the Euler Equations by Finite Volume Methods Using Runge-Kutta Time-Stepping Schemes. AIAA Paper 81-1259, 1981.
- [8] A. Jameson and T. J. Baker. Solutions of the Euler Equations for Complex Configurations. AIAA Paper 83-1929, 1983.
- [9] E. Turkel and V. N. Vatsa. Effect of Artificial Viscosity of Three Dimensional Flow Solutions. AIAA Paper 90-1444, 1990.
- [10] L. Martinelli. *Calculation of Viscous Flows with Multigrid Methods*. PhD thesis, MAE Dept., Princeton University, 1987.
- [11] R. C. Swanson and E. Turkel. Artificial Dissipation and Central Difference Schemes for the Euler and Navier-Stokes Equations. AIAA Paper 87-1107, 1987.
- [12] V. N. Vatsa, M. D. Sanetrik, and E. B. Parlette. Development of a Flexible and Efficient Multigrid-Based Multiblock Flow Solver. AIAA Paper 93-0677, 1993.
- [13] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations*. Prentice-Hall, Inc., 1971.
- [14] Hermann Schlichting. *Boundary Layer Theory, Seventh Edition*. McGraw-Hill Book Company, 1979.
- [15] Christopher L. Rumsey. Details of the Computed Flowfield Over A Circular Cylinder At Reynolds Number 1200. Presented at the Forum on Unsteady Flow Separation, Cincinnati, Ohio, June 15-17, 1987.



MGGHAT: ELLIPTIC PDE SOFTWARE WITH ADAPTIVE REFINEMENT,  
MULTIGRID AND HIGH ORDER FINITE ELEMENTS

William F. Mitchell  
GE Advanced Tech Labs  
Moorestown, NJ 08041

56-64  
197566  
P-10

SUMMARY

MGGHAT (MultiGrid Galerkin Hierarchical Adaptive Triangles) is a program for the solution of linear second order elliptic partial differential equations in two dimensional polygonal domains. This program is now available for public use. It is a finite element method with linear, quadratic or cubic elements over triangles. The adaptive refinement via newest vertex bisection and the multigrid iteration are both based on a hierarchical basis formulation. Visualization is available at run time through an X Window display, and a posteriori through output files that can be used as GNPLOT input. In this paper, we describe the methods used by MGGHAT, define the problem domain for which it is appropriate, illustrate use of the program, show numerical and graphical examples, and explain how to obtain the software.

INTRODUCTION

MGGHAT (MultiGrid Galerkin Hierarchical Adaptive Triangles) is a program for the solution of linear second order elliptic partial differential equations in two dimensional polygonal domains. It solves equations of the form:

$$(pu_x)_x + (qu_y)_y + ru = f \quad \text{in } \Omega$$

$$u = g \quad \text{on } \partial\Omega_1$$

$$\frac{\partial u}{\partial n} + cu = g \quad \text{on } \partial\Omega_2$$

where  $\Omega$  is a polygonal domain in  $\mathbb{R}^2$  and  $p, q, r, f, c$ , and  $g$  are functions of  $x$  and  $y$ , and  $n$  is the unit normal direction.

MGGHAT uses a finite element method with linear, quadratic or cubic elements over triangles. The adaptive refinement via newest vertex bisection and the multigrid iteration are both based on a hierarchical basis formulation. Visualization is available at run time through an X Window display, and for post-run analysis through output files that can be used as GNPLOT input. The program is now available in the public domain through mgnet and netlib.

NUMERICAL METHOD

The numerical method used by MGGHAT is a finite element method with adaptive refinement of the grid and a multigrid solution of the equations. In this section we briefly describe the method used. More details of the method can be found in [1], and a full description and analysis in [2], which is contained in the MGGHAT software package.

## Discretization

MGGHAT solves elliptic differential equations using the standard Galerkin finite element method. A triangular mesh is used over the 2D domain. The basis functions are  $C^1$  continuous piecewise polynomials of any specified degree. Currently, the program only handles linear, quadratic and cubic polynomials, but can be modified to handle higher order polynomials by defining a quadrature rule of the appropriate accuracy.

## Adaptive refinement

The program provides automatic adaptive refinement of the grid to ensure the highest accuracy for the number of nodes used. The refinement of triangles is performed using the newest vertex bisection method. This method divides pairs of triangles through the midpoint of their common edge, which is equivalent to enhancing the approximation space by one hierarchical basis function (in the linear case). The error estimate, used to determine which triangles should be divided, is based on an estimate of the coefficient of the new hierarchical basis function.

## Solution

The equations are solved using a hierarchical basis multigrid method. The relaxation phase consists of red-black Gauss-Seidel iterations on the nodal basis equations. The number of iterations can be user specified, but usually a red phase before coarse grid correction and a red and black phase after coarse grid correction suffices for optimal convergence rates. The grid transfers are a natural consequence of the transformation between the nodal and hierarchical bases, and can be shown to lead to a method equivalent to the "Galerkin" multigrid method in simple cases.

## MGGHAT SOFTWARE

MGGHAT is written in standard FORTRAN 77, and is callable as a subroutine. An example *main* program for MGGHAT is shown in Figure 1. The program has been tested on 3 computer configurations: 1) a Pyramid computer using the f77 compiler under a dual port of UNIX SysV Release 2.0, 2) a Sun workstation using the f77 compiler under SunOS 4.1.1, and 3) an i486 based PC using the f2c translator and gcc compiler under the Linux operating system. The program is easily installed with the makefile provided in the distribution, and requires only a FORTRAN compiler for the basic functionality. A C compiler is required for the UNIX dependent supplied timer routine (which can be replaced by the user). A C compiler and X Window libraries are required for the (optional) X Window graphics capability.

## Problem Definition

The differential equation, boundary conditions and domain are defined by user supplied subroutines. Figure 2 contains examples of these routines. The subroutine *pde* defines the equation by providing the value of the functions  $p$ ,  $q$ ,  $r$  and  $f$  at any point  $(x,y)$ . Subroutine *bcond* contains the boundary conditions. The boundary is partitioned into a set of pieces in the initial

triangulation. The piece containing the point  $(x,y)$  is passed to *bcond* through *ipiece*. *bcond* returns the functions *c* and *g* and sets *itype* to flag the boundary condition as Dirichlet or Mixed (including Neuman if  $c=0$ ). If the true solution is known, the user can supply functions *true*, *true<sub>x</sub>*, and *true<sub>y</sub>* to obtain error calculations. The initial triangulation (coarse grid) is defined by the user in subroutine *inittr* (not shown).

## Parameters

The user has control over the program through several parameters.

*mxvert*, *mxtri*, *mxlev*, *mxnode* and *mxtime*: maximum values for the number of vertices, triangles, refinement levels, nodes and execution time can be used as termination criteria.

*tol*: an error tolerance that can be used as a termination criterion.

*outlev*: controls the amount of printed output. Can be 0 for no output, 1 for summary at the end of execution, 2 for summary after each program phase, 3 for detailed information, and 4 and 5 for debugging level output. An extraction from a level 2 output is illustrated in Figure 3.

*iorder*: specifies the order (degree+1) of the piecewise polynomial basis functions.

*nu1* and *nu2*: number of (half) red-black Gauss-Seidel iterations to perform before and after coarse grid correction, respectively.

*ncyc*: number of multigrid cycles to perform in each solution phase.

*unifrm*: a logical variable to indicate a uniform refinement should be used rather than adaptive refinement.

## Graphics

Graphics support is provided in two forms: run time graphics on an X Window display, and output files suitable for input to GNUPLOT. The run time graphics use a small set of routines which call on the X Window graphics library. The user can expand this to support other graphics devices by writing equivalent routines (draw a point, draw a line, print some text, etc.) for the desired device. There are nine forms of run time graphics:

- 1) contour plot of computed solution with triangulation
- 2) contour plot of true solution with triangulation
- 3) contour plot of error with triangulation
- 4) color plot of computed solution
- 5) color plot of true solution
- 6) color plot of error
- 7) triangulation
- 8) graph of number of nodes vs. relative error in energy norm (or error estimate)
- 9) contour plot of both computed solution and true solution

Either one or two of these forms can be displayed during one run. When two are displayed, additional numerical information is printed on the display, including grid size information, norms of the error and error estimate, and execution time. Figure 4 contains an example of the run time graphic displays.

The user can select to save information in data files for later processing by GNUPLOT. These files contain the triangulation, computed and true solutions, and convergence data. Figures 5 and 6 contain plots generated by GNUPLOT.

## OBTAINING MGGHAT

MGGHAT is now available in the public domain. It can be obtained either from mgnet or netlib.

### mgnet

To obtain MGGHAT from mgnet (the multigrid network) ftp to casper.cs.yale.edu. Login as *anonymous* and use your email address as the password. Change to the mgghat directory by typing *cd mgnet/mgghat*. Then type *ls* to see what files are available, and *get filename* for each file you desire. To learn more about mgnet, also get the file *mgnet.README* from the *mgnet* directory.

### netlib

MGGHAT can be obtained from netlib using ftp, the mail server, or *xnetlib*. For ftp retrieval, ftp to research.att.com and follow the anonymous login procedure described above. Look for MGGHAT in the directory netlib/pdes/mgghat. To obtain MGGHAT via email, send a message to netlib@oml.gov, netlib@research.att.com, or one of the other netlib servers with the message *send index from pdes/mgghat*. To learn how to obtain materials from netlib through an X Window interface, send the message *send index from xnetlib* to one of the netlib mail servers. For more information on netlib, send the message *send index* to one of the netlib mail servers.

## REFERENCES

1. Mitchell, W. F.: Optimal Multilevel Iterative Methods for Adaptive Grids. *SIAM J. Sci. Statist. Comput.*, vol. 13, 1992, pp. 146-167.
2. Mitchell, W. F.: Unified Multilevel Adaptive Finite Element Methods for Elliptic Problems. Ph.D. thesis, Technical report UIUCDCS-R-88-1436, Department of Computer Science, University of Illinois, Urbana, IL, 1988.



```

program main
include 'commons'      ! all parameters are passed through common
c
c set maximum allowed values based on dimensions
c
    mxvert = ndvert
    mxtri  = ndtri
    mxlev  = ndlev
    mxnode = ndnode
c
c set program parameters
c
    mxtime = 12.*60.*60. ! maximum execution time in seconds
    ioutpt = 6           ! unit for printed output
    outlev = 2           ! amount (level) of printed output
    iorder = 2           ! polynomial order (linear in this case)
    nu1    = 1           ! number of relaxation iterations before
    nu2    = 2           ! and after coarse grid correction
    ncyc   = 1           ! number of multigrid cycles
    tol    = 0.001       ! error tolerance for termination
    mgfreq = 2.          ! how often to do multigrid cycle
    unifrml = .false.    ! flag for uniform/adaptive grid
    igrf1 = 0            ! run time graphics selections (no
    igrf2 = 0            ! graphics in this example)
    grf1st = 0.          ! a value for which a contour line is drawn
    grfsiz = .1          ! and the spacing between contours
    grffmn = 0.          ! bounds for determining the color
    grffmx = 2.          ! map for color contour plots
    gptri = 0            ! set to 1 to save triangulation for gnuplot
    gpsol = 0            ! set positive to save solution for gnuplot
    gpconv = 0           ! set to 1 to save convergence info for gnuplot
c
    call mgghat          ! invoke mgghat
    stop
end

```

Figure 1. Sample main program.

```

subroutine pde(x,y,p,q,r,f)
real x,y,p,q,r,f
c
c return the values of the pde coefficients at (x,y)
c
c  $-(p(x,y) * u_x) - (q(x,y) * u_y) + r(x,y) * u = f(x,y)$ 
c
  p=1.
  q=1.
  r = 0.
  f=-20.*(x**3 + y**3))
  return
end

c
subroutine bcond(x,y,ipiece,c,g,itype)
real x,y,c,g
integer ipiece,itype
c
c returns boundary condition coefficients at (x,y)
c
c  $u_x + c(x,y)*u = g(x,y)$  or  $u = g(x,y)$ 
c n
c In this example, the b.c. is Dirichlet on piece 1, and 0 Neuman on piece 2
c
  if (ipiece.eq.1) then
    itype = 1
    c = 0.
    g = true(x,y)
  else
    itype = 2
    c = 0.
    g = 0.
  endif
  return
end

c
real function true(x,y)      ! true solution of the pde
real x,y
true = x**5 + y**5
return
end

```

Figure 2. Examples of subroutines to define the problem.

# MULTIGRID GALERKIN HIERARCHICAL ADAPTIVE TRIANGLES (MGGHAT)

Version 0.9 (March 1993)

```
input parameters:
output level      2
polynomial order  2
number of cycles  1
relaxes before cgc 1
relaxes after cgc 2
multigrid frequency 2.00
error tolerance   0.0E+00
refinement        adaptive

begin initialization
initializations complete
time for initialization      .00

begin refinement
refinement complete

number of vertices  18
number of nodes     18
number of triangles 22
number of levels     3
time for refinement (this grid) .02
time for refinement (all grids) .02

begin solution
solution complete

norms of error:
max norm at vertices 1.20466471E-01
max norm at nodes    1.20466471E-01
max norm at quad pts 2.12660193E-01
continuous energy norm 3.30431342E-01
relative energy norm  1.49259701E-01

time for solution (this grid) .01
time for solution (all grids) .01
```

Figure 3. Sample level 2 output.

```

begin error indicators
error indicators and estimates complete

maximum error indicator 1.99157119E-01
error estimate          4.55372840E-01
effectivity index       1.37811637E+00
relative error estimate 1.96938977E-01
relative effect index   1.31943834E+00

time for error estimates (this grid)      .00
time for error estimates (all grids)      .00

time for this refinement/solution step    .03
total time so far                        .03

final solution complete

maximum error at vertices 7.39555359E-02
maximum error at nodes   7.39555359E-02
maximum error at quad pts 1.25789344E-01
continuous energy norm   2.70688415E-01
maximum error indicator   1.41879827E-01
error estimate            4.30013269E-01
effectivity index         1.58859134E+00
relative energy norm      1.87541485E-01
relative effect index     1.58822513E+00

number of vertices        32
number of nodes           32
number of triangles       45
number of levels          5

time for initializations  .00
time for refinement       .08
time for solution         .02
time for error estimates  .00
total time                .10

termination due to achieving maximum nodes
execution successful

```

Figure 3. Sample level 2 output (continued).

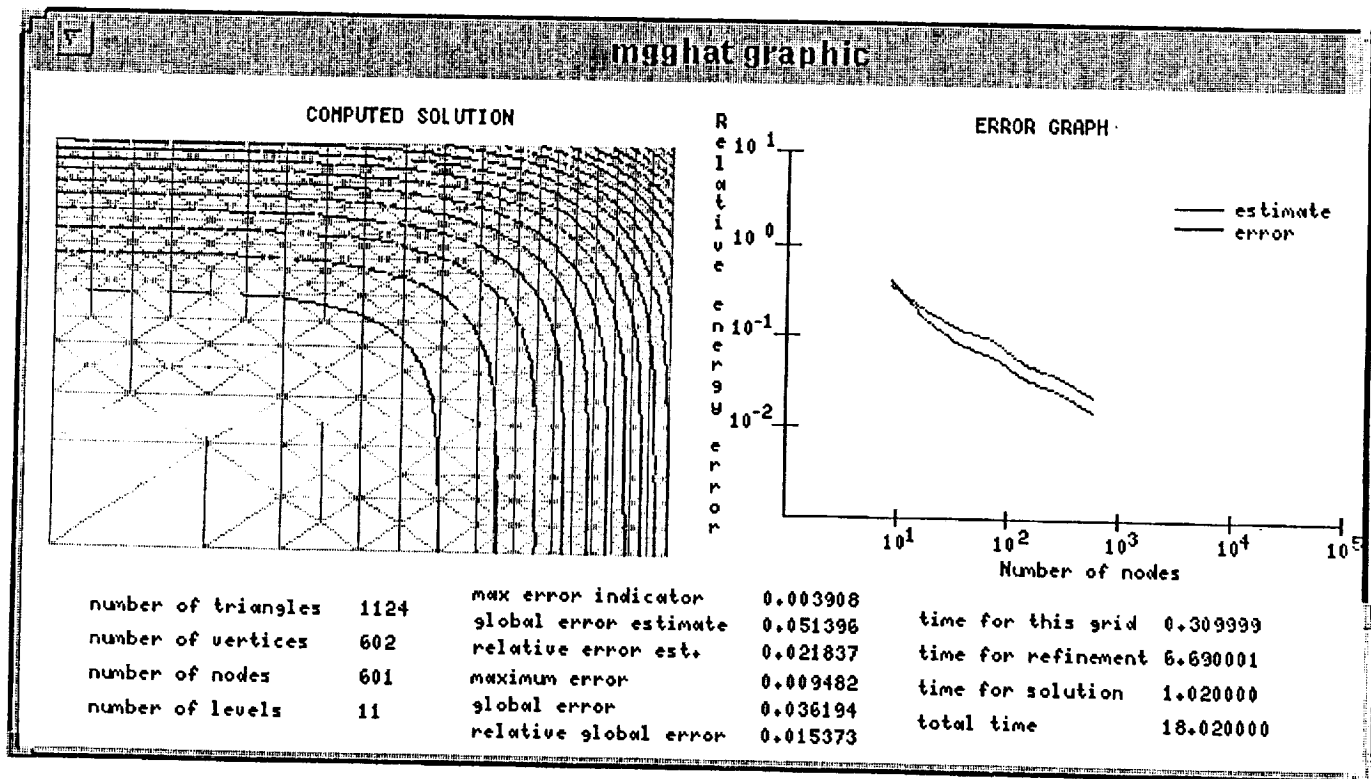


Figure 4. Sample run time graphics.

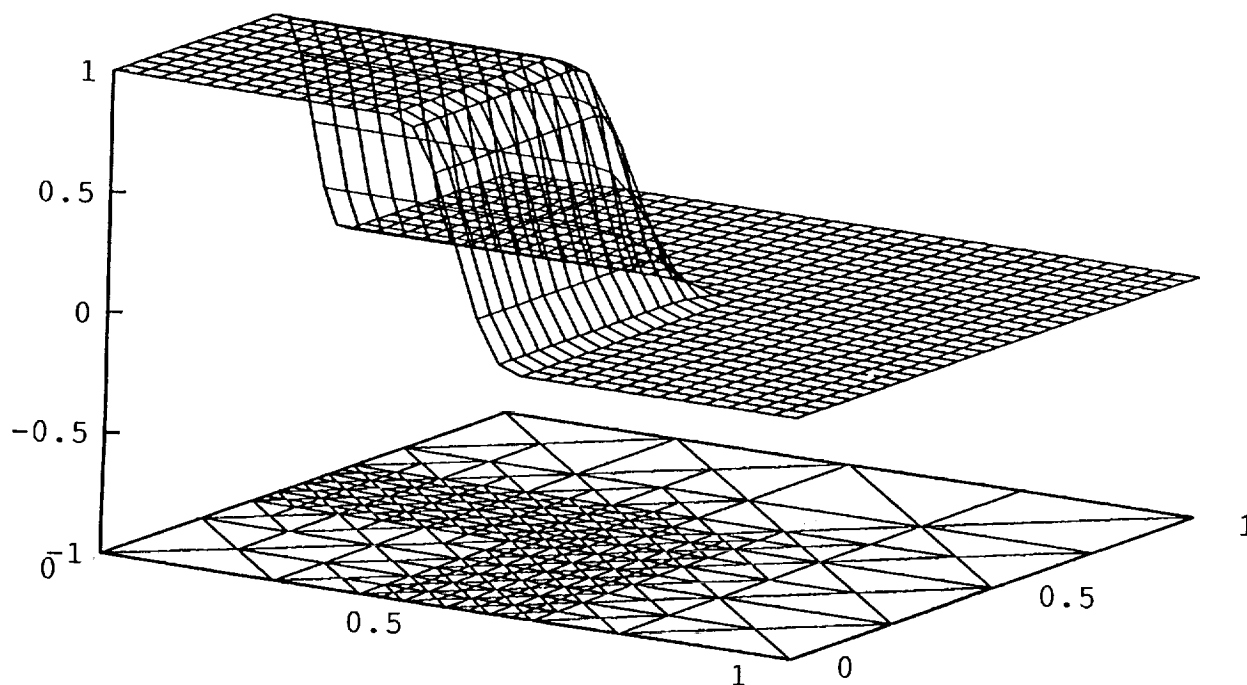


Figure 5. gnuplot plot of triangulation and solution.

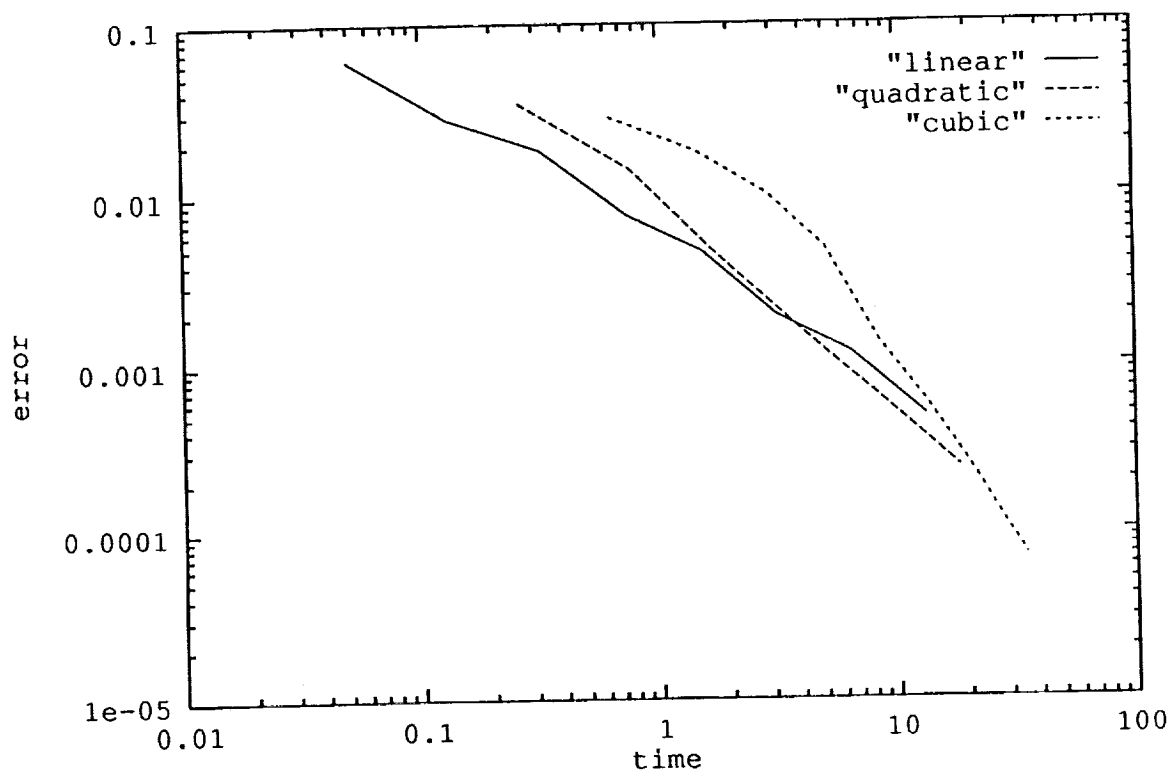


Figure 6. Convergence plot for 3 runs using gnuplot.

# LOOKING FOR $O(N)$ NAVIER-STOKES SOLUTIONS ON NON-STRUCTURED MESHES <sup>1</sup>

Eric MORANO <sup>2</sup>  
ICASE, NASA Langley Research Center  
Hampton, VA, USA

Alain DERVIEUX  
INRIA  
BP 93, 06902 Sophia Antipolis Cedex, France

57-34  
197567  
P-15

## SUMMARY

Multigrid methods are good candidates for the resolution of the system arising in Numerical Fluid Dynamics. However, the question is to know if those algorithms which are efficient for the Poisson equation on structured meshes will still apply well to the Euler and Navier-Stokes equations on unstructured meshes. The study of elliptic problems leads us to define the conditions where a Full Multigrid strategy has  $O(N)$  complexity. The aim of this paper is to build a comparison between the elliptic theory and practical CFD problems.

First, as an introduction, we will recall some basic definitions and theorems applied to a model problem. The goal of this section is to point out the different properties that we need to produce an FMG algorithm with  $O(N)$  complexity. Then, we will show how we can apply this theory to the fluid dynamics equations such as Euler and Navier-Stokes equations. At last, we present some results which are 2nd-order accurate and some explanations about the behaviour of the FMG process.

## INTRODUCTION

One first important element is the mesh independent convergence speed. Hackbush, in [1] for example, proposes a demonstration of this property. It is done in the special case of an elliptic problem on structured nested meshes. We want to evaluate the properties that we must keep in order to get the mesh independent convergence speed when we use unstructured non-embedded meshes.

The problem to be solved is the following:

$$\begin{cases} \Delta u = f \text{ on } \Omega \text{ convex polygonal domain} \\ u|_{\partial\Omega} = 0 \end{cases} \quad (1)$$

$$u \in H_1^0(\Omega) \text{ and } f \in L^2(\Omega)$$

The discretization is a usual linear P1-Galerkin finite element. Thus, we get a discrete space  $\mathcal{H}_h$  whose dimension is equal to  $N_h$  (number of nodes), and where the subscript  $h$  indicates the mesh

<sup>1</sup>Work partly supported by DRET Groupe 6 under contract.

<sup>2</sup>Supported by INRIA and "Région Provence-Alpes-Côte d'Azur" (France), and ICASE (USA).

size. The resulting problem consists now in solving the linear system:

$$A_h u_h = f_h \quad (2)$$

We may evaluate the discretization error thanks to the Aubin-Nitsche's theorem and usual regularity:

$$\|u - u_h\|_{L^2} \leq C_2 h^2 \|u\|_{H^2} \leq C_3 h^2 \|f\|_{L^2} \quad (3)$$

The linear system (2) may incur a lot of CPU time because of its size (large number of nodes). The idea is then to use a second finite element subspace  $\mathcal{H}_H$  whose dimension  $N_H$  is less than the previous one (usually  $H = 2h$ ). We have then the following relationship between both spaces (also called grids):

$$\begin{array}{ccc} & & A_H^{-1} \\ & \mathbb{R}^{N_H} & \longrightarrow \mathbb{R}^{N_H} \\ R \uparrow & & \downarrow P \\ & \mathbb{R}^{N_h} & \longrightarrow \mathbb{R}^{N_h} \\ & & A_h^{-1} \end{array}$$

where  $P$  and  $R$  are linear interpolations (transfer operators). The iterative process can be written as:

$$u_h^{n+1} = M_h u_h^n + N_h f_h \text{ where: } M_h = S_h^{\nu_2} (I - P A_H^{-1} R A_h) S_h^{\nu_1}, \quad S_h = I - \omega D_h^{-1} A_h$$

( $S$  defines the basic iterative smoother,  $\nu_1$  and  $\nu_2$  the number of pre- and post-relaxations). Such a process converges if  $\|M_h\| < 1$ . A very important property of this kind of method is that the convergence is independent of the mesh size. In order to simplify the notations (and the study) we rewrite  $M_h$  as the following ideal-2-grid operator [2]:

$$M_h = (A_h^{-1} - P A_H^{-1} R)(A_h S_h^{\nu}) \quad (4)$$

The norms of both factors of the right hand side of the equation (4) will determine the norm of  $M_h$ :

- The smoothing property:

$$\begin{aligned} \|A_h S_h^{\nu}\| &\leq 1/h^2 \eta(\nu) \\ \lim_{\nu \rightarrow \infty} \eta(\nu) &= 0 \end{aligned} \quad (5)$$

depends a lot on the basic smoothing process, and, we will not give any details.

- The approximation property is:

$$\|A_h^{-1} - P A_H^{-1} R\| = O(h^2) \quad (6)$$

Let us focus on (6): it takes into account the transfer operators, and overall, represents the difference that exists between the solution on the fine grid and the solution on the coarse grid.

An MG scheme that exhibits these properties will result in a convergence speed that is independent of the mesh size:

$$\forall \rho \exists \nu(\rho) \text{ such that } \|M_h v_h - u_h\| \leq \rho \|v_h - u_h\|$$

We may notice that demonstrating the approximation property leads to the evaluation of the following quantity:

$$\|p_h P A_H^{-1} R r_h - A^{-1}\|$$



For nested meshes, thanks to (3), one can easily derive this from the following equality:

$$p_h P = p_H$$

On the other hand, for unnested meshes,  $p_h P$  is not equal to  $p_H$  and this evaluation is more difficult. Actually, it is the same as evaluating the difference between two interpolated solutions. Zhang [3], thanks to Bank-Dupont's theory, proposes such an evaluation.

*Remark:* The multigrid iterative V-cycle algorithm can easily be deduced from the previous 2-grid algorithm recursively. It maintains the convergence speed independently of the mesh size and has an  $O(N \log N)$  complexity.

In order to apply the previous result of convergence, we propose to use a Full Multigrid (FMG) strategy (proposed by Brandt in [4]). A well known result is the one given by Hackbush in [1], which is given below:

*Theorem:* We note:  $\kappa$  the consistency order,  $C_2 = \max_{1 \leq k \leq l} (h_{k-1}/h_k)^\kappa$  the ratio of accuracy between two solutions,  $S$  the reduction factor of the MG process,  $i$  the number of MG iterations applied to reach the solution  $\tilde{u}_k^i$ . If  $u_k$  is the solution of the discrete problem, we have:

$$\|\tilde{u}_k^i - u_k\| \leq C_3^i C_1 h_k^\kappa \quad \text{with} \quad C_3^i = \frac{S^i}{1 - C_2 S^i}.$$

Assuming that  $C_2 = 2^\kappa$ , we deduce the exact number of cycles in each FMG phase to solve the 1st-order problem ( $S^i \leq 1/4$ ) and the 2nd-order problem ( $S^i \leq 1/8$ ). The number of cycles  $i$  in each phase is constant, which leads to an algorithm that has  $O(N)$  complexity.

Once again, the relative interpolation error [1] conditions the quality of the initialization in each phase. Thus, in order to stay close to the ideal scheme (where the different subspaces are nested), we propose to build meshes where:

- The mesh size ratio is close to 2,
- The triangles aspect ratio is locally comparable in the whole domain.

We have thus identified the different necessary ingredients to build an algorithm having  $O(N)$  complexity:

1. A sequence of grids,
2. A basic smoother (ex: Jacobi, Gauss-Seidel),
3. Intergrid transfer operators (ex: linear interpolations),
4. MG algorithm (ex: V-cycle, W-cycle),
5. FMG strategy.

We may now apply it to more complex fluid dynamics problems such as the resolution of the Navier-Stokes/Euler equations.

We recall first the formulation of the steady Navier-Stokes equations:

$$\begin{aligned} \frac{\partial F(W)}{\partial x} + \frac{\partial G(W)}{\partial y} &= \frac{1}{Re} \left( \frac{\partial R(W)}{\partial x} + \frac{\partial S(W)}{\partial y} \right), \quad W = (\rho, \rho u, \rho v, E)^T, \\ F(W) &= \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (E + p)u \end{pmatrix}, \quad G(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (E + p)v \end{pmatrix}, \quad \mathbb{F}(W) = \begin{pmatrix} F(W) \\ G(W) \end{pmatrix}, \\ R(W) &= \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ u\tau_{xx} + v\tau_{xy} + \frac{\gamma\kappa}{Pr} \frac{\partial e}{\partial x} \end{pmatrix}, \quad S(W) = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ u\tau_{xy} + v\tau_{yy} + \frac{\gamma\kappa}{Pr} \frac{\partial e}{\partial y} \end{pmatrix}, \\ p &= (\gamma - 1) \left( E - \frac{1}{2} \rho (u^2 + v^2) \right), \quad e = C_v T = \frac{E}{\rho} - \frac{1}{2} (u^2 + v^2), \end{aligned} \quad (7)$$

where  $\gamma = 1.4$  is the ratio of specific heats,  $T$  is the temperature,  $\mu$  and  $\kappa$  are the normalized viscosity and thermal conductivity coefficients. The components of the Cauchy stress tensor  $\tau_{xx}$ ,  $\tau_{xy}$  and  $\tau_{yy}$  are given by:

$$\tau_{xx} = \frac{2}{3}\mu \left( 2\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} \right), \quad \tau_{yy} = \frac{2}{3}\mu \left( 2\frac{\partial v}{\partial y} - \frac{\partial u}{\partial x} \right), \quad \tau_{xy} = \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)$$

$Re = \rho_0 U_0 L_0 / \mu_0$  is the Reynolds number and  $Pr = \mu_0 C_p / \kappa_0$  is the Prandtl number, where  $\rho_0$ ,  $U_0$ ,  $L_0$  and  $\mu_0$  denote respectively the characteristic density, velocity, length and diffusivity of the considered flow. It is easily seen that, if the right-hand side is equal to zero, then we recover the Euler equations. The inlet conditions are defined by the farfield flow. For Euler flows, we impose the slip condition on the wall ( $\vec{V} \cdot \vec{n} = 0$ ), and for Navier-Stokes flows, on the wall, the no-slip condition ( $\vec{V} = \vec{0}$ ) and the isothermal condition ( $T = T_b$ ). The discretization is given by a mixed FEM/FVM formulation [5], where the mesh is a finite-element type (triangles), on which we construct control-cells (FVM) in order to solve the variational formulation of the equations, such as, for the Euler flows:

$$\sum_{j \in K(i)} \left[ \int_{\partial C_i} \vec{v}_i \cdot \vec{F}(W) d\sigma \right] + \int_{\text{in} \partial C_i} \vec{v}_i \cdot \vec{F}(W) d\sigma = -\frac{1}{Re} \left[ \sum_{\Delta, i \in \Delta} \iint_{\Delta} \left( R \frac{\partial \Psi_i^\Delta}{\partial x} + S \frac{\partial \Psi_i^\Delta}{\partial y} \right) dx dy \right] \quad (8)$$

The computation of the fluxes, appearing in (8), between two cells, is managed by Roe's numerical flux vector splitting in the domain, and by the Steger-Warming numerical flux vector splitting for the farfield boundaries. The 2nd-order accurate scheme is obtained by the use of the MUSCL method developed by van Leer [6]. We solve the discrete equations with non-linear relaxation algorithms [6],

namely here the multistage Jacobi algorithm [2, 7]:

$$\begin{aligned}
 & W_j^{(0)} = W_j^\alpha \\
 & \text{For } ks = 1 \text{ to } nstage \\
 & \quad [D]_{i,j} = \left[ \frac{\partial \Phi}{\partial W_j} (W_i^\alpha, W_j^{(ks)}; \vec{\eta}_{i,j}) \right] \\
 & \quad W_j^{(ks+1)} = W_j^{(0)} - \omega C_{ks} \sum_{i \in K(j)} [D]_{i,j}^{-1} \Phi(W_i^\alpha, W_j^{(ks)}; \vec{\eta}_{i,j}) \\
 & \quad W_j^{\alpha+1} = W_j^{(nstage)}
 \end{aligned} \tag{9}$$

Let us now look at the meshes. We start with an initial given fine mesh Fig.1.a. Finer meshes are

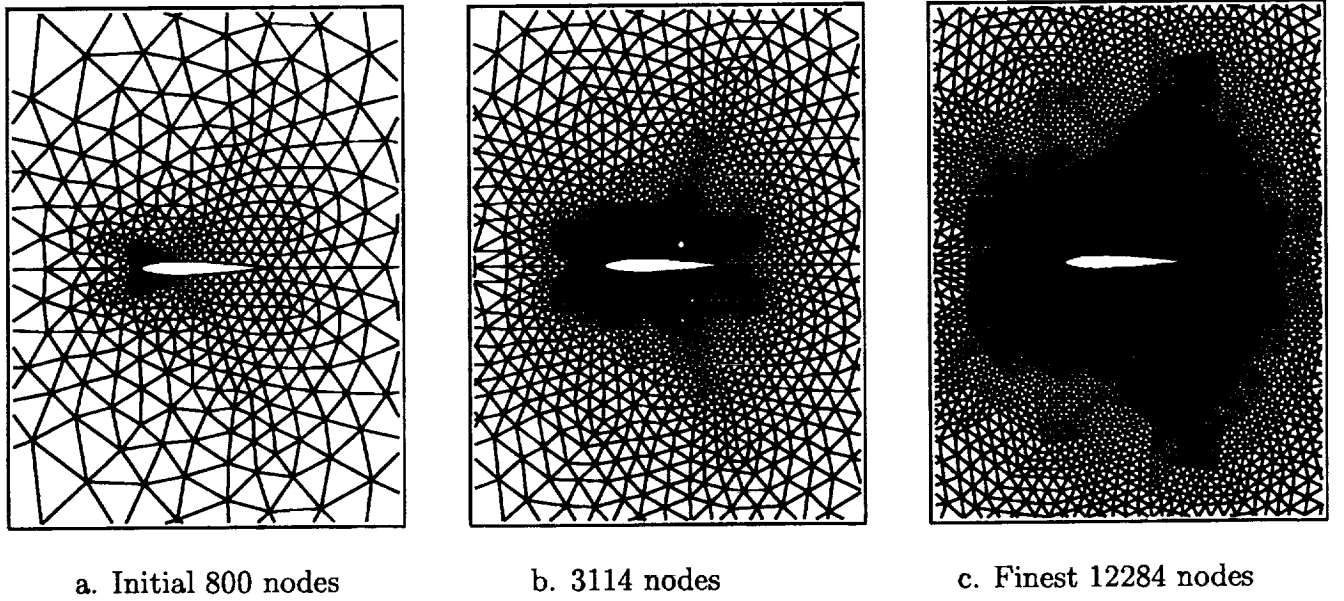
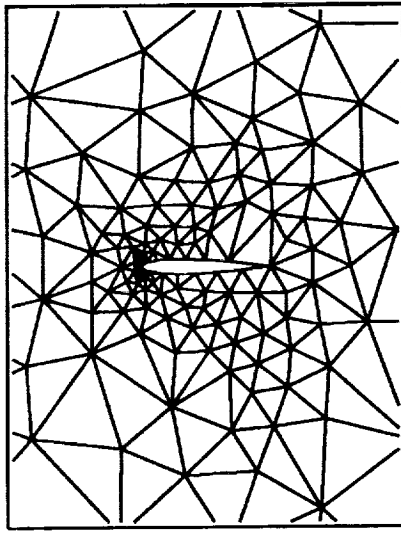


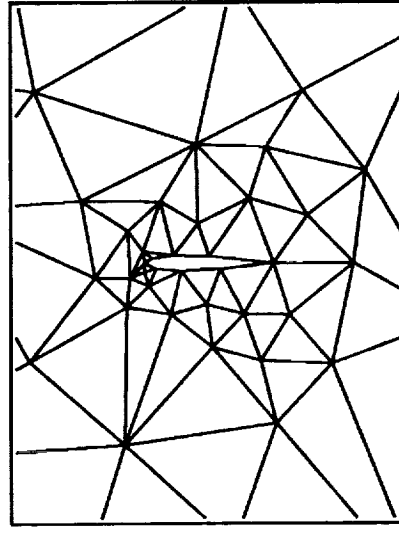
Figure 1: NACA0012 fine meshes.

obtained by triangle subdivision (Fig.1.b,c). Then, we use a coarsening algorithm due to Guillard [8] to build coarser meshes, from the initial one. This produces a sequence of node-embedded meshes (Fig.2.a,.b,.c). We get 6 meshes for the NACA0012 profile, where the finest has 12284 nodes and the coarsest 19 nodes. This method allows us to keep the mesh size ratio close to 2, and a comparable local mesh aspect ratio. The intergrid transfer operators [9] are linear interpolations, concerning the variables and the corrections, and linear distributions, concerning the residuals. The MG algorithm will be the W-cycle, because it is the natural extension of the ideal-2-grid scheme. Furthermore, there exist several ways to obtain 2nd-order accurate solutions:

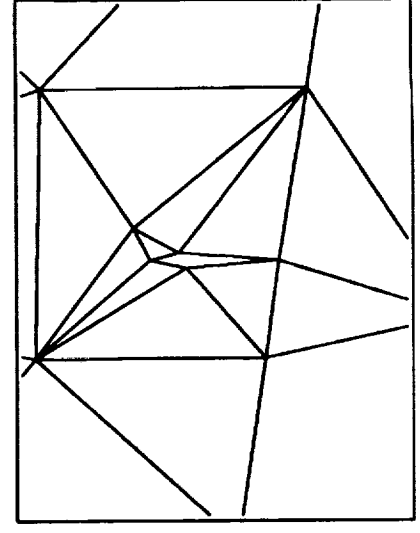
- Mavriplis [10] uses an FMG algorithm, where 1st-order accurate solutions are computed on the coarse levels, and 2nd-order accurate on the finest. Some experiments with our upwind schemes showed us that the convergence speed is hardly independent of the mesh size.
- Hemker-Koren [11] propose to get a 1st-order solution with an FMG strategy and then to compute a certain number of DeCV-cycles: They use the Defect Correction (DeC) algorithm



a. 223 nodes



b. 67 nodes



c. Coarsest 19 nodes

Figure 2: NACA0012 coarse meshes.

[12], in order to solve the 2nd-order accurate following problem:  $\mathcal{F}_2(W) = S$ . It is written:

$$\begin{aligned} \mathcal{F}_1(W^1) &= S, \\ \mathcal{F}_1(W^{N+1}) &= \mathcal{F}_1(W^N) - \mathcal{F}_2(W^N) + S, \quad N = 1, 2, \dots \end{aligned} \quad (10)$$

Actually, we define the DeCV-cycling method where the 1st-order problem in (10) is approximately solved with one V-cycle. However, we do not know how many DeCV-cycles are to be performed and we lose the  $O(N)$  complexity.

We propose here two different methods in order to obtain 2nd-order accurate solutions with an  $O(N)$  complexity algorithm [1].

- FMDeCV is an FMG strategy where we use DeCV-cycles in each phase, with two Jacobi sweeps per level (FMDeCV-2RK1).
- FMG2 is an FMG strategy where we use on each level of the different phases the good damping properties of the multistage schemes (see [13]) for smoothing directly the second order accurate problem.

A result of convergence of the DeCV method is given in [14] and assures that DeCV-cycling has a convergence speed independent of the mesh size:

$$\|DeCVu_h^\alpha - \bar{u}_h^2\| < S_2 \|u_h^\alpha - \bar{u}_h^2\|, \quad S_2 = S_1 + S_1 S_{DeC} + S_{DeC} < 1 \quad (11)$$

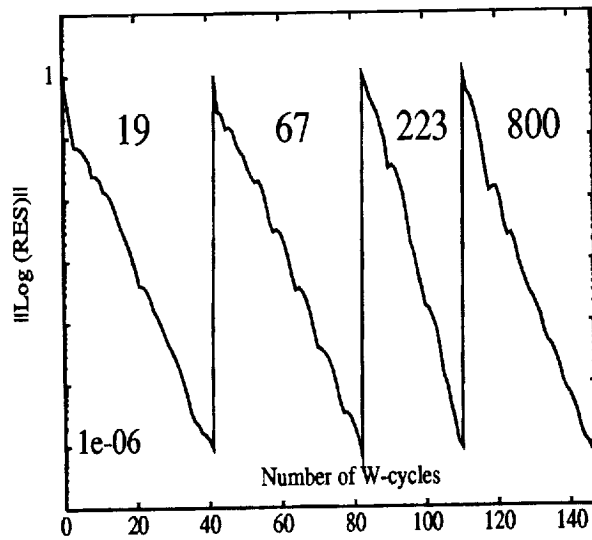
where  $DeCVu_h^\alpha$  is the  $\alpha$ -th iterate of the DeCV-cycling and  $\bar{u}_h^2$  is the solution of the 2nd-order accurate problem.

*Remark:* Desideri-Hemker in [12] show that the convergence speed  $S_{DeC}$  of the DeC process is at least equal to 1/2. From (11), we should use MG 1st-order accurate algorithms whose convergence

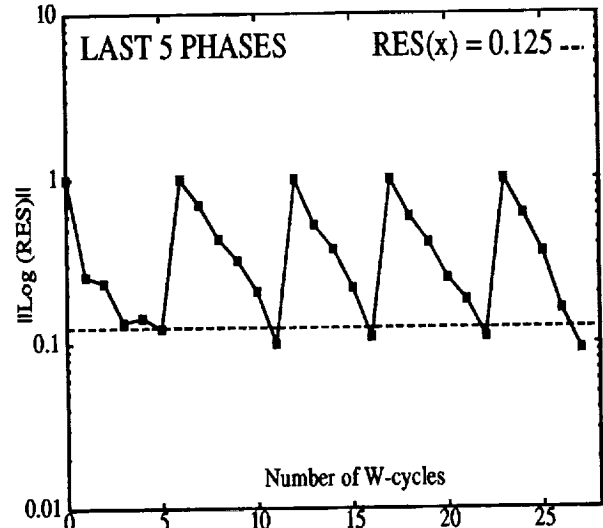
speed  $S_1$  (independent of the mesh size) is less than  $1/3$ . Actually, the experiments showed us that  $S_1 > 1/3$  does not induce difficulty.

## SECOND ORDER ACCURATE RESULTS

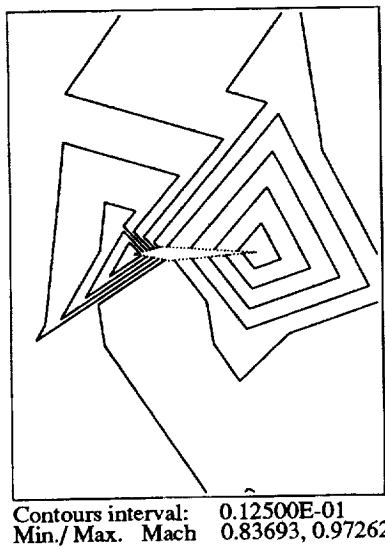
### Euler Flows around a NACA0012 profile



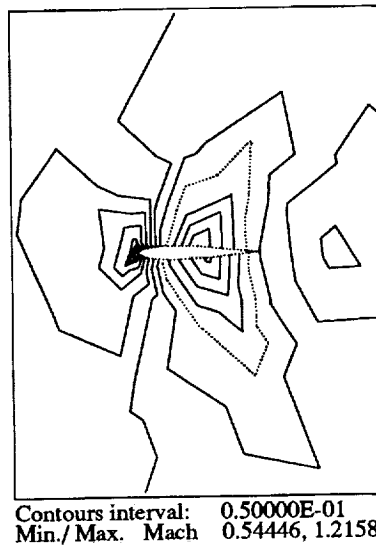
a. Mesh Independence



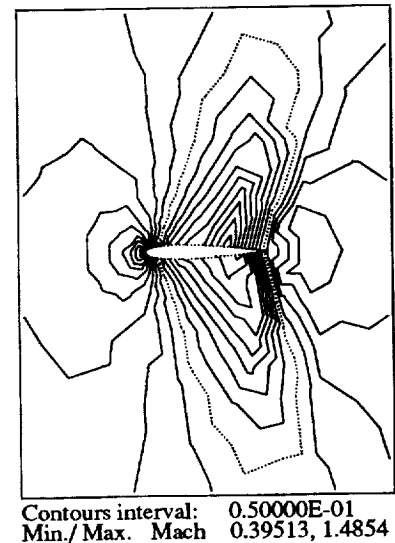
b. FMG Convergence



c. 19 nodes



d. 67 nodes



e. 223 nodes

Figure 3: Euler FMG2 phases, isomach contours,  $M_\infty = 0.9$ ,  $\alpha = 0^\circ$ .

The test-case depicted in Fig.3 to Fig.5 is defined by a farfield Mach number equal to 0.9, and a zero angle of attack. The smoother is the (4 stage) RKJ one, whose coefficients ( $\alpha_1 = 0.14$ ,  $\alpha_2 =$

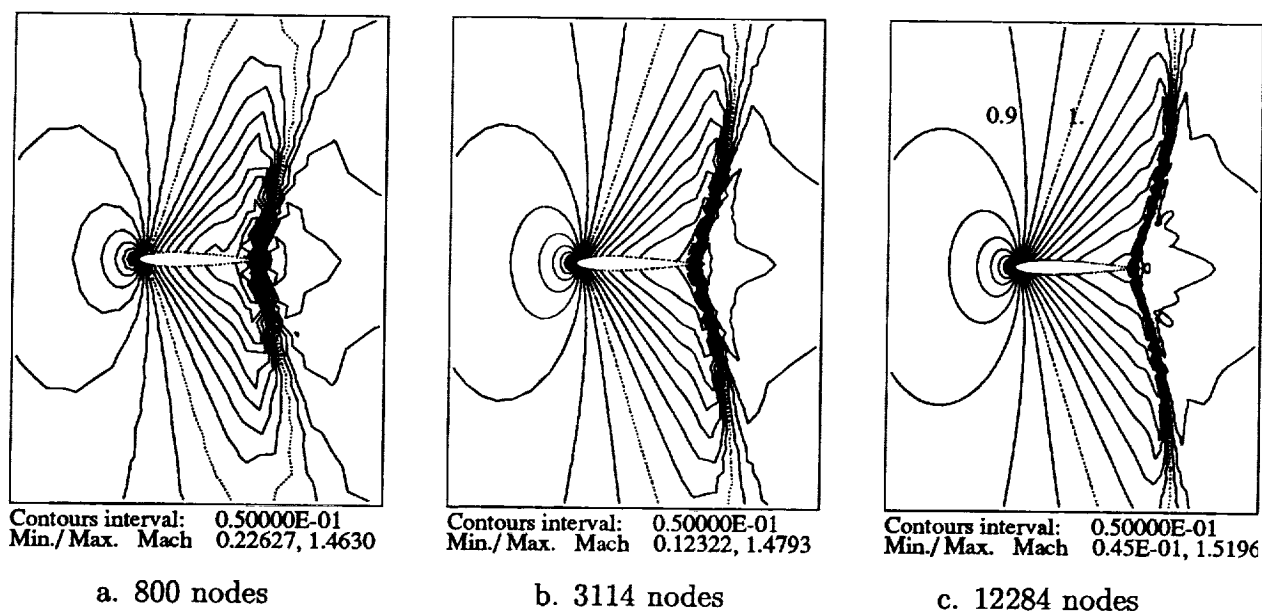
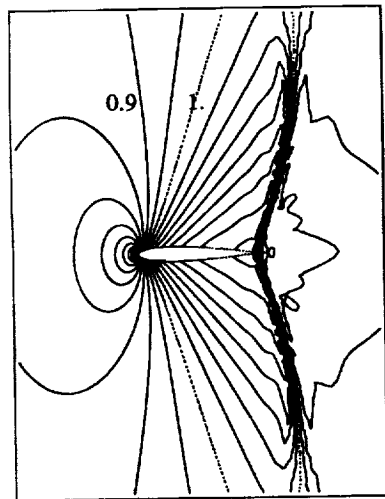


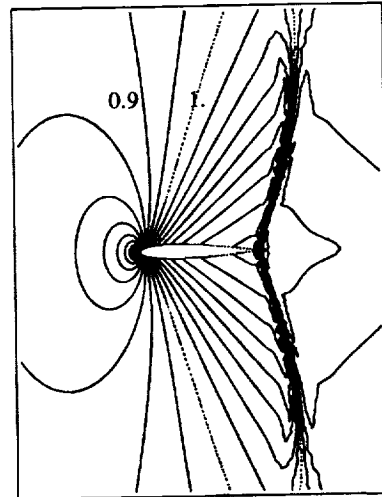
Figure 4: Euler FMG2 phases, isomach contours,  $M_\infty = 0.9$ ,  $\alpha = 0^\circ$ .

0.2939,  $\alpha_3 = 0.5252$ ,  $\alpha_4 = 1$ ) are due to van Leer [15], and defines the FMG2 strategy. In Fig.3.a, we present the convergence histories of the logarithm of the residual versus the number of cycles in each phase. The convergence is estimated as obtained when the residual decrease reaches  $10^{-6}$ . The first history is a 1-grid convergence on the coarsest mesh (19 nodes), the last (4-grid scheme) on the 800 node mesh. At the end of the convergence of each phase we produce an initialization of the next phase by interpolating the solution on the next finer mesh. We may notice that the different convergence histories tend to be straight lines with the same value of slope: this allows us to say that the convergence speed is independent of the mesh-size and that we may use an FMG strategy. In Fig.3.b the residual convergence histories of the last 5 phases are depicted (the first one is a 1-grid convergence history with a residual decrease equal to  $10^{-6}$ ). In order to neglect oscillatory non-linear phenomena we choose to impose a residual decrease equal to  $1/8$  (and not a defined number of cycles): the FMG convergence histories follow exactly the peaks of the (corresponding) phases on Fig.3.a, and, the solutions are not either changed. A solution on the finest grid (Fig.1.c) is reached after only 5 cycles (77 WU, 673 s on Convex C210 with non vectorized software). In Fig.3 and Fig.4 we show the different solutions obtained at the end of each phase: they are non-symmetrical solutions (Fig.3), due to the fact that the coarse meshes are non-symmetrical. However, this phenomenon vanishes when the different meshes become symmetrical (Fig.4). Another important remark is that the solution between the finest mesh and the next coarser one does not vary much: we may say that we get a nearly converged solution on the 3114 node mesh. In order to verify the previous assumption, we compare the FMG solution of Fig.5 with the solution obtained after a  $10^{-6}$  residual decrease: the Mach number extrema are approximately the same although the isomach lines are a little bit more oscillatory for the FMG solution.



Contours interval: 0.50000E-01  
Min./Max. Mach 0.45E-01, 1.5196

a. FMG2



Contours interval: 0.50000E-01  
Min./Max. Mach 0.47E-01, 1.5146

b. Converged (RES =  $10^{-6}$ )

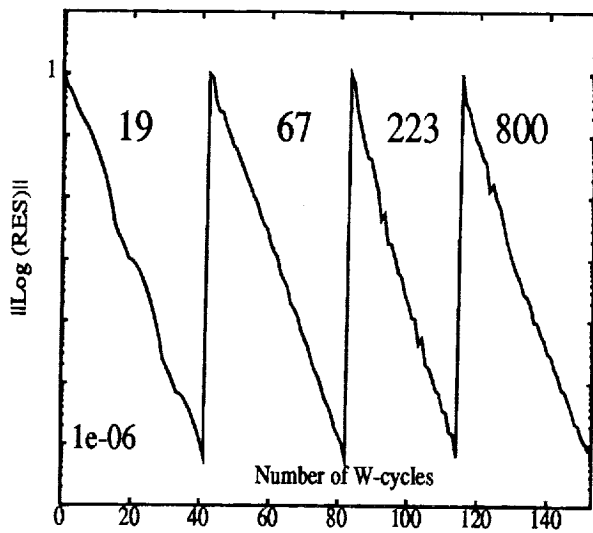
Figure 5: Euler FMG2 solutions, isomach contours,  $M_\infty = 0.9$ ,  $\alpha = 0^\circ$ .

### Navier-Stokes Flows around a NACA0012 profile

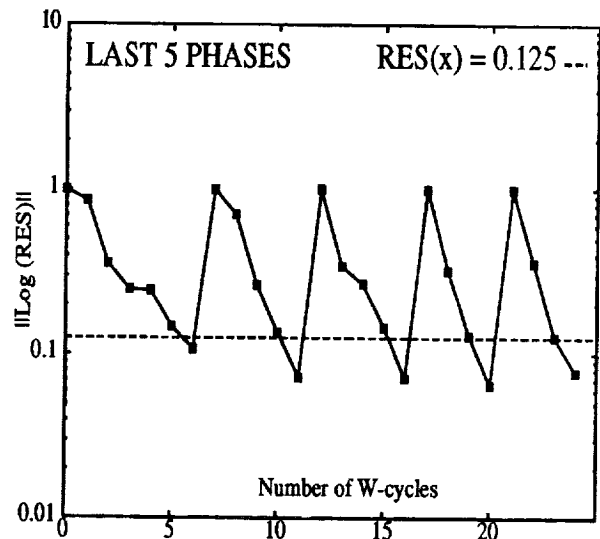
The first test-case is defined by a farfield Mach number equal to 0.8, an angle of attack equal to 10 degrees and a Reynolds number equal to 73. We use here an FMDeCV-2RK1 strategy (justified by the mesh independent convergence of Fig.6.a). The solution (Fig.6.c) is obtained on the finest mesh after 4 cycles that represent 42 WU computation and a total CPU time equal to 537 s. In Fig.6.d we illustrate the behavior of the pressure lift, CL (drag, CD) coefficient with a solid (dash) line, versus the number of finest-grid iterations, up to a residual decrease on the finest grid of  $10^{-12}$ . The points (cross) represent these coefficients during the FMG process, thus up to a 0.125 residual decrease. We can notice that the value of each of them is almost obtained at the end of the FMG phase (the error is equal to  $64 \cdot 10^{-5}$  for the CL coefficient and to  $16 \cdot 10^{-5}$  for the CD coefficient).

The second test-case, presented in Fig.7, is defined by a farfield Mach number equal to 2, an angle of attack equal to 10 degrees, and a Reynolds number equal to 106. This time we use an FMG2 strategy (more robust than FMDeCV-2RK1 that needs TVD limitation and implies that the residual stalls from the value of  $10^{-3}$ ); this produces a solution after 7 cycles (Fig.7.c, 109 WU, 1862 s), and we can make the same remarks as in the previous test-cases. The next coarser mesh results in CL and CD values within 1% of their final values which are obtained after 7 cycles on the finest mesh with a related error respectively of  $2 \cdot 10^{-6}$  and  $2 \cdot 10^{-5}$  (Fig.7.d),

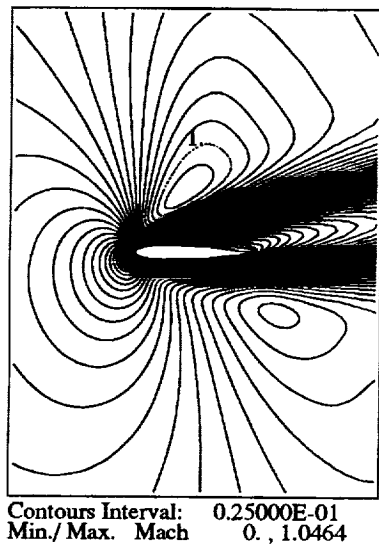
The last test-case shows us one limitation of this method. It is defined by a farfield Mach number equal to 0.8, an angle of attack equal to 10 degrees and a Reynolds number equal to 500. Here again, we use an FMDeCV-2RK1 strategy (Fig.8). We may note, once again, that the FMG convergence histories (Fig.8.b) look like the corresponding peaks on Fig.8.a, thus we think that the solution does not vary between a  $10^{-1}$  residual decrease and a  $10^{-6}$  one. However, on Fig.9.a we note that the isomach lines are deformed up until the last solution (Fig.9.c) which presents two bumps. Actually,



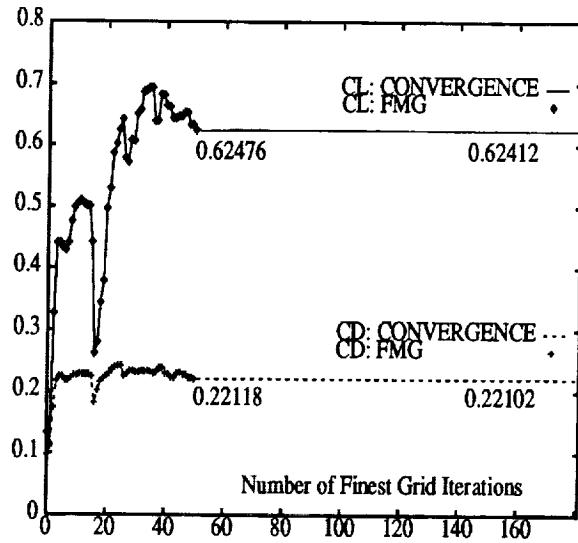
a. Mesh Independence



b. FMG Convergence



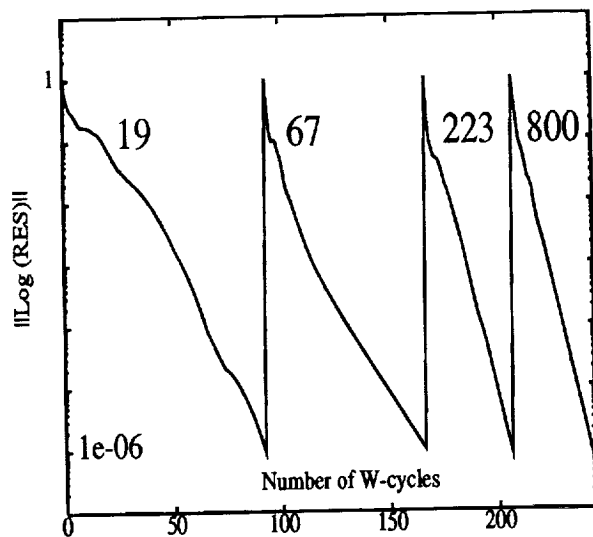
c. Isomach contours



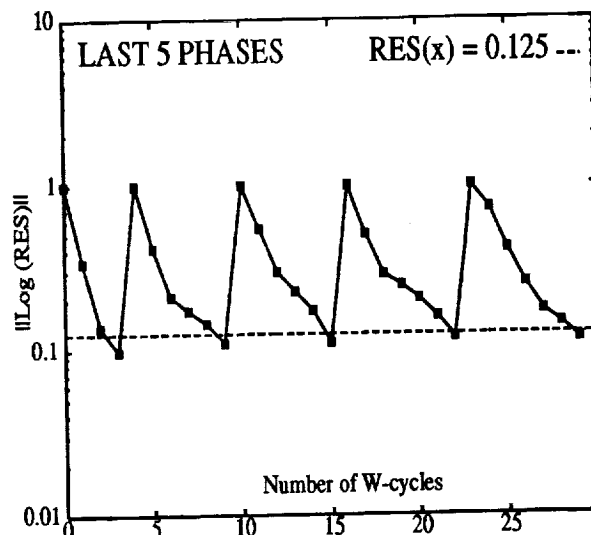
d. CL and CD behaviors

Figure 6: Navier-Stokes FMDeCV-2RK1 solution,  $M_\infty = 0.8$ ,  $\alpha = 10^\circ$ ,  $Re = 73$ .

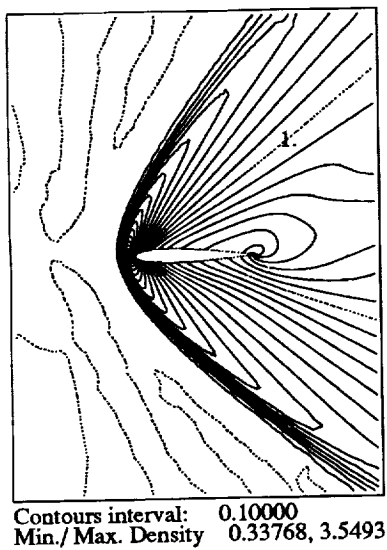




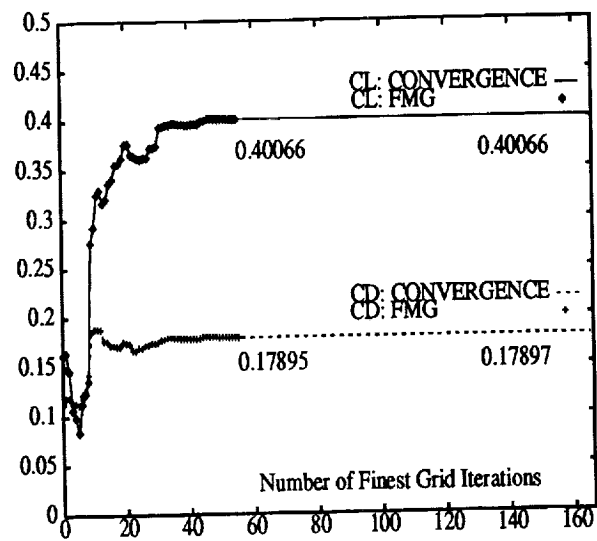
a. Mesh Independence



b. FMG Convergence

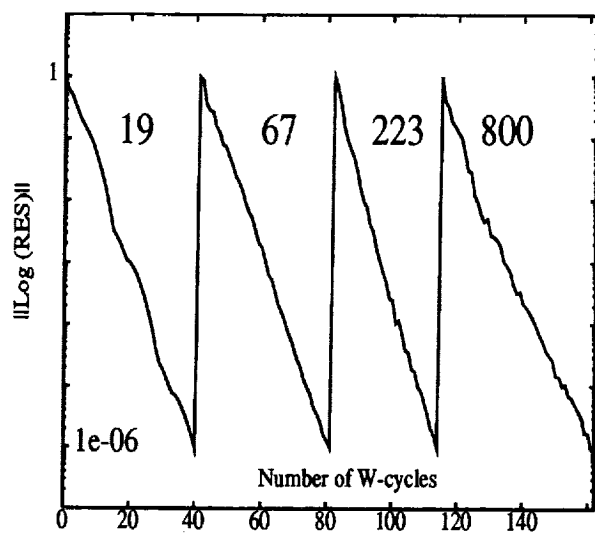


c. Density contours

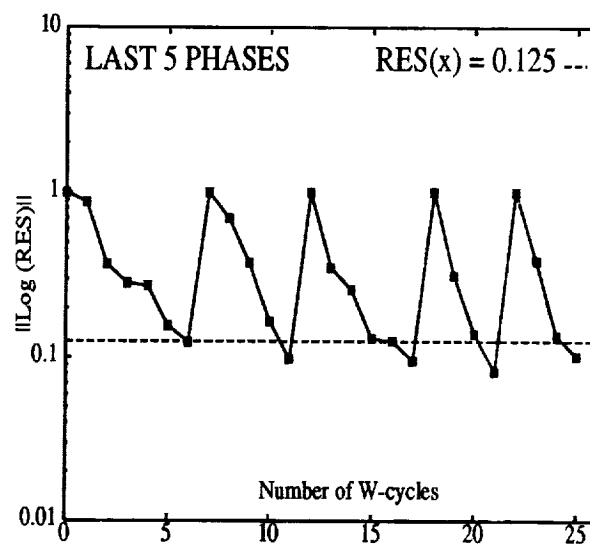


d. CL and CD behaviors

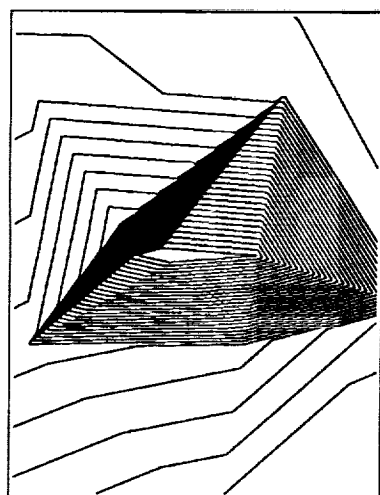
Figure 7: Navier-Stokes FMG2 solution,  $M_\infty = 2$ ,  $\alpha = 10^\circ$ ,  $Re = 106$ .



a. Mesh Independence

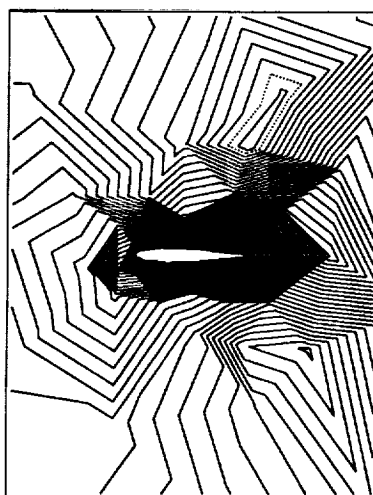


b. FMG Convergence



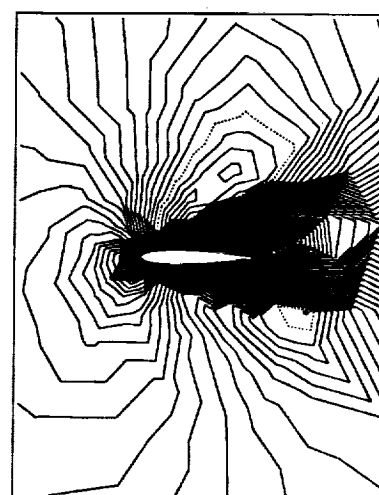
Contours interval: 0.25000E-01  
Min./Max. Mach 0., 0.79407

c. 19 nodes



Contours interval: 0.25000E-01  
Min./Max. Mach 0., 1.0496

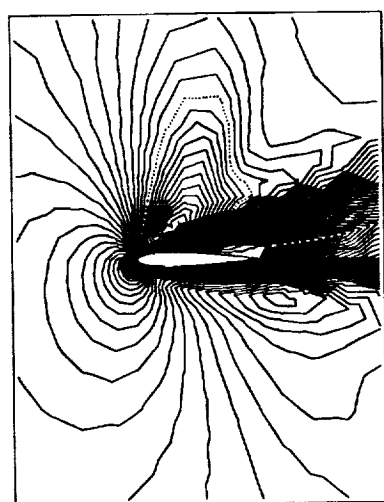
d. 67 nodes



Contours interval: 0.25000E-01  
Min./Max. Mach 0., 1.0901

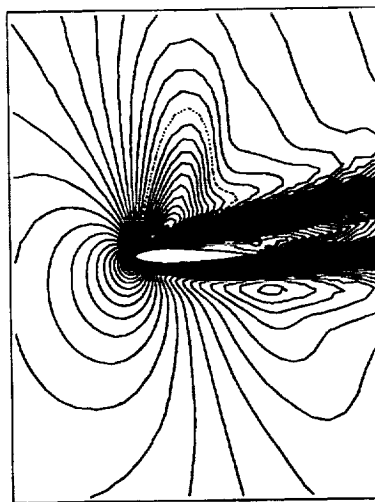
e. 223 nodes

Figure 8: Navier-Stokes FMDeCV-2RK1 phases, isomach contours,  $M_\infty = 0.8$ ,  $\alpha = 10^\circ$ ,  $Re = 500$ .



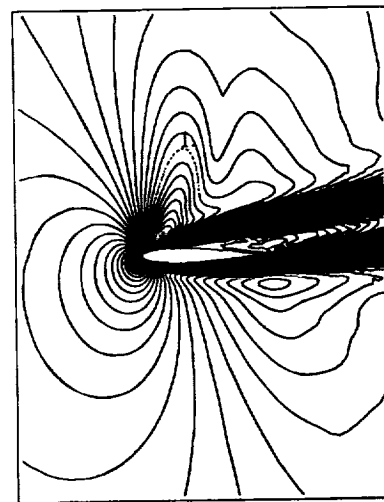
Contours interval: 0.25000E-01  
Min./Max. Mach 0. , 1.3009

a. 800 nodes



Contours interval: 0.25000E-01  
Min./Max. Mach 0. , 1.2843

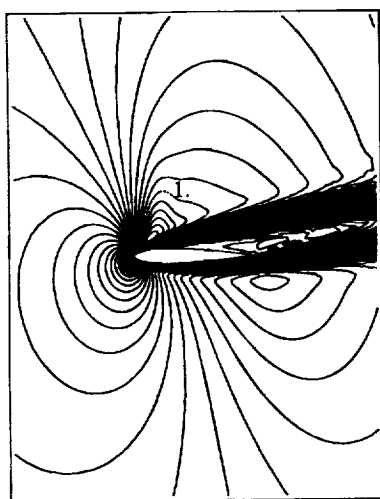
b. 3114 nodes



Contours interval: 0.25000E-01  
Min./Max. Mach 0. , 1.2028

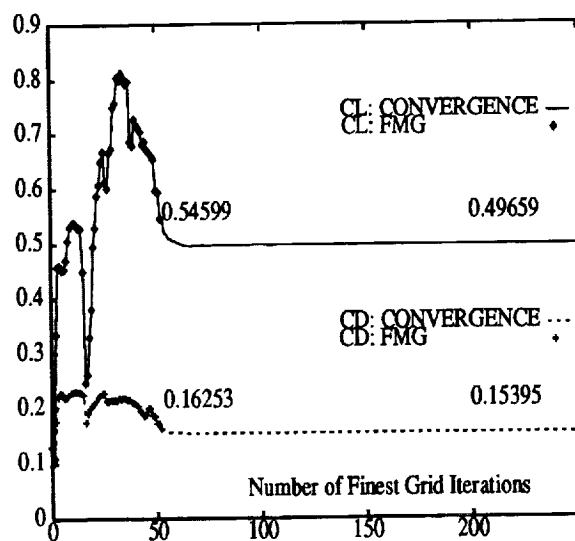
c. 12284 nodes

Figure 9: Navier-Stokes FMDeCV-2RK1 phases, isomach contours,  $M_\infty = 0.8$ ,  $\alpha = 10^\circ$ ,  $Re = 500$ .



Contours interval: 0.25000E-01  
Min./Max. Mach 0. , 1.1421

a. Isomach contours



b. CL and CD behaviors

Figure 10: Navier-Stokes FMDeCV-2RK1 solution,  $M_\infty = 0.8$ ,  $\alpha = 10^\circ$ ,  $Re = 500$ .

since the solution differs a lot between the one obtained on the finest mesh and the other on the next coarser mesh (Fig.9.c and .b), we may say that we did not obtain a grid-converged solution. Furthermore, these two humps may be justified because the meshes that we use (especially the coarse ones) are not adapted for such a viscous flow and may not capture the boundary layer. Our assumption is confirmed by the CL and CD behaviors (Fig.10.b), where we note an error for the CL coefficient equal to  $5 \cdot 10^{-2}$  and for the CD coefficient equal to  $8 \cdot 10^{-3}$ . Moreover, we get a solution (Fig.10.a) after 105 cycles (1107 WU, 14112 s), for a residual decrease of  $10^{-12}$  and which tends to confirm that the FMG solution is not a steady solution.

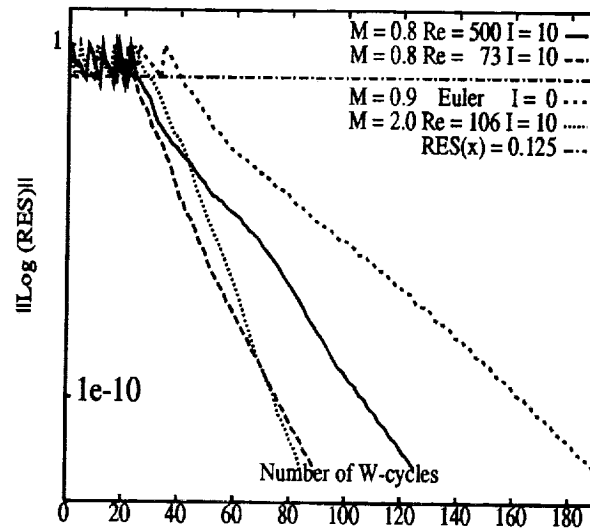


Figure 11: Convergences

## CONCLUSION

We want to point out that the use of FMG2 or FMDeCV strategy allowed us to get 2nd-order accurate solutions in most of cases with a limited number of operations ( $O(N)$  complexity). FMDeCV-2RK1 is more adapted to smooth problems and costs half as much as FMG2. Furthermore, we had to use an entropy correction technique [16] to get the above results, and occasionally a  $10^{-12}$  residual decrease required to increase the value of this correction to prevent the residual from stalling or to improve robustness on the finest grid (this increased slightly the number of cycles in each phase without changing the solution greatly). As depicted in Fig.11, these types of computations do not induce any stall during the convergence. The main two difficulties, non-embedded meshes and the requirement of 2nd-order accuracy, were remedied, respectively, by using a coarsening algorithm based on a Voronoï technique, and basic iteration techniques that were sufficient smoothers. The difficulty encountered in using an FMG strategy, with our meshes, increased as the Reynolds number was raised. Actually, it is obvious that our meshes are not adapted to these computations and that boundary layer problems will need the production of stretched meshes and different specialized smoothers, as suggested in [14].

## ACKNOWLEDGEMENTS

The meshes were kindly given by Hervé Guillard. A part of the MG codes was initiated by Marie-Pierre Leclercq and Bruno Stoufflet. We thank Piet Hemker, Barry Koren, Marie-Hélène Lallemand and Dimitri Mavriplis for the many helpful discussions we had with them.

## References

- [1] W. HACKBUSCH, *"Multi-Grid Methods and Applications"*, Springer Series in Comp. Math., Springer Verlag, Vol. 4, (1985).
- [2] E. MORANO, M.H. LALLEMAND, M.P. LECLERCQ, H. STEVE, B. STOUFFLET A. DERVIEUX, *"Local iterative upwind methods for steady compressible flows"*, GMD-Studien Nr. 189, Multigrid Methods: Special Topics and Applications II, 3rd Europ. Conf. on Multigrid Methods, Bonn 1990 (1991).
- [3] S. ZHANG, *"Multi-level iterative techniques"*, PhD Thesis, The Pennsylvania State University (1988).
- [4] A. BRANDT, *"Guide to multigrid development"*, Multigrid Methods, Proc. of the Köln-Porz Conf. on Multigrid Methods, (W. Hackbusch and U. Trottenberg, Eds.), Lect. Notes in Math. 960, Springer, Berlin, p. 220-312 (1982).
- [5] S. LANTERI, *"Simulation d'écoulements aérodynamiques instationnaires sur une architecture massivement parallèle"*, PhD Thesis, Université de Nice Sophia-Antipolis (1991).
- [6] B. van LEER, *"Flux vector splitting for the Euler equations"*, Lect. Notes in Phys., Vol. 170, p. 405-512 (1982).
- [7] E. DICK, K. RIEMSLAGH, *"Multi-stage Jacobi relaxation as smoother in a multigrid method for steady Euler equations"*, Proc. of ICFD Conf. on Num. Methods for Fluid Dynamics, Reading, England (1992).
- [8] E. MORANO, H. GUILLARD, A. DERVIEUX, M.P. LECLERCQ, B. STOUFFLET, *"Faster relaxations for non-structured MG with Voronoï coarsening"*, Comput. Fluid Dynamics '92, Proc. of the first Europ. Comput. Fluid Dynamics Conf., Brussels, Belgium, (Ch. Hirsch, J. Périaux, W. Kordulla Eds.), Vol. I, p. 69-74, ECCOMAS, ELSEVIER (1992).
- [9] M.-P. LECLERCQ, B. STOUFFLET, *"Characteristic Multigrid Method Application to solve the Euler equations with unstructured and unnested grids"*, Inter. Conf. on Hyperbolic Problems, Uppsala (1989).
- [10] D. MAVRIPLIS, A. JAMESON, *"Multigrid solution of the Navier-Stokes equations on triangular meshes"*, ICASE Report No. 89-11 (1989).
- [11] B. KOREN, P.W. HEMKER, *"Damped, direction-dependent multigrid for hypersonic flow computations"*, Appl. Num. Math., North-Holland, Vol. 7, p. 309-328 (1991).
- [12] J.A. DESIDERI, P. HEMKER, *"Analysis of the convergence of iterative implicit and Defect-Correction algorithms for hyperbolic problems"*, INRIA Research Report 1200 (1990).
- [13] M.H. LALLEMAND, *"Schémas décentrés multigrilles pour la résolution des équations d'Euler en éléments finis"*, PhD Thesis, Université de Marseille (1988).
- [14] E. MORANO, *"Résolution des équations d'Euler par une méthode multigrille stationnaire"*, PhD Thesis, Université de Nice Sophia-Antipolis (1992).
- [15] B. van LEER, C.H. TAI, and K.G. POWELL, *"Design of optimally-smoothing multi-stage schemes for the Euler equations"*, AIAA in 9th Comput. Fluid Dynamics Conf., 89-1933-CP (1989).
- [16] A. HARTEN, *"High Resolution Schemes for Hyperbolic Conservation Laws"*, J. Comput. Phys., Vol. 49, p. 357-393 (1983).



# THE BLOCK ADAPTIVE MULTIGRID METHOD APPLIED TO THE SOLUTION OF THE EULER EQUATIONS<sup>1</sup>

Nikos Pantelelis

PhD Student, National Technical University of Athens  
P.O.Box 64078, 15710 ATHENS, GREECE

58-02  
197568  
p. 15

## SUMMARY

In the present study, a scheme capable of solving very fast and robust complex nonlinear systems of equations is presented. The Block Adaptive Multigrid (BAM) solution method offers multigrid acceleration and adaptive grid refinement based on the prediction of the solution error. The proposed solution method was used with an implicit upwind Euler solver for the solution of complex transonic flows around airfoils. Very fast results were obtained (18-fold acceleration of the solution) using one fourth of the volumes of a global grid with the same solution accuracy for two test cases.

## INTRODUCTION

Although multigrid methods were introduced as grid adaptation techniques they have been established only as fast and efficient solvers for large scale computational problems. Up until today only a few adaptive multigrid schemes have been presented, e.g. the Multilevel Adaptive Technique MLAT (ref. 1), the Fast Adaptive Composite grid method FAC (ref. 2) and others (ref. 3), but the domain of applications has been mainly restricted to the solution of elliptic type equations. Regarding the development of adaptive schemes for hyperbolic systems of equations, only a few attempts have been made to take advantage of the favourable multigrid concept for the acceleration of the solution. On the other hand great advantages have been pointed out for the use of the truncation error prediction as a reliable error sensor for grid adaptation, though few studies exhibited numerical proofs (ref. 4,5).

The present study, a dynamically grid adaptive method, namely the Block Adaptive Multigrid (BAM) method, is presented, incorporating a reliable device for the prediction of the error and a composite multigrid solver. The method is based on a Full Multigrid scheme: starting from an acceptable coarse mesh the solution creates finer grid patches with block grid refinement. The refined regions are grouped into rectangular blocks defining a composite structure which is totally handled by the multigrid method. In this way an adapted non uniform domain is decomposed into regular subdomains solved with common solvers. Further, the communication between blocks and the

<sup>1</sup>This work was partially supported by CEC/ Brite-Euram contract AERO-0018C.

parallelization of the BAM method are based on domain decomposition ideas. For the integration and relaxation of the time marching Euler equations, an unfactored implicit upwind finite volume scheme is employed. The proposed method is tested for two complicated transonic inviscid cases for two different airfoils. Using the proposed method stable and accurate results are obtained in a small number of work units (18-fold acceleration with 4.5 times fewer volumes).

## FINITE VOLUME DISCRETIZATION

The general inviscid flow is described by the Euler equations that can be solved using the very popular time-marching conservative formulation. For the two dimensional case, conservation laws are used with body-fitted coordinates  $\xi, \eta$ :

$$\frac{\partial U}{\partial t} + \frac{\partial E}{\partial \xi} + \frac{\partial F}{\partial \eta} = 0 \quad (2.1)$$

The steady state solution is found when the time derivative of the solution vector becomes negligible. The solution vector and the fluxes normal to  $\xi = \text{const.}$  and  $\eta = \text{const.}$  faces are given respectively:

$$U = J \cdot \bar{U} \quad \text{and} \quad E = J \cdot (\bar{E} \xi_x + \bar{F} \xi_y), \quad F = J \cdot (\bar{E} \eta_x + \bar{F} \eta_y) \quad (2.2)$$

At the Cartesian coordinate system the corresponding solution vector and the inviscid fluxes are given by:

$$\bar{U} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \quad \text{and} \quad \bar{E} = \begin{bmatrix} \rho u^2 + p \\ \rho u v \\ (e + p)u \end{bmatrix}, \quad \bar{F} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ (e + p)v \end{bmatrix} \quad (2.3)$$

In equation (2.3)  $e$  is the total energy ( $e = \frac{p}{(\gamma-1)} + \frac{1}{2}\rho[u^2 + v^2]$ ),  $p$  and  $\rho$  are the pressure and the density respectively and  $J$  is the Jacobian of the inverse mapping.

For the discretisation of equation (2.1) a cell-centered finite volume method is used. For the pseudo-time evolution a Newton linearization scheme is adopted which, being an implicit scheme, allows high CFL numbers (100-200) to be used with local time stepping (different  $\Delta t$  for each volume):

$$\frac{\Delta U}{\Delta t} + (A^n \Delta U)_\xi + (B^n \Delta U)_\eta = - (E_\xi^n + F_\eta^n) \quad (2.4)$$

Or else:

$$L^n \cdot \Delta U = \left( \Delta t^{-1} + A_\xi^n + B_\eta^n \right) \cdot \Delta U = - \text{Res}^n(U^n) \quad (2.5)$$

Where  $\Delta U$  is the correction of the solution vector,  $A$  and  $B$  are the Jacobians



of the fluxes E and F respectively for the time level n:

$$U^{n+1} = U^n + \Delta U, \quad A^n = \left( \frac{\partial E}{\partial U} \right)^n \quad \text{and} \quad B^n = \left( \frac{\partial F}{\partial U} \right)^n \quad (2.6)$$

Upwind differencing of the flux vectors is used to achieve a diagonal dominant system of equations. Thus, using a symmetric collective point Gauss-Seidel relaxation scheme very good smoothing properties are attained for the multigrid calculations. For the flux calculations a linear locally one-dimensional Riemann solver (Godunov-type) is employed thus, the homogeneous property of the Euler fluxes [7] is guaranteed. The mean values of the conservative variables at both sides of the faces are used as input variables for the Riemann solver. For the calculation of the fluxes E and F the conservative variables are extrapolated using an upwind characteristic variable interpolation method (MUSCL-type). The interpolation scheme uses two volumes from both sides of each face. The accuracy of the scheme raises up to third order depending on the sign of the local eigenvalues of the Jacobians A and B. The local accuracy of the finite volume method is sensor-controlled so the monotonic behaviour of the solution is guaranteed. Boundary conditions are required for both sides of eq.(2.4), so for the RHS of eq.(2.4) characteristic boundary conditions are extracted from the Riemann solutions at the wall and at free surfaces. For the LHS of eq.(2.4) simple boundary conditions are prescribed on the  $\Delta U$  in phantom shells.

## THE BLOCK ADAPTIVE MULTIGRID METHOD

The Block Adaptive Multigrid (BAM) method is composed of three main parts: the fast nonlinear multigrid solver, the truncation error approximation for the prediction of the solution error and the block composite grid solver. Additionally an efficient solution strategy is required in order to achieve fast and robust accurate solutions.

### Multigrid Implementation

Concerning the multigrid method the Full Approximation Scheme (FAS) is employed as it is better than the Correction Scheme (CS) for Newton linearisation schemes. FAS has the major advantage in that it operates with the same solution vector of the initial algorithm so it is best suited for the solution of composite grid structures. The efficiency and the performance of the multigrid implementation are maintained adopting the "alternate point of view" of the FAS (ref. 1). Following this approach the finer grid levels are considered as devices to increase the spatial accuracy of the solution whereas the coarser grid levels are devices to accelerate the solution. The formulation of the solution is independent of the grid level (coarse or fine) and the type of grid (local or global) by simply adding to the RHS of eq.(2.5) the appropriate fine-to-coarse defect correction ( $\tau$ ). For the enumeration of the multigrid levels the classical mode is adopted. Thus, the current grid

level is denoted by  $n$ , its next finer level by  $n+1$  and its corresponding finest grid level by  $m$  ( $m \geq n \geq 1$ ) (1 is the coarsest grid level of the domain). The solution formulation for the current grid level  $n$  is given as:

$$L_n \cdot \Delta U_n = -\text{Res}_n(U_n) + \tau_n^{n+1} \quad (3.1)$$

considering that the fine-to-coarse defect correction is:

$$\tau_n^{n+1} = \Sigma_n^{n+1} \left[ L_{n+1} \cdot \Delta U_{n+1} \right] - L_n \cdot (I_n^{n+1} \Delta U_{n+1}) \quad \text{and} \quad \tau_m^{m+1} = 0 \quad (3.2)$$

Because one multigrid cycle equals one time step the time scale is not considered.

Because of the applied cell centered finite volume scheme the cellwise coarsening is adopted; each coarse volume is constructed by four consequently finer ones. The cellwise coarsening maintains the outer edges of the volumes (straight implementation of conservation laws) but the coarse grid centers are not a subset of the fine ones. Therefore, two different restriction operators are required. The restriction operator (I) for the physical variables is the simple average over the four fine volumes:

$$\Delta U_n = I_n^{n+1} \Delta U_{n+1} = \frac{\sum \Delta U_{n+1}}{4} \quad (3.3)$$

In contrast, the restriction operator ( $\Sigma$ ) for the generalized residuals,  $\text{Res}$  and  $\tau$ , is the summation of the residuals of the corresponding fine volumes. The fluxes of the inner common fine grid faces are canceled, thus flux conservation at the coarser grid levels is maintained:

$$\text{Res}_n = \Sigma_n^{n+1} \text{Res}_{n+1} = \sum \text{Res}_{n+1} \quad (3.4)$$

For the reverse direction of the multigrid cycle (coarse-to-fine direction) neither Euler solutions nor relaxation sweeps are required. Therefore,  $\Delta U$  variables are stored for all grid levels and only these are prolonged from the coarse-to-fine direction using the standard FAS prolongation formulation:

$$\Delta U_{n+1} = \Delta U_{n+1} + \Pi_{n+1}^n (\Delta U_n - I_n^{n+1} \Delta U_{n+1}) \quad (3.5)$$

For the prolongation operator (II) simple injection is adopted, i.e.:

$$U_{n+1} = \Pi_{n+1}^n U_n = U_n \quad (3.6)$$

Due to the composite grid structure, complicated interpolation schemes would have increased considerably the programming complexity with minor advantages. For the present multigrid implementation the V-cycle is applied with the improvement of increasing the number of relaxation sweeps as coarser levels are processed.

### Solution Error Approximation

To determine the erroneous regions of the computational domain where

increase of the accuracy is required, a reliable error indicator is required. Two approaches essentially exist: the first one is the physically based information of the problem, i.e. solution gradients, while the second one, numerically motivated, is the evaluation of the discretization error. The former approach may implicate the refinement process to reduce errors that have no influence on the global solution. In contrast, the evaluation of the truncation error approach indicates errors which can be confronted by the refinement procedure. On the basis of the Richardson extrapolation an estimation of the truncation error is formulated as:

$$T_n = Q(h) \cdot u \quad (3.6)$$

Whereas for a uniform Cartesian mesh holds:

$$T_n = c \cdot (h)^p \quad (3.7)$$

In equations (3.6), (3.7)  $Q$  is the differential operator,  $u$  is the physically correct solution,  $h$  is the mesh size of the finest grid level,  $p$  is the local order of accuracy and  $c$  is an unknown grid independent factor. Guided by the physical interpretation of the truncation error and the Richardson extrapolation concept, the difference of residuals between the two finest grid levels is used for the truncation error calculation. This error sensor provides a reliable local estimation of the solution error. Although the fine-to-coarse defect correction of eq.(3.2) is directly provided by the multigrid solution it was proved to be an unreliable error indicator. Fortunately the use of the Res(U) operator instead of the  $L \cdot \Delta U$  operator gives very good results. The explanation is that due to the Newton linearization scheme the Res(U) differential operator is insensitive to relaxation errors maintaining the accuracy of the solution. Therefore, the solution error evaluation for the grid level  $m-1$  ( $m$  is the current local finest grid level), is given by:

$$T_{m-1}^m = \sum_{m-1}^m T_m - T_{m-1} = \sum_{m-1}^m \text{Res}_m(U_m) - \text{Res}_{m-1}(I_{m-1}^m U_m) \quad (3.8)$$

For a totally converged solution equation (3.8) reduces to:

$$T_{m-1}^m = - \text{Res}_{m-1}(I_{m-1}^m U_m) \quad (3.9)$$

As this truncation error sensor is a vector, a reduction norm to a single value is required. At the present study the Euclidean norm has been adopted because it shows similar distribution to the pressure error of the solution. The proposed error sensor requires additional work of only one fourth of a simple flux calculation and it does not demand totally converged solution ( $\text{Res}_m(U_m) \neq 0$ ) although it converges from the early time steps to the steady state.

The prediction of the solution error for the new finer grid level  $m-1$  can be computed only if the assumption of a uniform Cartesian mesh is adopted ( $h_{m+1} = h_m/2$ ). Thus following equation (3.7) we get:

$$T_m^{m+1} = 2^{-p} T_{m-1}^m \quad (3.10)$$

For the determination of the order of accuracy ( $p$ ) of eq.(3.10), which varies from one to two depending on the flow features, the sensing functions that are used to control monotonicity at the integration routine are also used to calculate the local accuracy of the scheme. Unfortunately, equation (3.10) holds only for Cartesian grids, thus using this equation in arbitrary grids

errors are expected to the predicted error levels.

### Composite Grid Structure and Solution Procedure

For the optimal approximation of the most accurate solution within the minimum amount of work, several grid adaptive techniques and structures have been developed. The composite grid structure has many advantages by enabling the solution of a globally non uniform grid using a union of locally uniform subgrids (blocks) (ref. 2). Subgrid uniformity is essential to assure multigrid efficiency using simple solution routines (similar to the single grid solver). Moreover, significant simplifications to the data structure and to the interface manipulation are attained when the blocks are restricted to rectangles in the computational domain and the grid refinement ratio is 2 for each refined direction (ref. 5).

In a composite multigrid method the major problem is the requirement of special manipulation at the artificial boundaries to maintain the accuracy of the global solution. To suppress errors inconsistent with the solution method at the artificial boundaries (two fine volumes share the same edge with a coarse one) certain special boundary conditions must hold. Namely: accuracy preservation of the integration scheme, flux conservation and flux-splitting compatibility with respect to the global grid solver. To preserve the accuracy of the solution across an artificial boundary, the fluxes of the domain should be calculated at the block's finest grid level. For these calculations the standard integration routine has been employed. The basic idea is to properly construct false fine volumes at the coarse grid boundaries using all the available information near the interfaces. The modified scheme at the intergrid boundaries is depicted in fig.1 and fig.2 for the one and two dimensional analogs, respectively. Following these analogs the flux calculations should be started from Block 1 including the boundary interfaces. For the evaluation of the virtual volumes F3 and F4 (fig.1) the coarse volumes C1, C2 and the fine ones F1, F2 are considered adopting the same MUSCL-type extrapolation scheme used by the integration routine. At the two dimensional analog, the same couple of coarse volumes is used with the corresponding fine volumes (fig.2). In this way, the order of accuracy of the global solution scheme is maintained. After calculating the fluxes of the finest subgrid (Block 1 in fig.1), the boundary fluxes of the neighboring coarser subgrids (Block 2) can be calculated explicitly using conservation of fluxes. According to the multigrid restriction operator for the residuals, flux conservation across the artificial boundaries is achieved by addressing the summation of the fluxes of the fine volume faces to their adjacent coarse volume face (fig. 2).

With respect to the relaxation procedure, the modifications to the flux vector splitting and the relaxation scheme at the subgrid interfaces are totally handled by the multigrid algorithm. Adopting the "horizontal" communication mode among subgrids, the fine subgrids (i.e. block 1) are relaxed at the first grid level (fig.1). At the next multigrid level the coarse subgrids (i.e. block 2) are relaxed together with the restricted fine ones, while the block structure of the composite grid is preserved throughout the multigrid cycle.

Due to the block structure, the composite grid has the advantage of a straightforward implementation of vectorization and parallelization. This can be achieved when subgrids are considered totally independent from each other introducing ideas from the domain decomposition theory. Concerning the "vertical" communication mode (ref. 11), each subgrid is solved and relaxed independently for the complete multigrid cycle whereas data exchange among subgrids is permitted only at the start and/or the end of each cycle. The computational domain decomposition (different from the composite block structure) (ref. 8) should be performed according to load balancing and vectorization criteria taking into consideration the hardware configuration. The "vertical" communication mode is preferred from the "horizontal" mode when we deal with parallel processing and is used even though the latter mode has better convergence rates. Using the "vertical" mode the idle and communication time among processors can be considerably decreased.

With respect to the dynamically adaptive multigrid strategy a modified Full Multigrid scheme is applied, starting with a global coarse grid of acceptable grid resolution. After convergence (or after a fixed amount of work) at the current grid, the truncation error is calculated and the solution error is predicted. In regions where the prediction of the error is above a threshold the corresponding volumes are flagged and grouped into rectangular blocks. Afterwards, the domain is decomposed to the appropriate blocks where only those which contain the flagged volumes are refined to the next grid level injecting the coarse grid solution to the refined grid. The refinement procedure continues until the entire computational domain has local truncation errors below a given threshold. Clearly this strategy has the benefit of a continuous iterative procedure without wasting CPU-time on calculations that will not be used after the mesh refinement. Taking advantage of the most accurate available solution the proposed scheme converges straight to the most efficient solution.

## RESULTS AND DISCUSSION

In order to verify the accuracy and to validate the efficiency of the proposed method two transonic inviscid test cases were investigated. The first case is a NACA- 0012 airfoil for Mach 0.80 at angle 1.25 degrees while the second case is a RAE- 2822 airfoil for 0.73 Mach at 2.79 degrees. A Work Unit (W.U.) is defined by the CPU-time required for a global finest grid relaxation sweep in lexicographic order while for a single grid running one time step costs four work units.

For both test cases the grid adaptation procedure as described above was applied. Starting point is two global multigrid levels with  $64 \times 14$  volumes at the finest grid level. The user supplies only the maximum number of the additional grid levels which for both test cases were the same: two refinement grid levels. The truncation error threshold was defined explicitly targeting to a three- fold reduction of the initial error levels. For the convergence criterion the Euclidean norm of the correction vector was employed.

For the first test case (NACA-0012) a comparison between the computed and the predicted truncation error contours is given in figures 3 and 4, resp.. Some differences, not crucial, are due to the incorrect evaluation of the

local accuracy of the solution (p). For the same grid level the actual pressure error contours and the entropy contours in figures 5 and 6 are included. A comparison between the truncation errors and the pressure errors shows their similar distributions. Following the proposed method very accurate results were obtained in comparison to the global solutions as depicted in fig.7. In fig.7 the Mach- distributions along the airfoil are depicted for two global grids (256x56 and 128x28) and one composite grid sharing both grid levels. A comparison of these Mach distributions shows that at regions with the same mesh size the solutions between a global and a local refinement coincide. Additionally, in fig.8 the Mach contours together with the composite grid are given. In fig.9 the efficiency of the BAM method with respect to the single grid and the global multigrid schemes is clearly indicated. A 19-fold and 4-fold acceleration were achieved with respect to the single grid and to the multigrid cases, respectively. The final grid adaptive solution requires 4.5 times fewer volumes (from 14336 to 3200) for practically the same accuracy with a globally refined grid (0.35% discrepancy of the computed Cl).

Similar efficiency was achieved for the second test case. The domain decomposition into subgrids took place after 50 time steps spent on coarser grid levels. Using the same truncation error threshold as in the previous test case, a 17-fold acceleration was achieved with respect to the single grid and a 4.7- fold reduction of the volumes (from 14336 to 3056) for practically the same accuracy (Cl discrepancy is 0.2 %). The convergence histories of the error reduction and the lift coefficient are shown in fig.10 while in fig.11 the final composite grid together with the isomach contours are depicted. It is important that throughout the solution process the multigrid convergence rates were maintained while the overhead for the interface computations was negligible, i.e. only 2 % for a nine block structure with respect to an equivalent global grid.

As no parallel machine was currently available a simulation of the parallelization for the procedure has been attempted in order to evaluate the performance of the two different communication modes for the BAM method. To achieve this, the data exchange was properly adjusted among subgrids according to each communication mode. For the "horizontal" mode (semi-parallel) and the "vertical" mode (parallel) the convergence histories are shown in fig.12. A comparison between the parallel modes and the sequential BAM method shows only a small reduction of the convergence rates for the parallel ones. What is further required for a parallel implementation is a computational domain decomposition of the composite structure based on specific load balancing criteria in order to reduce the idle time among processors.

## CONCLUSIONS

The great advantages of the Block Adaptive Multigrid (BAM) method were exhibited. The incorporation of numerous efficient schemes into the BAM method makes the ultimate target of solving complex problems in just a few work units feasible. At the same time robustness, simplicity and accuracy of the single grid code were maintained at the new method. The extension to viscous and hypersonic three dimensional problems is straight forward though semi coarsening multigrid can be also included. On the other hand, in order to improve the adaptation capabilities, a moving grid point scheme should also be

considered as the grid alignment towards certain flow features is essential in some problems in combination with the present grid refinement procedure.

Although the basic features of the BAM method have been determined and verified, a few other issues remain to be settled. The first is the construction of a data structure which will handle more efficiently the block structure of the composite grid. The second is to approximate more precisely the local order of accuracy of the solution in such a way that the prediction of the solution error would be more accurate. Additionally, the implementation of the BAM method to other solution algorithms and equations is foreseen as the BAM method was designed in the general concept of the finite volume method.

## REFERENCES

1. A.Brandt: Multi-level Adaptive solutions to boundary-value problems, Math. Comp., 31, pp. 333-390.
2. S.McCormick: Multilevel Adaptive Methods for Partial Differential Equations (SIAM, Philadelphia, 1989).
3. W.Joppich: A multigrid algorithm with time-dependent, locally refined grids for solving the nonlinear diffusion equation on a nonrectangular geometry, in Proceedings of the 3rd European Conference in Multigrid methods, eds W.Hackbusch and U.Trottenberg, Int.Ser.Num. Math, 98 (Birkhaeuser Verlag, Basel, 1991) pp. 241-252.
4. K.Y.Fung, J.Tripp and B.Goble: Adaptive refinement with truncation error injection, Comp.Meth.Appl.Mech.Engng., 66 (1987), pp. 1-16.
5. M.J.Berger and A.Jameson: Automatic adaptive grid Refinement for the Euler Equations, AIAA J., 23 (1985), pp. 561.
6. A.Eberle: Characteristic flux averaging approach to the solution of the Euler equations, VKI Lecture series 1987.
7. A.Kanarachos and N.Pantelidis: Block Full Multigrid Adaptive Scheme for the Compressible Euler Equations, Proceedings of the 13th Intl.Conf. on Numerical Methods in Fluid Dynamics, eds. M.Napolitano, F.Sabeta, Lecture Notes in Physics V. 424 (Springer Verlag, Berlin, 1993) pp.265-269.
8. J.Scroggs and J.Saltz: Distributed Computing and Adaptive Solution to Nonlinear PDEs, Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations, eds. R. Glowinski et al, (SIAM, Philadelphia, 1990) pp.409-417.
9. C.Liu: The Finite Volume Element (FVE) and Fast Adaptive Composite Grid Methods (FAC) for the incompressible Navier-Stokes Equations, Proceedings of the Fourth Copper Mountain Conference on Multigrid Methods, eds. J.Mandel et al. (SIAM, Philadelphia, 1989) pp.319-337.

- |  |  |  |  |  |  |    |    |    |    | GRID LEVEL |  |   |
|--|--|--|--|--|--|----|----|----|----|------------|--|---|
|  |  |  |  |  |  | F1 | F2 | F3 | F4 | 1          |  |   |
|  |  |  |  |  |  |    |    | C1 | C2 |            |  | 2 |
|  |  |  |  |  |  |    |    |    |    |            |  | 3 |

474



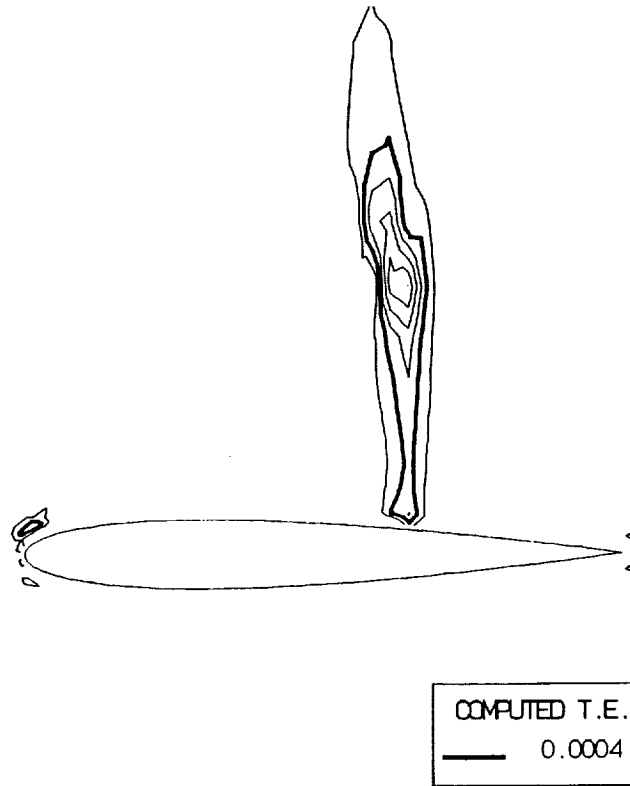


Figure 3. Computed truncation error contours for the finer grid level, CASE 1.

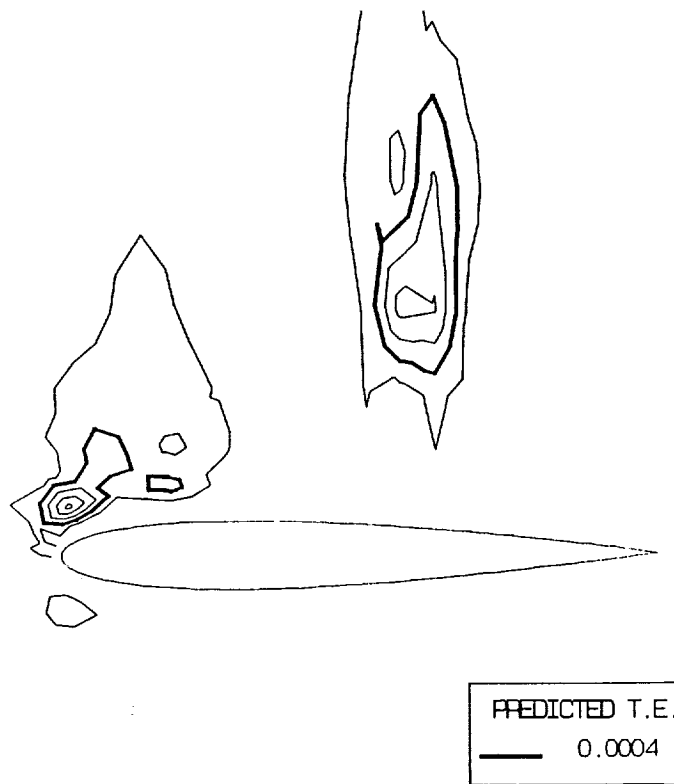


Figure 4. Predicted truncation error contours for the finer grid level, CASE 1.

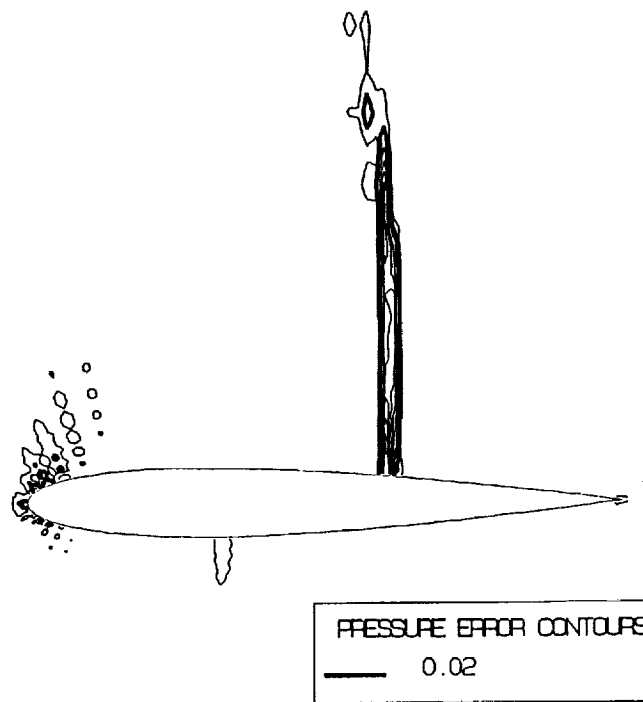


Figure 5. Pressure error contours for the finer grid level, CASE 1.

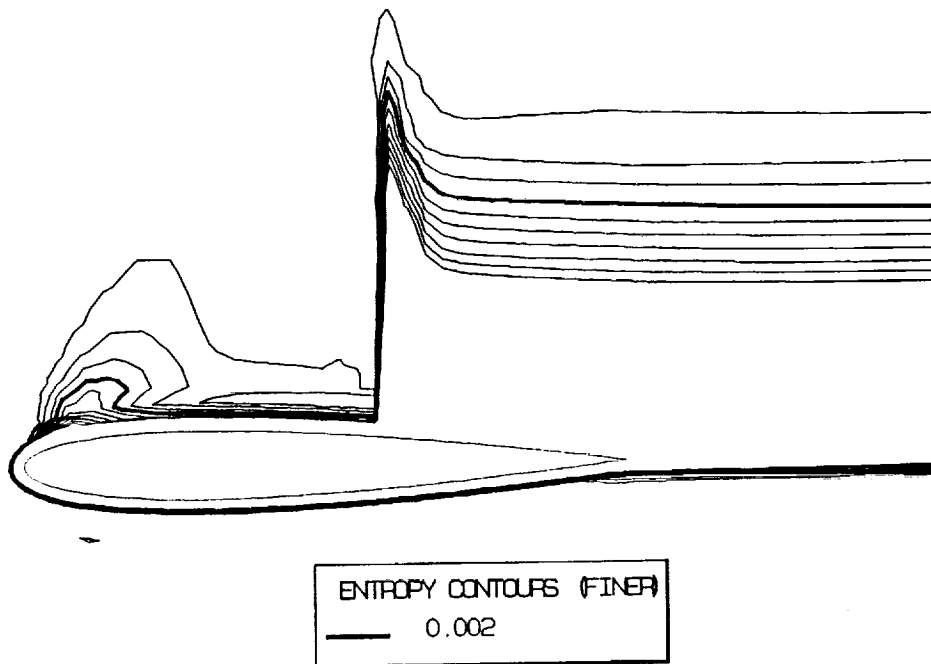


Figure 6. Entropy contours for the finer grid level, CASE 1.

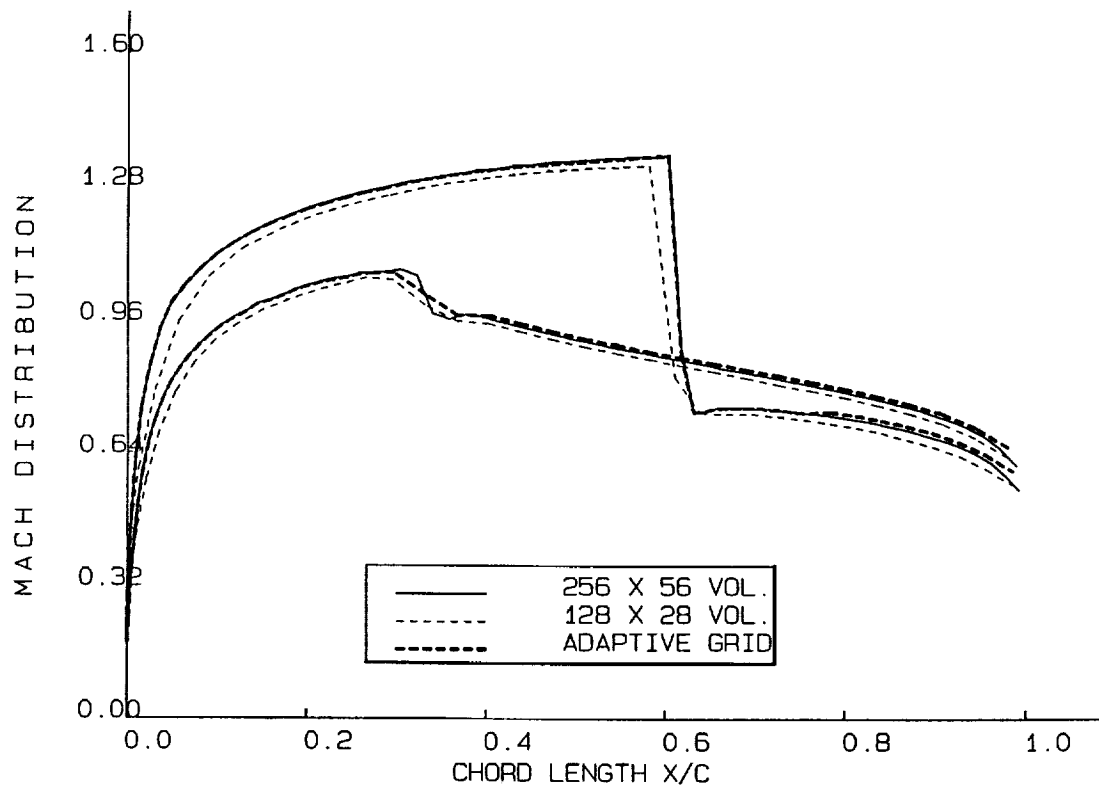


Figure 7. Mach distribution along the airfoil for a global fine, a global coarser and an adaptive composite grid (CASE 1).

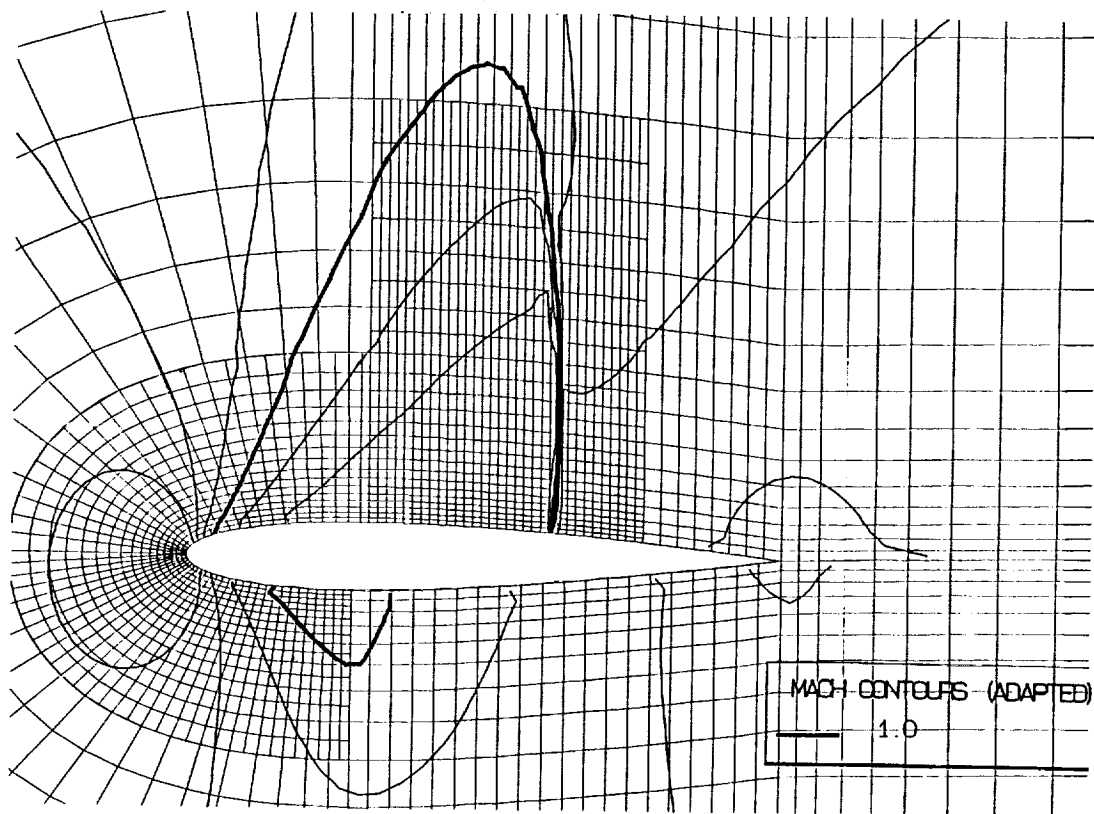


Figure 8. The adaptive composite grid and the corresponding Mach contours (CASE 1).

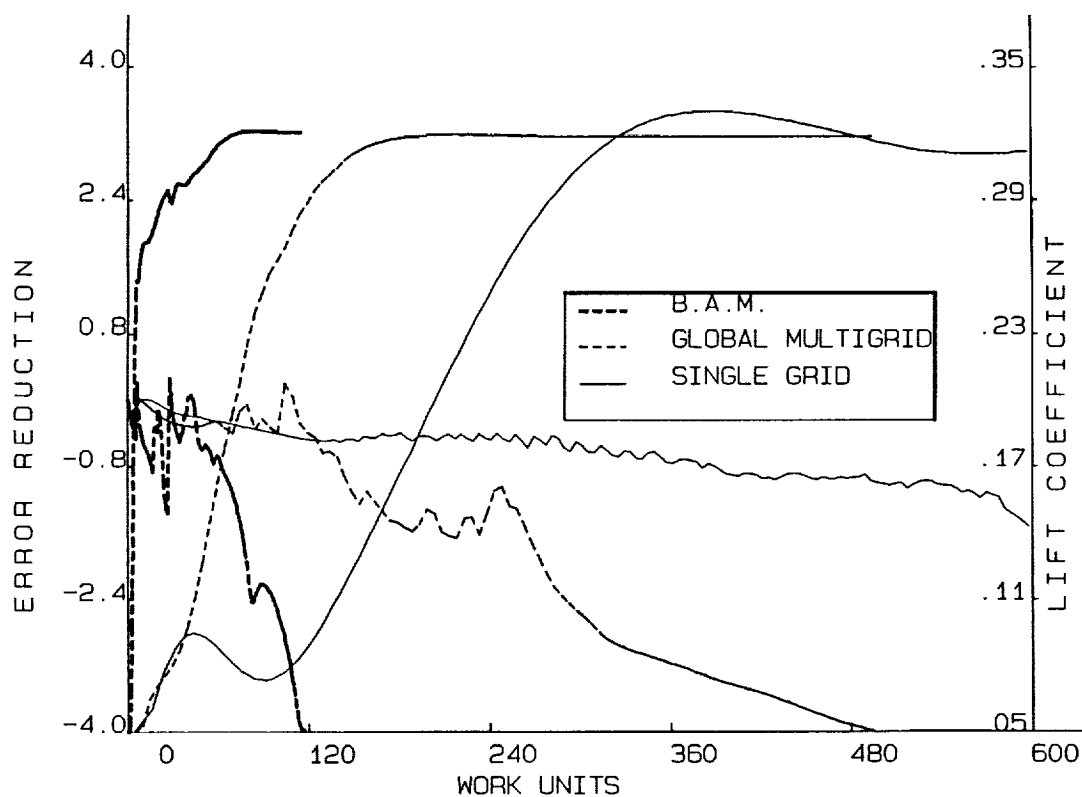


Figure 9. Convergence histories of the lift coefficient and the logarithmic Euclidean error reduction for the single grid, global multigrid and B.A.M. method (CASE 1).

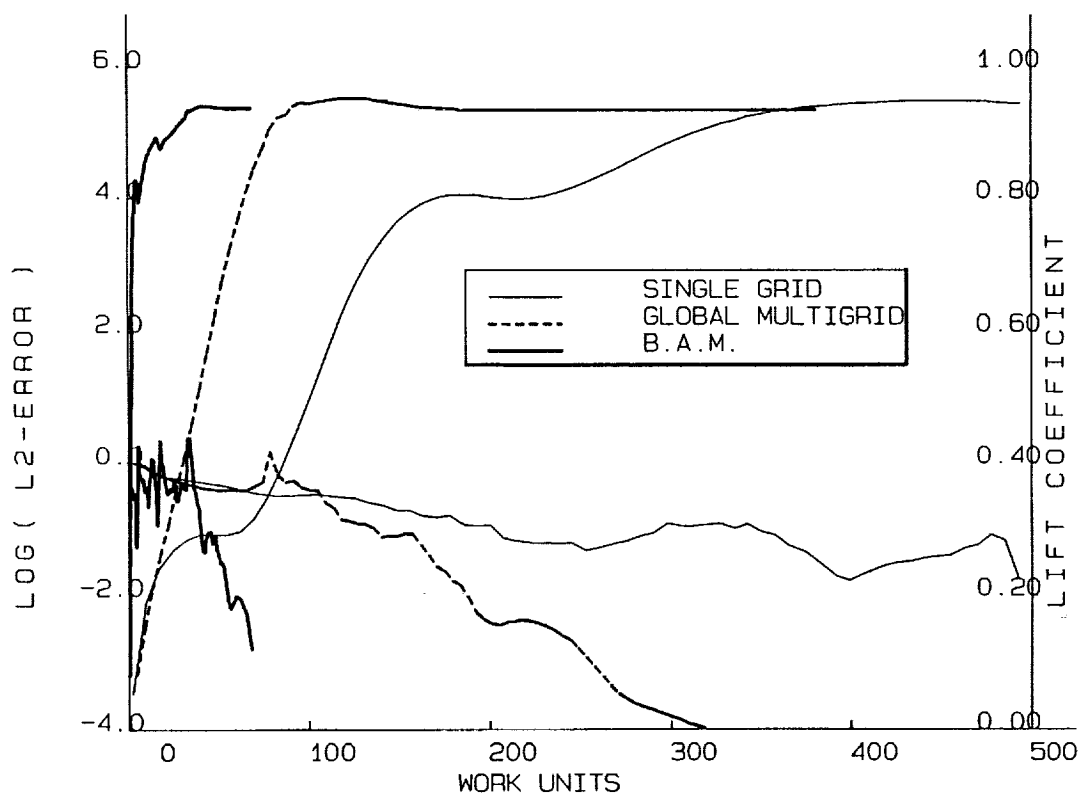


Figure 10. Convergence histories of the lift coefficient and the logarithmic Euclidean error reduction for the single grid, global multigrid and B.A.M. method (CASE 2).

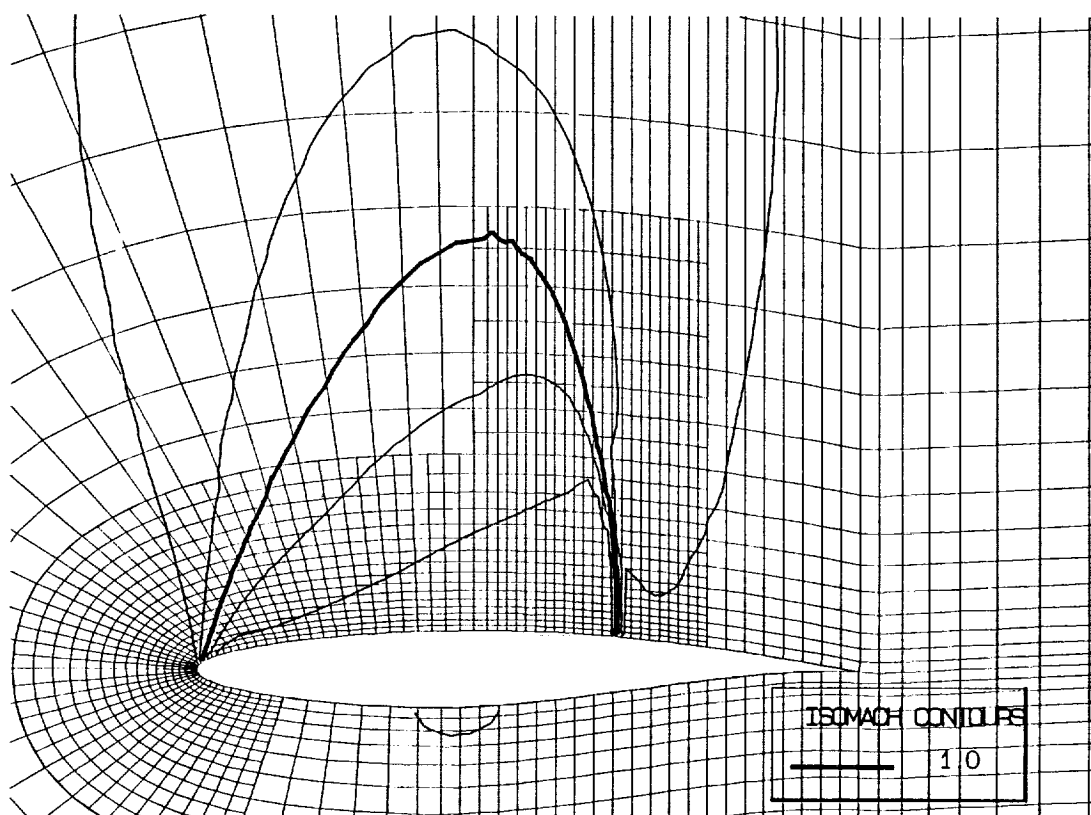


Figure 11. The adaptive composite grid and the corresponding Mach contours (CASE 2).

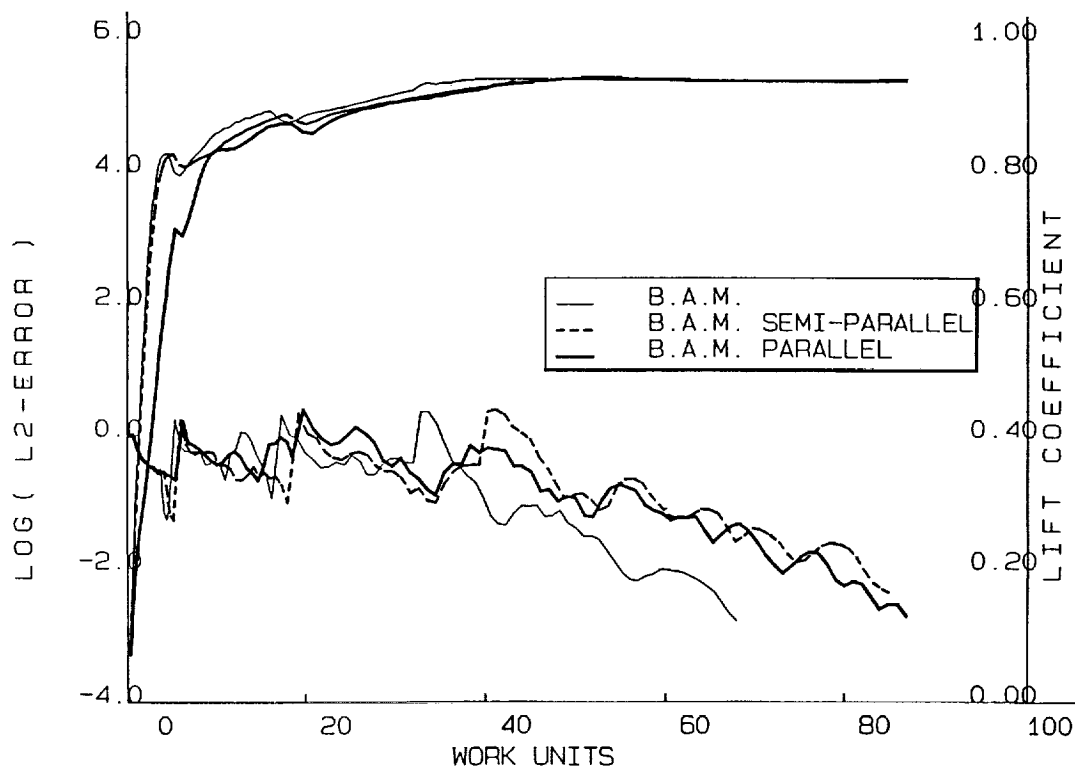
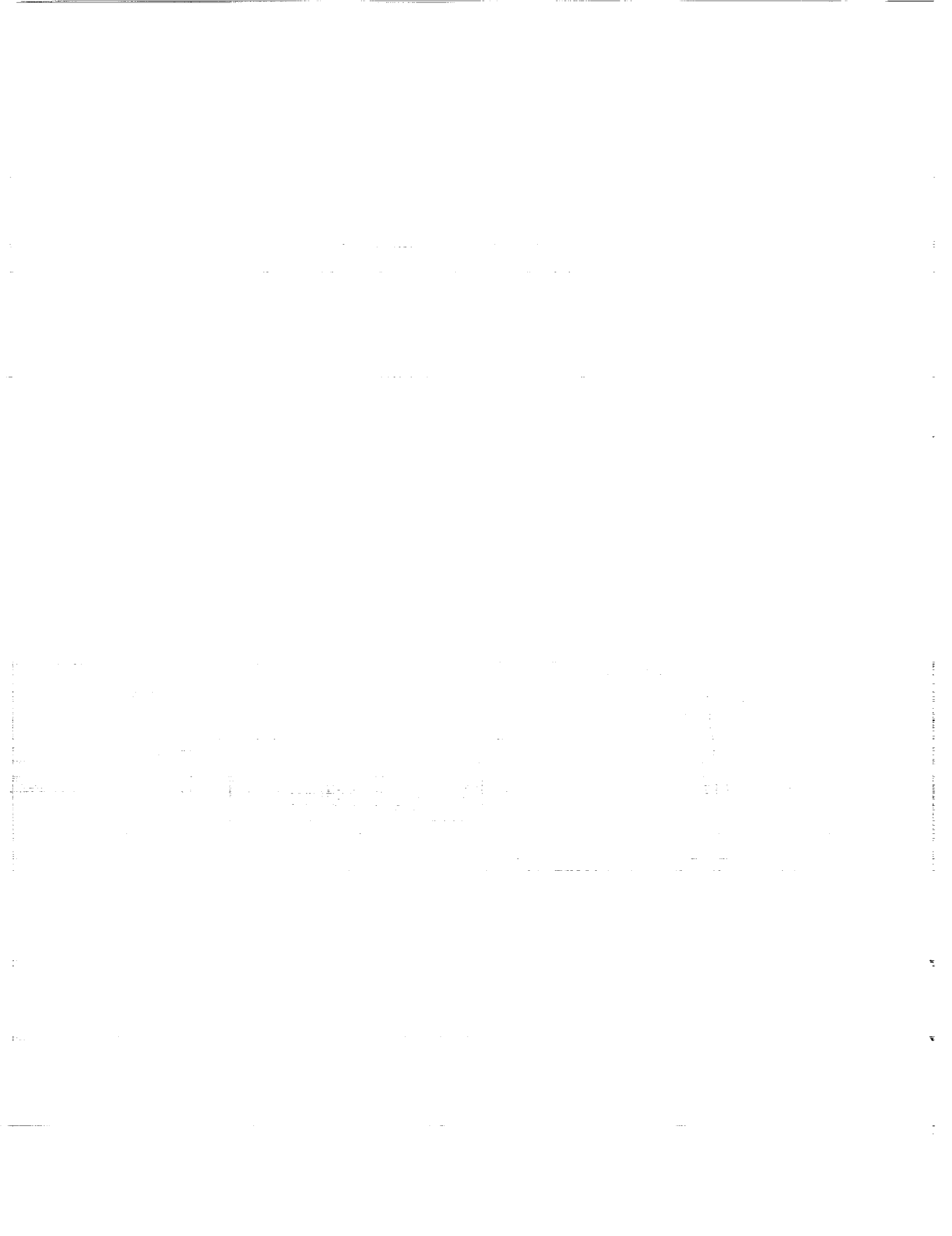


Figure 12. Convergence histories of the lift coefficient and the logarithmic Euclidean error reduction of the B.A.M. method (CASE 2) for sequential processing, "parallel" horizontal and "parallel" vertical schemes.



## MULTIGRID SCHEMES FOR VISCOUS HYPERSONIC FLOWS

R. Radespiel  
DLR, Braunschweig  
Germany

R. C. Swanson  
NASA Langley Research Center  
Hampton, VA 23681

S9-02  
197569  
P-15

## ABSTRACT

Several multigrid schemes are considered for the numerical computation of viscous hypersonic flows. For each scheme, the basic solution algorithm employs upwind spatial discretization with explicit multistage time stepping. Two-level versions of the various multigrid algorithms are applied to the two-dimensional advection equation, and Fourier analysis is used to determine their damping properties. The capabilities of the multigrid methods are assessed by solving two different hypersonic flow problems. Some new multigrid schemes, based on semicoarsening strategies, are shown to be quite effective in relieving the stiffness caused by the high-aspect-ratio cells required to resolve high Reynolds number flows. These schemes exhibit good convergence rates for Reynolds numbers up to  $200 \times 10^6$ .

## INTRODUCTION

In the past several years, multigrid has been used to accelerate the convergence of Navier-Stokes computations for a variety of flow problems at both subsonic and transonic speeds (refs. 1 and 2). More recently, multigrid methods with either central or upwind differencing have been applied to viscous hypersonic flows to achieve convergence rates that approach those obtained at lower Mach numbers and moderate Reynolds numbers ( $Re < 10^7$ ). However, at the higher  $Re$  values experienced by high-speed flight vehicles, a dramatic slowdown occurs in the convergence rate. One reason for this slowdown is the deterioration in the high-frequency damping of the multigrid driving scheme caused by the very high-aspect-ratio cells that occur in the computational mesh in order to resolve the thin boundary layers.

The present paper describes an effort to understand and improve the use of multigrid schemes for the computation of viscous hypersonic flows. First, various two-level multigrid schemes both with and without semicoarsening are introduced. Then we use a Fourier analysis of the schemes, applied to the two-dimensional convection equation, to reveal the behavior of their components. For each multigrid approach, the solver uses an upwind discretization combined with an explicit multistage scheme. We next consider the numerical solution of the Navier-Stokes equations for hypersonic flows. The basic elements of the flow solver for these equations are summarized. Some details concerning the application of

the time-stepping scheme to fine- and coarse-grid problems are presented. The extension of the two-level schemes to multilevel ones is then discussed. Elements of multigrid that are of particular importance for high-speed flow computations are given. In the results section, we consider two different hypersonic flow problems to assess the capabilities of the multigrid schemes. Semicoarsening is shown to be quite effective in relaxing the stiffness that arises from the resolution of thin boundary layers.

## MULTIGRID METHOD AND STRATEGIES

The multigrid approach is based on the full approximation scheme of Brandt (ref. 3). The grid transfer operators are those considered by Jameson (ref. 4). Coarser meshes are obtained by eliminating alternate mesh points in each coordinate direction. Both the solution and the residuals are restricted from fine to coarse meshes. A forcing function is constructed so that the solution on a coarse mesh is driven by the residuals collected on the next finer mesh. The corrections obtained on the coarse mesh are interpolated back to the fine mesh. The multigrid schemes investigated within the present work are displayed in Figure 1. Figure 1(a) shows a two-level scheme with full coarsening. Restriction of the solution from the fine mesh  $(m,n)$  to the coarse mesh  $(m/2,n/2)$  is done by injection, whereas full weighting is used for the restriction of the residuals. Prolongation of the corrections is done by bilinear interpolation. Figure 1(b) shows a scheme with semicoarsening in the different coordinate directions. Again, injection and full weighting are used in the restriction process. The corrections obtained on the coarse meshes are averaged before they are added to the current fine mesh solution which is indicated by the numbers at the "up" arrows. Because of this averaging, half of the individual corrections on the coarse meshes are lost. We, therefore, anticipate that the scheme in Figure 1(a) should be computationally more efficient, provided that enough high-frequency damping can be obtained with the smoothing scheme of the fine mesh. In order to overcome this deficiency of the semicoarsening scheme, two more variants are considered. For the scheme of Figure 1(c), the solutions on the coarse meshes are computed sequentially. Hence, the corrections obtained on the  $(m/2,n)$  mesh can be used to update the  $(m,n/2)$  mesh before time stepping (as indicated by the horizontal arrow). The sequential update of the second coarse mesh allows the full corrections to be passed up to the fine mesh. Note that this multigrid variant is not compatible with the idea of parallel computations. An interesting compromise between the schemes of Figures 1(b) and 1(c) was suggested by Van Rosendale based on the work of ref. 5 (Figure 1(d)). Here, only the corrections common to both of the coarse meshes,  $(m/2,n)$  and  $(m,n/2)$ , are averaged, whereas the corrections to the modes that live either on  $(m/2,n)$  or on  $(m,n/2)$  are passed to the fine mesh in full. This scheme does allow parallel computations for the coarse meshes.

## FOURIER ANALYSIS OF THE SCALAR ADVECTION EQUATION

A crucial factor in constructing an effective multigrid method is the selection of a smoothing or driving scheme. Local mode (Fourier) analysis is generally applied to evaluate possible smoothers on the basis of stability and high-frequency damping properties. The



screening of schemes is often performed with a single-grid analysis. Since a stable single-grid scheme may not be stable for the multigrid process, the behavior of a smoother with a particular multigrid strategy is needed. In addition, the multigrid process can have a substantial impact on the performance of the multigrid method. In fact, as we will demonstrate in this paper, semicoarsening can provide significant improvement, relative to full coarsening, in the damping of the multigrid, especially when a strong mesh anisotropy is present due to the high-aspect-ratio cells.

In ref. 4, Jameson models a multigrid scheme as a multilevel uniform scheme and analyzes the stability of this scheme when applied to the linear advection equation in one space dimension. With the multilevel uniform scheme, fine-grid and coarse-grid corrections are computed at all points of the fine grid. Then, a nonlinear filter is applied to remove the coarse-grid corrections at fine-grid points not contained in the coarse grid. The filtering produces additional errors in the form of a carrier wave with a frequency depending on the fine-mesh spacing. This approach does not allow for the coupling (aliasing) effects due to the restriction operator (fine to coarse grid transfer operator) in the multigrid method. However, it does offer the advantages of simplicity and application to more than two-level schemes. Thus, it allows the rapid comparison of multigrid algorithms. If a multigrid method is unstable or inefficient according to Fourier analysis of the multilevel uniform scheme, then it is probably not a reasonable scheme.

In ref. 6 we consider the scalar two-dimensional advection equation and perform a Fourier analysis of the multilevel uniform scheme for different multigrid strategies. The effects of mesh-cell aspect ratio are included in the analysis. For details of the analysis, see ref. 6. Here, as in ref. 6, a five-stage scheme with three weighted evaluations of the numerical dissipation is used for a solver. The explicit stability limit of this scheme is extended with variable-coefficient implicit residual smoothing, which results in a Courant-Friedrichs-Lewy (CFL) number of 5. A two-level analysis is applied to both full coarsening and semicoarsening strategies. Figure 2 presents contours of the amplification factor  $g$  as a function of Fourier phase angles for the full coarsening and sequential semicoarsening strategies when the mesh-cell aspect ratio ( $AR$ ) was set to 10. Even with this  $AR$ , one can clearly see the improved damping (reduced  $g$ ) in the direction of the long side of the cell with sequential semicoarsening.

## SPATIAL DISCRETIZATION

A finite-volume approach, where the flow quantities are stored at the cell vertices, is used for the spatial discretization of the Navier-Stokes equations. For the convective flux calculation, an auxiliary grid is used, which is defined by connecting the cell centers of the original cells (see Figure 3). The inviscid numerical flux is separated into the sum of an averaged term that corresponds to central differencing and a dissipative term that adapts the discretization stencil in accordance with local wave propagation. The dissipative flux function is based on the second-order-accurate upwind scheme of Yee and Harten (ref. 7). In the case of viscous flows the entropy correction for this scheme must be carefully designed, as discussed in ref. 6. The physical viscous fluxes are approximated by central differences with a local transformation from Cartesian to curvilinear coordinates (ref. 2).

## MULTISTAGE SCHEME FOR THE FINE AND COARSE MESHES

We have observed the need to pair spatial discretization and particular time-stepping schemes for the solution of the Navier-Stokes equation. The most robust choice of spatial discretizations found to this point is to use a second-order upwind scheme on the fine meshes and to set the limiter to zero everywhere on the coarse meshes. An alternative taken in refs. 8 and 9 is to use scalar second-difference dissipation terms on the coarse meshes. This approach turned out to be less robust because the second differences are less diffusive with respect to the acoustic modes; also, the central-difference scheme allows waves to travel upstream in supersonic flow. As indicated previously, a five-stage explicit scheme with three evaluations of dissipation is used for time advancement. Disturbances are most effectively expelled out of the computational domain by using local time stepping and implicit residual smoothing (refs. 8 and 10). The smoothing of the residuals allows the CFL number of the explicit scheme to be as high as 5.75, which extends the stability limit (CFL\*) by a factor of 2.5. The time step is proportional to the ratio of the cell volume to the sum of the spectral radii of the inviscid flux Jacobians in the different coordinate directions.

To stabilize the schemes in regions where the viscous stability limit is more restrictive than the inviscid limit, the coefficients of the implicit residual smoothing operator are locally increased, as outlined in refs. 8 and 9. At strong shocks, however, high Courant numbers are not appropriate. Consequently, an adaptive time step is employed. By using the nondimensional second difference of the pressure as a switch, the value of CFL is locally reduced to approximately 2 at the shock.

## MULTIGRID SCHEMES

For the numerical solution of the Navier-Stokes equations, the two-level strategies presented in Figure 1 are extended to multilevel schemes, as displayed in Figure 4. The only differences between the two-level schemes and the multilevel schemes occur in the restriction process. Whenever two "down" arrows meet at a coarse mesh, averaging is used to obtain the restricted variable. The multilevel arrangement of the coarse meshes, shown in Figure 4(b), was first given by Mulder (ref. 11), who used semicoarsening to solve the flow alignment problem. Suitable coordinate meshes for thin boundary layers exhibit mostly cells with high aspect ratios in the surface-aligned direction. In this paper, other variants of semicoarsening, which are computationally cheaper than the semicoarsening schemes shown in Figure 4, are also considered for these situations.

One may notice that the central restriction and prolongation operators discussed previously allow for upstream propagation of disturbances in supersonic flow. Furthermore, the corrections given by the standard multigrid scheme near strong shocks lead to divergence of the calculation, especially when free-stream initial conditions are used. Therefore, the restriction operator is damped by using

$$R_{i,j} = \max(1 - \epsilon_{i,j}^{(n)}, 0) \bar{R}_{i,j}, \quad (1)$$

where  $\bar{R}_{i,j}$  is the standard restriction operator and  $\epsilon_{i,j}^{(n)}$  is a switch to detect strong

shocks, and

$$\epsilon_{i,j}^{(n)} = k^{(n)} \max (\nu_i, \nu_{i+1}, \nu_{i-1}, \nu_j, \nu_{j+1}, \nu_{j-1}), \quad (2)$$

$$\nu_i = \left| \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{p_{i-1,j} + 2p_{i,j} + p_{i+1,j}} \right|, \quad \nu_j = \left| \frac{p_{i,j-1} - 2p_{i,j} + p_{i,j+1}}{p_{i,j-1} + 2p_{i,j} + p_{i,j+1}} \right|, \quad (3)$$

where  $p$  denotes pressure. The damping coefficient  $k^{(n)}$  is given a value of approximately 1 in the start-up phase of the multigrid process and is decreased to a value of about 0.4 at later cycle numbers to allow for good asymptotic convergence rates. Such a local damping with a  $k^{(n)}$  that does not vanish is in line with the restriction damping of Koren and Hemker (ref. 12), who based their damping coefficients on a more physical analysis.

A fixed V-type cycle with time stepping only on the way down is used to execute the multigrid strategies described above. The robustness of the overall scheme is improved by smoothing the resultant coarse-mesh corrections before they are passed to the finest mesh. The smoothing reduces the high-frequency oscillations introduced by the linear interpolation of the coarse-mesh corrections. The implicit residual smoothing procedure with constant coefficients of around 0.1 is used for this smoothing. Also, the application of full multigrid (FMG) provides a well-conditioned starting solution for the finest mesh that is considered.

## NUMERICAL RESULTS

Two different hypersonic flow cases are used to assess the capabilities of the multigrid schemes. These are laminar Mach 10 ( $M = 10$ ) flow over a compression ramp and turbulent flow over a slender forebody at high Reynolds numbers. Table 1 gives a summary of the geometries and the flow parameters of the test cases. In this table,  $T_{inf}$  is the dimensional free-stream temperature, and  $T_w$  is the specified wall temperature. Also, the finest grid used for each flow computation is characterized by the streamwise and normal leading-edge spacings  $\Delta s_{le}$ ,  $\Delta n_{le}$ , with the normal spacing  $\Delta n_{te}$  at the end of the geometry.

The flow over the compression ramp is identical to case 3.2 of the Workshop on Hypersonic Flows for Reentry Problems, Part II, held in Antibes, France, in 1991. This allows comparisons with the performance of other computational methods published in ref. 13. Figure 5 displays the coordinate mesh generated for this test case. The low Reynolds number allows for a mesh with moderate aspect ratios between 5 and 50 near the wall. The  $129 \times 81$  mesh is successively coarsened down to  $9 \times 6$ , which yields 9 grid levels with semicoarsening and 5 levels with full coarsening. The semicoarsening strategy is expected to eliminate most of the stiffness associated with aspect ratio. The converged flow solution is shown in Figure 6 for the  $129 \times 81$  and  $65 \times 41$  grids. The computed extent of separation in the corner is somewhat smaller for the coarse mesh than for the fine mesh. The fine-mesh results agree well with grid-converged computations published in ref. 14.

In the next figures, we investigate the performance of the different multigrid schemes. For this purpose, computations were started from a solution that was converged to about plotting accuracy. Results from the different schemes of Figure 4 are compared in Figure 7. The numbers indicate the final convergence rate  $r$  of the schemes and the rate of data processing ( $RDP$ ) on a CRAY-YMP to advance one grid point by one multigrid cycle.

The sequential semicoarsening scheme (Figure 4(c)) gives by far the best convergence rate. For this scheme, the effect of the modifications in the multigrid strategies of Figure 4 is investigated in Figure 8. The meshes obtained by full coarsening and by semicoarsening in the direction normal to the wall are both important in achieving good convergence rates. From Figures 7 and 8, we conclude that semicoarsening with a selected number of coarse meshes is most effective for this flow problem. Semicoarsening is about 2.4 times faster than full coarsening, which does a surprisingly good job because of its low work count. In particular, the multigrid scheme with sequential semicoarsening converges (i.e., the residual was reduced 3 orders of magnitude) in roughly 33 sec (CPU time) on a Cray-YMP, which is 10 times faster than the single-mesh scheme.

The flow over a slender forebody is chosen to represent a generic configuration that corresponds to a high-speed civil transport aircraft or an air-breathing space transportation system with low wave drag. The high Reynolds numbers yield thin boundary layers, which can only be resolved with highly clustered coordinate meshes and large-aspect-ratio cells. The mesh used for the present investigations is displayed in Figure 9. The cells near the wall have aspect ratios up to 25000. The flow computations were done with fixed transition at 2 percent chord and with the assumption of an adiabatic wall. Figure 10 shows the Mach contours for the mesh of  $256 \times 96$  cells, and Figures 11 and 12 show the solution obtained on three successively refined meshes. Both the distributions of the skin friction and the wall temperature are accurately computed, even with only 25 points in the normal direction.

Next we examine the convergence behavior of the multigrid schemes. The fine mesh with  $257 \times 97$  points allows 11 grid levels to be used with semicoarsening. The full diamond-shaped tree of coarse meshes cannot be run because the time-stepping scheme is not well suited to handle the extreme aspect ratios that occur on the coarse meshes. With the proper half of the diamond, which includes the meshes with relatively low aspect ratios, the numerical solution converges. Figure 13 displays a comparison of the different multigrid strategies. The computations are started from a preconverged solution. Again, the scheme with sequential semicoarsening converges best. The differences between the multigrid schemes for this case, which has cells with very high aspect ratios, are larger than for the ramp flow. The final convergence rate of the scheme with sequential semicoarsening is 15 times better than the rate with full coarsening. A comparison of the performance for the complete FMG process is given in Figure 14. The sequential semicoarsening scheme takes 194 cycles and 570 sec to reduce the averaged residuals to  $10^{-2}$  on the fine mesh. The scheme with full coarsening takes 1024 cycles and 1430 sec, and the single mesh code takes 7762 time steps and 6190 sec to achieve the same convergence level. Note that residuals of  $10^{-2}$  correspond to a solution that is converged within plotting accuracy. If we compared computer times to reach lower levels of residuals instead, then the results would have been even better for the multigrid scheme with semicoarsening.

## CONCLUSIONS

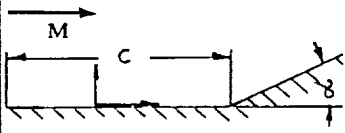
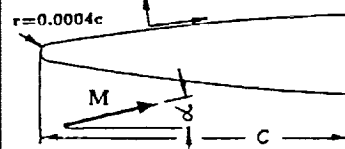
New multigrid schemes for hypersonic flow computations have been investigated. The basic solution algorithm employs upwind discretization and explicit multistage time stepping. Various multigrid schemes with semicoarsening are introduced to overcome the

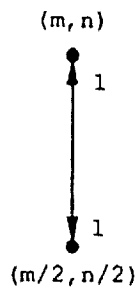
stiffness that results from the high-aspect-ratio mesh cells used to resolve viscous flows. The basic components of the algorithm are examined with a Fourier stability analysis applied to the two-dimensional advection equation. Both the results of the Fourier analysis and the computations of high Reynolds number flows suggest that the semicoarsening approach is effective. The convergence rates shown for hypersonic viscous flows are similar or even better than those previously published for the transonic regime in refs. 1 and 2. Further work is required to make the computational scheme less expensive. This need for more research is particularly true for the coarse meshes used within the semicoarsening approach, which make up the major portion of the overall work count of the scheme.

## REFERENCES

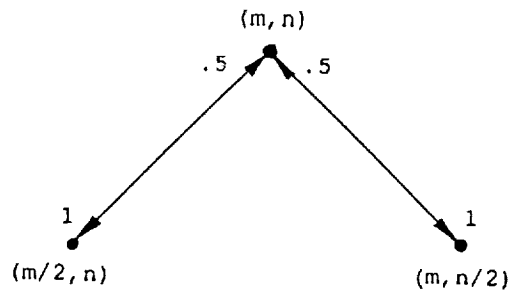
1. Radespiel, R.; Rossow, C.; and Swanson, R. C.: An Efficient Cell-Vertex Multigrid Scheme for the Three-Dimensional Navier-Stokes Equations. *AIAA J.*, vol. 28, no. 8, 1990, pp. 1464-1472.
2. Swanson, R. C.; and Radespiel, R.: Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations. *AIAA J.*, vol. 29, no. 5, 1991, pp. 697-703.
3. Brandt, A.: Multi-Level Adaptive Solutions to Boundary-Value Problems. *Math. Comp.*, vol. 31, no. 138, 1977, pp. 333-390.
4. Jameson, A.: Multigrid Algorithms for Compressible Flow Calculations. *Lecture Notes in Mathematics*, vol. 1228, Springer-Verlag, 1986, pp. 166-201.
5. Naik, N. H.; and Van Rosendale, J.: The Improved Robustness of Multigrid Elliptic Solvers Based on Multiple Semicoarsened Grids. *SIAM J. Numer. Anal.*, vol. 30, no. 1, 1993, pp. 215-229.
6. Radespiel, R.; and Swanson, R. C.: Progress with Multigrid Schemes for Hypersonic Flow Problems. NASA CR-189579, Dec. 1991.
7. Yee, H. C.; and Harten, A.: Implicit TVD Schemes for Hyperbolic Conservation Laws in Curvilinear Coordinates. *AIAA J.*, vol. 25, 1987, pp. 266-274.
8. Turkel, E.; Swanson, R. C.; Vatsa, V. N.; and White, J. A.: Multigrid for Hypersonic Viscous Two- and Three-dimensional Flows. AIAA Paper 91-1572-CP, 1991.
9. Radespiel, R.; and Kroll, N.: Multigrid Schemes with Semicoarsening for Accurate Computations of Hypersonic Viscous Flows. Tagungsband 7, DGLR-Fachsymposium "Strömungen mit Ablösung", Aachen, Nov. 7-9, 1990, DGLR-Bericht 91-5, 1990.
10. Swanson, R. C.; Turkel, E.; and White, J. A.: An Effective Multigrid Method for High-Speed Flows. *Commun. Appl. Numer. Methods*, vol. 8, no. 9, 1992, pp. 671-681.
11. Mulder, W. A.: A New Multigrid Approach to Convection Problems. *J. of Comput. Phys.*, vol. 83, 1989, pp. 303-323.
12. Koren, B.; and Hemker, P. W.: Damped, Direction-Dependent Multigrid for Hypersonic Flow Computations. *Appl. Numer. Math.*, vol. 7, 1991, pp. 309-328.
13. Abgrall, R. et al., ed.: *Hypersonic Flows for Reentry Problems*, vol. 3. Springer-Verlag, 1992.
14. Thomas, J. L.: An Implicit Multigrid Scheme for Hypersonic Strong-Interaction Flowfields. *Commun. Appl. Numer. Methods*, vol. 8, no. 9, 1992, pp. 683-693.

Table 1. Flow and geometric parameters for test cases

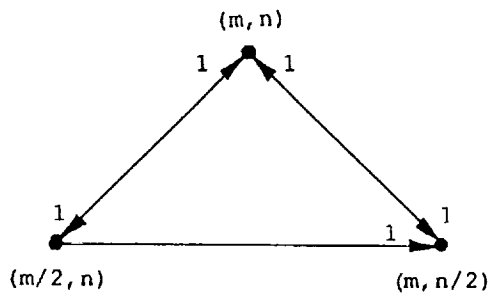
Flow Case	$M$	$\alpha$	$Re_c$	$T_{inf}$	$T_w/T_{inf}$	No.Pts.	$\Delta s_{le}/c$	$\Delta n_{le}/c$	$\Delta n_{te}/c$
	10	20°	18119	52K	5.57	129x81	0.004	0.0008	0.0008
	7	5°	2.E8	100K	adiab.	257x97	4.4E-5	2.E-7	2.E-6



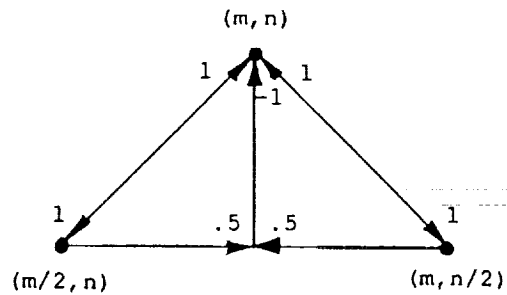
(a) Full coarsening.



(b) Semicoarsening with simple averaging.

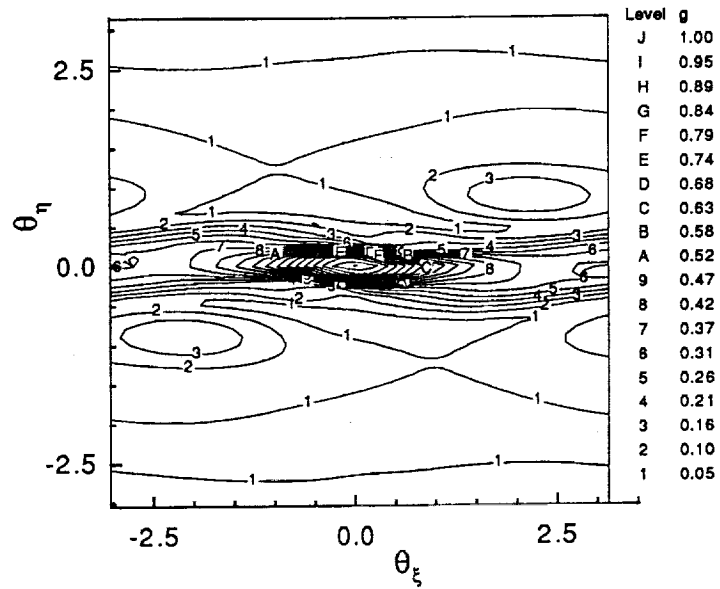


(c) Sequential semicoarsening.

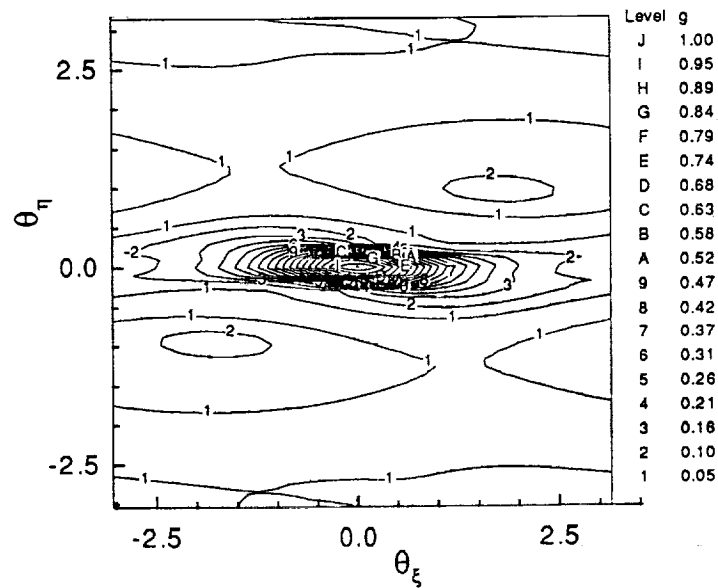


(d) Semicoarsening with selective averaging.

Figure 1. Two-level multigrid schemes investigated in present work.



(a) Two levels, full coarsening, AR = 10 (CFL = 5.0, CFL\* = 2.4).



(b) Two levels, sequential semicoarsening, weights = 1.0, AR = 10 (CFL = 5.0, CFL\* = 2.4).

Figure 2. Contour plots of amplification factor for 5-stage Runge-Kutta scheme with first-order upwind approximation and 3 evaluations of dissipation (coefficients : 0.2742, 0.2067, 0.5020, 0.5142, 1.0).

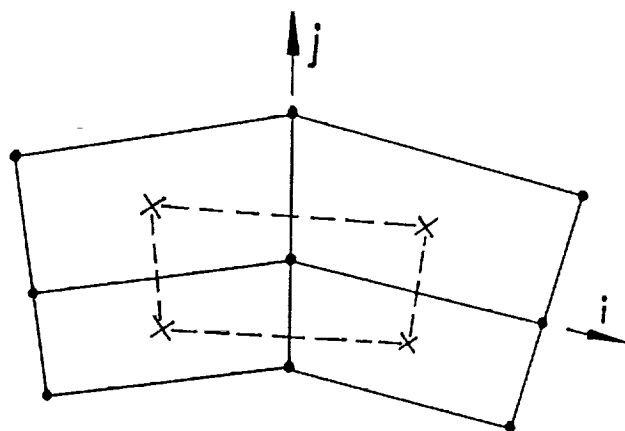
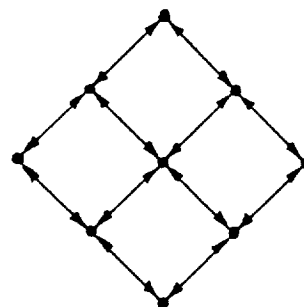


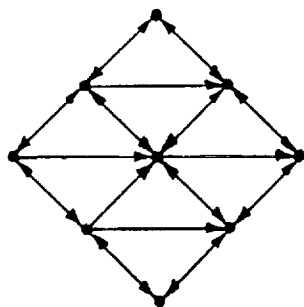
Figure 3. Control volume for nodal-point scheme.



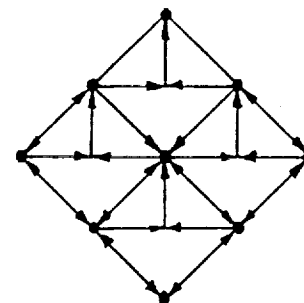
(a) Full coarsening.



(b) Semicoarsening with simple averaging.



(c) Sequential semicoarsening.



(d) Semicoarsening with selective averaging.

Figure 4. Multilevel schemes.



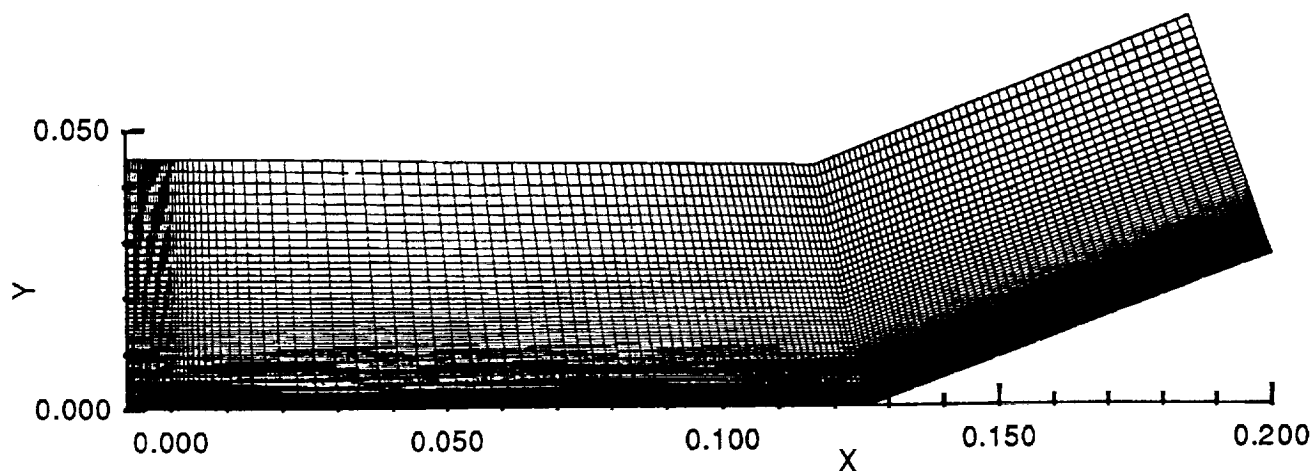
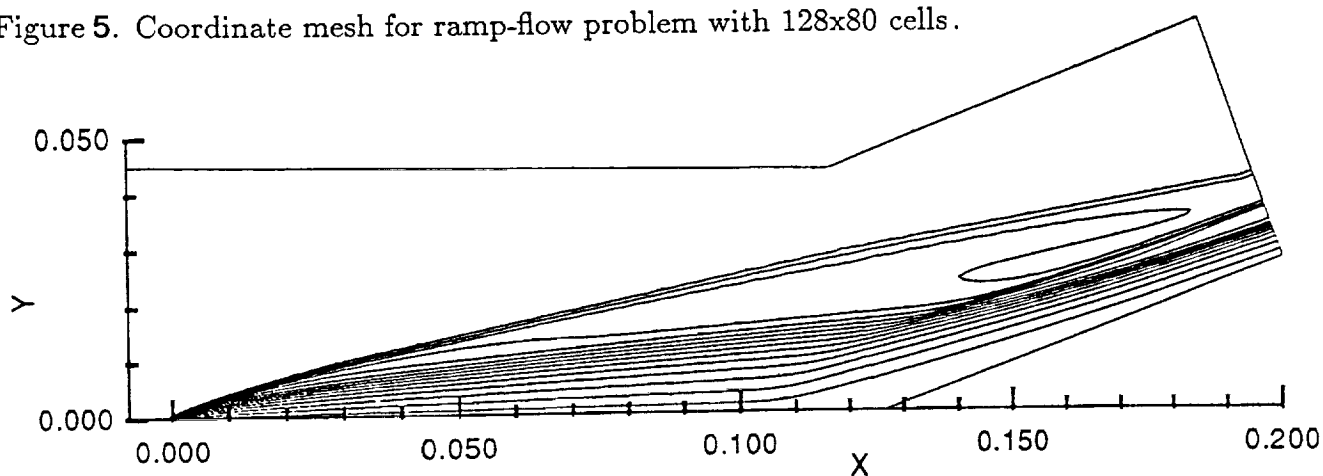
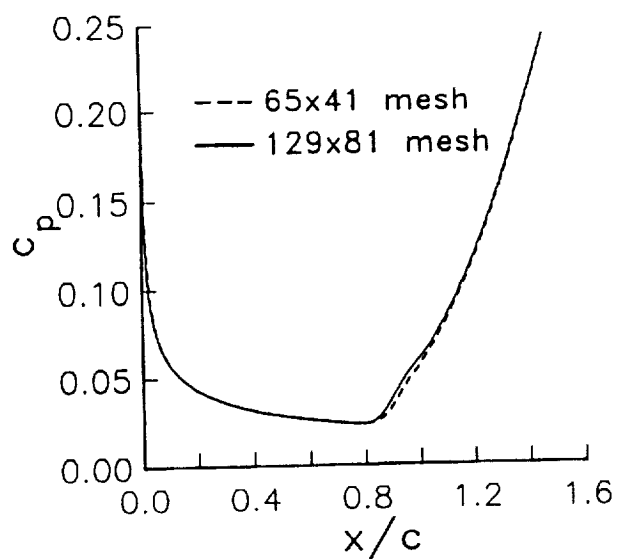


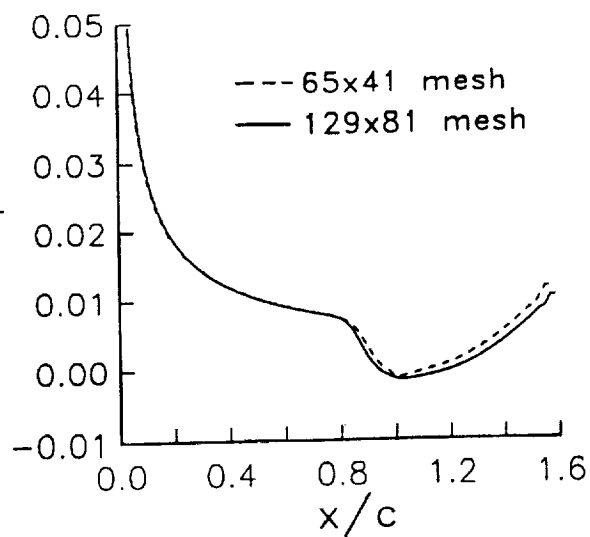
Figure 5. Coordinate mesh for ramp-flow problem with 128x80 cells.



(a) Mach contours.



(b) Pressure coefficient.



(c) Skin friction.

Figure 6. Flow solution for ramp-flow problem.

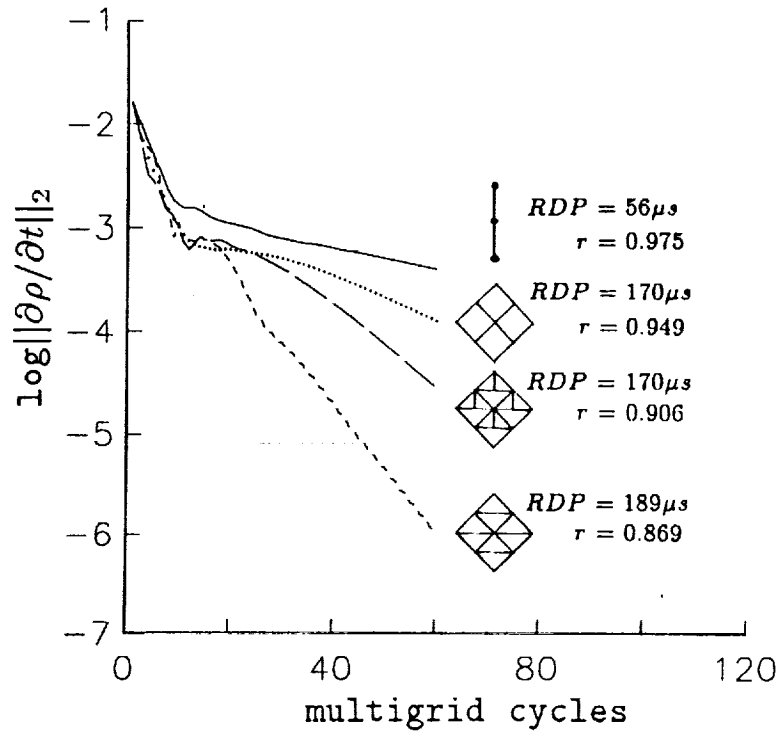


Figure 7. Influence of multigrid strategies on convergence for ramp-flow problem.

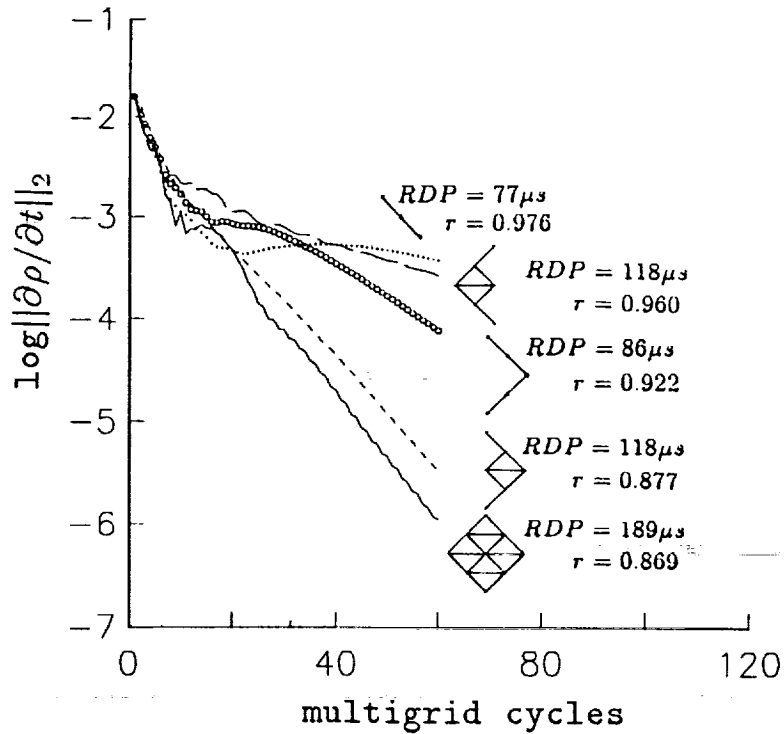


Figure 8. Influence of selected coarse meshes on convergence for ramp-flow problem.

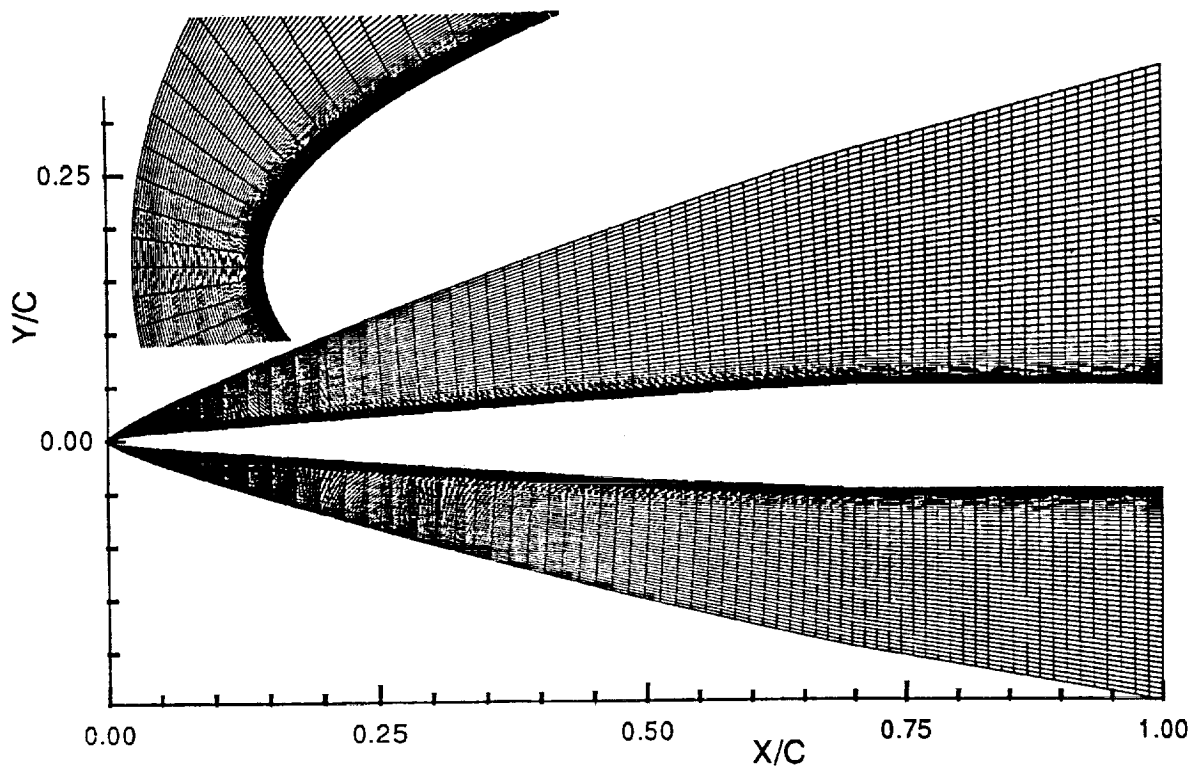


Figure 9 . Coordinate mesh for forebody with 256x96 cells .

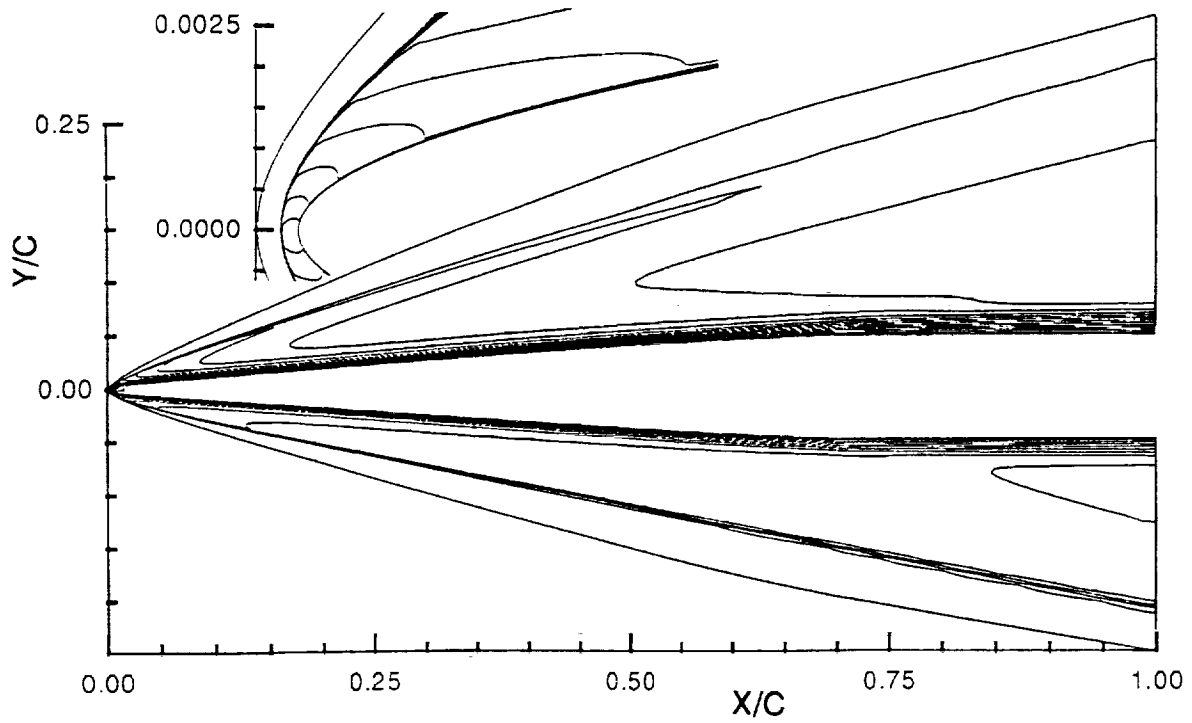


Figure 10. Mach contours for turbulent forebody.

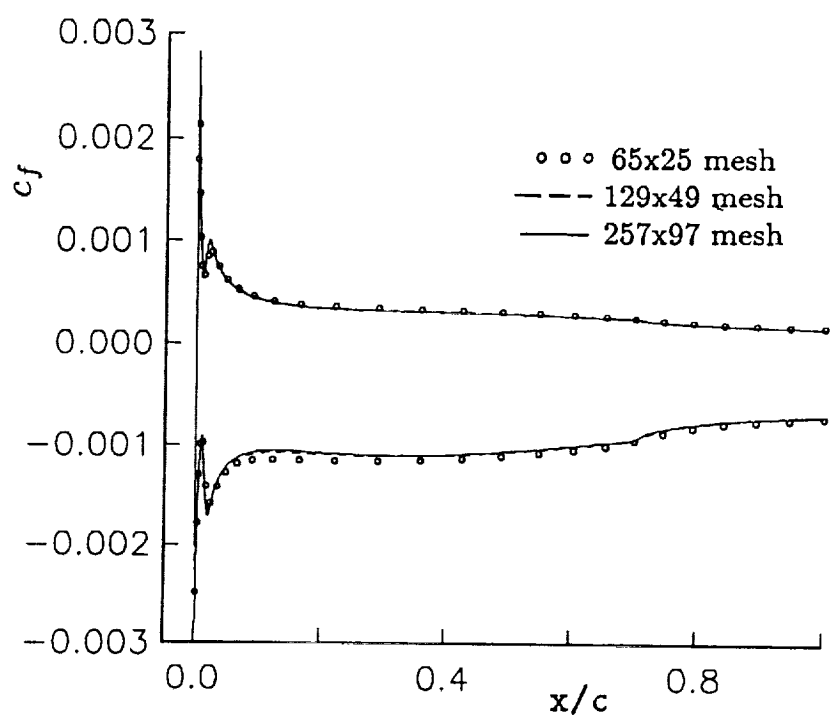


Figure 11. Distribution of skin friction along forebody.

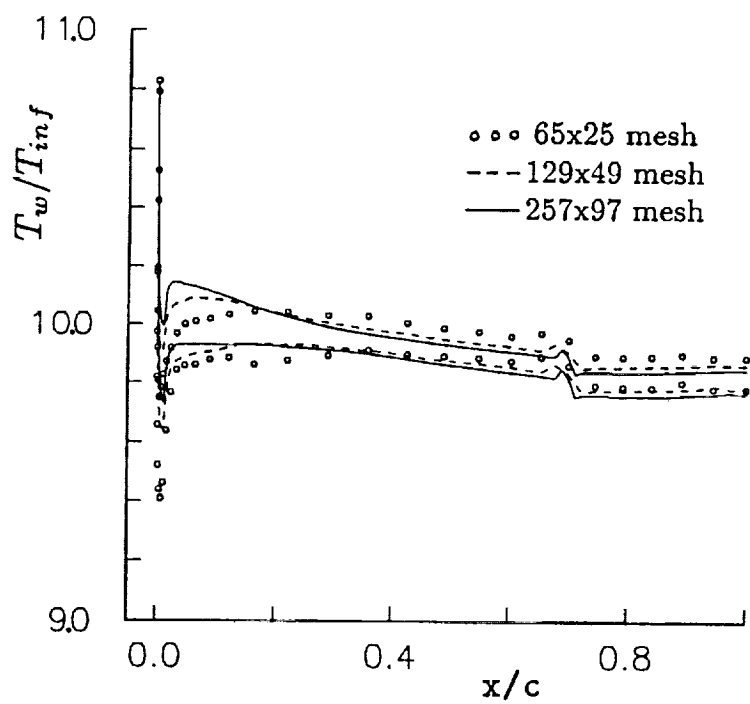


Figure 12. Distribution of adiabatic wall temperature along forebody.

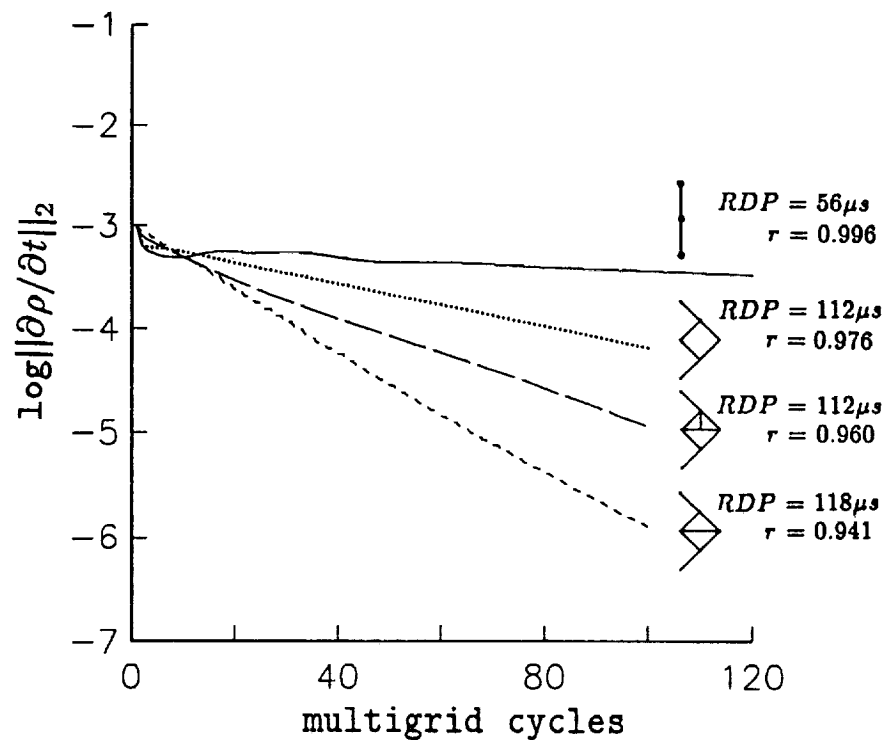


Figure 13. Influence of multigrid strategy on convergence for forebody, mesh 256x96.

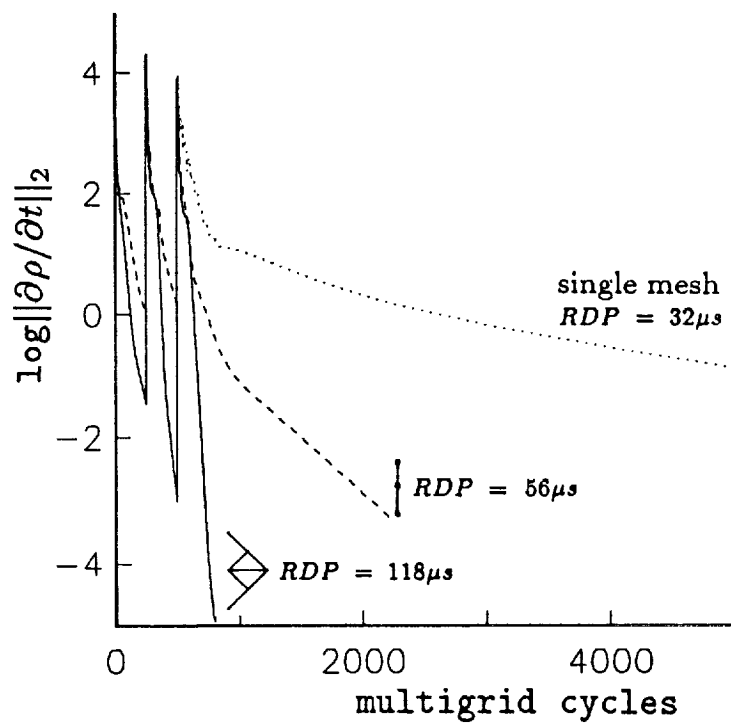


Figure 14. Convergence histories for single-mesh time stepping and multigrid with sequential semicoarsening.



LAYOUT OPTIMIZATION WITH ALGEBRAIC  
MULTIGRID METHODS

Hans Regler and Ulrich Rüde  
 Institut für Informatik  
 Technische Universität München  
 Arcisstr. 21, D-8000 München 2, Germany  
 e-mail: regler/ruede @informatik.tu-muenchen.de

510-64  
 197570  
 p. 15

## SUMMARY

Finding the optimal position for the individual *cells* (also called functional *modules*) on the chip surface is an important and difficult step in the design of integrated circuits. This paper deals with the problem of *relative placement*, that is the minimization of a quadratic functional with a large, sparse, positive definite system matrix. The basic optimization problem must be augmented by constraints to inhibit solutions where cells overlap. Besides classical iterative methods, based on *conjugate gradients* (CG), we show that *algebraic multigrid methods* (AMG) provide an interesting alternative. For moderately sized examples with about 10000 cells, AMG is already competitive with CG and is expected to be superior for larger problems. Besides the classical "multiplicative" AMG algorithm where the levels are visited sequentially, we propose an "additive" variant of AMG where levels may be treated in parallel and that is suitable as a preconditioner in the CG algorithm.

## THE PLACEMENT PROBLEM IN INTEGRATED CIRCUIT LAYOUT OPTIMIZATION

In this paper we present some results of research in *algebraic multigrid methods* (AMG). Our interest in these methods is motivated by an application arising in the layout optimization for integrated circuits. Modern integrated circuits consist of several millions of transistors. The layout optimization for an integrated circuit is usually based on grouping the transistors into *cells* (also called functional *modules*) like NAND/NOR-gates. This leads to the problem of finding the optimal location (placement) for hundreds of thousands of such cells on the chip surface. The goal of this optimization is to find a design that uses as little surface area as possible and that minimizes the time delay caused by long connections between cells. Short connections are desirable, because they permit higher clock rates and thus faster chips.

Generally, finding the *optimal* layout for a given functional description of an integrated circuit is a formidable task. From a mathematical point of view the problem begins with the modeling of the above informal optimality conditions. Furthermore, cells cannot be positioned freely on the chip surface. Clearly, they must not overlap, so that we must consider their individual size and shape. Additionally, the manufacturing process introduces constraints on the locations permitted.

\*This research is supported by the SFB 0342 of the Deutsche Forschungsgemeinschaft (DFG)

Our research is done in the context of GORDIAN, a state-of-the-art layout synthesis package<sup>†</sup> that has been developed at the *Institute for Electronic Design Automation, Technische Universität München*, see Kleinhans, Sigl, and Johannes [1, 2]. Within this package, the placement problem is handled by breaking it into two separate steps, the *relative placement* and the *final placement*.

The purpose of the relative placement step is to provide a good initial guess for the final placement by finding the *global* optimum of a sequence of problems with a simplified optimality condition. After relative placement, only *local effects* are considered in the final placement, much simplifying the task of positioning a cell within the constraints.

The global relative placement optimization is based on a *force model* where connections between cells are weighted according to their Euclidean length. Modules are connected by *signals* that can be interpreted as abstract connections of the cells. Implicitly, the positions of the signals are also subject to the optimization process.

The functional in the relative placement optimization is quadratic with a positive definite M-matrix  $C$  whose entries represent the graph of connections between the cells and signals. Mathematically, the problem can be stated as

$$\min_{x \in \mathbb{R}^n} x^T C x - 2b^T x, \quad (1)$$

where  $x, b \in \mathbb{R}^n$ , and  $C \in \mathbb{R}^{n \times n}$ .

An unaugmented minimization of (1), however, tends to cluster the cells in the center of the chip. This is unrealistic, because there is too much overlap between the cells so that the final placement step would not be able to find acceptable positions for the cells. Therefore, the optimization is augmented with linear constraints of the form

$$Ax = d, \quad (2)$$

that specify *centers of gravity* for groups of cells. These constraints are introduced successively by recursively *partitioning* the cells into groups with equal overall cell surface area, and assigning their center of gravity to subdomains of the chip surface. This is illustrated in Figs. 1 and 2, where the results after five successive partitioning steps with 1, 2, 4, 8, and 16 constraints are shown; see also Regler [3].

## THE AMG ALGORITHM OF RUGE AND STÜBEN

Our application leads to a large, sparse, positive definite system of equations, which is in no way related to a partial differential equation. A typical matrix structure is displayed in Figure 4. The placement optimization program GORDIAN presently uses a preconditioned conjugate gradient method for the minimization in the relative placement step. We now study the suitability of algebraic multigrid as an alternative. Classical, *geometric* multigrid methods have been very

<sup>†</sup>In fact, GORDIAN compared favorably at the 1992 "TimberWolf Hunt", an international competition for placement algorithms



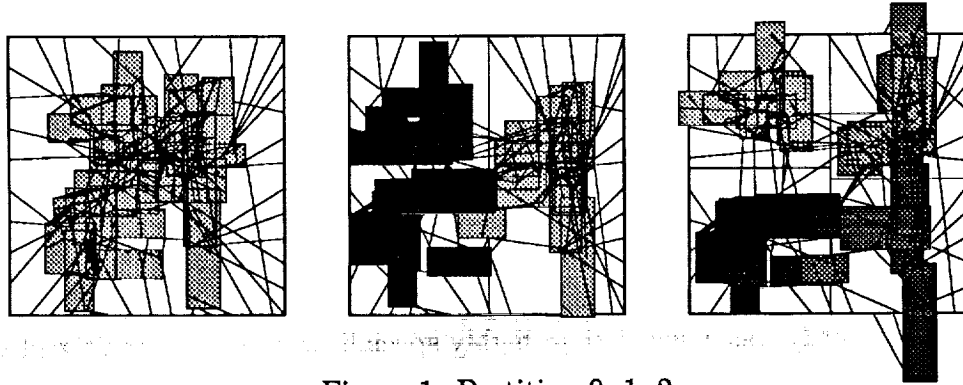


Figure 1: Partition 0, 1, 2

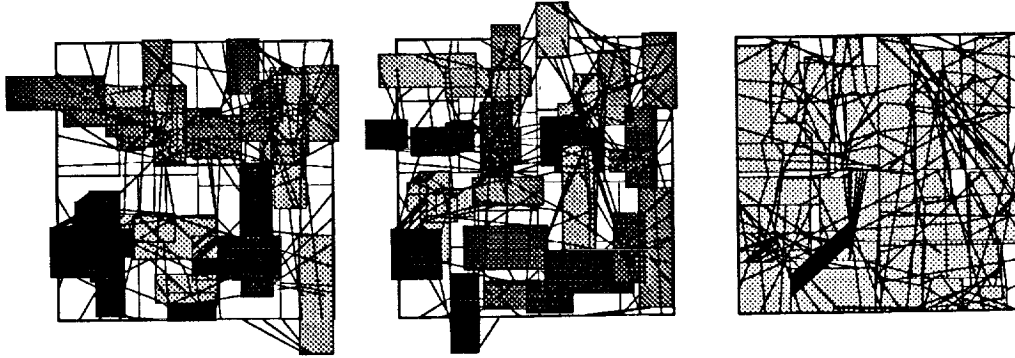


Figure 2: Partition 3, 4, and final placement

successful for solving (1) when the matrix originates from the discretization of elliptic partial differential equations. Here, however, we need an *algebraic multigrid method* that works as a black-box solver given only the system matrix  $C$  and the right hand side vector  $b$ .

In general, the key to multigrid methods is a family of smaller, *coarse level* systems

$$\min_{x^k \in \mathbb{R}^{n^k}} (x^k)^T C^k x^k - 2(b^k)^T x^k, \quad (3)$$

for  $k = 1, 2, \dots, K$ , where the superscript denotes the level and where  $x^k, b^k \in \mathbb{R}^{n^k}$ , and  $C^k \in \mathbb{R}^{n^k \times n^k}$ , and where the dimensions  $n^k$  form a decreasing sequence

$$n^1 > n^2 > \dots > n^K \geq 1.$$

The original system coincides with the first and largest problem in the family,  $C = C^1$ ,  $b = b^1$ .

For an AMG algorithm, the sequence of matrices  $C^k$  must be constructed algebraically. The smaller  $C^k$  are computed successively by selecting a subset of the unknowns of the level  $k - 1$  system and by evaluating the *strength of the connections* between the unknowns in  $C^{k-1}$ . The basis

for this paper is the AMG method of Ruge and Stüben [4] that uses the assumptions

$$\begin{aligned} C &= (\gamma_{ij})_{1 \leq i, j \leq n} \text{ symmetric positive definite,} \\ \gamma_{ij} &\leq 0 \quad \text{for } 1 \leq i, j \leq n, i \neq j, \\ \sum_{j=1}^n \gamma_{ij} &\geq 0 \quad \text{for } 1 \leq i \leq n. \end{aligned} \tag{4}$$

With (4) the effect of Gauss-Seidel iterations on  $C$  is well understood and can be used to guide the construction of the coarser level systems  $C^k$  for  $k = 2, 3, \dots, K$ .

AMG methods were first introduced in the early eighties by Brandt, McCormick, and Ruge [5, 6, 7]. AMG is necessarily less efficient than highly specialized geometric multigrid solvers for elliptic problems on uniform rectangular grids. However, for more complicated cases with complex domains, AMG has been shown to behave quite favorably in terms of operation count and CPU time. AMG also works for problems where geometric multigrid methods are impossible to design. In this paper we will show that AMG works very satisfactorily even for the matrices in chip design.

The generality of AMG must be paid for by a setup phase that may take 80% or more of the overall time. This setup is needed to construct the sequence of reduced matrices  $C^k$  together with appropriate transfer operators from level  $k$  to level  $k + 1$ .

$$I_k^{k+1} : \mathbf{R}^{n^k} \rightarrow \mathbf{R}^{n^{k+1}}. \tag{5}$$

This step is quite expensive and contains code that does not vectorize or parallelize well.

We will briefly review the AMG algorithm, as introduced by Ruge and Stüben [4, 8]. The most interesting part may be the setup routine to build the family of systems (3) with the transfer operators (5).

The matrices  $C^k$  are constructed such that each of the unknowns  $x_i^k$  on level  $k$ , ( $k > 1$ ), will represent an unknown on the next finer level  $k - 1$ . The level  $k - 1$  unknown represented by  $x_i^k$  on level  $k$  is denoted by  $x_{j(i)}^{k-1}$ , the *corresponding* finer level unknown. This naturally partitions the unknowns on each level (except the coarsest) into those that correspond to a coarser level unknown, and those that do not. These will be called the  $C$ - and  $F$ -unknowns of a level, respectively. The partitioning is performed in two phases on each level. At the beginning of the first phase, the unknowns with strictly diagonal dominant matrix rows are determined. These unknowns are not restricted to a coarser level.

# SETUP PHASE I:

1. Set  $F_d = \emptyset$
2.  $\forall i \in \Omega$  : **If**  $\bar{\alpha}\gamma_{ii} \geq \sum_{j \neq i} |\gamma_{ij}|$  **then** set  $F_d = F_d \cup \{i\}$  **endif**
3. Set  $C = \emptyset$  and set  $F = \emptyset$
4. **While**  $C \cup F \neq (\Omega \setminus F_d)$  **do**
  - Pick  $i \in (\Omega \setminus F_d) \setminus (C \cup F)$  with maximal  $|S_i^T| + |S_i^T \cap F|$
  - If**  $|S_i^T| + |S_i^T \cap F| = 0$ 
    - then** set  $F = (\Omega \setminus F_d) \setminus C$
    - else** set  $C = C \cup \{i\}$  and set  $F = F \cup (S_i^T \setminus C)$ ;
  - endif**

Next, in a second phase the final  $C$ -point choice is made.

# SETUP PHASE II:

1. Set  $T = \emptyset$
2. **While**  $T \subset F$  **do**
  - Pick  $i \in F \setminus T$  and set  $T = T \cup \{i\}$
  - set  $\tilde{C} = \emptyset$  and set  $S_i^I = S_i \cap C$
  - set  $P = S_i \setminus S_i^I$
  - While**  $P \neq \emptyset$  **do**
    - Pick  $j \in P$  and set  $P = P \setminus \{j\}$
    - If**  $d(j, S_i^I) \leq \beta d(i, \{j\})$ 
      - then if**  $|\tilde{C}| = 0$ 
        - then** set  $\tilde{C} = \{j\}$  and set  $S_i^I = S_i^I \cup \{j\}$
        - else** set  $C = C \cup \{i\}$ , set  $F = F \setminus \{i\}$  and **Goto 2**
      - endif**
    - endif**
    - set  $C = C \cup \tilde{C}$ , set  $F = F \setminus \tilde{C}$
  3. (Set  $F = F \cup F_d$ )

In these algorithms we use

$$d(i, S) := \frac{1}{\max_{k \neq i} \{-\gamma_{ik}\}} \sum_{j \in S} -\gamma_{ij}$$

and  $S_i := \{j \in N_i \mid d(i, \{j\}) \geq \alpha\}$ ,  $S_i^T := \{j \mid i \in S_j\}$ , where  $N_i := \{j \mid j \neq i, \gamma_{ij} \neq 0\}$  is the set of neighbors of  $i$ .

After the unknowns of the level have been partitioned, the *interpolation* operator  $I_{k+1}^k : \mathbf{R}^{n^{k+1}} \rightarrow \mathbf{R}^{n^k}$  is defined by  $v^k = I_{k+1}^k v^{k+1}$

$$v_{j(i)}^k \stackrel{\text{def}}{=} \begin{cases} v_i^{k+1} & \text{for } i \in C^k \\ -\sum_{l \in C^k} \gamma_{il}^k v_l^{k+1} / \gamma_{ii}^k & \text{for } i \in F^k \setminus F_d^k \\ 0 & \text{for } i \in F_d^k \end{cases} . \quad (6)$$

The coarse level system for level  $k+1$  is now defined by the so-called *Galerkin* or *variational* conditions. The *restriction* operator is the transpose of the *interpolation* operator

$$I_k^{k+1} = (I_{k+1}^k)^T \quad (7)$$

and the reduced system matrix is

$$C^{k+1} = I_k^{k+1} C^k I_{k+1}^k. \quad (8)$$

Note that all coarse level matrices inherit the positive definiteness from  $C$ , provided all  $I_k^{k+1}$  have full rank.

The AMG algorithm can now be described as follows.

1. set  $k = 0$
2. **Do** set  $k = k + 1$ ; SETUP PHASE I and SETUP PHASE II; **until**  $|\Omega^k| = 1$
3. **While**  $\|b - Cx\| \geq \delta$   
     MGSTEP(1)

**MGSTEP(k):**

1. **If**  $k = K$  **then** solve (11)
2. **else** SMOOTH( $x^k$ )  
     set  $b^{k+1} = I_k^{k+1}(b^k - C^k x^k)$   
     MGSTEP(k+1)  
     set  $x^k = x^k + I_{k+1}^k x^{k+1}$   
     SMOOTH( $x^k$ )
3. **endif**

## VARIANTS OF AMG

We now discuss the handling of constraints in the AMG-algorithm. Just like the system matrix, the constraints (2) must be transferred to the coarse levels. Equation (2) thus becomes a family of constraints

$$A^k x^k = d^k, \quad (9)$$

for  $k = 1, 2, \dots, K$  corresponding to the reduced systems (3), where

$$A^{k+1} = A^k I_{k+1}^k. \quad (10)$$

The original matrix coincides with the first and largest problem in the family,  $A = A^1$ ,  $d = d^1$ .

The algorithm is modified such that (9) is satisfied on each level. On the coarsest level this is accomplished by solving the system with constraints directly using a Lagrange multiplier approach

$$\begin{bmatrix} C^K & (A^K)^T \\ A^K & 0 \end{bmatrix} \begin{bmatrix} x^K \\ \lambda \end{bmatrix} = \begin{bmatrix} b^K \\ d^K \end{bmatrix}. \quad (11)$$

The definition of the coarse level equations and constraints by a Galerkin condition has the effect that the finer level equations remain satisfied after a coarse grid correction, provided the coarse level constraints have been satisfied.

After each smoothing step, the constraints will be violated. This is compensated by an additional projection that enforces the constraints. Note that for general constraints the transfer can lead to coarse grid problems that are not well defined. This has been studied in detail in Bungartz [9]. Even if both  $A^k$  and  $I_{k+1}^k$  have full rank,  $A^k I_{k+1}^k$  may not. In this case constraints have become linearly dependent and the subspace determined by  $A^{k+1} x^k = d^{k+1}$  is either overdetermined or empty. In the case of overdetermined constraints, the number of constraints should be reduced. Numerically, however, detecting and treating this situation is difficult. Ideally, the matrix of constraints  $A^k$  should already be considered in the coarse level setup.

Here, we concentrate on the type of situation arising in the placement problem. With each constraint, a group of cells is assigned to a subdomain. Each cell is uniquely assigned to one such subdomain and the coefficients of the matrix  $A^k$  are determined by the relative surface area of the corresponding cells. Clearly, the rows of  $A^k$  are orthogonal. The coarse level constraints will remain consistent, if the interpolation  $I_{k+1}^k$  is constructed such that a coarse level variable only interpolates variables belonging to the same subdomain. Unfortunately, the constraints are still unknown in the (first) setup phase. In practice inconsistencies rarely arise, if we guarantee that the dimension of the coarsest level is larger than the number of constraints.

On the coarsest level the Lagrange multipliers  $\lambda$  must be calculated. This requires the solution of a full system of a dimension that is equal to the number of constraints. The number of constraints doubles with each partitioning step. Thus the coarsest permissible level may be quite large and expensive to solve exactly, making the algorithm unacceptable for large chips.

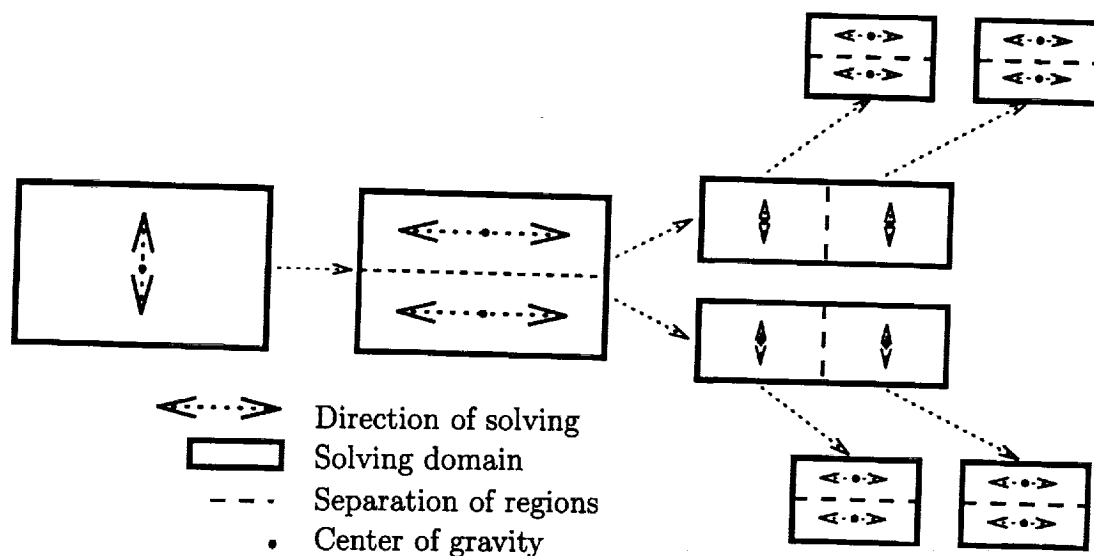


Figure 3: The double arrow shows in which direction the solution is calculated, here it is started in y direction. The box indicates the subdomains for which the computation is performed. The dashed line indicates a separation of the regions; cells cannot cross such a line during the overall placement calculation.

Experience shows that the influence of cells in different subdomains is rather small and may be neglected. Additionally, earlier experiments with GORDIAN have shown that the quadratic objective functional is only a crude approximation to the true one. It can be argued that the usual routing of connections in the final layout induces a measure of distances that is modeled better by an  $L_1$ -like norm and a linear objective functional (see Sigl [10]). This motivates an algorithm that recursively splits the problem into independent ones by partitioning into subdomains. A *solution subdomain* is defined as two neighboring subdomains that have been obtained by partitioning a single subdomain of the previous iterations. We can now simulate the effect of a linear objective functional by keeping the cells fixed in all subdomains except those in the current solution subdomain. This must be repeated for all solution subdomains. Thus, though the above simplification changes the mathematical model, the modified algorithm may help to produce better overall layouts. This is indicated by experimental results.

The algorithm is illustrated in Figure 3. The first two calculations are performed as before, without any change. After the second partitioning, the computation of the overall chip is split into an upper and a lower solution subdomain. When the new solution for the upper solution subdomain is computed, the cells of the lower solution subdomain are kept fixed. Simultaneously, the lower solution subdomain is computed with fixed upper domain cell positions. This is repeated recursively until the partitioning is completed. Clearly, this algorithm can be easily parallelized because each solution subdomain can be computed independently. Because no data exchange between the different solution subdomain is necessary, this is a plain *divide-and-conquer* algorithm inducing a natural parallelization. Note, that we have to solve systems with at most two simultaneous constraints. This leads to an algorithm, where it is sufficient to perform the setup once at the beginning of the computation. Before each optimization step the (at most two) constraints are tested for linear dependencies. In the case of inconsistent constraints the previous

level is taken for the coarsest level.

The conventional setup of the coarse level matrices is *variational* in the sense that (7) and (8) are satisfied. Experience shows that the coarse level matrices tend to fill up rather quickly. On the other hand, the definition by equation (8) often leads to *small* matrix entries, so that one may have the idea to modify the coarser matrices by dropping small entries. More precisely, we may perturb each  $C^k$  to

$$\tilde{C}^k = C^k + B^k, \quad (12)$$

such that the matrix remains sparse. This will not only speed up each individual iteration, but also simplify coarser matrix setups. We suggest performing the perturbation such that the matrix remains symmetric and such that dropped values are added to the diagonal with the opposite sign. For an analysis of these perturbations see Muszynski, Rude, and Zenger [11], Bungartz [9], and Chang and Wong [12].

Classical AMG is used with a single sweep of Gauss-Seidel smoothing on each level. Alternatively, we may use Jacobi-type smoothers. As usual, the Jacobi method must be damped to obtain good smoothing. Though the Jacobi method is usually a less efficient smoother than Gauss-Seidel (even with optimal damping), it may be an interesting alternative, because it has a symmetric error propagation matrix without performing sweeps in reverse order. Jacobi-AMG may thus be used directly as a preconditioner for the conjugate gradient method. Another advantage of Jacobi is parallelization. To parallelize Gauss-Seidel we would have to find a coloring scheme for a general unstructured matrix that permits the parallel execution of relaxation steps. Our experimental results (see Figure 5) indicate that two optimally damped Jacobi iterations are about as good a smoother as a single sweep of Gauss-Seidel. This is in agreement with experience for the solution of partial differential equations. In future work we intend to experiment with other smoothers, like conjugate residuals or incomplete LU decomposition; see e.g. Bank and Douglas [13].

We denote the diagonal part of  $C^k$  by  $D^k$  and can thus write a damped Jacobi iteration for level  $k$  as

$$x^k \leftarrow x^k + \omega(D^k)^{-1}(b^k - C^k x^k), \quad (13)$$

where  $\omega$  is the relaxation parameter. For the error  $e = x - C^{-1}b$  in the original system, a relaxation on level  $k$  has an effect that can be described by

$$e \leftarrow (I - I_k^1(D^k)^{-1}I_1^k C)e, \quad (14)$$

where

$$I_1^k = \prod_{j=1}^{k-1} I_j^{j+1}. \quad (15)$$

The AMG algorithm in its simplest form (with a single sweep of Jacobi on each level) has an error propagation

$$e \leftarrow \prod_{k=1}^K (I - I_k^1(D^k)^{-1}I_1^k C)e. \quad (16)$$

This is a typical *multiplicative* method.

All conventional multigrid methods, including AMG, are *multiplicative* algorithms in the sense that the levels are visited sequentially in a predetermined order. The recent development of multilevel methods has led to the formulation of a class of *additive* multilevel methods. These include the AFAC type algorithms (see McCormick [14]), the BPX method (see Bramble, Pasciak and Xu [15]), and the multilevel additive Schwarz methods (see Dryja and Widlund [16]). Formally, these methods do not form a product of operators as in (16), but a sum, whose terms can — in principle — be computed simultaneously.

With some exceptions (like the AFAC method), additive methods provide only preconditioners that are divergent when used as iterations by themselves. However, they define operators with improved condition numbers, and so they will lead to fast convergence when suitably damped or when they are used in combination with self-scaling iterative methods, most notably the conjugate gradient algorithm. Recent results have shown that these methods can have typical multigrid efficiency with convergence rates independent of the problem size.

We will show that for our problems

$$P^k \stackrel{\text{def}}{=} \sum_{j=1}^k I_j^1 (D^j)^{-1} I_1^j C \quad (17)$$

also has a better condition number than the original matrix  $C$ . Note that an application of  $P^k$  does not require the explicit construction of the corresponding matrix, but only the restriction of the residuals to all levels, just like in conventional AMG. An iteration based on  $P^k$ , like

$$x = x + \omega \sum I_k^1 (D^k)^{-1} I_1^k (b - Cx) \quad (18)$$

will only converge, when suitably damped with  $\omega < 1$ . Preferably (18) is used as a preconditioner for a conjugate gradient iteration.

## NUMERICAL EXPERIMENTS

Our first example is a typical benchmark chip called *Primary I* with 752 cells, 81 fixed cells, and 902 signals. Figure 4 shows the corresponding matrix structure, and Figure 5 displays the convergence history of (multiplicative) AMG using different smoothers for the solution of (1). Clearly two sweeps of damped Jacobi are almost as good a smoother as Gauss-Seidel (GS). In Table 1 the minimal and the maximal eigenvalue ( $\lambda_{min}, \lambda_{max}$ ) plus the condition number  $\kappa = \lambda_{max}/\lambda_{min}$  of  $P^k$  are shown. On the coarsest level ( $k = 6$ )  $D^k$  is replaced by  $C^k$ . This means that the coarsest level equations are solved exactly. In Figure 6 we present the corresponding spectrum of the eigenvalues for  $k = 1, 3, 6$ . In each case, the first few eigenvalues are marked by asteriks(\*) and diamonds( $\diamond$ ), respectively. In column 5 and 6 of Table 1, the density and dimension of the coarse level system  $C^k$  are displayed additionally.

In Figure 7 we show the convergence history for preconditioned CG in analogy to Figure 5 in comparison to the AMG-solver with Gauss-Seidel smoothing. Conventional AMG is superior to AMG-preconditioned CG, partly because Gauss-Seidel is a better smoother than Jacobi. However,



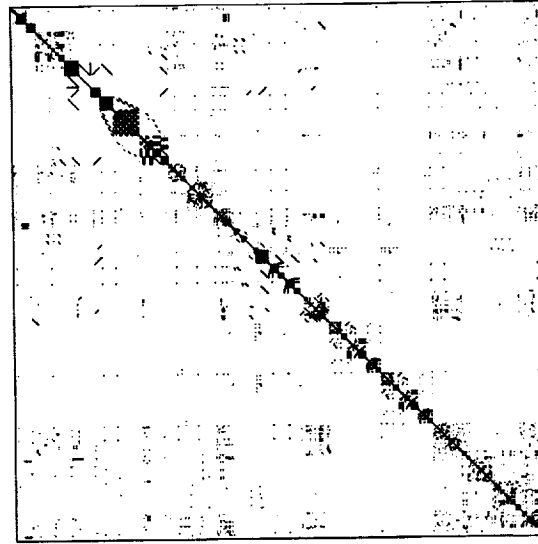


Figure 4: Sparsity pattern of system matrix of Primary I

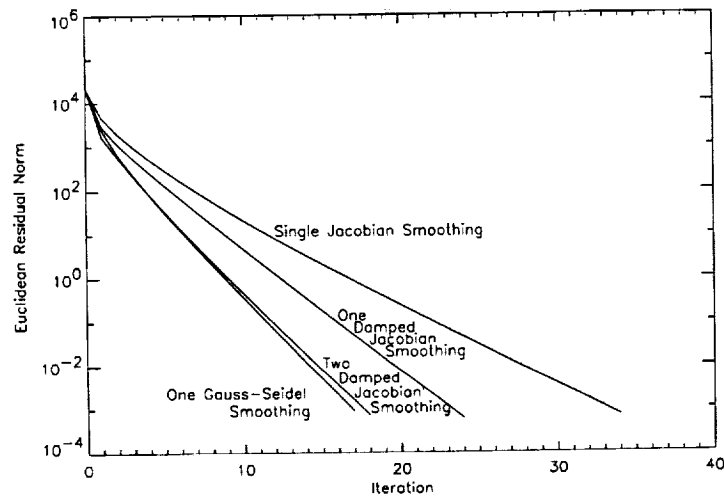


Figure 5: Convergence history for AMG with different smoothers

k	$\lambda_{min}$	$\lambda_{max}$	$\kappa$	density	dim
1	0.0085	1.8301	215.3	0.02	752
2	0.0302	3.4020	133.4	0.09	343
3	0.0598	4.2936	71.8	0.33	147
4	0.1117	4.9196	44.0	0.72	59
5	0.2500	5.9495	23.8	0.95	26
6	0.4060	6.1167	15.1	1.00	10

Table 1: Eigenvalues and characteristics of Primary I

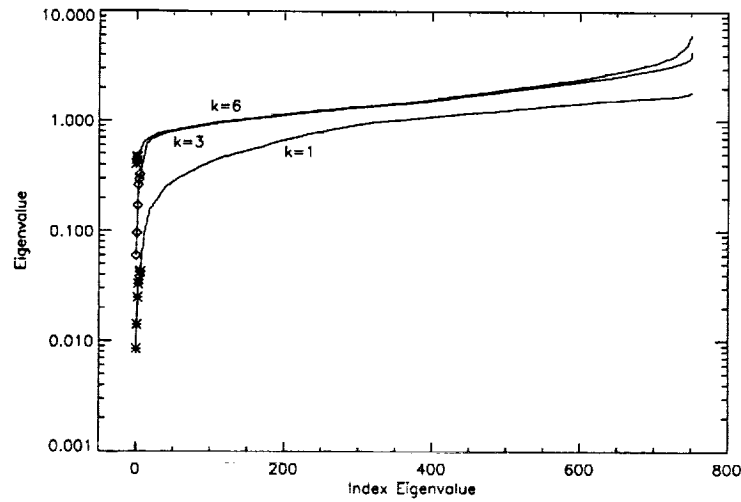


Figure 6: Eigenvalues of  $P^k$  for Primary I matrix

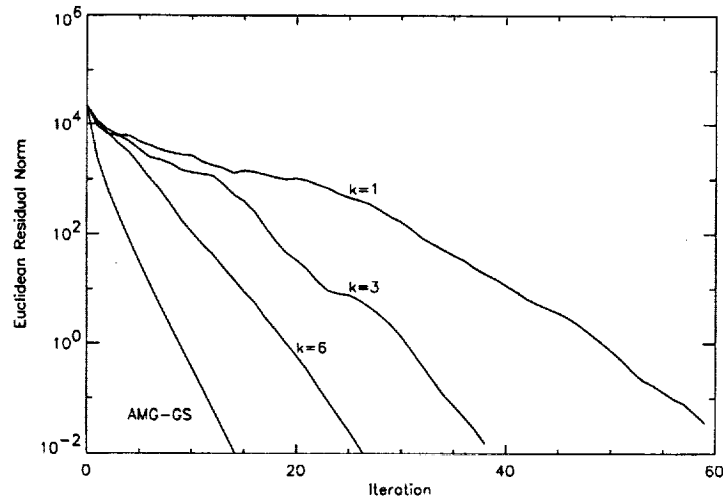


Figure 7: Convergence history of preconditioned CG for Primary I

AMG-preconditioned CG is an interesting alternative when we consider its potential for parallelization.

In further tests we have applied the AMG algorithm to a problem of similar size arising from the discretization of a partial differential equation and have found that the behavior is surprisingly similar.

Finally, we present results for a real-life chip with 25178 cells. The original preconditioned CG solver (CG) in GORDIAN is replaced by the AMG routines combined with the *divide and conquer* strategy. In Table 2 we compare the CPU times for CG and AMG for the optimization after each partitioning step. The first AMG step includes the setup time, which is 9 times as expensive as the iteration itself, but still faster than CG. AMG outperforms conventional CG for almost all subproblems, except the very last six partitions. In the overall time AMG is still clearly superior to CG.

Partition	CG	AMG	sol	par
0	126.3	43.7	4.2	
1	104.5	5.4	4.9	5.4
2	91.1	33.8	8.3	15.2
3	85.1	29.7	8.2	6.9
4	79.2	26.7	7.8	3.4
5	76.5	45.0	25.8	19.7
6	61.7	35.4	17.8	9.6
7	58.1	24.2	9.0	0.6
8	58.2	50.4	34.3	9.4
9	115.0	47.9	30.3	3.0
10	91.5	38.2	15.2	0.9
11	72.9	114.4	33.4	0.5
12	30.4	137.0	37.8	0.4
13	15.7	41.7	13.2	0.1
14	12.7	19.4	11.0	0.0
15	6.2	11.2	9.5	0.0
16		9.0	8.7	0.0
total	1084.9	713.2	279.4	—

Table 2: avq : time [s] spent per partition

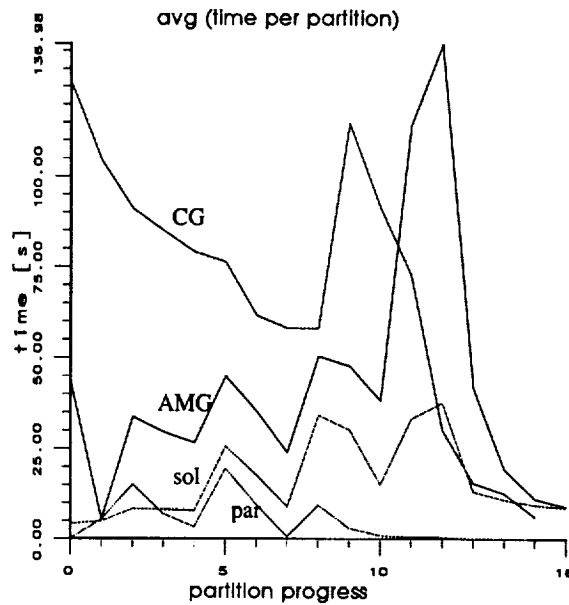


Figure 8: avq 1: time [s] spent per partition

Following the divide and conquer strategy, we transform the system into separately solvable subproblems after the second partition. This requires a transformation of the data that is not yet optimally implemented. The column labeled "sol" therefore shows the time for the AMG solution process without the overhead for this data transformation. The overhead for the transformation increases with the number of partitions, adding to the cost of the AMG method.

However, each subdomain can be computed in parallel. To illustrate the potential for parallelization, the "par" column shows the maximal time needed for computation of a subdomain, thus simulating the effect of an optimal parallelization. The example chip for this calculation is a *standard cell chip*. This type of chip has a fixed number of rows of cells. Thus subdomains with height below a certain minimum are not permitted. To avoid this, GORDIAN computes the partition for both directions until the maximal number of rows is reached. Here, this applies to partition 9,10,11 during conventional CG; for the AMG method this happens during partition 11,12,13. As the partition progresses, the original AMG setup may not be suitable any more and must be repeated for the subdomains that cause trouble. In our example this has been the case in partitions 5,8, and 9. For further discussion see Regler [3].

## CONCLUDING REMARKS

We have discussed the application of algebraic multigrid methods and have proposed several variants and extensions of the classical AMG method of Ruge and Stuben, including constrained optimization and a new additive algorithm. We have shown that the AMG method is a highly competitive alternative for the layout optimization of real life chips.

Acknowledgements: We wish to thank H. Bungartz, K. Doll, F. M. Johannes, G. Sigl, and C. Zenger for many helpful discussions.

## REFERENCES

- [1] J. Kleinhans, G. Sigl, F. Johannes, and K. Antreich. Gordian: VLSI placement by quadratic programming and slicing optimization. *IEEE Trans. Computer-Aided Design*, CAD-10:356-365, March 1991.
- [2] G. Sigl. Platzierung der Zellen bei der Layoutsynthese mittels Partitionierung und quadratischer Optimierung. Dissertation, Lehrstuhl für Rechnergestütztes Entwerfen, Technische Universität München, 1992.
- [3] H. Regler. Algebraic multilevel methods in chip design. In *Proceedings of the GAMM-Seminar on Multigrid Methods, Sept. 21 - 25, 1992 in Gosen, Germany*, Berlin, 1993. Institut für Angewandte Analysis und Stochastik. Report 5, ISSN 0942-9077.
- [4] J. Ruge and K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). *Arbeitspapiere der GMD*, 89, 1984.

- [5] A. Brandt. Algebraic multigrid theory: The symmetric case. In S. McCormick and U. Trottenberg, editors, *Preliminary Proceedings of the International Multigrid Conference, Copper Mountain, Colorado, April 6-8, 1983*, 1983.
- [6] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic algorithm design and problem solution. Report, . Comp. Studies, Colorado State University, Ft. Collins, 1982.
- [7] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with applications to geodetic computations. In Evans, editor, *Sparsity and its Applications*. Cambridge University Press, 1984.
- [8] J. Ruge and K. Stüben. Algebraic multigrid (AMG). Arbeitspapiere der GMD, Gesellschaft für Mathematik und Datenverarbeitung, 1986.
- [9] H. Bungartz. Beschränkte Optimierung mit algebraischen Mehrgittermethoden. Diplomarbeit, Institut für Informatik, Technische Universität München, 1988.
- [10] G. Sigl, K. Doll, and F. Johannes. Analytical placement: A linear or a quadratic objective function? In *ACM/IEEE Proceedings 28th Design Automation Conference*, 1991.
- [11] P. Muszynski, U. Rüde, and C. Zenger. Application of algebraic multigrid (AMG) to constrained quadratic optimization. Bericht I-8801, Institut für Informatik, TU München, January 1988.
- [12] Qianshun Chang and Yau Shu Wong. Recent developments in algebraic multigrid methods. In T. Manteuffel and S. McCormick, editors, *Preliminary proceedings of the 2nd Copper Mountain Conference on Iterative Methods, Copper Mountain, April 9-14, 1992*. University of Colorado at Denver, 1992.
- [13] R. Bank and C. Douglas. Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *SIAM J. Numer. Anal.*, 22:617–633, 1985.
- [14] S.F. McCormick. *Multilevel Adaptive Methods for Partial Differential Equations*, volume 6 of *Frontiers in Applied Mathematics*. SIAM, Philadelphia, 1989.
- [15] J. Bramble, J. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 31:333–390, 1990.
- [16] M. Dryja and O. Widlund. Multilevel additive methods for elliptic finite element problems. In W. Hackbusch, editor, *Parallel Algorithms for Partial Differential Equations, Proceedings of the Sixth GAMM-Seminar, Kiel, January 19-21, 1990*, Braunschweig, 1991. Vieweg-Verlag.



OPTIMAL CONVOLUTION SOR ACCELERATION  
OF WAVEFORM RELAXATION WITH APPLICATION  
TO SEMICONDUCTOR DEVICE SIMULATION

511-61

Mark Reichelt

Research Laboratory of Electronics, Massachusetts Institute of Technology  
Cambridge, MA197571  
p. 14

## SUMMARY

In this paper we describe a novel generalized SOR algorithm for accelerating the convergence of the dynamic iteration method known as waveform relaxation. A new convolution SOR algorithm is presented, along with a theorem for determining the optimal convolution SOR parameter. Both analytic and experimental results are given to demonstrate that the convergence of the convolution SOR algorithm is substantially faster than that of the more obvious frequency-independent waveform SOR algorithm. Finally, to demonstrate the general applicability of this new method, it is used to solve the differential-algebraic system generated by spatial discretization of the time-dependent semiconductor device equations.

## INTRODUCTION

To achieve highest performance on a parallel computer, a numerical algorithm must avoid frequent parallel synchronization [1]. The waveform relaxation approach to solving time-dependent initial-value problems is just such a method, as the iterates are waveforms over an interval, rather than single timepoints [2, 3, 4]. Like any relaxation scheme, efficiency depends on rapid convergence, and there have been several investigations into how to accelerate WR [2, 5], including using multigrid [6] and conjugate direction techniques [7].

In this paper, we investigate using successive overrelaxation (SOR) to accelerate WR convergence. In particular, we show that the pessimistic results about waveform SOR derived in [2] can be substantially improved by replacing multiplication with a fixed SOR parameter by convolution with an SOR kernel. We derive the optimal SOR kernel using

\* This work was supervised by Professors Jacob White and Jonathan Allen and supported by a grant from IBM, the Defense Advanced Research Projects Agency contract N00014-91-J-1698, and the National Science Foundation contract MIP-8858764 A02.

Fourier analysis techniques and then demonstrate the effectiveness of the approach for a model parabolic problem. Finally, we demonstrate the general applicability of the approach by using the method to solve the time-dependent drift-diffusion equations associated with modeling semiconductor devices.

We begin in Section 2 by reviewing waveform SOR, and in Section 3 we relate the algorithm to pointwise SOR to demonstrate the difficulty in accelerating WR with a fixed SOR parameter. In Section 4, we use Fourier analysis to derive the SOR kernel for the continuous WR algorithm, and give a proof of optimality. In Section 5 we briefly consider the effect of time-discretization, and in Section 6 we apply the method to device simulation. Finally, conclusions and acknowledgements are given in Section 7.

## WAVEFORM SOR

In this section, we consider applying waveform relaxation methods to the model linear initial-value problem

$$(1) \quad \left( \frac{d}{dt} + A \right) x(t) = b(t) \quad \text{with} \quad x(0) = x_0,$$

where  $A \in \mathbb{R}^{n \times n}$ ,  $b(t) \in \mathbb{R}^n$  is a given time-dependent right-hand side vector,  $x(t) \in \mathbb{R}^n$  is the unknown vector to be computed over simulation interval  $t \in [0, T]$ , and  $x_0 \in \mathbb{R}^n$  is an initial condition.

Given the relaxation splitting  $A = D - L - U$ , and subtracting successive waveform relaxation iterations, the waveform Gauss-Jacobi (WGJ) and waveform Gauss-Seidel (WGS) iteration equations, respectively, may be written as:

$$(2) \quad \left( \frac{d}{dt} + D \right) \Delta x^{k+1}(t) = (L + U) \Delta x^k(t)$$

$$(3) \quad \left( \frac{d}{dt} + D - L \right) \Delta x^{k+1}(t) = U \Delta x^k(t),$$

where  $\Delta x^{k+1}(t) = x^{k+1}(t) - x^k(t)$  is used to eliminate the right hand side  $b(t)$ .

The waveform SOR method for acceleration of WGS is a simple extension of algebraic SOR. To derive the waveform SOR iteration equation, compute a waveform  $\hat{x}_i^{k+1}(t)$  on  $t \in [0, T]$ , as in WGS:

$$(4) \quad \left( \frac{d}{dt} + a_{ii} \right) \hat{x}_i^{k+1}(t) = b_i(t) - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1}(t) - \sum_{j=i+1}^n a_{ij} x_j^k(t) \quad \text{with} \quad \hat{x}_i^{k+1}(0) = x_{i0},$$

and then update  $x_i^k(t)$  in the iteration direction by multiplication with an overrelaxation parameter  $\omega$ ,

$$(5) \quad x_i^{k+1}(t) \leftarrow x_i^k(t) + \omega \cdot [\hat{x}_i^{k+1}(t) - x_i^k(t)].$$



Combining equations (4) and (5) yields

$$(6) \quad \left( \frac{d}{dt} + a_{ii} \right) x_i^{k+1}(t) = (1 - \omega) \left[ \left( \frac{d}{dt} + a_{ii} \right) x_i^k(t) \right] + \omega \left[ b_i(t) - \sum_{j=1}^{i-1} a_{ij} x_j^{k+1}(t) - \sum_{j=i+1}^n a_{ij} x_j^k(t) \right],$$

which, after subtracting successive waveform relaxation iterations, leads to

$$(7) \quad \left( \frac{d}{dt} + D - \omega L \right) \Delta x^{k+1}(t) = [(1 - \omega) \left( \frac{d}{dt} + D \right) + \omega U] \Delta x^k(t),$$

where  $\Delta x^{k+1}(t) = x^{k+1}(t) - x^k(t)$ .

Note that the iteration matrices implied by equations (2), (3) and (7) correspond exactly to the standard algebraic relaxation and SOR matrices with diagonal matrix  $D$  replaced by  $\left( \frac{d}{dt} + D \right)$ . Also note that waveform SOR as defined by (7) is *not* the same as the dynamic SOR iteration considered in [2], because, unlike WGJ or WGS, the waveform SOR iteration equations are not of the form

$$(8) \quad \frac{d}{dt} \Delta x^{k+1} + M \Delta x^{k+1} = N \Delta x^k$$

where  $M, N \in \mathbb{R}^{n \times n}$ .

## RELATION TO POINTWISE SOR

Discretizing (1) in time using a multistep integration method yields

$$(9) \quad \sum_{j=0}^s \alpha_j x[m-j] = h \sum_{j=0}^s \beta_j (b[m-j] - A x[m-j]),$$

where  $\alpha_0 = 1$  and  $x[m]$  denotes  $x(t)$  at timepoint  $t = mh$  with timestep  $h$ . Thus, the time-discretized model problem can be rewritten as a sequence of linear algebraic problems

$$(10) \quad [I + h\beta_0 A] x[m] = h\beta_0 b[m] - \sum_{j=1}^s \alpha_j x[m-j] + h \sum_{j=1}^s \beta_j (b[m-j] - A x[m-j]).$$

We now compare the convergence of the waveform SOR method to the convergence of pointwise SOR, in which algebraic SOR is used to solve the matrix problem at each timepoint.

The pointwise SOR iteration equations are derived by applying the relaxation splitting  $A = D - L - U$  to equation (10) and taking the difference between the  $(k+1)$ st and  $k$ th

iterations. More precisely, the pointwise SOR iteration equation applied to solve (10) for  $\Delta x^{k+1}[m] = x^{k+1}[m] - x^k[m]$  is

$$(11) \quad \begin{aligned} & \left[ (I + h\beta_0 D) - \omega h\beta_0 L \right] \Delta x^{k+1}[m] = \\ & \left[ (1 - \omega) (I + h\beta_0 D) + \omega h\beta_0 U \right] \Delta x^k[m], \end{aligned}$$

where  $\omega$  is the SOR parameter. It follows that the spectral radius of the iteration matrix generated by pointwise SOR at the  $m$ th timestep is

$$(12) \quad \rho \left( \left[ (I + h\beta_0 D) - \omega h\beta_0 L \right]^{-1} \left[ (1 - \omega) (I + h\beta_0 D) + \omega h\beta_0 U \right] \right).$$

If waveform SOR is used to solve the model problem (1), and a multistep method is used to solve iteration equation (7), then  $\Delta x^{k+1}[m]$ , now denoting the discretized difference in waveform iterates, satisfies

$$(13) \quad \begin{aligned} & \sum_{j=0}^s \alpha_j \left[ \Delta x^{k+1}[m-j] - (1 - \omega) \Delta x^k[m-j] \right] = \\ & h \sum_{j=0}^s \beta_j \left\{ - (D - \omega L) \Delta x^{k+1}[m-j] + [(1 - \omega) D + \omega U] \Delta x^k[m-j] \right\}. \end{aligned}$$

This can be rewritten as the discrete-time analogue of (7):

$$(14) \quad \begin{aligned} & \sum_{j=0}^s \left[ (\alpha_j I + h\beta_j D) - \omega h\beta_j L \right] \Delta x^{k+1}[m-j] = \\ & \sum_{j=0}^s \left[ (1 - \omega) (\alpha_j I + h\beta_j D) + \omega h\beta_j U \right] \Delta x^k[m-j]. \end{aligned}$$

As the similarities of equations (11) and (14) suggest, if the time interval is finite, i.e. the number of timesteps is some finite  $L$ , then for a given timestep  $h$  and a given SOR parameter  $\omega$ , the time-discretized waveform SOR method has the same asymptotic convergence rate as pointwise SOR.

*Theorem 3.1.* On a finite simulation interval, the iterations defined by (11) and (14) have the same asymptotic convergence rate.

*Proof.* Let  $y^k$  denote the large vector consisting of the concatenation of vectors  $\Delta x^k[m]$  at all  $L$  discrete timepoints, i.e.  $y^k = [\Delta x^k[1]^T, \dots, \Delta x^k[L]^T]^T$ . Collecting together the equations (14) generated at each timepoint into one large matrix equation in terms of vectors  $y^{k+1}$  and  $y^k$  yields  $M \Delta y^{k+1} = N \Delta y^k$  where  $M, N \in \mathbb{R}^{Ln \times Ln}$  are block lower triangular banded matrices, with blocks of size  $n \times n$ , and with block bandwidth  $s$ . It is then easily seen that  $M^{-1}N$  is block lower triangular, with diagonal blocks equal to

$$(15) \quad \left[ (I + h\beta_0 D) - \omega h\beta_0 L \right]^{-1} \left[ (1 - \omega) (I + h\beta_0 D) + \omega h\beta_0 U \right].$$

Therefore,  $\rho(M^{-1}N)$  is given by (12), implying that the iterations defined by (11) and (14) have identical asymptotic convergence rates.  $\square$

Theorem 3.1 suggests that parameter  $\omega$  for waveform SOR should be chosen to be precisely equal to the optimum parameter for the pointwise SOR method. However, this does not necessarily lead to fast convergence, as the following example illustrates.

*Example 3.1.* Let  $t \in [0, 2048]$ ,  $x(0) = 0$ , and let matrix  $A \in \mathbb{R}^{32 \times 32}$  and time-dependent input vector  $b(t) \in \mathbb{R}^{32}$  of the model problem (1) be given by

$$(16) \quad A = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix}$$

$$b(t) = \begin{bmatrix} b_1(t) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{where} \quad b_1(t) = \begin{cases} 1 - \cos\left(\frac{2\pi t}{256}\right) & \text{if } t \leq 256 \\ 0 & \text{otherwise.} \end{cases}$$

Consider the four problems generated by discretizing in time with the first-order backward difference formula, using 64, 128, 256, and 512 uniform timesteps of size  $h = 32, 16, 8$  and 4 respectively.

Since the tridiagonal matrix  $A$  is symmetric and is consistently ordered [8, 9], the matrix  $(I + h\beta_0 A)$  of the pointwise time-discretized model problem (10) is also consistently ordered, and the optimum pointwise SOR parameter  $\omega_{opt}$  is given by

$$(17) \quad \omega_{opt} = \frac{2}{1 + \sqrt{1 - \mu_1^2}}$$

where  $\mu_1 = \rho(H_{GJ})$  is the spectral radius of the pointwise Gauss-Jacobi iteration matrix  $H_{GJ} = (I + h\beta_0 D)^{-1}(h\beta_0 L + h\beta_0 U)$ . For the four problems with 64, 128, 256 and 512 timesteps, the optimum pointwise parameters  $\omega_{opt}$  are 1.669, 1.586, 1.482 and 1.364 respectively.

Curves PT64, PT128, PT256 and PT512 of Figure 1 show the convergence of the waveform SOR method versus iteration for the four problems with their optimum pointwise SOR parameters  $\omega_{opt}$ . Note that as the total number of timesteps is increased, the initial convergence rate is slower, approaching a limiting value of the convergence rate of the continuous Gauss-Seidel WR algorithm (shown as WR in Figure 1). In each case, the convergence rate of the waveform SOR eventually approaches the expected asymptotic value of  $\omega_{opt} - 1$ . Note that with a reasonable error accuracy tolerance such as  $10^{-6}$  as a stopping point, the asymptotic convergence rate is *never* reached. For comparison, Figure 1

also shows the superposition of four convergence plots (CSOR) of the new convolution SOR method to be introduced in the following sections.

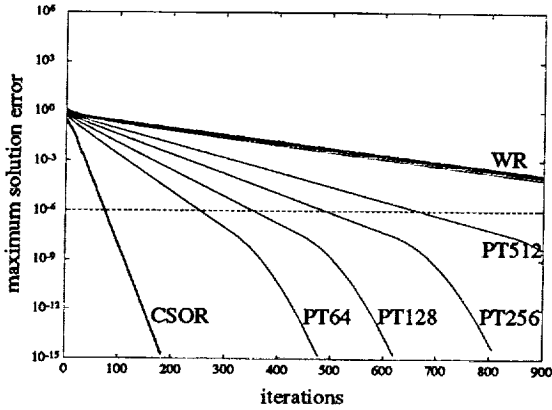


FIG. 1. Convergence of waveform SOR using the pointwise optimal parameter (PT) compared to waveform relaxation (WR), and convolution SOR (CSOR), with 64, 128, 256 and 512 timesteps.

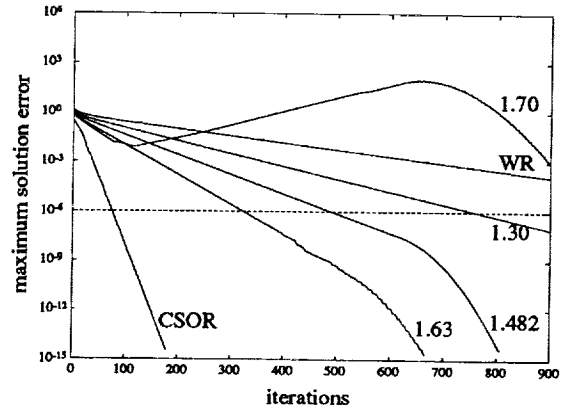


FIG. 2. Effect on convergence of the 256-timestep waveform SOR of varying the SOR parameter from the pointwise optimum  $\omega_{opt} = 1.482$ .

To illustrate the effect of choosing a different SOR parameter  $\omega$ , Figure 2 shows the convergence versus iteration of the 256-timestep example for waveform SOR with values of the SOR parameter  $\omega$  not equal to the pointwise optimum  $\omega_{opt} = 1.482$ . When  $\omega = 1.30 < \omega_{opt}$ , the convergence curve lies between the pointwise optimum curve and the WR convergence curve, i.e. both initial and asymptotic convergence rates are slower. By increasing the SOR parameter to  $\omega = 1.63 > \omega_{opt}$ , the initial convergence rate can be made faster at the expense of slowing down the asymptotic convergence rate. But as the  $\omega = 1.70$  curve shows, once the SOR parameter is increased beyond some point, the waveform SOR method may appear to diverge before eventually converging. Also, the solution produced by the  $\omega = 1.70$  example contains spurious oscillations, as shown in Figure 3. Note both the growth and translation of the oscillation with iteration.

The optimum pointwise SOR parameter  $\omega_{opt}$  does not dramatically improve the convergence rate of waveform SOR because the matrix  $M^{-1}N$  which describes the waveform SOR convergence is far from normal. This suggests that although the spectral radius of the iteration matrix determines the *asymptotic* convergence rate of waveform SOR, it does not determine the practically observable convergence rate. The convergence rate could be characterized, for example, by computing the pseudo-eigenvalues [10] of the waveform SOR iteration matrix. In the following section, we take an alternate approach.

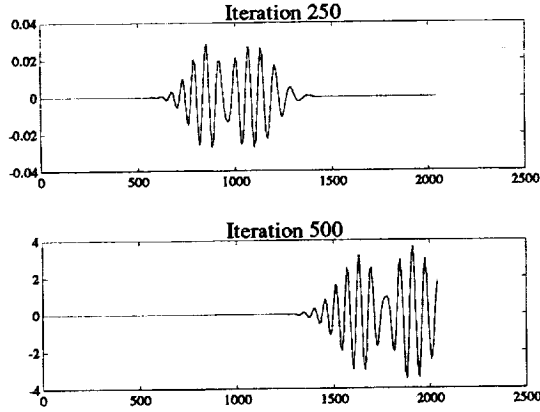


FIG. 3. Delta waveform  $\Delta x_{16}^{k+1}(t) = x_{16}^{k+1}(t) - x_{16}^k(t)$  versus time after iterations 250 and 500, for the 256-timestep waveform SOR method using  $\omega = 1.70$ , showing the growth and translation of an oscillating region.

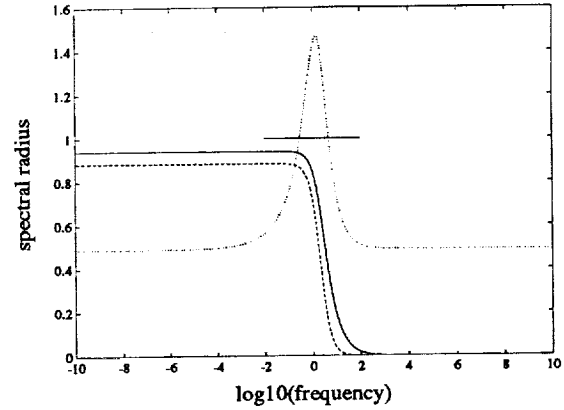


FIG. 4. The spectral radii as functions of frequency  $\Omega$  of the Gauss-Jacobi WR (solid), Gauss-Seidel WR (dashed) and waveform SOR (dotted) iteration matrices for an  $8 \times 8$  version of the continuous-time problem of Example 3.1.

## FOURIER ANALYSIS

In [2], the spectral radius of dynamic iteration operators which map  $x^k$  to  $x^{k+1}$ , such as those given by equations (2), (3), and (8), was related to their Fourier transform. In this section, we make a more detailed use of Fourier analysis to derive a frequency-dependent SOR parameter for the waveform SOR operator of equation (7).

The Fourier transform of  $x^k(t)$  is given by

$$(18) \quad x^k(i\Omega) = \int_{-\infty}^{\infty} x^k(t) e^{-i\Omega t} dt = \mathcal{F}\{x^k(t)\},$$

where  $\Omega$  is frequency. Standard Fourier identities can be used to show that  $\Delta x^{k+1}(i\Omega) = H(i\Omega) \Delta x^k(i\Omega)$ , where for WGJ (2), WGS (3) and waveform SOR (7), the iteration operator  $H(i\Omega)$  is given by

$$(19) \quad H_{GJ}(i\Omega) = (i\Omega I + D)^{-1}(L + U)$$

$$(20) \quad H_{GS}(i\Omega) = (i\Omega I + D - L)^{-1}U$$

$$(21) \quad H_{SOR}(i\Omega) = (i\Omega I + D - \omega L)^{-1}[(1 - \omega)(i\Omega I + D) + \omega U]$$

respectively. The obvious interpretation of equations (19)–(21) is that the spectral radius  $\rho(H(i\Omega))$  yields the asymptotic convergence rate for errors in the frequency component  $\Omega$ .

Figure 4 is a plot of the spectral radii of  $H_{GJ}(i\Omega)$ ,  $H_{GS}(i\Omega)$  and  $H_{SOR}(i\Omega)$  for an  $8 \times 8$  version of the continuous-time problem given in Example 3.1, using  $\omega = 1.49$  for  $H_{SOR}(i\Omega)$ . From the plot it is clear that very high frequency components of the error are damped much more quickly than low frequency components. However, the spectral radius  $\rho(H_{SOR}(i\Omega))$  is greater than one over a range of frequencies, and therefore the waveform SOR iteration magnifies errors in this frequency range. This effect was predicted in [2] and is easily seen in Figure 3.

This situation can be remedied by using a generalized SOR algorithm, in which equation (5) is replaced by an overrelaxation convolution with a time-dependent SOR parameter  $\omega(t)$ ,

$$(22) \quad x_i^{k+1}(t) \leftarrow x_i^k(t) + \int_{-\infty}^{\infty} \omega(t - \tau) \cdot [\hat{x}_i^{k+1}(\tau) - x_i^k(\tau)] d\tau.$$

The Fourier transform of the SOR operator is then given by

$$(23) \quad H_C(i\Omega) = [i\Omega I + D - \omega(i\Omega)L]^{-1} [(1 - \omega(i\Omega))(i\Omega I + D) + \omega(i\Omega)U],$$

where  $\omega(i\Omega)$  is the Fourier transform of the time-dependent  $\omega(t)$ . We refer to the SOR algorithm represented by iteration matrix (23) as the *convolution SOR* algorithm (CSOR). The theorem below, which is the main result of this paper, gives a formula for determining the optimal frequency-dependent SOR parameter  $\omega(i\Omega)$ .

*Theorem 4.2.* If the spectrum of  $H_{GJ}(i\Omega)$  lies on the line segment  $[-\mu_1(i\Omega), \mu_1(i\Omega)]$  with  $|\mu_1| < 1$ , then the spectral radius of  $H_C(i\Omega)$  is minimized at frequency  $\Omega$  by a unique optimum  $\omega(i\Omega) = \omega_{opt}(i\Omega) \in \mathbb{C}$  given by

$$(24) \quad \omega_{opt}(i\Omega) = \frac{2}{1 + \sqrt{1 - \mu_1(i\Omega)^2}}$$

where  $\sqrt{\cdot}$  denotes the root with the positive real part.

*Proof.* For brevity, the argument  $(i\Omega)$  will be omitted in the following, and  $H_C(\omega)$  will denote the convolution SOR operator (at frequency  $\Omega$ ) computed using SOR parameter  $\omega$ .

Let  $\mu_i = r_i \mu_1$  denote each eigenvalue of  $H_{GJ}$ , where  $r_i \in [-1, 1]$ . Classical SOR theory [8, 9] guarantees that for each  $\mu_i = r_i \mu_1$ , there is an eigenvalue  $\lambda_i$  of  $H_C(\omega)$  which satisfies

$$(25) \quad \lambda_i - \omega r_i \mu_1 \sqrt{\lambda_i} + (\omega - 1) = 0,$$

and therefore, from the quadratic formula,

$$(26) \quad \sqrt{\lambda_i} = \frac{r_i \mu_1 \omega}{2} + \sqrt{\left(\frac{r_i \mu_1 \omega}{2}\right)^2 - \omega + 1}.$$

Let  $\omega$  to be the conjectured optimal  $\omega_{opt}$ . Combining equation (24) with (26) yields

$$(27) \quad \sqrt{|\lambda_i|} = \left| \frac{1}{2} \mu_1 \omega_{opt} \left[ r_i + \sqrt{r_i^2 - 1} \right] \right| = \left| \frac{1}{2} \mu_1 \omega_{opt} \right|,$$

where the rightmost equality follows from the fact that  $\left| r_i + \sqrt{r_i^2 - 1} \right| = 1$  for  $r_i \in [-1, 1]$ . And as (27) holds for all  $i$ ,

$$(28) \quad \rho(H_C(\omega_{opt})) = |\lambda_i| = \left| \left( \frac{1}{2} \mu_1 \omega_{opt} \right)^2 \right| = |\omega_{opt} - 1|.$$

Equation (28) implies that  $\rho(H_C(\omega))$  cannot be decreased below  $\rho(H_C(\omega_{opt}))$  by using an  $\omega$  such that  $|\omega - 1| > |\omega_{opt} - 1|$ . This follows from the fact that, in general [8, 9],

$$(29) \quad \rho(H_C(\omega)) \geq |\omega - 1|$$

for any  $\omega$ .

To show that  $\rho(H_C(\omega))$  also cannot be decreased by choosing a value of  $\omega$  such that  $|\omega - 1| < |\omega_{opt} - 1|$ , consider the eigenvalue  $\lambda_j$  corresponding to  $\mu_1$ :

$$(30) \quad \sqrt{\lambda_j} = f_+(\omega) = \frac{\mu_1 \omega}{2} + \sqrt{\frac{\mu_1^2 \omega^2}{4} - \omega + 1}$$

and note that  $f_+ : \mathbb{C} \rightarrow \mathbb{C}$ , given by equation (30), is a single-valued, continuous function that is analytic except at

$$(31) \quad \omega_1, \omega_2 = \frac{2}{1 \pm \sqrt{1 - \mu_1^2}}.$$

Since  $|\mu_1| < 1$ , points  $\omega_1$  and  $\omega_2$  lie in the interior and exterior, respectively, of the circle  $|\omega - 1| = 1$  in the complex  $\omega$ -plane. Note that  $\omega_1$  equals the conjectured  $\omega_{opt}$  from equation (24).

Let  $D$  denote the interior of the curve given by the perimeter of the circle  $|\omega - 1| = 1$ , except with a cut along the line defined by the circle's center and  $\omega_1$ . The cut follows the line from the perimeter down to  $\omega_1$ , and then back up the other side to the perimeter, as shown in Figure 5. The function  $f_+$  is nonzero everywhere within  $D$ , since equation (25) implies that a zero can occur only at  $\omega = 1$ , and  $f_+(1) = \mu_1$ . Therefore, the minimum modulus theorem [11] implies that  $|f_+(\omega)|$  attains its minimum value somewhere on the boundary of  $D$ . Finally, the lower bound in (29) implies that  $\omega_1 = \omega_{opt}$  in (24) is the only point on  $D$  which can achieve as low a  $\rho(H_C(\omega))$  as given in (28), completing the proof.  $\square$

Note that when the eigenvalues  $\mu$  lie on a real line segment, this is yet another alternative proof of a classic SOR Theorem [8, 9, 12]. Also note that, in general, the optimal overrelaxation parameter  $\omega(i\Omega)$  is complex.

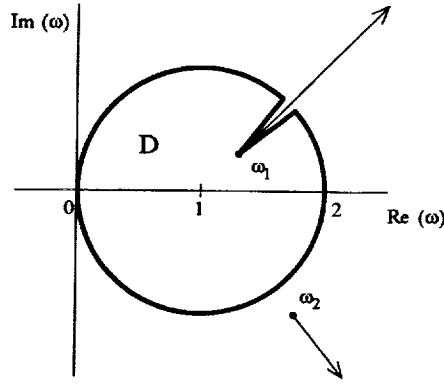


FIG. 5. The region  $D$  and branch cuts in the complex  $\omega$ -plane.

The conditions of optimal SOR parameter Theorem 4.2 are satisfied by a large class of matrices.

*Corollary 4.3.* If  $A$  in (1) is consistently ordered, symmetric, and has constant diagonal  $D = dI$ , then the optimal SOR parameter is given by (24),

$$(32) \quad \omega_{opt}(i\Omega) = \frac{2}{1 + \sqrt{1 - \left(\frac{d\mu_1}{d + i\Omega}\right)^2}},$$

where  $\mu_1$  denotes the spectral radius of  $D^{-1}(L + U)$ .

*Proof.* To show that Theorem 4.2 applies, note that (19) implies that for a constant diagonal  $A$ , the  $H_{GJ}(i\Omega)$  eigenvalues  $\mu(i\Omega)$  are given by  $\mu(i\Omega) = d\mu_0/(d + i\Omega)$ , where  $\mu_0$  are the eigenvalues of  $D^{-1}(L + U)$ . Since  $\mu_0$  lie on the real axis, the  $\mu(i\Omega)$  lie on a line rotated in the complex plane.  $\square$

*Corollary 4.4.* If  $A$  in (1) is consistently ordered, symmetric, and has constant diagonal  $D = dI$ , then the optimal time-dependent SOR convolution waveform  $\omega(t)$  is real.

*Proof.* Equation (32) implies that  $\omega_{opt}(i\Omega)$  is a conjugate-symmetric function of  $\Omega$ .  $\square$

## DISCRETE-TIME MODIFICATION

For the sake of brevity, we consider only the first-order backward difference formula, in which case equation (14) becomes

$$(33) \quad \Delta x^{k+1}[m] + h(D - \omega L)\Delta x^{k+1}[m] - \Delta x^{k+1}[m - 1] = (1 - \omega)\Delta x^k[m] + [(1 - \omega)hD + h\omega U]\Delta x^k[m] - (1 - \omega)\Delta x^k[m - 1],$$



where  $h$  is the uniform timestep. The  $z$ -transform of  $x[m]$ , defined by

$$(34) \quad x(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n} = \mathcal{Z}\{x[n]\},$$

may be used to show that  $\Delta x^{k+1}(z) = H_C(z) \Delta x^k(z)$ , where the  $z$ -dependent convolution SOR operator is

$$H_C(z) = \left( \frac{1-z^{-1}}{h}I + D - \omega(z)L \right)^{-1} \left[ (1-\omega(z)) \left( \frac{1-z^{-1}}{h}I + D \right) + \omega(z)U \right].$$

Since  $\omega(z)$  depends on  $z$ , overrelaxation becomes a convolution sum

$$(35) \quad x_i^{k+1}[m] \leftarrow x_i^k[m] + \sum_{j=-\infty}^{\infty} \omega[m-j] \cdot (\hat{x}_i^{k+1}[j] - x_i^k[j]),$$

where  $\omega(z) = \mathcal{Z}\{\omega[m]\}$ . To determine the optimal  $\omega(z)$ , we have the following theorem, whose proof is analogous to that of Theorem 4.2.

*Theorem 5.5.* If the spectrum of  $H_{GJ}(z)$  lies on the line segment  $[-\mu_1(z), \mu_1(z)]$  with  $|\mu_1| < 1$ , then the spectral radius of  $H_C(z)$  is minimized at  $z$  by the unique optimum  $\omega(z) = \omega_{opt}(z) \in \mathbb{C}$  given by

$$(36) \quad \omega_{opt}(z) = \frac{2}{1 + \sqrt{1 - \mu_1(z)^2}}$$

where  $\sqrt{\cdot}$  denotes the root with the positive real part.

In Example 3.1, matrix  $A$  has constant diagonal  $D = dI$ , so that

$$(37) \quad \omega_{opt}(z) = \frac{2}{1 + \sqrt{1 - \left( \frac{d\mu_1}{d + \frac{1-z^{-1}}{h}} \right)^2}},$$

where  $\mu_1$  denotes the spectral radius of  $D^{-1}(L + U)$ . Thus, to compute the optimal convolution SOR sequence  $\omega[m]$  for the four CSOR plots of Figure 1, equation (37) was used to compute  $\omega(z)$ , and then the inverse  $z$ -transform of  $\omega(z)$  was computed analytically by series expansion.

## DEVICE TRANSIENT SIMULATION

A device is assumed to be governed by the Poisson equation, and the electron and hole continuity equations:

$$(38) \quad \nabla^2 u + c_1 (p - n + N) = 0$$

$$(39) \quad \nabla^2 n - \nabla n \nabla u - n \nabla^2 u = c_2 \frac{\partial n}{\partial t}$$

$$(40) \quad \nabla^2 p + \nabla p \nabla u + p \nabla^2 u = c_3 \frac{\partial p}{\partial t}$$

where  $u$  is the normalized electrostatic potential,  $n$  and  $p$  are the electron and hole concentrations,  $N$  is a background concentration, and  $c_1, c_2, c_3$  are physical constants [13].

Given a rectangular mesh that covers a two-dimensional slice of a MOSFET, a common approach to spatially discretizing the device equations is to use a finite-difference formula to discretize the Poisson equation, and an exponentially-fit finite-difference formula to discretize the continuity equations [13]. On an  $N$ -node rectangular mesh, the spatial discretization yields a differential-algebraic system of  $3N$  equations in  $3N$  unknowns.

The convolution SOR method was implemented in the WR-based device transient simulation program WORDS [14]. WORDS uses red/black block Gauss-Seidel WR, where the blocks correspond to vertical mesh lines. The equations governing nodes in the same block are solved simultaneously using the first order backward-difference formula. The implicit algebraic systems generated by the backward difference formula are solved with Newton's method, and the linear equation systems generated by Newton's method are solved with sparse Gaussian elimination.

The three MOS devices of Figure 6 were used to construct six simulation examples, each device being subjected to either a drain voltage pulse with the gate held high (the **D** examples), or a gate voltage pulse with the drain held high (the **G** examples). All examples ranged from low to high drain current, and in the **G** examples, the gate displacement current was substantial because the applied voltage pulses changed at a rate of  $.2 \sim 2$  volts per picosecond.

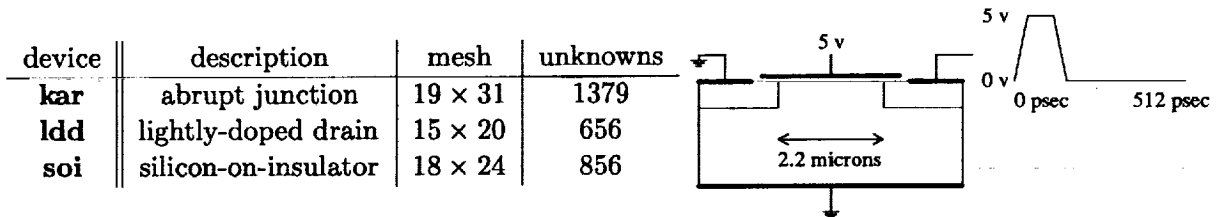


FIG. 6. Description of devices and illustration of the drain-driven **karD** example.

Figure 7 shows the convergence of the six examples as a function of iteration for WR, ordinary waveform SOR (using the pointwise optimum parameter), and the convolution SOR algorithm. The convolution SOR sequence  $\omega[m]$  was calculated by linearizing (38)–(40) about the initial condition, estimating the spectral radius of the iteration matrix as a function of  $z$ , applying Theorem 5.5 and inverse transforming. Both overrelaxation methods were applied only to the potential variable  $u$ . All simulations began with 64 initial WR iterations, and used 256 equally-spaced timesteps. In Figure 7, convergence was measured using the terminal current error.

Despite the nonlinearity of the semiconductor equations, the convolution SOR algorithm converged substantially faster than either WR or ordinary waveform SOR, demonstrating the robustness of the approach.

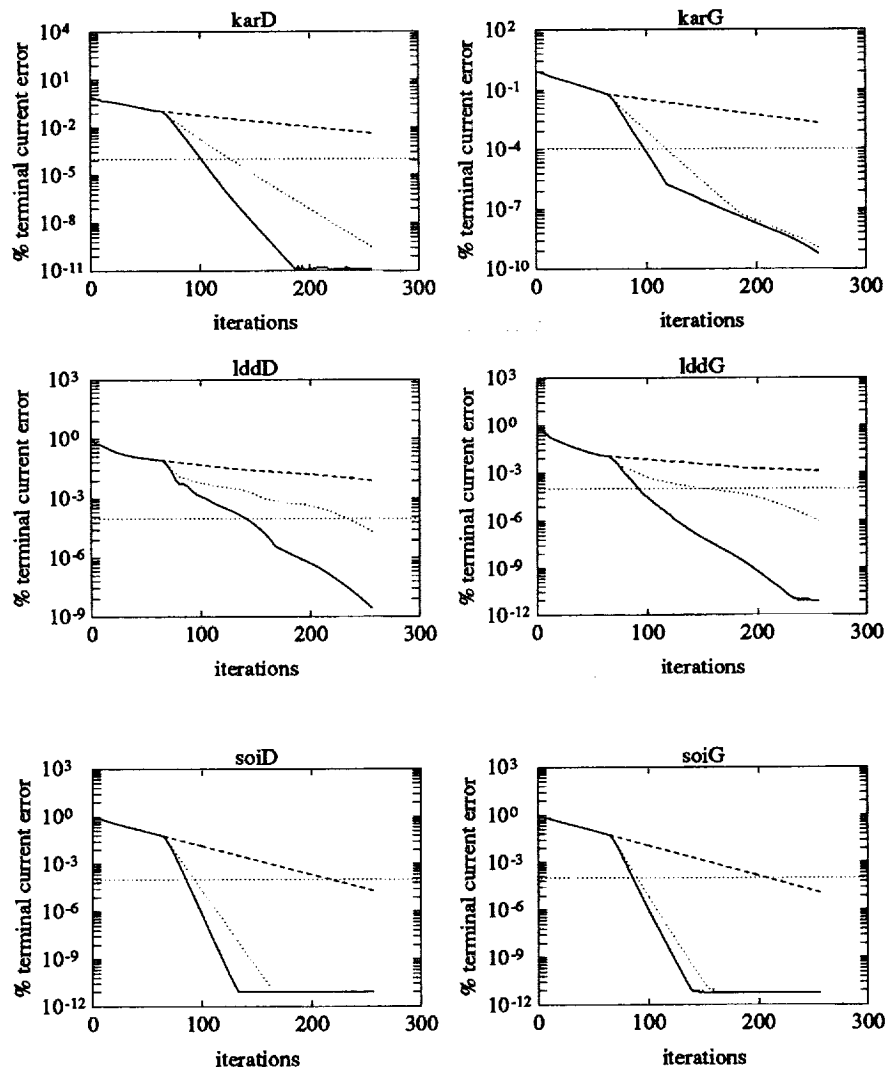


FIG. 7. Terminal current error of the six examples as a function of iteration for WR (dashed), ordinary waveform SOR (dotted), and convolution SOR (solid).

## CONCLUSION

In this paper, a new waveform overrelaxation algorithm was presented and applied to solving the differential-algebraic system generated by spatial discretization of the time-dependent semiconductor device equations. In the experiments included, the convolution SOR algorithm converged robustly, and substantially faster than ordinary WR.

The author would like to acknowledge extensive conversations with his advisor, Professor Jacob White, and also thank Professors Alar Toomre, Donald Rose, Paul Lanzcron, Andrew Lumsdaine and Olavi Nevanlinna for many valuable suggestions.

## REFERENCES

- [1] A. Agarwal, "Limits on interconnection network performance," *IEEE Trans. Parallel Distrib. Sys.*, pp. 398–412, October 1991.
- [2] U. Miekkala and O. Nevanlinna, "Convergence of dynamic iteration methods for initial value problems," *SIAM J. Sci. Statist. Comput.*, vol. 8, pp. 459–482, July 1987.
- [3] J. K. White and A. Sangiovanni-Vincentelli, *Relaxation Techniques for the Simulation of VLSI Circuits*. The Kluwer International Series in Engineering and Computer Science, Boston: Kluwer Academic Publishers, 1987.
- [4] E. Lelarsmee, A. E. Ruehli, and A. L. Sangiovanni-Vincentelli, "The waveform relaxation method for time domain analysis of large scale integrated circuits," *IEEE Trans. CAD*, vol. 1, pp. 131–145, July 1982.
- [5] R. D. Skeel, "Waveform iteration and the shifted Picard splitting," *SIAM J. Sci. Statist. Comput.*, vol. 10, no. 4, pp. 756–776, 1989.
- [6] C. Lubich and A. Osterman, "Multi-grid dynamic iteration for parabolic equations," *BIT*, vol. 27, pp. 216–234, 1987.
- [7] A. Lumsdaine, *Theoretical and Practical Aspects of Parallel Numerical Algorithms for Initial Value Problems, with Applications*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1992.
- [8] D. M. Young, *Iterative Solution of Large Linear Systems*. Orlando, FL: Academic Press, 1971.
- [9] R. S. Varga, *Matrix Iterative Analysis*. Automatic Computation Series, Englewood Cliffs, NJ: Prentice-Hall, 1962.
- [10] L. N. Trefethen, "Pseudospectra of matrices," in *Numerical Analysis 1991* (D. F. Griffiths and G. A. Watson, eds.), Harlow, Essex, England: Longman Scientific & Technical, 1992.
- [11] E. B. Saff and A. D. Snider, *Fundamentals of Complex Analysis for Mathematics, Science, and Engineering*. New Jersey: Prentice-Hall, Inc., 1976.
- [12] R. C. Y. Chin and T. A. Manteuffel, "An analysis of block successive overrelaxation for a class of matrices with complex spectra," *SIAM J. Numer. Anal.*, vol. 25, pp. 564–585, June 1988.
- [13] S. Selberherr, *Analysis and Simulation of Semiconductor Devices*. New York: Springer-Verlag, 1984.
- [14] M. Reichelt, J. White, and J. Allen, "Waveform relaxation for transient two-dimensional simulation of MOS devices," in *Proc. International Conference on Computer-Aided Design*, (Santa Clara, CA), pp. 412–415, November 1989.

# A MULTIGRID METHOD FOR STEADY EULER EQUATIONS ON UNSTRUCTURED ADAPTIVE GRIDS

Kris Rienslagh and Erik Dick  
Universiteit Gent  
Sint-Pietersnieuwstraat 41, 9000 Gent  
Belgium

5/2-64  
197572  
p. 15

## SUMMARY

A flux-difference splitting type algorithm is formulated for the steady Euler equations on unstructured grids. The polynomial flux-difference splitting technique is used. A vertex-centered finite volume method is employed on a triangular mesh.

The multigrid method is in defect-correction form. A relaxation procedure with a first order accurate inner iteration and a second-order correction performed only on the finest grid, is used. A multi-stage Jacobi relaxation method is employed as a smoother. Since the grid is unstructured a Jacobi type is chosen. The multi-staging is necessary to provide sufficient smoothing properties.

The domain is discretized using a Delaunay triangular mesh generator. Three grids with more or less uniform distribution of nodes but with different resolution are generated by successive refinement of the coarsest grid. Nodes of coarser grids appear in the finer grids. The multigrid method is started on these grids. As soon as the residual drops below a threshold value, an adaptive refinement is started. The solution on the adaptively refined grid is accelerated by a multigrid procedure. The coarser multigrid grids are generated by successive coarsening through point removal. The adaption cycle is repeated a few times.

Results are given for the transonic flow over a NACA-0012 airfoil.

## THE SPACE DISCRETIZATION

### Flux-Difference Splitting

Figure 1 shows part of an unstructured triangular grid. In the vertex-centered finite volume method nodes are located at the vertices of the grid. Every node has a control volume, constructed by connecting the centers of gravity of the cells surrounding the node. To close the control volumes on the boundary, the midpoints of the boundary edges are chosen as vertices of the control volumes.

To define the flux through a side of a control volume, use is made of the flux-difference splitting

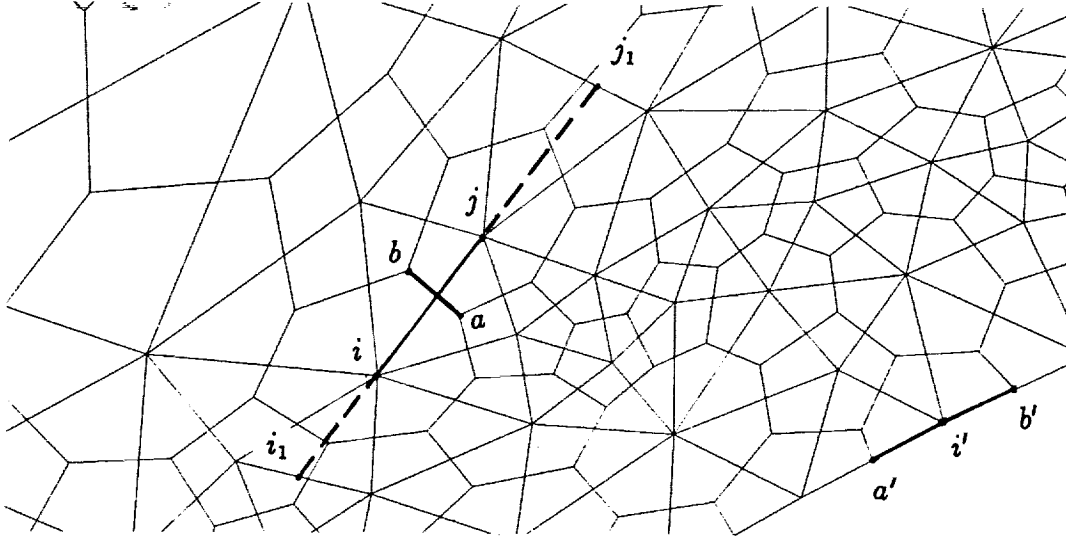


Figure 1: Vertex-centered discretization.

principle. The Euler equations in two dimensions take the form

$$\frac{\partial U}{\partial t} + \frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} = 0, \quad (1)$$

where  $U$  is the vector of conserved variables,  $f$  and  $g$  the Cartesian flux vectors, given by

$$U = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, f = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uH \end{pmatrix}, g = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vH \end{pmatrix}, \quad (2)$$

where  $\rho$  is the density;  $u$  and  $v$  are the Cartesian velocity components in the  $x$  and  $y$  directions, respectively;  $p$  is the pressure;  $E = p/(\gamma - 1)\rho + u^2/2 + v^2/2$  is the total energy;  $H = \gamma p/(\gamma - 1)\rho + u^2/2 + v^2/2$  is the total enthalpy; and  $\gamma$  is the adiabatic constant.

For the side  $ab$  of the node  $i$  and node  $j$  control volumes (figure 1), the flux-difference can be written as

$$\Delta F_{i,j} = (n_x \Delta f_{i,j} + n_y \Delta g_{i,j}) \Delta s_{i,j}, \quad (3)$$

where  $\Delta f_{i,j}$  and  $\Delta g_{i,j}$  denote the differences of the Cartesian flux vectors,  $n_x$  and  $n_y$  are the components of the unit normal to the side  $ab$  in the sense  $i$  to  $j$ , and  $\Delta s_{i,j}$  is the length of the side. The differences of the Cartesian flux vectors are

$$\Delta f_{i,j} = f_j - f_i, \quad \Delta g_{i,j} = g_j - g_i, \quad (4)$$

where  $f_i, g_i$  and  $f_j, g_j$  are the flux vectors calculated with the flow variables in node  $i$  and node  $j$ , respectively.

The flux-difference defined by equation (3) can be written as

$$\Delta F_{i,j} = A_{i,j} (U_j - U_i) \Delta s_{i,j} = A_{i,j} \Delta U_{i,j} \Delta s_{i,j}. \quad (5)$$

To define the discrete Jacobian  $A$ , the polynomial flux-difference splitting is used. This splitting technique was introduced by the second author [1]. Full details are given in [2, 3]. The polynomial flux-difference splitting is a Roe-type technique, i.e. it satisfies the primary requirements formulated by Roe [4]. However, it is simpler. Its simplicity follows from dropping the secondary requirement of having a unique definition of averaged flow variables. This secondary requirement defines the original Roe-splitting within the class of methods allowed by Roe's primary requirements. However, the secondary requirement is not necessary. The precise splitting used is not really relevant for the method we describe here. Therefore, we do not detail the method any further. The only important result is equation (5).

The discrete Jacobian  $A_{i,j}$  in equation (5) has real eigenvalues that are discrete analogs of the normal velocity through the side of the control volume, and the normal velocity plus and minus the velocity of sound. The Jacobian also has a complete set of eigenvectors. These properties are a direct consequence of the hyperbolic character of the Euler equations with respect to time. The Jacobian matrix  $A$  can be written as

$$A = R \Lambda L, \quad (6)$$

where  $\Lambda$  denotes the eigenvalue matrix and where  $R$  and  $L$  denote the right and left eigenvector matrices in orthonormal form. The matrix  $A$  can be split into positive and negative parts by

$$A^+ = R \Lambda^+ L, \quad A^- = R \Lambda^- L. \quad (7)$$

### Upwind Flux Definition

For the side  $ab$  of the node  $i$  and node  $j$  control volumes (figure 1), the first order upwind flux is defined by

$$F_{i,j}^1 = \frac{1}{2}(F_i + F_j) - \frac{1}{2}(A_{i,j}^+ - A_{i,j}^-) \Delta U_{i,j} \Delta s_{i,j}, \quad (8)$$

where

$$F_i = (n_x f_i + n_y g_i) \Delta s_{i,j}, \quad F_j = (n_x f_j + n_y g_j) \Delta s_{i,j}.$$

Using equation (5), equation (8) can be written as

$$F_{i,j}^1 = F_i + A_{i,j}^- \Delta U_{i,j} \Delta s_{i,j}. \quad (9)$$

This way of writing the flux shows the incoming wave components.

In order to define a second-order flux, the second part in the right hand side of equation (8), which contains the positive and negative parts of the flux-difference, is decomposed into components along the eigenvectors of the Jacobian, according to

$$\Delta F_{i,j}^\pm = A_{i,j}^\pm \Delta U_{i,j} \Delta s_{i,j} = \sum_n r_{i,j}^n \lambda_{i,j}^{\pm n} l_{i,j}^n \Delta U_{i,j} \Delta s_{i,j}, \quad (10)$$

where  $r^n$  and  $l^n$  are right and left eigenvectors of  $A$  associated to the eigenvalue  $\lambda^n$ .  $r^n$  and  $l^n$  are components of  $R$  and  $L$ , respectively. By denoting the projection of  $\Delta U_{i,j}$  on the  $n^{th}$  eigenvector by

$$\sigma_{i,j}^n = l_{i,j}^n \Delta U_{i,j}, \quad (11)$$

the parts of the flux-difference become

$$\Delta F_{i,j}^{\pm} = \sum_n r_{i,j}^n \lambda_{i,j}^{n\pm} \sigma_{i,j}^n \Delta s_{i,j} = \sum_n \Delta F_{i,j}^{n\pm}. \quad (12)$$

The second-order flux is then defined by

$$F_{i,j}^2 = \frac{1}{2}(F_i + F_j) - \frac{1}{2} \sum_n \Delta F_{i,j}^{n+} + \frac{1}{2} \sum_n \Delta F_{i,j}^{n-} + \frac{1}{2} \sum_n \Delta \tilde{F}_{i,j}^{n+} - \frac{1}{2} \sum_n \Delta \tilde{F}_{i,j}^{n-}, \quad (13)$$

where  $\Delta \tilde{F}^{n\pm}$  are limited combinations of the flux-difference components  $\Delta F^{n\pm}$  and the shifted differences  $\Delta \tilde{F}^{n\pm}$ , in the sense of  $i$  for positive components and in the sense of  $j$  for negative components. The limiter used here is the minmod-limiter. The shifted flux-difference components are constructed based on shifted differences of conservative variables. These are obtained by extending the line segment  $i, j$  into the adjacent triangles and constructing intersections with the opposing sides (point  $i_1$  and point  $j_1$  in figure 1). The shifted flux-differences are defined by

$$\Delta \tilde{F}_{i,j}^{n\pm} = r_{i,j}^n \lambda_{i,j}^{n\pm} \tilde{\sigma}_{i,j}^{n\pm} \Delta s_{i,j} = r_{i,j}^n \lambda_{i,j}^{n\pm} l_{i,j}^n \Delta \tilde{U}_{i,j}^{n\pm} \Delta s_{i,j}, \quad (14)$$

where  $\Delta \tilde{U}_{i,j}^{n\pm}$  denotes the shifted differences. The employed technique to define the second-order flux commonly is called the flux-extrapolation technique. This concept was introduced by Chakravarthy and Osher [5]. For examples illustrating the quality of this second-order formulation on structured grids, the reader is referred to [2, 3, 6].

### Boundary Conditions

The examples to follow are either channel flows or flows around airfoils. The internal-type flows have solid, inlet, and outlet boundary conditions. The external-type flows have solid boundaries and far-field boundaries. Figure 2 gives an example of an external flow. The grid generation is detailed in a section below. The far-field boundary is a hexagon with sides 100 chord lengths away from the airfoil.

Inflow and outflow boundary conditions are imposed through the definition of the boundary flux. The boundary flux is calculated by equation (8) with the flux Jacobian  $A_{i,j}$  determined with the values of the variables of the boundary node and the difference  $\Delta U_{i,j}$  taken as the difference between values in the boundary node and in a fictitious node. The variables in this fictitious node are calculated by the classic extrapolation procedure. At a subsonic inlet, the Mach number is extrapolated while stagnation conditions and flow direction are imposed. For outflow, the stagnation properties and flow direction are extrapolated while the Mach number is imposed.

At solid boundaries, impermeability is imposed by setting the convective part of the flux equal to zero. Thus a special flux definition is used for the boundary edges of the control volumes of boundary



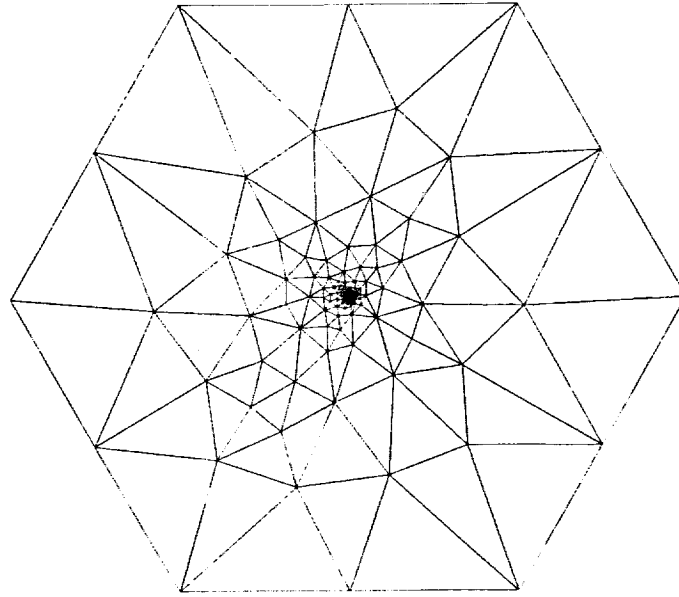


Figure 2: Triangulation around an airfoil with a hexagon far-field boundary 100 chords away from the airfoil. The large distance between boundary and airfoil makes that the airfoil itself is not visible on this figure.

nodes. For a point on the boundary ( $i'$  in figure 1), the flux through the boundary side can be written as

$$F_{i,j'} = F'_{i'} \quad (15)$$

where  $F'_{i'}$  is the flux calculated with the variables in the node  $i'$ , and where the convective part of the flux is set equal to zero. To give the same appearance to a boundary flux as to an interior flux, all flux expressions can be written as

$$F_{i,j} = F_i + A_{i,j}^- \Delta U_{i,j} \Delta s_{i,j} + S.O., \quad (16)$$

where  $S.O.$  denotes the second-order correction. For a boundary node the point  $j$  does not exist. This can be introduced by taking the values of the variables in the fictitious node  $j$  equal to the values of the variables in the node  $i$ . So, the first order difference of the variables  $\Delta U_{i,j}$  and the second-order correction term  $S.O.$  in the r.h.s. of equation (16) vanish. The matrix  $A_{i,j}^-$  in equation (16) is then calculated with the values of the variables in the node  $i$ . The impermeability is introduced in the term  $F_i$ . As will be discussed in the next section, the matrix  $A_{i,j}^-$  at a solid boundary plays an important role in the relaxation method, although it is multiplied with a zero term.

For an external-type flow, fluxes through edges on the far-field boundary are defined by equation (9). The Jacobian is calculated with the values in the boundary node. The values in the fictitious outside node are taken from the uniform far-field flow. By the flux difference splitting this procedure is equivalent to the use of Riemann invariants.

## THE MULTI-STAGE RELAXATION METHOD

In our earlier multigrid formulations for steady Euler equations using structured grids, the Gauss-Seidel relaxation method was always used [2, 3, 6]. Gauss-Seidel relaxation was preferred to Jacobi-relaxation because it has much better smoothing properties (effectiveness of the coarse grid correction) and much better speed of convergence (effectiveness of the relaxation method itself). For structured grid applications the sequential Gauss-Seidel relaxation is very natural. However, it is not simple to use a sequential relaxation method on an unstructured grid, since this requires the construction of paths through the grid. On an unstructured grid, a much more natural method is a simultaneous method instead of a successive one. A simultaneous relaxation method, like the Jacobi relaxation, also has the further advantage of being easily vectorizable and parallelizable. The only drawback is that a simultaneous relaxation method, at least in its basic form, is much less effective than a sequential method.

To repair this, we bring multi-staging into the Jacobi method in the same way as multi-staging is used for time-stepping methods and we use the optimization results with respect to smoothing known for time-stepping schemes. This was first suggested by Morano et al. [7], but not worked out in detail. A more detailed analysis on the possible multigrid performance was made by the authors [8].

We do not intend here to enter a principal discussion on the possibilities of multi-stage Jacobi relaxation. We only want to illustrate that it can be used and that it is sufficiently effective, certainly on unstructured grids, to be attractive in a multigrid context. We choose here a priori the defect correction multigrid procedure as was used in [2, 3, 6]. This means that the second-order part of the flux (S.O. in equation (16)) is updated only on the finest grid and is frozen on all other grids. We choose this configuration mainly for simplicity. The inner, multigrid, iteration cycle is then based on a linear discretization scheme for the partial differential equations. As a consequence of the linearity, this cycle can be rigorously optimized. The outer, defect correction, cycle does not involve any parameters and does not require optimization. In the outer cycle, a second-order accurate flux definition is used. The second-order part of the flux is highly non-linear due to the TVD-limiter. It is certainly possible to use second-order operators in the multigrid formulation as shown in [7] and [8], but it is not clear how to optimize. Therefore, the difficulties associated with non-linear discretization schemes in the multigrid formulation are avoided and only first order discretizations are considered.

For the time-dependent Euler equations, the first order discrete set of equations associated with the node  $i$  reads

$$Vol_i \frac{dU_i}{dt} + \sum_j A_{i,j}^-(U_j - U_i) \Delta s_{i,j} = 0, \quad (17)$$

where the index  $j$  loops over the faces of the control volume and the surrounding nodes and where  $Vol_i$  is the area of the two dimensional control volume. A single-stage time-stepping method with local time-stepping applied to equation (17) gives

$$\left( \frac{Vol_i}{\Delta t_i} \right) (U_i^{n+1} - U_i^n) + \sum_j A_{i,j}^{-n} (U_j^n - U_i^n) \Delta s_{i,j} = 0, \quad (18)$$

where the superscript  $n$  denotes the time level.

Using increments  $\delta U_i = U_i^{n+1} - U_i^n$ , equation (18) is written as

$$\left( \frac{Vol_i}{\Delta t_i} \right) \delta U_i + \sum_j A_{i,j}^{-n} (U_j^n - U_i^n) \Delta s_{i,j} = 0. \quad (19)$$

The Jacobi-relaxation applied to the steady part of equation (17) reads

$$\sum_j A_{i,j}^{-n} (U_j^n - U_i^{n+1}) \Delta s_{i,j} = 0. \quad (20)$$

Using increments  $\delta U_i = U_i^{n+1} - U_i^n$ , this gives

$$\left( - \sum_j A_{i,j}^{-n} \Delta s_{i,j} \right) \delta U_i + \sum_j A_{i,j}^{-n} (U_j^n - U_i^n) \Delta s_{i,j} = 0. \quad (21)$$

The 4x4 matrix coefficient of  $\delta U_i$  in equation (21) is non-singular. In equations (18) to (21), the matrices  $A_{i,j}^{-n}$  are at the time or relaxation level  $n$ . The difference between (single-stage) Jacobi relaxation equation (21) and single-stage time-stepping equation (18) is seen in the matrix coefficient of the vector of increments  $\delta U_i$ . In the time-stepping method, the coefficient is a diagonal matrix. In the Jacobi method, the matrix is composed of parts of the flux-Jacobians associated with the different faces of the control volume. The collected parts correspond to waves incoming to the control volume. In the time-stepping, the incoming waves contribute to the increment of the flow variables all with the same weight factor. In the Jacobi relaxation the corresponding weight factors are proportional to the wave speeds. As a consequence, Jacobi relaxation can be seen as a time-stepping in which all incoming wave components are scaled to have the same effective speed. This is very important for the optimization in the sequel.

For a node on a solid boundary, an expression similar to equation (21) is obtained provided that for a face on the boundary the flux expression of equation (16) is used and that the difference in the first order flux-difference part is introduced as  $U_i^n - U_i^{n+1}$ , (similar to the term  $U_j^n - U_i^{n+1}$  which is used for a flux on an interior face). In order to avoid a singular matrix coefficient of the vector of increments in equation (21), this special treatment at boundaries is necessary. A boundary node can then be updated precisely in the same way as a node in the interior.

To bring in multi-staging is now very simple. For instance, a three-stage Jacobi relaxation is given by

$$\begin{aligned} U_0 &= U_i^n \\ U_1 &= U_0 + \alpha_1 \delta U_i^0 \\ U_2 &= U_0 + \alpha_2 \delta U_i^1 \\ U_3 &= U_0 + \alpha_3 \delta U_i^2 \\ U_i^{n+1} &= U_3, \end{aligned}$$

with

$$\left( - \sum_j A_{i,j}^{-l} \Delta s_{i,j} \right) \delta U_i^l = R_i^l = - \sum_j A_{i,j}^{-l} \Delta U_{i,j}^l \Delta s_{i,j}. \quad (22)$$

The parameters  $\alpha_1, \alpha_2$  and  $\alpha_3$  are to be chosen. The increment  $\delta U_i^l$  is obtained from the single-stage Jacobi relaxation method. The method is converted to a multi-stage time-stepping if  $\delta U_i^l$  is replaced by the increment obtained from the single-stage time-stepping method with *CFL*-number equal to 1. In the sequel, we use the optimization results obtained by Van Leer et al. [9]. For three-stage relaxation, the parameters are  $\alpha_1 = 0.1481$ ,  $\alpha_3 = 0.40$ ,  $\alpha_2 = 0.40$  and  $\alpha_3 = 1.5$ . Since by the Jacobi relaxation all wave components are scaled to have the same speed, the tuning of the smoothing is correct for all wave components.

To illustrate the performance of multi-stage Jacobi relaxation, we consider a test problem in which no adaption is used. Figure 3 shows the test geometry discretized with two unstructured grids with 10557 and 2657 nodes. The grids have a more or less uniform distribution of the mesh size. Two coarser grids (not shown) with 683 and 182 nodes are also used. The bump has a height of 4% of its chord. The channel height is equal to the bump chord.

Figure 4 shows the iso-Mach line results obtained for a transonic case with an outlet Mach number of 0.79 and for a supersonic case with inlet Mach number of 1.4

Figure 5 shows the convergence history. A full multigrid method with *W*-cycles is employed. The calculation starts from uniform flow on the coarsest grid. On each level one three-stage Jacobi relaxation is done. One Jacobi stage and one residual evaluation for all nodes on the finest grid, are both counted as one work unit. The corresponding work on a coarse grid is counted proportional to the number of nodes on that grid. A second-order correction is also counted as one work unit. The work associated to intergrid transfer is neglected. Coarse grid points also appear in the finer grids. So injection is used for function value restriction. Defect restriction is obtained through weighting by

$$\frac{R_{i'}}{Vol_{i'}} = \frac{1}{2} \left( \frac{R_i}{Vol_i} + \frac{1}{n} \sum_j^n \frac{R_j}{Vol_j} \right),$$

where the node  $i$  on the finer grid corresponds with the node  $i'$  on the coarser grid and where  $j$  loops over the  $n$  neighboring nodes of  $i$ .  $R_i$  stands for the residual as given by equation (22) and  $Vol_i$  is the volume of the control volume. The prolongation is constructed by direct transfer of the coarse grid correction to the corresponding points in the finer grid. Fine grid points that do not correspond to a coarse grid point are given a correction value based on an average of the corrections at the neighboring nodes that do appear on the coarser grid.

The convergence history of the maximum residual is shown in figure 5. The convergence rate is about one order of magnitude per 250 work units. This is an acceptable performance. The best convergence rate on structured grids using Gauss-Seidel relaxation on comparable problems is about one magnitude per 50 work units [2, 3, 6, 10]. Due to the use of a Gauss-Seidel type smoother the convergence rate in the structured case is better.

## THE MESH GENERATION

The automatic triangulation of an arbitrary set of points can be achieved using Delaunay triangulation. Robust algorithms to construct this triangulation in 2D are available.

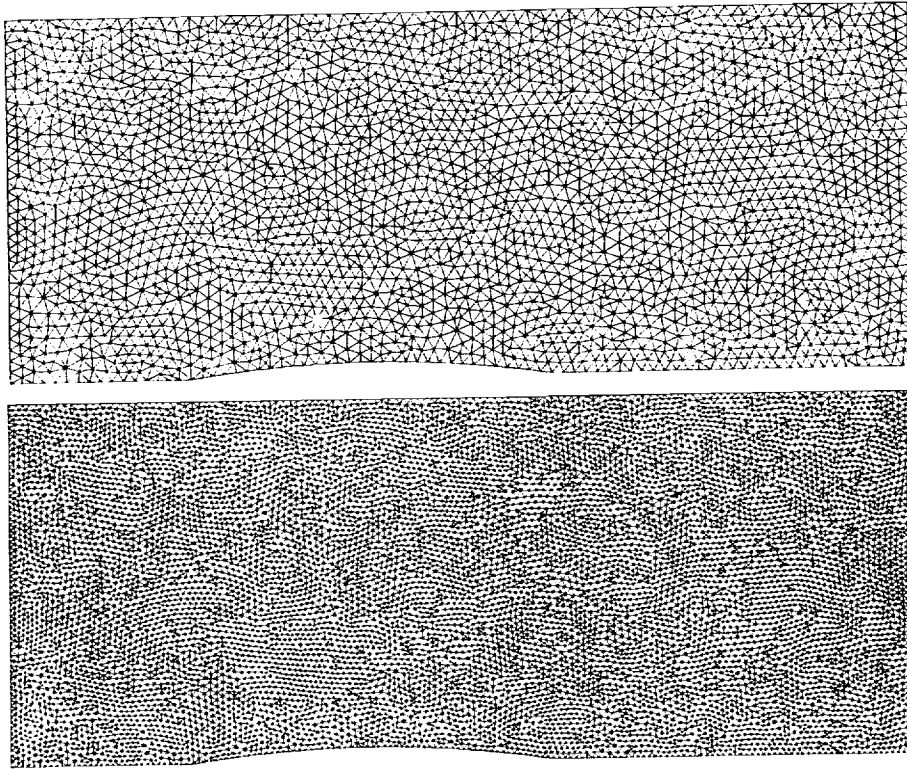


Figure 3: Top : Unstructured grid with 2657 nodes; Bottom : Finest grid with 10557 nodes.

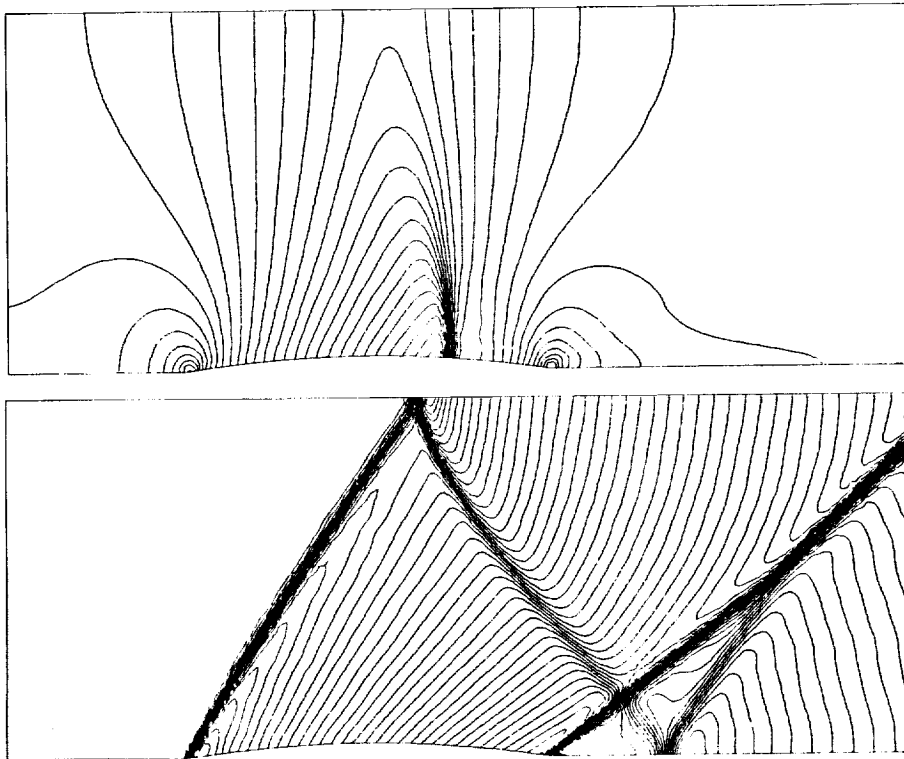


Figure 4: Iso-Mach lines calculated for the transonic test case Mach 0.79 (top) and the supersonic test case Mach 1.4 (bottom). Iso-mach lines per 0.02. Defect correction result with minmod-limiter.

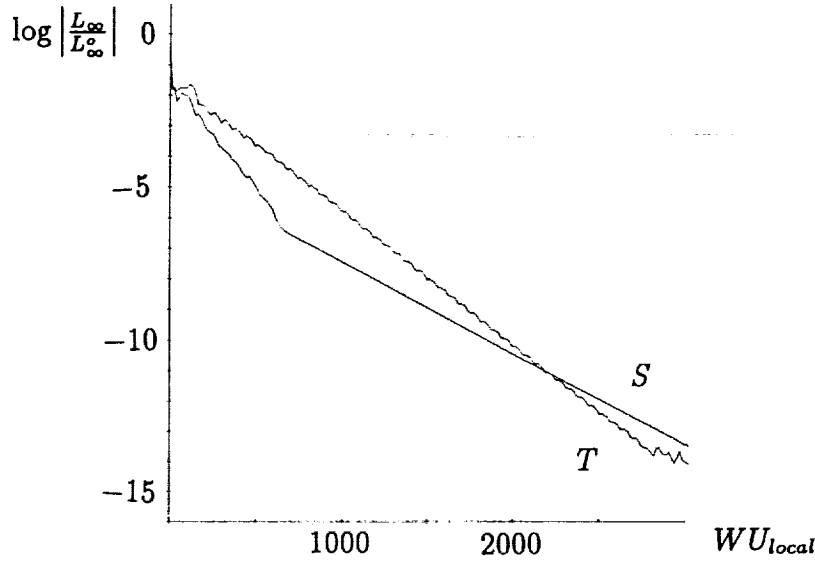


Figure 5: Convergence behaviour for the second-order defect correction scheme (TVD minmod-limiter) using three-stage Jacobi relaxation. Transonic (T) and supersonic (S) test cases.

The grid generator is built with two basic algorithms. The first one is based on the advancing front method, developed by Tanemura et al. [11], simplified to work on a set of nodes which are already connected. The nodes have to lie on the boundary of a polygon. With this algorithm a given, not necessarily convex, polygon can be triangulated. No extra nodes are added. The result is a constrained Delaunay triangulation of the given polygonal boundary. This algorithm was used for the generation of the initial triangulation of the domain. The domain was defined by discretizing the boundaries based on local curvature and local grid spacing.

The second basic algorithm used in the grid generation allows the addition of a node to an existing triangulation. The new node is connected to three or four existing nodes: three nodes if the new node lies inside a triangle, four nodes if the new node lies on an edge. The triangulation is made Delaunay by the use of a diagonal swapping algorithm [12].

For a sequence of multigrid grids, our formulation of the intergrid transfer operators demands that a node of a coarse grid appears in the finer grids. Two strategies can be used to satisfy this condition. A sequence of grids can be constructed by refining a coarse grid (adding nodes), or by coarsening a fine grid (deleting nodes). To add a node to a triangulation the second basic algorithm is used. To delete a node, the node (together with all the connected edges) is removed from the triangulation and then the remaining cavity is retriangulated using the first basic algorithm.

To allow stretching in the construction of the grid, the above described Delaunay triangulation is performed in a transformed space. Every node has its own transformation parameters,  $(\theta, s_{x'}, s_{y'})$ . The transformation is a rotation over the angle  $\theta$ , followed by a rescaling of the new  $x'$  and  $y'$  axes by  $s_{x'}$  and  $s_{y'}$ . The criteria by which the Delaunay triangulation is built are based on transformed properties.

For the first test case, the bump in a channel of figure 3, multiple grids are built from coarse to fine by adding nodes. An initial grid is generated based on the discretization of the boundary.

Transformation parameters are calculated from boundary mesh spacing and boundary edge direction. The initial grid is refined to form the coarsest grid in the computation. Interior nodes are added in the middle of selected edges, until all edges satisfy an edge length criterion in the locally transformed space. Finer meshes are generated by taking a smaller value of the threshold on the edge length.

The final mesh is smoothed, moving all the nodes of all the grids. It is possible that by this smoothing the triangulation is no longer Delaunay. Therefore, an edge swapping algorithm is used to restore the Delaunay property on all the grids.

For the second test case, the NACA-0012 airfoil, the first multigrid sequence is built in the same way. Then a solution is determined. A new mesh is now generated based on flow adaptive refinement, in which three refinement criteria are used. The first criterion is based on the pressure difference over an edge. If  $|p_i - p_j| L_{i,j} \geq L_{ref} |p_{max} - p_{min}| / C_p$ , then the edge  $i, j$  is refined by placing a new node into the center of the edge. In this formulation  $p_i$  is the pressure at node  $i$  and  $L_{i,j}$  is the length of the edge. The variables at the right hand side are minimum or maximum values taken over all the nodes.  $L_{ref}$  is a problem dependent reference length and  $C_p$  is the sensitivity constant for the pressure criterion. This criterion triggers shockwaves and stagnation regions. The second criterion is based on the entropy difference over an edge. It is similar to the first criterion, however  $p$  is replaced by the entropy  $s$ . This criterion triggers shock waves and tangential discontinuities. Further, edges are refined where the flow passes through Mach number equal to 1 from supersonic to subsonic flow. This criterion also triggers shock waves.

The adaptively refined mesh can be included directly in the multigrid sequence but large parts of the mesh might coincide with the next coarser mesh. This results in a degradation of the multigrid performance. Therefore, it is better to generate coarser meshes by coarsening the new mesh. The simple criterion, removing a node only if no neighbors of this node are removed, is used. The number of nodes is decreased by roughly 1/4 by doing this. This coarsening is repeated twice to get a coarser mesh.

## MULTIGRID RESULTS USING ADAPTIVITY

Starting from the grid shown in figure 2, two successive stages of refinement were done using a threshold value on the edge length in the locally transformed space. Figure 6 shows part of the final grid generated this way. On this grid a solution is calculated using multigrid. By the use of the foregoing adaption criteria, this mesh is refined. Three coarser grids are generated from this grid with the above described coarsening procedure. A solution is calculated using the four grids. The adaption is repeated three times. The solution on the final mesh is given in figure 7 in the form of iso-Mach lines. The NACA-0012 profile has an angle of attack of 1.25 degrees and the Mach number of the incoming flow is 0.80. In total, 4 fine meshes (with for every fine mesh 2 or 3 coarser meshes) were used with 1302 (figure 6) (594,300), 2243 (1256,720,421), 2834 (1577,911,522), 3236 (figure 7) (1782,1010,580) nodes. Figure 8 shows the final grid structure in the shock regions. Figure 9 shows the final grid structure in the leading edge and the trailing edge regions. The refinement of the fourth grid is shown in the left part of figure ?? where only the edges selected for refinement are shown. The right part of figure 10 shows the pressure distribution over the a irfoil.

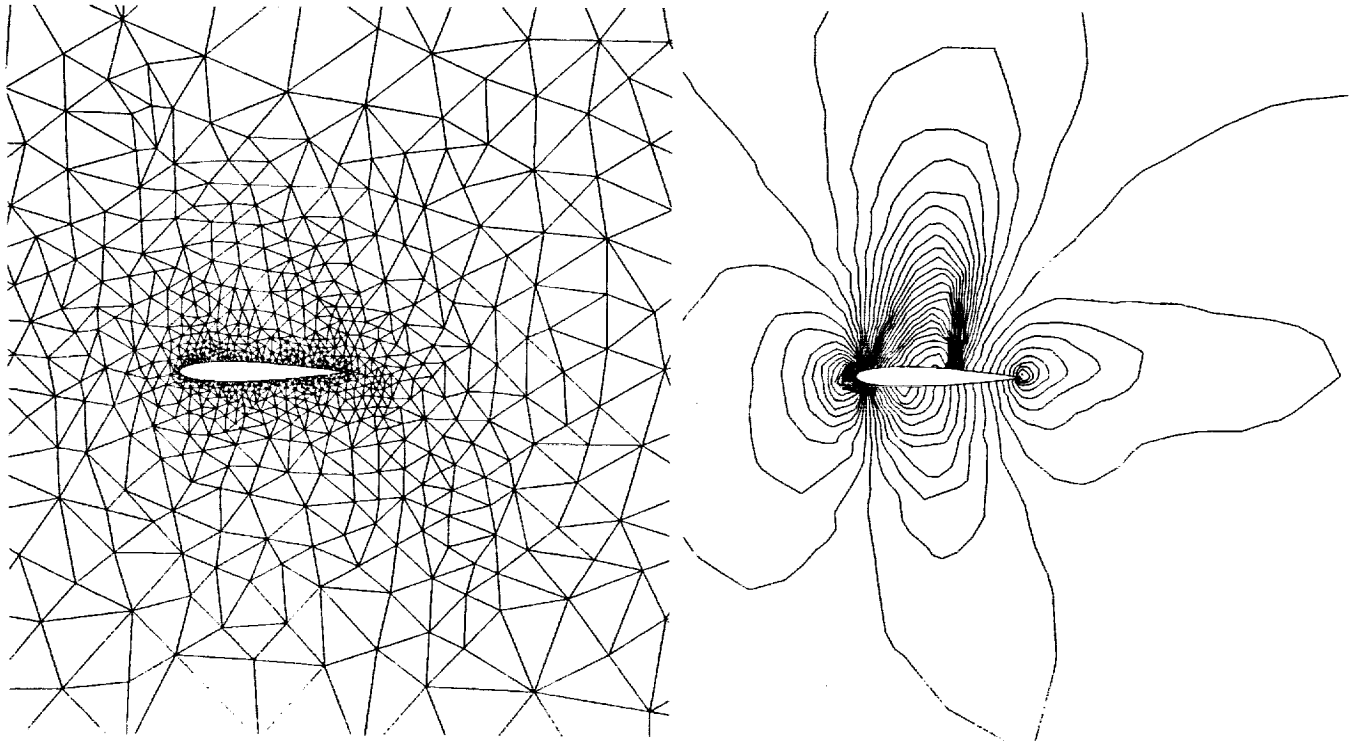


Figure 6: Left: First grid (1302 nodes), Right: Iso-Mach lines per 0.02.

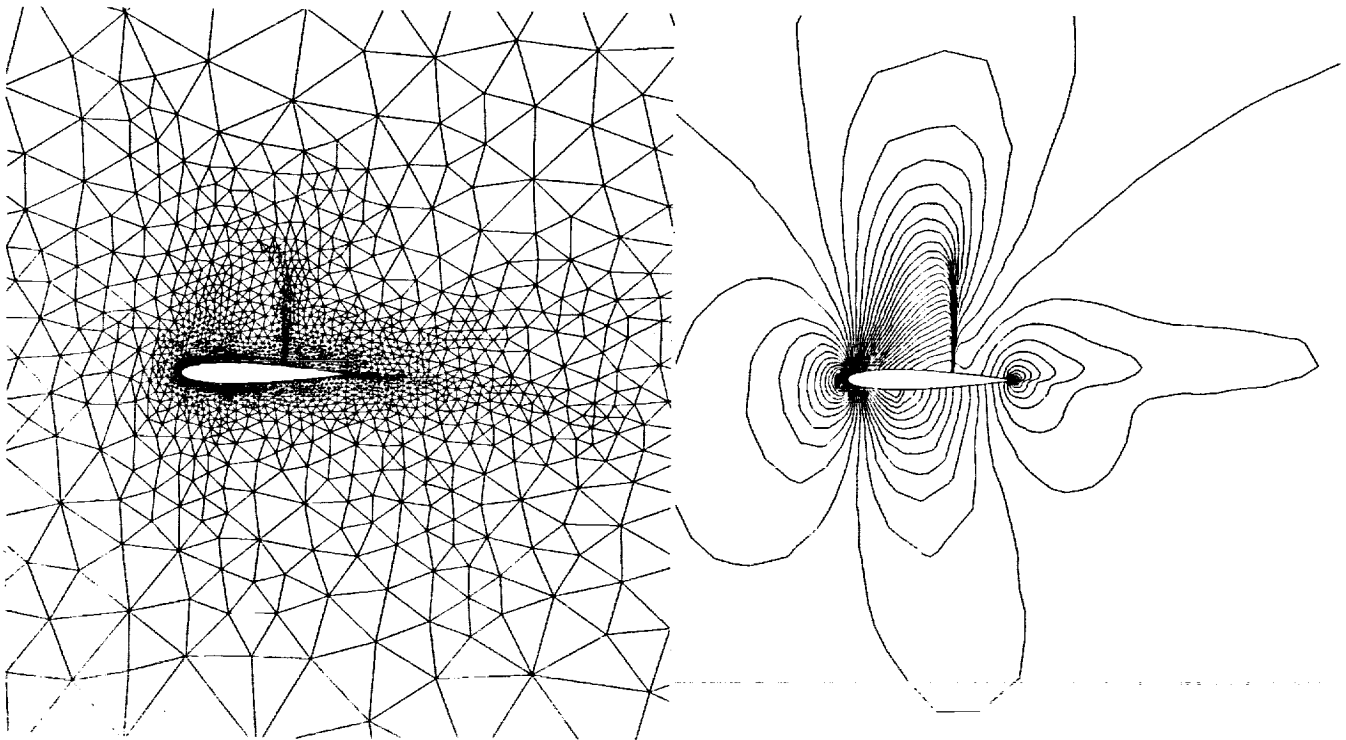


Figure 7: Left: Fourth and final grid (3236 nodes), Right: Iso-Mach lines per 0.02.



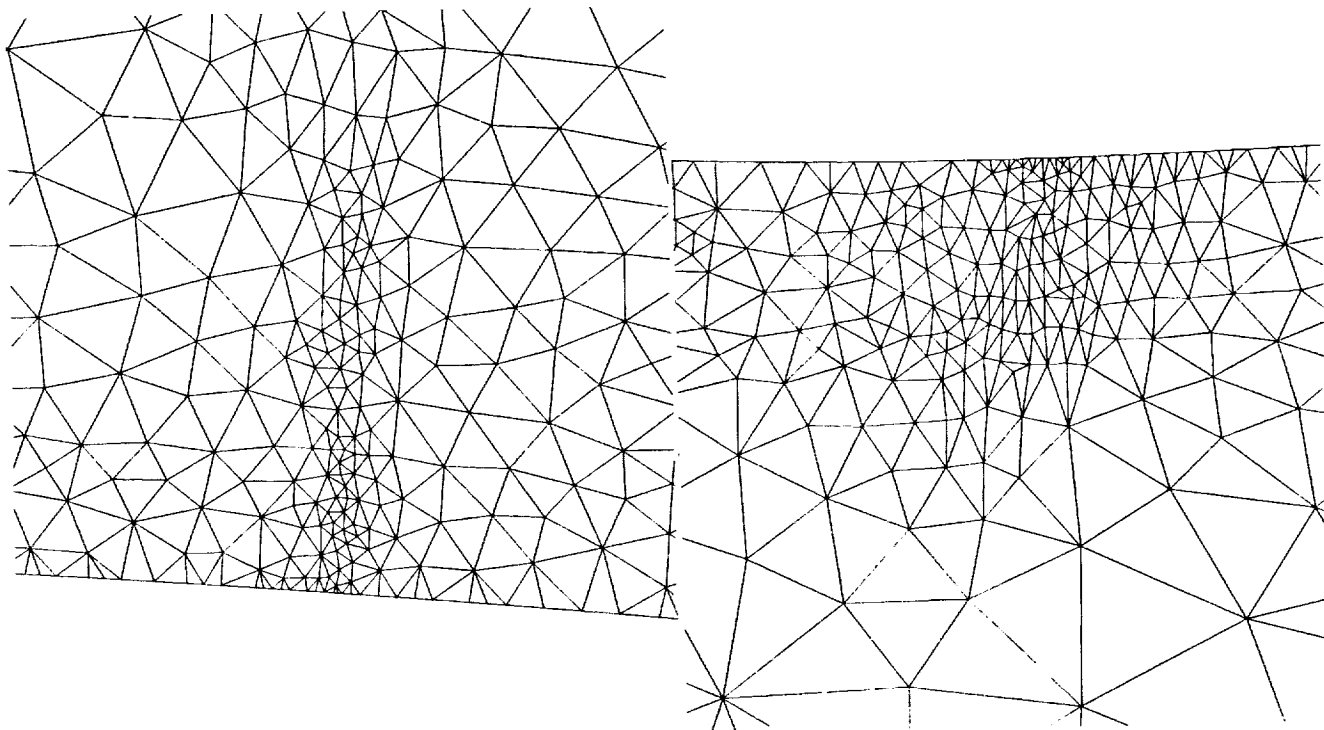


Figure 8: Exploded view of the final grid, shock regions.

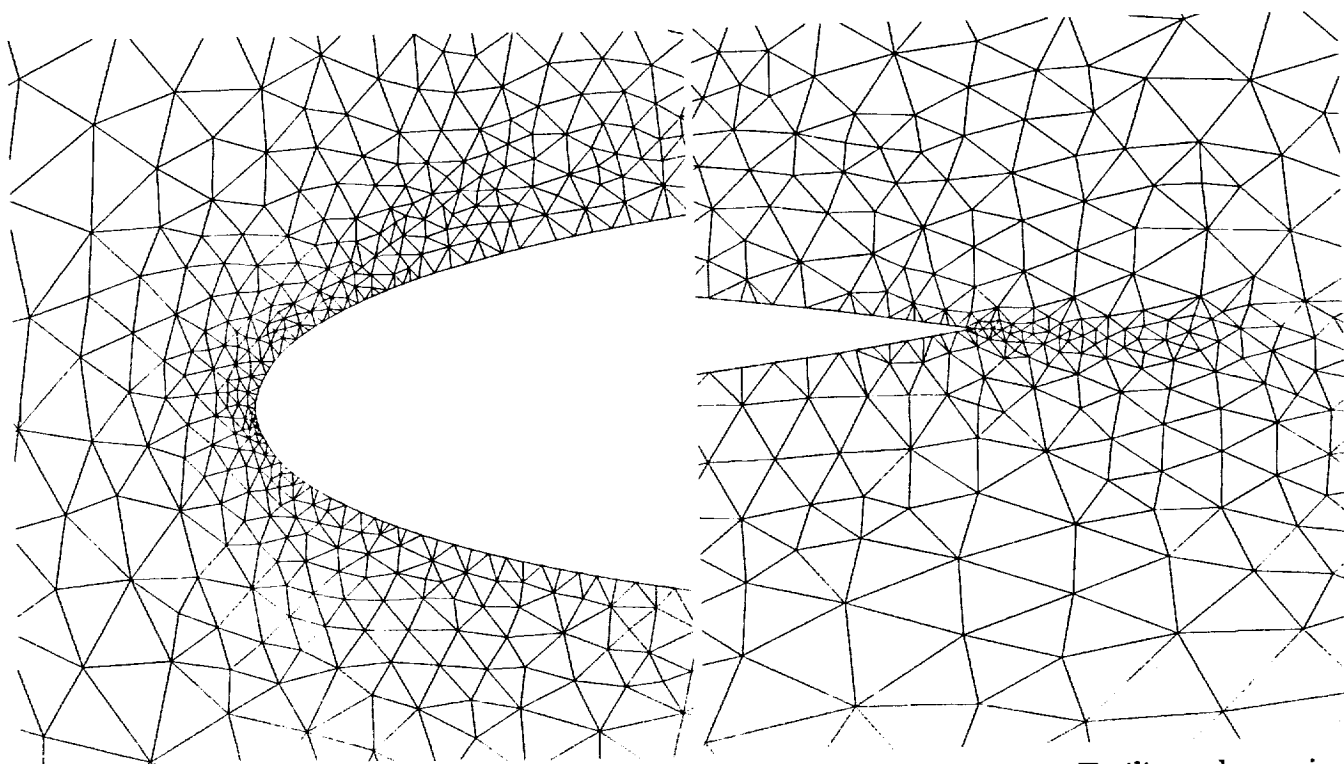


Figure 9: Exploded view of the final grid. Left : Leading edge region; Right : Trailing edge region.

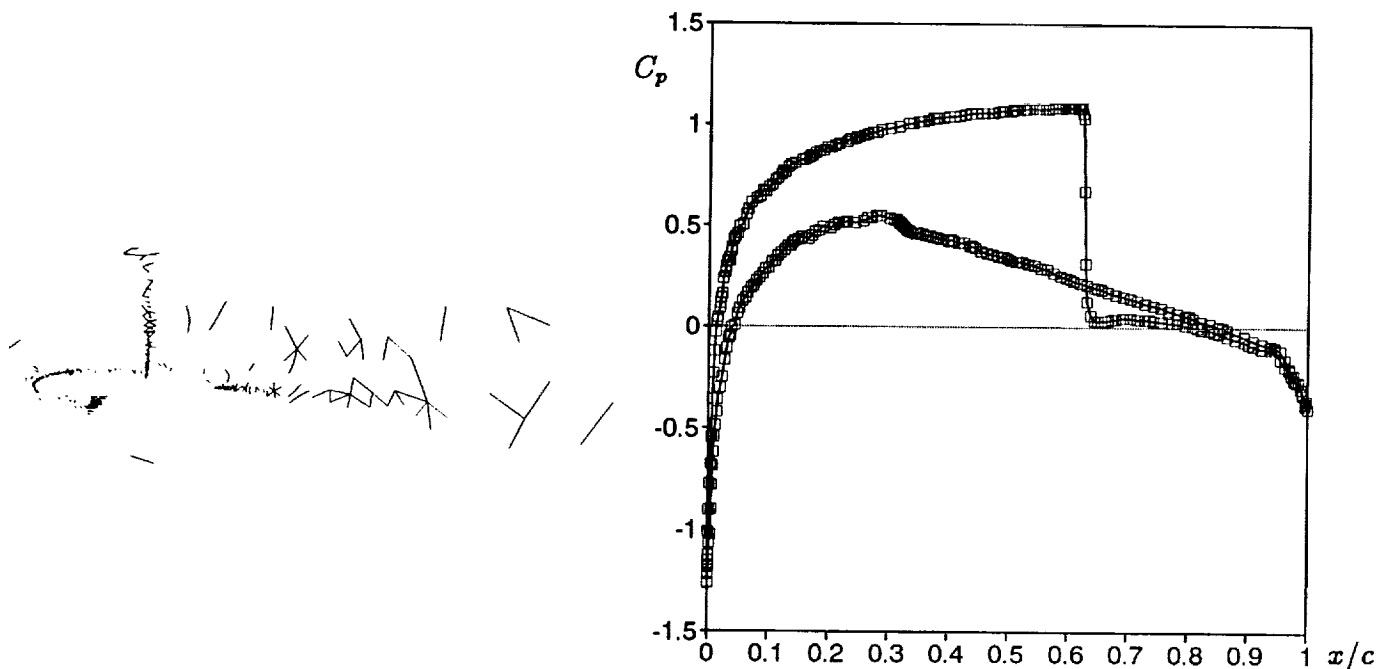


Figure 10: Left : Edges selected for refinement from third to fourth grid; Right : Pressure distribution over the airfoil.

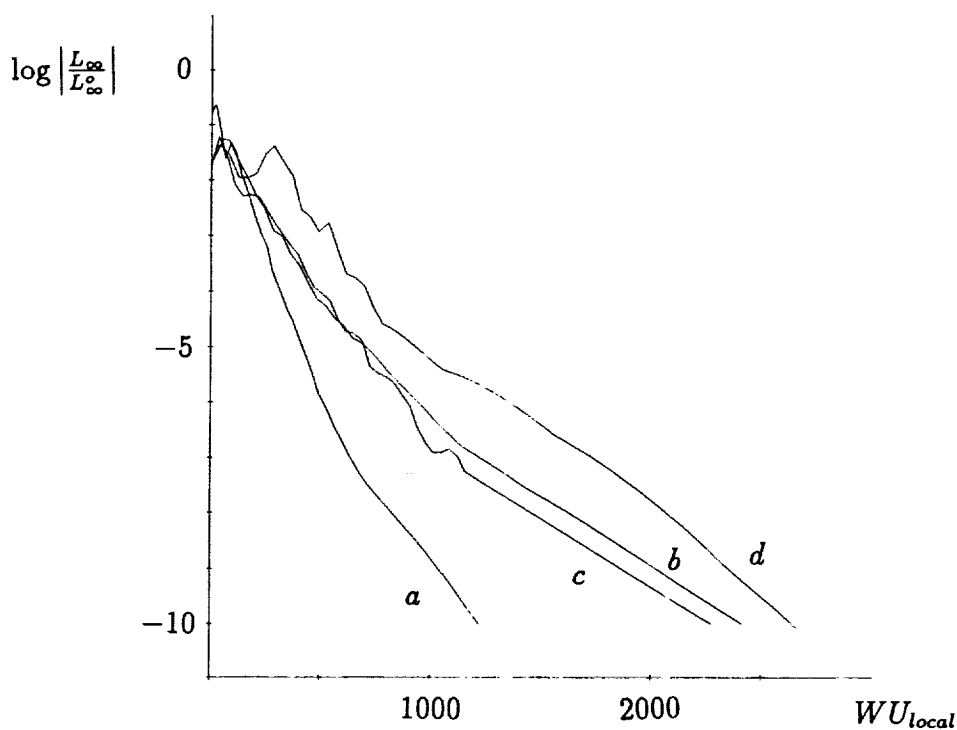


Figure 11: Convergence behaviour for the second-order defect correction scheme ( TVD minmod limiter) on the NACA-0012 test case, using three-stage Jacobi. Logarithm of the residual as function of local work units. Multigrid with fine grids 1302 (a), 2243 (b), 2834(c), and 3236 (d) nodes.

Figure 11 shows the convergence behaviour. For each multigrid phase, the work unit is defined based on the number of nodes in the finest grid during this phase. This way of representing the convergence is chosen to demonstrate the mesh independency of the convergence.

## ACKNOWLEDGEMENT

The research reported here was granted under contract IT/SC/13, as part of the Belgian National Programme for Large Scale Scientific Computing and under contract IUAP/17 as part of the Belgian National Programme on Interuniversity Poles of Attraction, both initiated by the Belgian State, Prime Minister's Office, Science Policy Programming.

## REFERENCES

- [1] Dick, E. (1988). A flux-difference splitting method for steady Euler equations. *J. Comp. Phys.*, 76, 19-32.
- [2] Dick, E. (1990). Multigrid formulation of polynomial flux-difference splitting for steady Euler equations. *J. Comp. Phys.*, 91, 161-173.
- [3] Dick, E. (1991). Multigrid methods for steady Euler and Navier-Stokes equations based on polynomial flux-difference splitting. *Multigrid Methods III*, Int. Series on Numerical Mathematics, 98, Birkhauser Verlag, Basel, 1-20.
- [4] Roe, P. L. (1981). Approximate Riemann solvers, parameter vectors and difference schemes. *J. Comp. Phys.*, 43, 357-372.
- [5] Chakravarthy, R. S., Osher, S. (1985). A new class of high accurate TVD schemes for hyperbolic conservation laws. *AIAA paper*, 85-0363.
- [6] Dick, E. (1991). Multigrid solution of steady Euler equations based on polynomial flux-difference splitting. *Int. J. Num. Methods Heat Fluid Flow*, 1, 51-62.
- [7] Morano, E., Lallemand, M.-H., Leclercq, M.-P., Steve, H., Stoufflet, B. and Dervieux, A. (1991). Local iterative upwind methods for steady compressible flows. *GMD-Studien*, 189.
- [8] Dick, E., Rienslagh, K. (1993). Multi-stage Jacobi relaxation as smoother in a multigrid method for steady Euler equations. *Proc. of ICFD Conference on Numerical Methods for Fluid Dynamics*, Reading, to appear.
- [9] Van Leer, B., Tai, C. H. and Powell, K. G. (1989). Design of optimally smoothing multi-stage schemes for the Euler equations. *AIAA-paper*, 89-1933.
- [10] Hemker, P. W. and Spekrijse, S. P. (1986). Multiple grid and Osher's scheme for the efficient solution of the steady Euler equations. *Appl. Numer. Math.*, 2, 475-493.
- [11] Tanemura, M., Ogawa, T., Ogita, N (1983). A new algorithm for three-dimensional Voronoi tessellation. *Journal of Computational Physics*, 51, 191-207.
- [12] Lawson, C. L. (1972). Generation of a triangular grid with application to contour plotting. *California Institute of Technology, Jet Propulsion Laboratory, Technical Memorandum*, 299.



TWO-LEVEL SCHWARZ METHODS FOR  
NONCONFORMING FINITE ELEMENTS  
AND DISCONTINUOUS COEFFICIENTS\*

Marcus Sarkis  
Courant Institute  
New York, NY

5/3-64  
197573  
p-23

## SUMMARY

Two-level domain decomposition methods are developed for a simple nonconforming approximation of second order elliptic problems. A bound is established for the condition number of these iterative methods, which grows only logarithmically with the number of degrees of freedom in each subregion. This bound holds for two and three dimensions and is independent of jumps in the value of the coefficients.

## INTRODUCTION

The purpose of this paper is to develop domain decomposition methods for second order elliptic partial differential equations approximated by a simple nonconforming finite element method, the nonconforming  $P_1$  elements. We consider a variant of a two-level additive Schwarz method introduced in 1987 by Dryja and Widlund [1] for a conforming case. In these methods, a preconditioner is constructed from the restriction of the given elliptic problem to overlapping subregions into which the given region has been decomposed. In addition, in order to enhance the convergence rate, the preconditioner includes a coarse mesh component of relatively modest dimension. The construction of this component is the most interesting part of the work. Here we have been able to draw on earlier multilevel studies, cf. Brenner [2], Oswald [3], as well as on recent work by Dryja, Smith, and Widlund [4]. Our main result shows that the condition number of our iterative methods is bounded by  $C(1 + \log(H/h))$ , where  $H$  and  $h$  are the mesh sizes of the global and local problems, respectively. We also note that this bound is independent of the variations of the coefficients across the subregion interfaces.

The *face based* and the Neumann-Neumann coarse spaces, that we are introducing, have the following characteristics. The nodal values are constant on each edge (or face) of the subregions and the values at the other nodes are given by a simple but nonstandard interpolation formula. Thus the value at any node in the interior of a subregion is a convex combination of three (or four) values given on the boundary, in case of triangular (or tetrahedral) substructures. We note that an

\*This work was supported by a graduate student fellowship from Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq, in part by the National Science Foundation under Grant NSF-CCR-9204255, in part by the U. S. Department of Energy under contract DE-FG02-92ER25127.

important difference between nonconforming and the conforming case is that there are no nodes at the vertices (or wire basket) of the subregions.

We note that ideas similar to ours have been used recently in other studies of domain decomposition methods for nonconforming elements; cf. Cowsar [5,6] and Cowsar, Mandel and Wheeler [7]. In particular, an isomorphism similar to ours was independently introduced by Cowsar. We point out that by using these isomorphisms, we can analyze any nonconforming version of domain decomposition methods which have already been analyzed for conforming cases. In this paper, we focus on the case where there are great variations in the coefficients across subdomains boundaries for both two and three dimensions. We define and analyze new coarse spaces and obtain condition numbers with just one log factor.

A short version of this paper was entered into Copper Mountain student competition in mid-December 1992. The present paper is a slight modification of a technical report [8].

## DIFFERENTIAL AND FINITE ELEMENT MODEL PROBLEMS

To simplify the presentation, we assume that  $\Omega$  is an open, bounded, polygonal region of diameter 1 in the plane, with boundary  $\partial\Omega$ . In a separate section, we extend all our results to the three dimensional case.

We introduce a partition of  $\Omega$  as follows. In a first step, we divide the region  $\Omega$  into nonoverlapping triangular substructures  $\Omega_i, i = 1, \dots, N$ . Adopting common assumptions in finite element theory, cf. Ciarlet [9], all substructures are assumed to be shape regular, quasi uniform and to have no dead points; i.e. each interior edge is the intersection of the boundaries of two triangular regions. We can show that the theory also holds if we choose nontriangular substructures, where the boundary of each substructure is a composition of several curved edges, and each curved edge is the intersection of two substructures. Naturally, we need assumptions related to the quasi uniformity and nondegeneracy of this partition. Initially, we restrict our exposition to the case of triangular substructures since the main ideas are seen in this case. This partition induces a coarse mesh and we introduce a mesh parameter  $H := \max\{H_1, \dots, H_N\}$  where  $H_i$  is the diameter of  $\Omega_i$ . We denote this triangulation by  $\mathcal{T}^H$ . Later, we extend the results to nontriangular substructures.

In a second step, we obtain the elements by subdividing the substructures into triangles in such a way that they are shape regular, and quasi uniform. We define a mesh parameter  $h$  as the diameter of the smallest element and denote this triangulation by  $\mathcal{T}^h$ . Similarly, we assume the triangulation  $\mathcal{T}^h$  does not have any dead points.

We study the following selfadjoint second order elliptic problem:

Find  $u \in H_0^1(\Omega)$ , such that

$$a(u, v) = f(v), \forall v \in H_0^1(\Omega), \quad (1)$$

where

$$a(u, v) = \int_{\Omega} a(x) \nabla u \cdot \nabla v \, dx \quad \text{and} \quad f(v) = \int_{\Omega} f v \, dx \quad \text{for } f \in L^2.$$

We assume that  $a(x) \geq \alpha > 0$  and that it is a piecewise constant function with jumps occurring only across the substructure boundaries. This includes cases where there is a great variation in the value of the coefficient  $a(x)$ . We remark that there is no difficulty in extending the analysis and the results to the case where  $a(x)$  does not vary greatly inside each substructure.

**Definition 1** *The nonconforming  $P_1$  element spaces (cf. Crouzeix and Raviart [10]) on the  $h$ -mesh and  $H$ -mesh is given by*

$$\begin{aligned} V^h := \{v \mid & v \text{ linear in each triangle } T \in \mathcal{T}^h, \\ & v \text{ continuous at the midpoints of the edges of } \mathcal{T}^h, \text{ and} \\ & v = 0 \text{ at the midpoints of edges of } \mathcal{T}^h \text{ that belong to } \partial\Omega\}, \end{aligned}$$

and

$$\begin{aligned} V^H := \{v \mid & v \text{ linear in each triangle } T \in \mathcal{T}^H, \\ & v \text{ continuous at the midpoints of the edges of } \mathcal{T}^H, \text{ and} \\ & v = 0 \text{ at the midpoints of edges of } \mathcal{T}^H \text{ that belong to } \partial\Omega\}. \end{aligned}$$

These spaces are nonconforming; in fact  $V^H \not\subset V^h$  and  $V^h \not\subset H_0^1(\Omega)$ .

Let  $\Sigma$  be a region contained in  $\Omega$  such that  $\partial\Sigma$  does not cut through any element. Denote by  $V_{|\Sigma}^h$  and  $\mathcal{T}_{|\Sigma}^h$  the space  $V^h$  and the triangulation  $\mathcal{T}^h$  restricted to  $\bar{\Sigma}$ , respectively.

Given  $u \in V_{|\Sigma}^h$ , we define the discrete weighted energy semi norm by:

$$|u|_{H_{a,h}^1(\Sigma)}^2 := a_{\Sigma}^h(u, u), \quad (2)$$

where

$$a_{\Sigma}^h(u, v) = \sum_{T \in \mathcal{T}_{|\Sigma}^h} \int_T a(x) \nabla u \cdot \nabla v \, dx. \quad (3)$$

In a similar fashion, we define the inner product  $a_{\Omega}^H(u, v)$  and the semi norm  $|u|_{H_{a,H}^1(\Omega)}$  for  $u, v \in V^H(\Omega)$ . In order not to use an unnecessary notation, we drop the subscript  $\Omega$  when the integration is over  $\Omega$  and the subscript  $a$  when  $a = 1$ .

The discrete problem associated with (1) is given by:

Find  $u \in V^h$ , such that

$$a^h(u, v) = f(v), \quad \forall v \in V^h(\Omega). \quad (4)$$

Note that  $|\cdot|_{H_{a,h}^1(\Omega)}$  is a norm, because if  $|u|_{H_{a,h}^1(\Omega)} = 0$ , then  $u$  is constant in each element. By the continuity at the midpoints of the edges and the zero boundary conditions, we obtain  $u = 0$ . Note also that  $f$  is a continuous linear form. Therefore, we can apply the Lax-Milgram theorem and find that there exists one and only one solution of the discrete equation (4).

We also define the weighted  $L^2$  norm by:

$$\|u\|_{L_a^2(\Sigma)}^2 := \int_{\Sigma} a(x) |u(x)|^2 \, dx \quad \text{for } u \in (V^h + V^H + L_a^2)_{|\Sigma}. \quad (5)$$

We introduce the following notation:  $x \preceq y$ ,  $f \succeq g$  and  $u \asymp v$  meaning

$$x \leq C y, \quad f \geq c g \quad \text{and} \quad c v \leq u \leq C v, \quad \text{respectively.}$$

Here  $C$  and  $c$  are positive constants independent of the variables appearing in the inequalities and the parameters related to meshes, spaces and, especially, the weight  $a(x)$ .

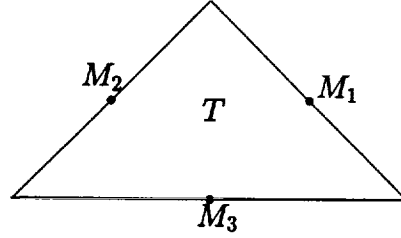


Figure 1.

Sometimes it is more convenient to evaluate a norm of a finite element function in terms of the values of this function at the nodal points. By first working on a reference element and then using the assumption that the elements are shape regular, we obtain the following lemma:

**Lemma 1** For  $u \in V_{|\Sigma}^h$ ,

$$\|u\|_{L_{a,h}^2(\Sigma)} \asymp h^2 \sum_{T \in \mathcal{T}_h|_{\Sigma}} a(T) (u^2(M_1) + u^2(M_2) + u^2(M_3)) \quad (6)$$

and

$$|u|_{W_{a,h}^1(\Sigma)} \asymp \sum_{T \in \mathcal{T}_h|_{\Sigma}} a(T) \{ (u(M_1) - u(M_2))^2 + (u(M_2) - u(M_3))^2 + (u(M_3) - u(M_1))^2 \}, \quad (7)$$

where  $M_1, M_2, M_3$  are the midpoints of the edges of the triangle  $T$  as in Figure 1.

An inverse inequality can be obtained by using only local properties. It is easy to see that for  $u \in V^h$ ,

$$|u|_{H_{a,h}^1} \preceq h^{-1} \|u\|_{L_a^2}. \quad (8)$$

## ADDITIVE SCHWARZ SCHEMES

We now describe the special additive Schwarz method introduced by Dryja and Widlund; see e.g. [11,12]. In this method, we cover  $\Omega$  by overlapping subregions obtained by extending each substructure  $\Omega_i$  to a larger region  $\Omega'_i$ . We assume that the overlap is  $\delta_i$ , where  $\delta_i$  is the distance between the boundaries  $\partial\Omega_i$  and  $\partial\Omega'_i$ , and we denote by  $\delta$  the minimum of the  $\delta_i$ . We also assume



that  $\partial\Omega'_i$  does not cut through any element. We make the same construction for the substructures that meet the boundary except that we cut off the part of  $\Omega'_i$  that is outside of  $\Omega$ .

For each  $\Omega'_i$ , a  $P_1$  nonconforming finite element subdivision is inherited from the  $h$ -mesh subdivision of  $\Omega$ . The corresponding finite element space is defined by

$$V_i^h := \{v \mid v \in V^h, \text{ support of } v \subset \Omega'_i\}, \quad i = 1, \dots, N. \quad (9)$$

The coarse space  $V_0^h \subset V^h(\Omega)$  is given as the range of  $I_H^h$  (or  $\tilde{I}_H^h$ ) where the *prolongation operator*  $I_H^h$  (or  $\tilde{I}_H^h$ ) will be defined later.

Our finite element space is represented as a sum of  $N + 1$  subspaces

$$V^h = V_0^h + V_1^h + \dots + V_N^h. \quad (10)$$

We introduce operators  $P_i : V^h \rightarrow V_i^h$ ,  $i = 0, \dots, N$ , by

$$a^h(P_i w, v) = a^h(w, v), \quad \forall v \in V_i^h, \quad (11)$$

and the operator  $P : V^h \rightarrow V^h$ , by

$$P = P_0 + P_1 + \dots + P_N. \quad (12)$$

In matrix notation,  $P_0$  is given by

$$P_0 = I_H^h (I_H^h)^T K I_H^h)^{-1} I_H^h)^T K \quad (13)$$

where  $K$  is the global stiffness matrix associated with  $a_h(\cdot, \cdot)$ .

We replace the problem (4) by

$$Pu = g, \quad g = \sum_{i=0}^N g_i \quad \text{where } g_i = P_i u. \quad (14)$$

By construction, (4) and (14) have the same solution. We point out that  $g_i$  can be computed, without knowledge of  $u$ , since we can find  $g_i$  by solving

$$a^h(g_i, v) = a^h(u, v) = f(v), \quad \forall v \in V_i^h. \quad (15)$$

The operator  $P$  is positive definite and symmetric with respect to  $a^h(\cdot, \cdot)$ . We can therefore solve (14) by a conjugate gradient method. In order to estimate the rate of convergence, we need to obtain upper and lower bounds for the spectrum of  $P$ . A lower bound is obtained by using the following lemma: cf. Zhang [13,14].

**Lemma 2** Let  $P_i$  be the operators defined in equation (11) and let  $P$  be given by (12). Then

$$a^h(P^{-1}v, v) = \min_{v = \sum v_i} \sum a^h(v_i, v_i), \quad v_i \in V_i^h. \quad (16)$$

Therefore, if a representation  $v = \sum v_i$  can be found such that

$$\sum_{i=0}^N a^h(v_i, v_i) \leq C_0^2 a^h(v, v), \quad \forall v \in V^h, \quad (17)$$

then

$$\lambda_{\min}(P) \geq C_0^{-2}.$$

An upper bound on the spectrum is obtained by bounding

$$a^h(Pv, v) = a^h(P_0v, v) + a^h(P_1v, v) + \cdots + a^h(P_Nv, v) \quad (18)$$

from above in terms of  $a^h(v, v)$ . Using Schwarz's inequality, the fact that the  $P_i$  are projections, and that the maximum number of regions that intersect at any point is uniformly bounded, it is easy to show that the spectrum of  $P$  is bounded above by

$$\max_{p \in \Omega} \{ \#(i : p \in \Omega'_i) + 1 \}.$$

## PROPERTIES OF THE $P_1$ NONCONFORMING FINITE ELEMENT SPACE

We first define two local equivalence maps in order to obtain some inequalities and local properties for our nonconforming space. Through these mappings, we can extend some results that are known for the piecewise linear conforming elements to our nonconforming case.

We use a bar to denote conforming spaces. Let  $\bar{V}^{\frac{h}{2}}|_{\bar{\Omega}_i}$  be the conforming space of piecewise linear functions in  $\bar{\Omega}_i$ , where the  $h/2$ -mesh is obtained by joining midpoints of the edges of elements of  $\mathcal{T}^h|_{\bar{\Omega}_i}$ .

We define the *local equivalence map*  $\mathcal{M}_i : V^h|_{\bar{\Omega}_i} \rightarrow \bar{V}^{\frac{h}{2}}|_{\bar{\Omega}_i}$ , as follows:

**Isomorphism 1** Given  $u \in V^h|_{\bar{\Omega}_i}$ , define  $\bar{u} = \mathcal{M}_i u$  by the values of  $\bar{u}$  at the three sets of points (cf. Figure 2.):

i) If  $P$  is a midpoint of an edge of a triangle in  $\mathcal{T}^h$ , then

$$\bar{u}(P) := u(P).$$

ii) If  $P$  is a vertex of an element in  $\mathcal{T}^h$  and belongs to the interior of  $\Omega_i$ , and the  $T_j$  are the elements that have  $P$  as a vertex, then

$$\bar{u}(P) := \text{mean of } u|_{T_j}(P).$$

Here  $u|_{T_j}(P)$ , is the limit value of  $u(x)$  when  $x \in T_j$  approaches  $P$ .

iii) If  $Q$  is a vertex of  $\mathcal{T}^h|_{\partial\Omega_i}$ , and  $Q_l$  and  $Q_r$  the two midpoints of  $\mathcal{T}^h|_{\partial\Omega_i}$  that are next neighbors of  $Q$ , then

$$\bar{u}(Q) := \frac{|Q_l Q|}{|Q_l Q_r|} u(Q_l) + \frac{|Q_r Q|}{|Q_l Q_r|} u(Q_r).$$

Here  $|Q_r Q|$  is the length of the segment  $Q_r Q$ .

Case ii) is illustrated in Figure 2., where

$$\bar{u}(P) = \frac{1}{6} \sum_{i=1}^6 u|_{T_i}(P).$$

Case iii) is required in order to have property (21), which will be very important in our analysis.

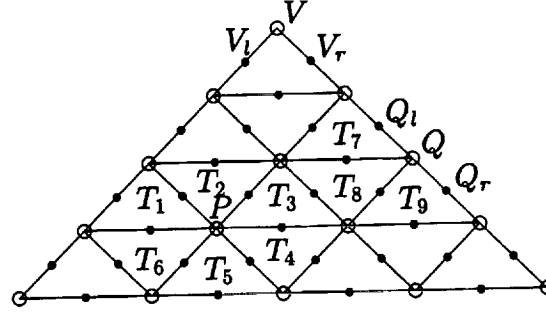


Figure 2.

**Lemma 3** Given  $u \in V^h|_{\Omega_i}$ , let  $\bar{u} \in \bar{V}^{\frac{h}{2}}|_{\Omega_i}$  given by  $\bar{u} = \mathcal{M}_i u$ . Then

$$|\bar{u}|_{H_a^1(\Omega_i)} \asymp |u|_{H_{a,h}^1(\Omega_i)}, \quad (19)$$

$$\|\bar{u}\|_{L_a^2(\Omega_i)} \asymp \|u\|_{L_a^2(\Omega_i)}, \quad (20)$$

and

$$\int_{\partial\Omega_i} \bar{u}(s) ds = \int_{\partial\Omega_i} u(s) ds. \quad (21)$$

Here  $|\cdot|_{H_a^1(\Omega_i)}$  is the standard weighted energy semi norm for conforming functions.

*Proof.* We first note that we have results similar to (6) and (7) for the conforming space  $\bar{V}^{\frac{h}{2}}|_{\Omega_i}$ , where now  $M_1, M_2$  and  $M_3$  are the vertices of a triangle in  $\mathcal{T}^{\frac{h}{2}}$ . In order to prove (19), we compare (7) with the analogous formula for the piecewise linear conforming space.

For instance (see Figure 2.),

$$|\bar{u}(Q) - \bar{u}(Q_r)|^2 = \frac{|Q_l Q|}{|Q_l Q_r|} |u(Q_l) - u(Q_r)|^2.$$

The right hand side can be controlled by the energy semi norm of  $u$  restricted to the union of the triangles  $T_7, T_8$  and  $T_9$ .

We also prove that if we take next two neighboring vertices of  $\mathcal{T}^{\frac{h}{2}}$  in the interior of  $\Omega_i$ , the energy semi norm can be bounded locally. If  $a(x)$  does not vary a great deal, we can work with weighted semi norms. Using the fact that our arguments are local, it is easy to obtain the upper bound of (19).

The lower bound is easy to obtain since the degrees of freedom of  $V^h$  are contained in those of  $\bar{V}^{\frac{h}{2}}$ .

Similar arguments can also be used to obtain (20).

Finally, it is easy to see that (21) follows directly from iii) even if the refinement is not uniform.  $\square$

We define another local equivalence map  $\mathcal{M}_i^E : V^h|_{\bar{\Omega}_i} \rightarrow \bar{V}^{\frac{h}{2}}|_{\bar{\Omega}_i}$ , by:

**Isomorphism 2** Given  $u \in V^h|_{\bar{\Omega}_i}$  and an edge  $E$  of  $\partial\Omega_i$ , define  $\bar{u} = \mathcal{M}_i^E u$  by the values of  $\bar{u}$  at the three sets of points (cf. Figure 2.):

i) Same as step i) of Isomorphism 1.

ii) Same as step ii) of Isomorphism 1.

iii) If  $V$  is a vertex  $T^h|_{\partial\Omega_i}$  and an end point of  $E$ , and  $V_r$  the midpoint of  $T^h|_E$  that is the next neighbor of  $V$ , then

$$\bar{u}(V) := u(V_r).$$

iv) If  $Q$  is a vertex of  $T^h|_{\partial\Omega_i}$  and we are not in case iii), then

$$\bar{u}(Q) := \frac{|Q_l Q|}{|Q_l Q_r|} u(Q_l) + \frac{|Q_r Q|}{|Q_l Q_r|} u(Q_r).$$

Using the same ideas as in Lemma 3, we can prove:

**Lemma 4** Given  $u \in V^h|_{\bar{\Omega}_i}$ , let  $\bar{u} \in \bar{V}^{\frac{h}{2}}|_{\bar{\Omega}_i}$  given by  $\bar{u} = \mathcal{M}_i^E u$ . Then

$$|\bar{u}|_{H_a^1(\Omega_i)} \asymp |u|_{H_{a,h}^1(\Omega_i)}, \quad (22)$$

$$\|\bar{u}\|_{L_a^2(\Omega_i)} \asymp \|u\|_{L_a^2(\Omega_i)}, \quad (23)$$

and

$$\int_E \bar{u}(s) ds = \int_E u(s) ds. \quad (24)$$

## THE INTERPOLATION OPERATOR

Let  $v \in V^h$  and let  $P_{ij}$  be the midpoint of the edge  $E_{ij}$  common to  $\bar{\Omega}_i$  and  $\bar{\Omega}_j$ .

**Definition 2** The Interpolation operator  $I_h^H : V^h \rightarrow V^H$ , is given by:

$$(I_h^H v)(P_{ij}) := \frac{1}{|E_{ij}|} \int_{E_{ij}} v|_{\bar{\Omega}_i}(x) dx = \frac{1}{|E_{ij}|} \int_{E_{ij}} v|_{\bar{\Omega}_j}(x) dx. \quad (25)$$

The second equality follows from the fact that the mean of  $v$  on each edge of an element of  $T^h$  is equal to  $v(M_1)$ , where  $M_1$  is the midpoint of the edge. It is important to note that the value of  $(I_h^H v)(P_{ij})$  depends only on the values of  $v$  on the interface  $E_{ij}$ . This allows us to obtain stability properties that are independent of the differences of  $a(x)$  across the substructure interfaces.

Before studying the stability properties of this operator, we need two lemmas for the piecewise linear conforming finite element space.

The following lemma is a Poincaré-Friedrichs inequality. The idea of the proof can be found in Ciarlet (Theorem 6.1) [9] and in Nečas (Chapter 2.7.2) [15].

**Lemma 5** Let  $\Gamma$  be a subset of  $\partial\Omega_i$ , such that  $\Gamma$  and  $\partial\Omega_i$  have measures of order  $H$ . Then,

$$\|\bar{u}\|_{L^2(\Omega_i)}^2 \preceq H^2 |\bar{u}|_{H^1(\Omega_i)}^2 + \left( \int_{\Gamma} \bar{u}(x) dx \right)^2, \quad \forall \bar{u} \in H^1(\Omega_i). \quad (26)$$

As a consequence, if  $\int_{\Gamma} \bar{u}(x) dx = 0$ , we have the Poincaré inequality

$$\|\bar{u}\|_{L^2(\Omega_i)} \preceq H |\bar{u}|_{H^1(\Omega_i)}. \quad (27)$$

The next lemma is a Poincaré-Friedrichs inequality for nonconforming  $P_1$  elements. It is obtained by using Lemmas 3, 4 and 5.

**Lemma 6** Let  $u \in H_{a,h}^1(\Omega_i)$ , where  $\Omega_i$  is a triangular substructure of diameter  $O(H)$ . Let  $\Gamma$  be  $\partial\Omega_i$  (or an edge of  $\partial\Omega_i$ ). Then,

$$\|u\|_{L^2(\Omega_i)}^2 \preceq H^2 |u|_{H_h^1(\Omega_i)}^2 + \left( \int_{\Gamma} u(x) dx \right)^2, \quad \forall u \in H_h^1(\Omega_i). \quad (28)$$

As a consequence, if  $\int_{\Gamma} u(x) dx = 0$ , we have the Poincaré inequality

$$\|u\|_{L_{a,h}^2(\Omega_i)} \preceq H |u|_{H_{a,h}^1(\Omega_i)}. \quad (29)$$

The next lemma gives an example of an operator that is  $L_a^2$ - and  $H_a^1$ -stable.

**Lemma 7** Let  $\bar{u} \in H_a^1(\Omega_i)$ , where  $\Omega_i$  is a triangular substructure of diameter of  $O(H)$ . Define a linear function  $\bar{u}_H$  in  $\Omega_i$  by

$$\bar{u}_H(P_{ij}) := \frac{1}{|E_{ij}|} \int_{E_{ij}} \bar{u}(x) dx, \quad j = 1, 2, 3, \quad (30)$$

where the  $E_{ij}$  are the edges of  $\Omega_i$ , and  $P_{ij}$  is the midpoint of  $E_{ij}$ . Then,

$$|\bar{u}_H(P_{ij})|^2 \preceq \frac{1}{H^2} \|\bar{u}\|_{L^2(\Omega_i)}^2 + |\bar{u}|_{H^1(\Omega_i)}^2, \quad (31)$$

$$|\bar{u}_H|_{H_a^1(\Omega_i)} \preceq |\bar{u}|_{H_a^1(\Omega_i)}, \quad (32)$$

and

$$\|\bar{u}_H - \bar{u}\|_{L_a^2(\Omega_i)} \preceq H |\bar{u}|_{H_a^1(\Omega_i)}. \quad (33)$$

*Proof.* Consider initially a region  $\Omega$  with a diameter of 1. Using that  $|E_{ij}| = O(1)$ , the Cauchy-Schwarz inequality and a trace theorem, we have

$$\begin{aligned} |\bar{u}_H(P_{ij})|^2 &\asymp \left| \int_{E_{ij}} \bar{u}(x) dx \right|^2 \preceq \|\bar{u}\|_{L^2(E_{ij})}^2 \\ &\preceq \|\bar{u}\|_{H^{\frac{1}{2}}(E_{ij})}^2 \preceq \|\bar{u}\|_{H^1(E_{ij})}^2 \preceq \|\bar{u}\|_{L^2(E_{ij})}^2 + |\bar{u}|_{H^1(E_{ij})}^2. \end{aligned}$$

We obtain (31) by returning to a region of diameter  $H$ .

Note that for any constant  $c$

$$\begin{aligned} |\bar{u}_H|_{H_h^1(\Omega_i)}^2 &\asymp \\ |\bar{u}_H(P_{i1}) - \bar{u}_H(P_{i2})|^2 &+ |\bar{u}_H(P_{i2}) - \bar{u}_H(P_{i3})|^2 + |\bar{u}_H(P_{i3}) - \bar{u}_H(P_{i1})|^2 \\ &\leq \|\bar{u} - c\|_{H^1(\Omega_i)}^2. \end{aligned} \quad (34)$$

By choosing  $c = \bar{u}(P_{i1})$  and  $\Gamma = E_{i1}$ , we can apply Lemma 5 and obtain the  $H^1$ -stability (32).

We now prove the  $L^2$ -stability. Since  $\bar{u} - \bar{u}_H$  has mean zero on  $\partial\Omega_i$ , we can apply the Poincaré inequality (27) and obtain

$$\|\bar{u} - \bar{u}_H\|_{L^2(\Omega_i)} \leq H |\bar{u} - \bar{u}_H|_{H^1(\Omega_i)}.$$

Using the first part of this lemma, we obtain the  $L^2$ -stability (33).  $\square$

The next lemma shows that the interpolation operator  $I_h^H$ , defined by (25), is locally  $L_a^2$ - and  $H_a^1$ -stable.

**Lemma 8** *Let  $u \in V^h(\Omega)$ . Then  $u_H = I_h^H u$  satisfies the following properties*

$$|u_H|_{H_{a,h}^1(\Omega_i)} \preceq |u|_{H_{a,h}^1(\Omega_i)}, \quad (36)$$

and

$$\|u_H - u\|_{L_a^2(\Omega_i)} \preceq H |u|_{H_{a,h}^1(\Omega_i)}, \quad i = 1, \dots, N. \quad (37)$$

*Proof.* Let  $u_H = I_h^H u$  and let  $\bar{u} \in H^1(\Omega_i)$  be given by  $\bar{u} = \mathcal{M}_i^{E_{i1}} u$  and let  $\bar{u}_H(P_{i1})$  be given by (30). Using the properties (24) and (25), we have

$$u_H(P_{i1}) = \bar{u}_H(P_{i1}). \quad (38)$$

Therefore, by (38), (31) and Lemma 4, we have

$$\begin{aligned} |u_H(P_{i1})|^2 &= |\bar{u}_H(P_{i1})|^2 \preceq \frac{1}{H^2} \|\bar{u}\|_{L^2(\Omega_i)}^2 + |\bar{u}|_{H^1(\Omega_i)}^2 \\ &\preceq \frac{1}{H^2} \|u\|_{L^2(\Omega_i)}^2 + |u|_{H^1(\Omega_i)}^2. \end{aligned} \quad (39)$$

We also obtain the same estimate for  $|u_H(P_{i2})|$  and  $|u_H(P_{i3})|$ .

The rest of the proof is similar to that of Lemma 7. We now use the Poincaré inequality for nonconforming elements.  $\square$

## THE PROLONGATION OPERATOR

In this section, we introduce several prolongation operators and establish that they are stable. The range of each of these operators will serve as a coarse space in our algorithms.

**Definition 3** The Prolongation Operator  $I_H^h : V^H \rightarrow V^h$ , is given by:

- i) For all nodal points  $P$  of  $T^h$  that belongs to an edge  $E_{ij}$  common to  $\bar{\Omega}_i$  and  $\bar{\Omega}_j$ , let  $(I_H^h u_H)(P) := u_H(P_{ij})$ , where  $P_{ij}$  is the midpoint of the edge  $E_{ij}$ .
- ii) Given  $I_H^h u_H$  at the nodal points of  $\Gamma = \cup_i \partial\Omega_i$  from i), let  $I_H^h u_H(\Omega)$  be the  $P_1$ -nonconforming harmonic extension inside each  $\Omega_i$ .

It is easy to check that  $u_h = I_H^h u_H \in V^h(\Omega)$ . A disadvantage of step ii) is that we have to solve exactly a local Dirichlet problem for each substructure in order to obtain the harmonic extension. Other extensions can be used, which we call *approximate harmonic extensions*. They are given by simple explicit formulas and have the same  $L_a^2$  and  $H_{a,h}^1$  stability properties as the harmonic one.

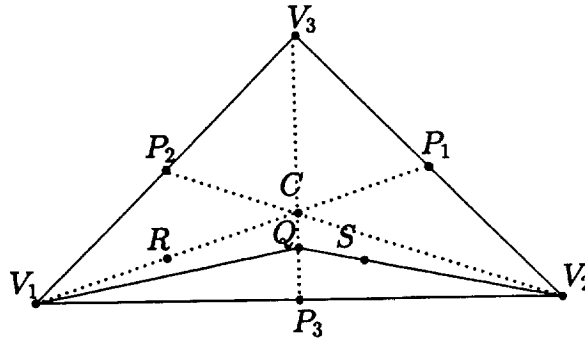


Figure 3.

Our first construction is a natural generalization of the partition of unity introduced by Dryja and Widlund in [11]; this partition of unity will provide the basis functions of our approximate extensions. Let  $P_j$ ,  $j = 1, 2, 3$ , be the midpoints of the edges of  $\Omega_i$ , and let  $V_j$  be the vertex of  $\Omega_i$  that is opposite to  $P_j$ . Let  $C$  be the barycenter of the triangle  $\Omega_i$ , i.e. the intersection of the line segment connecting  $V_j$  to  $P_j$ .

**Extension 1** The construction of an approximate harmonic extension is defined by the following steps (see Figure 3.):

- i) Let

$$\bar{u}(C) := \frac{1}{3} \{u_H(P_1) + u_H(P_2) + u_H(P_3)\}.$$

- ii) For a point  $R$  that belongs to a line segment that connects  $C$  to a vertex  $V_j$ , let

$$\bar{u}(R) := \bar{u}(C).$$

- iii) For a point  $Q$  that belongs to a line segment connecting  $C$  to  $P_j$ , define  $\bar{u}(Q)$  by linear interpolation between the values  $\bar{u}(C)$  and  $u_H(P_j)$ , i.e by

$$\bar{u}(Q) := \lambda(Q)\bar{u}(C) + (1 - \lambda(Q))u_H(P_j).$$

Here  $\lambda(Q) = \text{distance}(Q, P_j) / \text{distance}(C, P_j)$ .

iv) For a point  $S$  that belongs to the line segment connecting the previous point  $Q$  to a vertex  $V_k$ , with  $k \neq j$ , let

$$\bar{u}(S) := \bar{u}(Q).$$

v) Finally, let  $I_H^h u_H = I_h \bar{u}$ , where  $I_h$  is the interpolation operator into the space  $V^h$  that preserves the values of a function at the midpoints of the edges of the elements.

Note that the function  $\bar{u}$  just constructed is continuous except at the vertices  $V_j$  of  $\Omega_i$ . The step i) can be viewed as emulating the mean value theorem for harmonic functions. However, near the vertices,  $\bar{u}$  is a bad approximation of the harmonic extension. We know that the local behavior of the harmonic extension near a vertex  $V_j$  depends primarily on the boundary values in the vicinity of  $V_j$ . For instance, if  $u_H(P_1) = 0$ ,  $u_H(P_3) = 0$ , and  $u_H(P_2) = 1$ , we should obtain  $u_h \simeq 0$  near  $V_2$ ; in addition, by using symmetry arguments, we should have  $u_h \simeq 1/2$  for points near  $V_1$  that lie on the bisector that passes through  $V_1$ . With this in mind, we now construct an alternative approximate harmonic extension.

We change notation in order to be able to use Figure 3. Now let  $C$  be the point where the three bisectors intersect.

**Extension 2** The construction of the approximate harmonic extension is defined by (see Figure 3.):

i) Same as Step i) of Extension 1.

ii) Define  $\bar{u}(V_j) = \frac{1}{2} \sum_{l \neq j} \bar{u}(P_l)$ . For a point  $R$  that belongs to a line segment connecting  $C$  to  $V_j$ , define  $\bar{u}(R)$  by linear interpolation between the values  $\bar{u}(C)$  and  $\bar{u}(V_j)$ .

iii) Same as Step iii) of Extension 1.

iv) For a point  $S$  that belongs to a line segment connecting the previous point  $Q$  to  $V_k$ ,  $k \neq j$ ,  $\bar{u}(S)$  is defined by linear interpolation between the values  $\bar{u}(Q)$  at  $Q$  and  $f(Q, j, k)$  at  $V_k$ . Here,

$$f(Q, j, k) = \lambda(Q) \bar{u}(V_k) + (1 - \lambda(Q)) \bar{u}(P_j).$$

v) Same as Step v) of Extension 1.

A disadvantage of this extension is that we cannot just work in a reference triangle, since the angles are not preserved under a linear transformation. This is similar to the fact that under a linear transformation a harmonic function does not necessarily remain harmonic. We can construct other approximate harmonic extensions which combine the properties of the two extensions, given so far, and working, for instance, with the barycenter  $C$  as in Extension 2 and replacing the weight  $1/2$  in Step ii).

The next lemma shows that the extensions given above have quasi-optimal energy stability. Using ideas of Dryja and Widlund[11], we prove the following lemma.



**Lemma 9** Let  $u_H \in V^H(\Omega)$ . Then

$$|I_H^h u_H|_{H_{a,h}^1(\Omega_i)} \preceq (1 + \log(H/h))^{\frac{1}{2}} |u_H|_{H_{a,H}^1(\Omega_i)} \quad (40)$$

and

$$\|I_H^h u_H - u_H\|_{L_a^2(\Omega_i)} \preceq H |u_H|_{H_{a,H}^1(\Omega_i)}. \quad (41)$$

*Proof.* Let  $\theta_h^j \in V^h|_{\Omega_i}$ ,  $j = 1, 2, 3$ , be the approximate harmonic extensions constructed from the boundary values  $\theta_h^j = 1$  at the  $h$ -mesh nodes on the edge  $E_{ij}$ , and  $\theta_h^j = 0$  at the other boundary nodes of  $\partial\Omega_i$ . It is easy to see that the  $\theta_h^j$  form a basis of all approximate harmonic extensions that take constant values on the edges of the substructure. It is easy to show that if a point  $x$  belongs to the interior of an element of  $\Omega_i$ , then  $|\nabla \theta_h^j(x)|$  is bounded by  $C/r$ , where  $r$  is the minimum distance from  $x$  to any vertex of  $\Omega_i$ . Note that any element that touches a vertex of  $\Omega_i$  provides an order one contribution to the energy semi norm. To estimate the contribution to the energy semi norm from the rest of the substructure, we introduce polar coordinate systems centered at the vertices of  $\Omega_i$ . Then,

$$|\theta_h^j|_{H_h^1(\Omega_i)}^2 \preceq 1 + \int \int_h^H r^{-2} r dr d\varphi \preceq 1 + \log(H/h). \quad (42)$$

Since the partition of unity  $\theta_h^j$  forms a basis, it is easy to see that

$$|I_H^h u_H|_{H_h^1(\Omega_i)}^2 \preceq (1 + \log(H/h)) \{|u_H(P_1)|^2 + |u_H(P_2)|^2 + |u_H(P_3)|^2\} \quad (43)$$

and using ideas similar to that of Lemma 7, we have

$$\begin{aligned} |I_H^h u_H|_{H_h^1(\Omega_i)}^2 &\preceq (1 + \log(H/h)) \{|u_H(P_1) - u_H(P_2)|^2 + \\ &|u_H(P_2) - u_H(P_3)|^2 + |u_H(P_3) - u_H(P_1)|^2\} \\ &\asymp (1 + \log(H/h)) |u_H|_{H_h^1(\Omega_i)}^2. \end{aligned}$$

By construction, it is easy to see that

$$|(I_H^h u_H)(x)| \leq \max_{i=1,2,3} |u_H(P_i)|.$$

Therefore

$$\|I_H^h u_H - u_H\|_{L^2(\Omega_i)}^2 \preceq \sum_i H^2 |u_H(P_i)|^2,$$

and by using (39) and (29), we obtain (41).

Since  $a(x)$  varies little in each  $\Omega_i$ , these arguments are also valid for the weighted norms and we obtain (40).  $\square$

Using Lemmas 6 and 9 and the triangular inequality, we have:

**Theorem 1** *Let  $u \in V^h(\Omega)$ . Then*

$$\|I_H^h I_h^H u - u\|_{L^2_0(\Omega_i)} \leq H |u|_{H^1_{a,h}(\Omega_i)} \quad (44)$$

and

$$|I_H^h I_h^H u|_{H^1_{a,h}(\Omega_i)} \leq (1 + \log(H/h))^{1/2} |u|_{H^1_{a,h}(\Omega_i)}. \quad (45)$$

**Remark 1** *It is easy to see that we do not need to use the fact that  $u_H \in V_H(\Omega)$ ; we only need to calculate values  $V_H(P_{ij})$  by formula (25) at the midpoint  $P_{ij}$  of the edge  $E_{ij}$ . The next step is to provide the constant value  $V_H(P_{ij})$  to all nodes of the interface and perform an approximate harmonic extension.*

**Remark 2** *The extensions also can be constructed for nontriangular substructures. In a first step, we construct a partition of unity in  $\Omega_i$ . This can be done by using ideas similar to those of the triangular case. By using the same technique as in the proof of Lemma 9, we can show that*

$$\begin{aligned} |I_H^h u_H|_{H^1_{a,h}(\Omega_i)}^2 &\leq \\ (1 + \log(H/h)) \sum_{j=1}^{N_e^i} a(\Omega_i) |u_H(P_{ij}) - u_H(P_{i(j-1)})|^2 \end{aligned} \quad (46)$$

where  $P_{ij}$  and  $P_{i(j-1)}$  are neighboring midpoints of edges of  $\partial\Omega_i$  and  $N_e^i$  is the number of edges of  $\partial\Omega_i$ . We obtain (44) by noting that each term of the sum is bounded by  $|u|_{H^1_{a,h}(\Omega_i)}^2$ .

## THE NEUMANN-NEUMANN BASIS

In this section, we consider a Neumann-Neumann coarse space. This is the  $P_1$  nonconforming version of a coarse space studied in Dryja and Widlund [16], and Mandel and Brezina [17]. However, here we use an approximate harmonic extension inside the substructures. We note that the coarse spaces considered by these authors differ only in how certain weights are chosen. Mandel and Brezina use weights that are convex combinations of the coefficient  $a(x)$ , while Dryja and Widlund use  $a^{1/2}(x)$ . Here we show that any convex combination of  $a^\beta(x)$ , for  $\beta \geq 1/2$ , leads to stability. We point out that the choice  $\beta = 1/2$  can be viewed as a  $L^2$ -average, while  $\beta = 1$  is an average in the  $L^1$  sense.

We call the coarse space of the previous section, *face based*. There are some differences between Neumann-Neumann and face based coarse spaces. A Neumann-Neumann coarse space has one degree of freedom per substructure, while a face based uses one degree of freedom per edge. A Neumann-Neumann basis function associated with the substructure  $\Omega_i$ , has support in  $\Omega_i$  and its neighboring substructures, while a face based function basis, associated with an edge of a

substructure, has support in just two substructures. The face based coarse space appears to be more stable since all the estimates, related to the jumps of the coefficients, are tight. In the lemmas that we have proved for the face based methods, all the stability results were derived in individual substructures, while in the Neumann-Neumann case, we need to work in an extended subdomain.

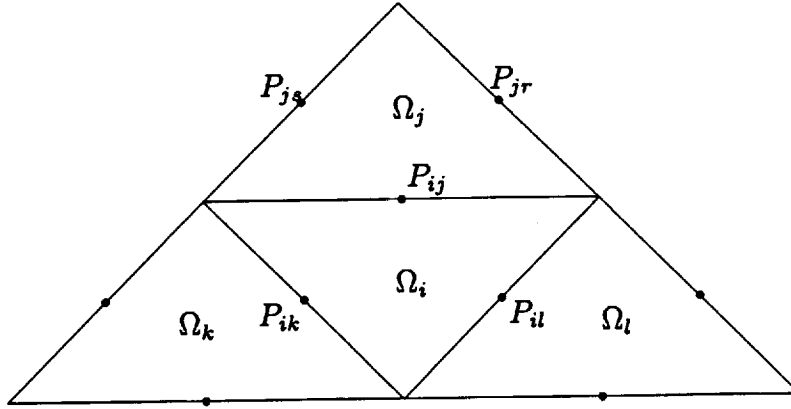


Figure 4.

**Definition 4** The Neumann-Neumann interpolation operator,  $I_{NN} : V^h \rightarrow V^h$ , as follows:

i) For each substructure  $\Omega_i$ , calculate the mean value on  $\partial\Omega_i$ , i.e.

$$m_i u := \frac{1}{|\partial\Omega_i|} \int_{\partial\Omega_i} u(s) ds.$$

Here  $|\partial\Omega_i|$  is the length size of  $\partial\Omega_i$ .

ii) For all nodal points  $P$  of  $\mathcal{T}^h$  that belong to the edge  $E_{ij}$ , let  $(I_{NN}u)(P) = (\tilde{I}_h^H u)(P_{ij})$ , where

$$(\tilde{I}_h^H u)(P_{ij}) := \frac{a^\beta(\Omega_i)}{a^\beta(\Omega_i) + a^\beta(\Omega_j)} m_i u + \frac{a^\beta(\Omega_j)}{a^\beta(\Omega_i) + a^\beta(\Omega_j)} m_j u.$$

Here  $P_{ij}$  is the midpoint of the edge  $E_{ij}$ .

iii) Perform an approximate harmonic extension to define  $I_{NN}u$  inside the substructures.

Note that we can also calculate  $m_i u$  by:

$$m_i u = \sum_j \frac{|E_{ij}|}{|\partial\Omega_i|} (I_h^H u)(P_{ij}). \quad (47)$$

Therefore, there exists a linear transformation  $I_H^H : V_H \rightarrow V_H$ , such that  $\tilde{I}_h^H u = I_H^H I_h^H u$ . The next lemma establishes stability properties for  $I_H^H$ .

**Lemma 10** Let  $u_H \in V^H(\Omega)$  and  $\beta \geq 1/2$ . Then

$$|I_H^H u_H|_{H_{a,H}^1(\Omega_i)} \leq C(\beta) |u_H|_{H_{a,H}^1(\Omega_i^{ext})}, \quad (48)$$

and

$$\|I_H^H u_H - u_H\|_{L_a^2(\Omega_i)} \leq C(\beta) H |u_H|_{H_{a,H}^1(\Omega_i^{ext})}. \quad (49)$$

Here the extended domain  $\Omega_i^{ext}$  is the union of  $\Omega_i$  and the substructures that share an edge with  $\Omega_i$ .

*Proof.* Let us first prove the  $L_a^2$  stability. Note that (see Figure 4.)

$$|u_H(P_{ij}) - (I_H^H u_H)(P_{ij})|^2 = |u_H(P_{ij}) - \frac{a^\beta(\Omega_i) m_i + a^\beta(\Omega_j) m_j}{a^\beta(\Omega_i) + a^\beta(\Omega_j)}|^2.$$

By using (47) and simple calculations, this quantity is equal to

$$\begin{aligned} & \frac{1}{|a^\beta(\Omega_i) + a^\beta(\Omega_j)|^2} * \\ & |a^\beta(\Omega_i) \left\{ \frac{|E_{ik}|}{|\partial\Omega_i|} (u_H(P_{ij}) - u_H(P_{ik})) + \frac{|E_{il}|}{|\partial\Omega_i|} (u_H(P_{ij}) - u_H(P_{il})) \right\} + \\ & a^\beta(\Omega_j) \left\{ \frac{|E_{js}|}{|\partial\Omega_j|} (u_H(P_{ij}) - u_H(P_{js})) + \frac{|E_{jr}|}{|\partial\Omega_j|} (u_H(P_{ij}) - u_H(P_{jr})) \right\}|^2. \end{aligned}$$

Using the shape regularity of the subdomains, it is easy to see that

$$a(\Omega_i) |u_H(P_{ij}) - (I_H^H u_H)(P_{ij})|^2 \preceq \quad (50)$$

$$\frac{a^{2\beta}(\Omega_i)}{|a^\beta(\Omega_i) + a^\beta(\Omega_j)|^2} |u_H|_{H_{a,H}^1(\Omega_i)}^2 + \frac{a(\Omega_i) a^{2\beta-1}(\Omega_j)}{|a^\beta(\Omega_i) + a^\beta(\Omega_j)|^2} |u_H|_{H_{a,H}^1(\Omega_j)}^2$$

and using the fact that  $\beta \geq 1/2$ , we can bound this quantity by

$$\leq C(\beta) |u_H|_{H_{a,H}^1(\Omega_i \cup \Omega_j)}^2.$$

We obtain (49) by adding all the contributions (50) to the  $L_a^2(\Omega_i)$  norm.

We prove (48) by using the triangular inequality, an inverse inequality, and (49).  $\square$

**Theorem 2** Let  $u \in V^h(\Omega)$  and  $\beta \geq 1/2$ . Then

$$\|I_{NN} u - u\|_{L_a^2(\Omega_i)} \leq C(\beta) H |u|_{H_{a,h}^1(\Omega_i^{ext})}, \quad (51)$$

and

$$|I_{NN} u|_{H_{a,h}^1(\Omega_i)} \leq C(\beta) (1 + \log(H/h))^{\frac{1}{2}} |u|_{H_{a,h}^1(\Omega_i^{ext})}. \quad (52)$$

*Proof.* Using Lemmas 9, 10 and 8, we have

$$\begin{aligned} |I_{NN}u|_{H^1_{a,h}(\Omega_i)} &\leq (1 + \log(H/h))^{\frac{1}{2}} |I_H^H I_h^H u|_{H^1_{a,H}(\Omega_i)} \leq \\ &C(\beta) (1 + \log(H/h))^{\frac{1}{2}} |I_h^H u|_{H^1_{a,H}(\Omega_i^{xt})} \leq \\ &C(\beta) (1 + \log(H/h))^{\frac{1}{2}} |u|_{H^1_{a,h}(\Omega_i^{xt})}. \end{aligned}$$

The  $L^2_a$ -stability is obtained by

$$\begin{aligned} \|I_{NN}u - u\|_{L^2_a(\Omega_i)} &\leq \|I_{NN}u - I_H^H I_h^H u\|_{L^2_a(\Omega_i)} + \\ &\|I_H^H I_h^H u - I_h^H u\|_{L^2_a(\Omega_i)} + \|I_h^H u - u\|_{L^2_a(\Omega_i)}, \end{aligned}$$

and by using Lemmas 9, 10 and 8.  $\square$

**Remark 3** We can also prove Theorem 2 for the case of nontriangular substructures; cf. Remarks 1 and 2.

## THE THREE DIMENSIONAL CASE

We show in this section that the methods developed before can be extended to three dimensions.

For simplicity, we assume that  $\Omega$  is a polyhedral region of diameter 1 in three dimensional space. As before, we introduce a nonoverlapping partition composed of tetrahedra  $\Omega_i$  of diameter of order  $H$ . This defines a coarse space and a triangulation  $\mathcal{T}^H$ . We further subdivide the substructures into tetrahedra which results in a triangulation  $\mathcal{T}^h$  and define the nonconforming  $P_1$  finite element spaces  $V^h$  and  $V^H$  as in Definition 1. Here, the continuity is enforced at the barycenter of the faces of the triangulations.

The local equivalence maps are given by the following procedure. In each tetrahedral element of  $\mathcal{T}^h$  (cf. Figure 5.), we connect its centroid to the four vertices and to the barycenters of the four faces. We also connect each barycenter to the three vertices. In other words, we subdivide each tetrahedral element into twelve subtetrahedra. We denote this new triangulation by  $\mathcal{T}^{\tilde{h}}$ . The vertices of  $\mathcal{T}^{\tilde{h}}$  are the vertices, barycenters, and centroids of the elements of  $\mathcal{T}^h$ .

Let  $\bar{V}^{\tilde{h}}|_{\tilde{\Omega}_i}$  be the conforming space of piecewise linear functions of the triangulation  $\mathcal{T}^{\tilde{h}}|_{\tilde{\Omega}_i}$ .

We define the local equivalence map  $\mathcal{M}_i : V^h|_{\tilde{\Omega}_i} \rightarrow \bar{V}^{\tilde{h}}|_{\tilde{\Omega}_i}$ , as follows:

**Isomorphism 3** Given  $u \in V^h|_{\tilde{\Omega}_i}$ , define  $\bar{u} = \mathcal{M}_i u$  by the values of  $\bar{u}$  at the following sets of points:

i) If  $P$  is a vertex of an element of  $\mathcal{T}^h$  and belongs to the interior of  $\Omega_i$ , and the  $K_j$  are the elements in  $\mathcal{T}^h|_{\Omega_i}$  that have  $P$  as a vertex, then

$$\bar{u}(P) := \text{mean of } u|_{K_j}(P).$$

Here  $u|_{K_j}(P)$  is the limit value of  $u(x)$  when  $x \in K_j$  approaches  $P$ .

ii) If  $P$  is a barycenter of a triangle in  $\mathcal{T}^h|_{\partial\Omega_i}$ , then

$$\bar{u}(P) := u(P).$$

iii) If  $P$  is a vertex of a triangle in  $\mathcal{T}^h|_{\partial\Omega_i}$  and  $T_j$ ,  $j = 1, \dots, N_P$ , are the triangles of  $\mathcal{T}^h|_{\partial\Omega_i}$  that have  $P$  as a vertex, then

$$\bar{u}(P) := \sum_{k=1}^{N_P} \frac{|T_k|}{|\cup_{j=1}^{N_P} T_j|} u(C_i).$$

Here  $C_i$  and  $|T_i|$  are the barycenter and the area of the triangle  $T_i$ , respectively.

It is easy to check that the Lemma 3 holds, if we replace  $\bar{V}^{h/2}|_{\Omega_i}$  by  $\bar{V}^h|_{\Omega_i}$ .

We define another local equivalence map  $\mathcal{M}_i^F : V^h|_{\Omega_i} \rightarrow \bar{V}^h|_{\Omega_i}$ , by:

**Isomorphism 4** Given  $u \in V^h|_{\Omega_i}$  and a face  $F$  of  $\partial\Omega_i$ , define  $\bar{u} = \mathcal{M}_i^F u$  by the values of  $\bar{u}$  at the following sets of points:

i) Same as step i) of Isomorphism 3.

ii) Same as step ii) of Isomorphism 3.

iii) Let  $P$  be a vertex of a triangle in  $\mathcal{T}^h|_{\partial\Omega_i}$  that belongs to  $\partial F$ , and let  $T_j$ ,  $j = 1, \dots, N_P^F$ , be the triangles of  $\mathcal{T}^h|_F$  that have  $P$  as a vertex. Then

$$\bar{u}(P) := \sum_{k=1}^{N_P^F} \frac{|T_k|}{|\cup_{j=1}^{N_P^F} T_j|} u(C_i).$$

iv) Let  $P$  be a vertex of a triangle in  $\mathcal{T}^h|_{\partial\Omega_i}$  that does not belong to  $\partial F$ , and let  $T_j$ ,  $j = 1, \dots, N_P$ , be the triangles of  $\mathcal{T}^h|_F$  that have  $P$  as a vertex. Then

$$\bar{u}(P) := \sum_{k=1}^{N_P} \frac{|T_k|}{|\cup_{j=1}^{N_P} T_j|} u(C_i).$$

It is easy to check that Lemma 4 holds, if we replace  $\bar{V}^{h/2}|_{\Omega_i}$  by  $\bar{V}^h|_{\Omega_i}$ , and let the faces play the role previously played by the edges.

Let  $v \in V^h$  and let  $C_{ij}$  be the barycenter of the face  $F_{ij}$  common to  $\bar{\Omega}_i$  and  $\bar{\Omega}_j$ .

**Definition 5** The interpolation operator  $I_h^H : V^h \rightarrow V^H$ , is given by:

$$(I_h^H v)(C_{ij}) := \frac{1}{|F_{ij}|} \int_{F_{ij}} v|_{\Omega_i}(x) dx = \frac{1}{|F_{ij}|} \int_{F_{ij}} v|_{\Omega_j}(x) dx,$$

where  $|F_{ij}|$  is the area of the face  $F_{ij}$ .

Using the same ideas as in two dimensions, we can prove lemmas analogous to Lemmas 5-8.

The prolongation operator  $I_H^h : V^H \rightarrow V^h$ , is defined as in the two dimensional case. In a first step, we define  $(I_H^h u_H)(P) := u_H(C_{ij})$  for all barycenters  $P$  of triangles in  $\mathcal{T}^h|_{F_{ij}}$ . Finally, we perform a  $P_1$ -nonconforming harmonic or approximate harmonic extension.

We describe the three dimensional version of Extension 1. This is a generalization of the partition of unity introduced by Dryja, Smith, and Widlund [14]. Let  $C_j$ ,  $j = 1, \dots, 4$ , be the barycenters of the faces  $F_j$  of  $\partial\Omega_i$ , and let  $V_j$  be the vertex of  $\Omega_i$  that is opposite to  $C_j$ . Let  $C$  the centroid of  $\Omega_i$ , i.e. the intersection of the line segments connecting the  $V_j$  to the  $C_j$ . Let  $E_{jk}$ ,  $k = 1, 2, 3$ , be the edges of  $\partial F_j$ .

**Extension 3** The construction of an approximate harmonic extension  $I_H^h u_H$  is defined by the following steps (see Figure 5.):

i) Let

$$\bar{u}(C) := \frac{1}{4} \sum_{j=1}^4 u_H(C_j).$$

ii) For a point  $Q$  that belongs to a line segment connecting  $C$  to  $C_j$ , define  $\bar{u}(Q)$  by linear interpolation between the values  $\bar{u}(C)$  and  $u_H(C_j)$ , i.e. by

$$\bar{u}(Q) := \lambda(Q)\bar{u}(C) + (1 - \lambda(Q))u_H(C_j).$$

Here  $\lambda(Q) = \text{distance}(Q, C_j) / \text{distance}(C, C_j)$ .

iii) For a point  $S$  that belongs to any of the three triangles defined by the previous  $Q$ , and the edges  $E_{jk}$ ,  $k = 1, \dots, 3$ , let

$$\bar{u}(S) := \bar{u}(Q).$$

iv) Finally, let  $I_H^h u_H = I_h \bar{u}$ , where  $I_h$  is the interpolation operator into the space  $V^h$  that preserves the values of a function at the barycenter of the faces of elements in  $\mathcal{T}^h$ .

We can also construct an approximate harmonic extension similar to that of Extension 2. This gives a better approximate harmonic extension near the edges.

The prolongation operator  $I_H^h$  in three dimensions has the same stability properties as in the two dimensional case, i.e. Lemma 9 still holds.

The idea of the proof is the following. Consider the case where  $u_H(\Omega_i)$  is given by  $u_H(P_{i1}) = 1$  and  $u_H(P_{i2}) = u_H(P_{i3}) = 0$ . This gives the partition of the unity introduced by Dryja, Smith, and Widlund [4]. The energy semi norm of  $u_H$  is of order  $H$ .

Let  $\theta_h^{i1} = I_H^h u_H(\Omega_i)$ . We note that  $|\nabla \theta_h^{i1}(x)|$  is bounded by  $C/r$ , where  $r$  is the distance to the nearest edge of  $\Omega_i$ . The contribution to the energy semi norm from the union of the elements with at least one vertex on the edge of the substructure can be bounded by  $CH$ , since the extension is given by a convex combination of the boundary values. To estimate the contribution to the energy from the rest of the substructure, we introduce cylindrical coordinates using the appropriate substructure edge as the  $z$ -axis. Integrating  $|\nabla \theta_h^{i1}(x)|^2$  over this region, we find that it is bounded by  $C(1 + \log(H/h))H$ .

To prove Lemma 9 for a general  $u_H$ , we use the same ideas as for two dimensions. Similarly, we can extend the results to nontriangular substructures and to the Neumann-Neumann case.

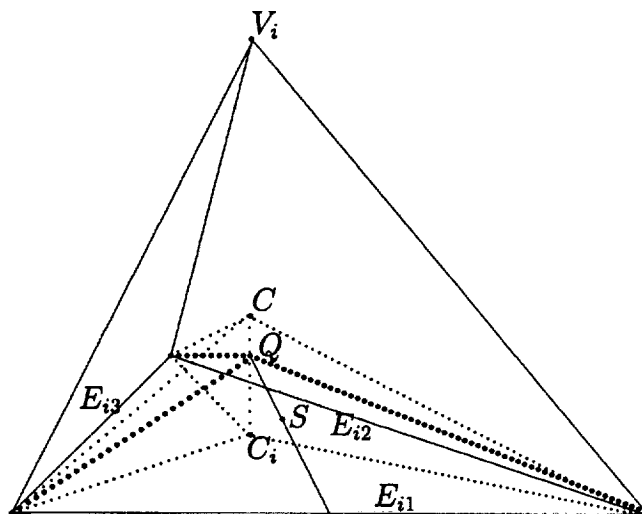


Figure 5.

## MAIN RESULT

In this section, we consider the Schwarz method introduced in the previous sections and prove the following result.

**Theorem 3** *The operator  $P$  of the additive Schwarz algorithm, defined by the spaces  $V_0^h$  and  $V_i^h$ , satisfies:*

$$\kappa(P) \leq (1 + \log(\frac{H}{h})) (1 + \frac{H}{\delta}).$$



Here  $\kappa(P)$  is the condition number of  $P$ . Therefore, if we use a generous overlapping, then

$$\kappa(P) \leq 1 + \log\left(\frac{H}{h}\right).$$

*Proof.* The proof of this theorem is essentially the same as in the case of a conforming space; see Dryja and Widlund [12].

As we have seen before, the upper bound is very easy to obtain. The lower bound is obtained by using Lemma 2. We partition the finite element function  $u \in V_h$  as follows. We first choose  $u_0 = I_H^h I_h^H u$  or  $I_{NN} u$ , i.e. apply a face based or Neumann-Neumann interpolation operator. Let  $w = u - u_0$ . The other terms in the representation of  $u$  are defined by  $u_i = I_h(\theta_i w)$ ,  $i = 1, \dots, N$ . Here  $I_h$  is the linear interpolation operator into the space  $V^h$  that preserves the values at the midpoints of the edges of the elements and  $\{\theta_i\}$  is a partition of unity with  $\theta_i \in C_0^\infty(\Omega'_i)$  and  $\sum \theta_i(x) = 1$ .

For a relatively generous overlap of the subdomains, these functions can be chosen so that  $\nabla \theta_i$  is bounded by  $C/H$ . By using the linearity of  $I_h$ , we can show that we have a correct partition of  $u$ . In order to estimate the semi norm of  $u_i$ , we work on one element  $K$  at a time. We obtain

$$|u_i|_{H_{a,h}^1(K)}^2 \leq 2 |\bar{\theta}_i w|_{H_{a,h}^1(K)}^2 + 2 |I_h((\theta_i - \bar{\theta}_i)w)|_{H_{a,h}^1(K)}^2$$

Here  $\bar{\theta}_i$  is the average value of  $\theta_i$  over  $K$ . It is easy to see, by using the inverse inequality (8), that

$$|I_h((\theta_i - \bar{\theta}_i)w)|_{H_{a,h}^1(K)}^2 \leq h^{-2} \|I_h((\theta_i - \bar{\theta}_i)w)\|_{L_a^2(K)}^2.$$

We can now use the fact that on  $K$ ,  $\theta_i$  differs from its average by at most  $Ch/H$ . After summing over all elements of  $\Omega'_i$ , we arrive at the inequality

$$|u_i|_{H_{a,h}^1(\Omega'_i)}^2 \leq |w|_{H_{a,h}^1(\Omega'_i)}^2 + H^{-2} \|w\|_{L_a^2(\Omega'_i)}^2.$$

We sum over all  $i$  and use that each point in  $\Omega$  is covered only a fixed number of times and obtain a uniform bound on  $C_0^2$ . We conclude the proof by estimating the two terms of

$$|w|_{H_{a,h}^1(\Omega)}^2 + H^{-2} \|w\|_{L_a^2(\Omega)}^2$$

by  $|u|_{H_{a,h}^1(\Omega)}$ . The bounds follow by using the stability results of Theorem 1 or 2.

For the case of small overlap, the proof is similar to that of the case of piecewise linear conforming space considered in Dryja and Widlund [12].  $\square$

**Acknowledgments.** I would like to thank my advisor, Olof Widlund, for all his friendship, guidance, help, and time he has been devoting to me. The author is also indebted to professors Peter Oswald, Jan Mandel and Max Dryja for many suggestions on this work.

## 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, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.
- [2] S.C. Brenner. An optimal-order multigrid method for P1 nonconforming finite elements. *Math. Comp.*, 53:1–15, 89.
- [3] Peter Oswald. On a hierarchical basis multilevel method with nonconforming P1 elements. *Numer. Math.*, 62:189–212, 92.
- [4] Maksymilian Dryja, Barry F. Smith, and Olof B. Widlund. Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions. Technical report, Department of Computer Science, Courant Institute, 1993. In preparation.
- [5] Lawrence C. Cowsar. Dual variable Schwarz methods for mixed finite elements. Technical Report TR93-09, Department of Mathematical Sciences, Rice University, March 1993.
- [6] Lawrence C. Cowsar. Domain decomposition methods for nonconforming finite elements spaces of lagrange-type. Technical Report TR93-11, Department of Mathematical Sciences, Rice University, March 1993.
- [7] Lawrence C. Cowsar, Jan Mandel, and Mary F. Wheeler. Balancing domain decomposition for mixed finite elements. Technical Report TR93-08, Department of Mathematical Sciences, Rice University, March 1993.
- [8] Marcus Sarkis. Two-level Schwarz methods for nonconforming finite elements and discontinuous coefficients. Technical Report 629, Department of Computer Science, Courant Institute, March 1993.
- [9] Philippe G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, 1978.
- [10] M. Crouzeix and P.A. Raviart. Conforming and non-conforming finite element methods for solving the stationary Stokes equations. *RAIRO Anal. Numer.*, 7:33–76, 73.
- [11] Maksymilian Dryja and Olof B. Widlund. Some domain decomposition algorithms for elliptic problems. In Linda Hayes and David Kincaid, editors, *Iterative Methods for Large Linear Systems*, pages 273–291, San Diego, California, 1989. Academic Press. Proceeding of the Conference on Iterative Methods for Large Linear Systems held in Austin, Texas, October 19 - 21, 1988, to celebrate the sixty-fifth birthday of David M. Young, Jr.
- [12] Maksymilian Dryja and Olof B. Widlund. Domain decomposition algorithms with small overlap. Technical Report 606, Department of Computer Science, Courant Institute, May 1992. To appear in *SIAM J. Sci. Stat. Comput.*

- [13] Xuejun Zhang. Multilevel Schwarz methods. *Numerische Mathematik*, 63(4):521–539, 1992.
- [14] Xuejun Zhang. *Studies in Domain Decomposition: Multilevel Methods and the Biharmonic Dirichlet Problem*. PhD thesis, Courant Institute, New York University, September 1991.
- [15] Jindřich Nečas. *Les méthodes directes en théorie des équations elliptiques*. Academia, Prague, 1967.
- [16] Maksymilian Dryja and Olof B. Widlund. Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems. Technical Report 626, Department of Computer Science, Courant Institute, March 1993. Submitted to Comm. Pure Appl. Math.
- [17] Jan Mandel and Marian Brezina. Balancing domain decomposition: Theory and computations in two and three dimensions. Technical report, Computational Mathematics Group, University of Colorado at Denver, 1992. Submitted to Math. Comp.



N 94-21478

# AN AUTOMATIC MULTIGRID METHOD FOR THE SOLUTION OF SPARSE LINEAR SYSTEMS

Yair Shapira

Mathematics Department, Technion -  
Israel Institute of Technology, Haifa 32000, Israel.

514-64  
197574  
p. 16

Moshe Israeli and Avram Sidi

Computer Science Department, Technion -  
Israel Institute of Technology, Haifa 32000, Israel.

## SUMMARY

An automatic version of the multigrid method for the solution of linear systems arising from the discretization of elliptic PDE's is presented. This version is based on the structure of the algebraic system solely, and does not use the original partial differential operator. Numerical experiments show that for the Poisson equation the rate of convergence of our method is equal to that of classical multigrid methods. Moreover, the method is robust in the sense that its high rate of convergence is conserved for other classes of problems: non-symmetric, hyperbolic (even with closed characteristics) and problems on non-uniform grids. No double discretization or special treatment of sub-domains (e.g. boundaries) is needed. When supplemented with a vector extrapolation method, high rates of convergence are achieved also for anisotropic and discontinuous problems and also for indefinite Helmholtz equations. A new double discretization strategy is proposed for finite and spectral element schemes and is found better than known strategies.

## 1 INTRODUCTION

The multigrid method is a powerful tool for the solution of linear systems which arise from elliptic PDE's [1] [2]. This is an iterative method, in which the equation is first relaxed on the original fine grid in order to smooth the error; then the residual equation is sent to a coarser grid to be solved there and to supply a correction term. Recursion is used to solve the coarser grid problem in the same way the original equation is handled. In order to apply this procedure, the differential operator has to be discretized on all grids, and restriction and prolongation operators have to be defined in order to pass from coarse to fine grids and vice versa. The multigrid method works well for the Poisson equation in the square, but difficulties arise with non-symmetric problems, indefinite problems and problems with discontinuous coefficients or non-uniform grids. In those cases, it is not easy to discretize the differential operator on coarse grids and to generate the restriction and prolongation operators appropriately. Some suggestions about how to handle discontinuous coefficients are given in [4] and [5], while the singularly perturbed case is discussed in [6]. Slightly indefinite problems are discussed in [7]. These approaches involve special treatment of problems according to the original PDE, and the need for a uniform approach is not yet fulfilled.

In principle, the multigrid procedure is problem-dependent, and cannot serve as a "black box" that solves every problem. Special attention has to be given to the neighborhood of the boundary and to lines of discontinuity. In [8] [9] [10] an algebraic multigrid method for symmetric problems is developed. Though this method is automatic in the sense that it depends on the linear system of equations solely, it suffers from the disadvantage of the coefficient matrices for coarse grids being of 9-diagonal type, even when the original matrix is of 5-diagonal type [11]. An algebraic version of multigrid which overcomes this difficulty is presented in [12], and generalized to nonsymmetric problems in [13]. This version, however, does not improve the classical multigrid in cases of indefinite or hyperbolic problems and of non-uniform grids.

The algorithm which is presented in this work, and which we denote Multi Block Factorization (*MBF*) (the reason for this terminology will become clear in the next section), gives a uniform approach that enables one to handle the above difficulties. It relies on the algebraic system of equations solely, and not on the original PDE. The operators for coarse grids, as well as restriction and prolongation operators, are automatically defined when the coefficient matrix is given. It seems to be more robust than the classical multigrid method, as it solves non-symmetric problems (even hyperbolic or with closed characteristics) as quickly as classical multigrid solves the Poisson equation. Moreover, it is applicable to non-uniform grids as well, and does not require any special treatment of sub-domains. For anisotropic, discontinuous or indefinite problems *MBF* by itself is not always sufficient. However, it can cope with such problems successfully when it is applied in conjunction with vector extrapolation methods. In our numerical examples in the present work we have employed the Reduced Rank Extrapolation (*RRE*) of [17] and [18]. This and other related methods have been surveyed in [19] and their analysis provided in [20], [21] and [22]. The numerical implementation that we have used is the one given in [23].

The *MBF* algorithm is described in Section 2. In Section 3 numerical results are presented. In Section 4 the algorithm and the numerical results are discussed.

## 2 DESCRIPTION OF ALGORITHMS

### 2.1 Definition of the *TBF* Method

Let  $A$  be an  $N \times N$  matrix. Let  $x$  and  $b$  be  $N$ -dimensional vectors. Consider the problem

$$Ax = b \quad (1)$$

An iteration of the Two-Block Factorization (*TBF*) method is defined by

$$\begin{aligned} TBF(x_{in}, A, b, x_{out}) : \\ \begin{aligned} x_0 &= x_{in} \\ x_{i+1} &= x_i - Z_i(Ax_i - b) \quad 0 \leq i < i_1 \\ Q\bar{e} &= R(Ax_{i_1} - b) \\ x_{i_1+1} &= x_{i_1} - P\bar{e} \\ x_{i+1} &= x_i - Z_i(Ax_i - b) \quad i_1 < i \leq i_1 + i_2 \\ x_{out} &= x_{i_1+i_2+1} \end{aligned} \end{aligned} \quad (2)$$

where the  $Z_i$  are some preconditioning operators,  $i_1$  and  $i_2$  are nonnegative integers denoting the number of presmoothings and post-smoothings respectively and  $R$ ,  $P$  and  $Q$  are operators to be defined later. Define

$$e_{in} \equiv x_{in} - x, \quad e_{out} \equiv x_{out} - x$$

Then

$$e_{out} = \left[ \prod_{i=i_1+1}^{i_1+i_2} (I - Z_i A) \right] (I - PQ^{-1}RA) \left[ \prod_{i=0}^{i_1-1} (I - Z_i A) \right] e_{in}$$

Consequently,

$$Q = RAP \Rightarrow e_{out} = 0 \Rightarrow x_{out} = x. \quad (3)$$

In the sequel, practical choices for  $Q$  will be considered. An iterative application of  $TBF$  is given by

$$\begin{aligned} & x_0 = 0, \quad i = 0 \\ & \text{while } \|\text{residual}\| \geq \varepsilon \\ & \quad TBF(x_i, A, b, x_{i+1}) \\ & \quad i \leftarrow i + 1 \\ & \text{endwhile} \end{aligned}$$

## 2.2 Definition of the MBF Method

The Multi-Block Factorization ( $MBF$ ) method is a modification of the  $TBF$  method, in which the system (2) is not solved directly, but is divided into several independent subsystems, which are solved directly or recursively by  $MBF$  itself. For simplicity, we first write the algorithm for tridiagonal systems. The operators  $P$ ,  $R$ ,  $Q$  and  $D$  will be defined later.

$$\begin{aligned} & MBF(x_{in}, A, b, x_{out}) : \\ & \quad \text{if } A \text{ is diagonal} \\ & \quad \quad x_{out} = A^{-1}b \\ & \quad \text{otherwise:} \\ & \quad \quad D \equiv \text{diag}(d_1, \dots, d_N) \\ & \quad \quad MBF \quad (0, D^{-1}Q, D^{-1}R(Ax_{in} - b), \bar{e}) \\ & \quad \quad x_{out} = x_{in} - P\bar{e}. \end{aligned}$$

We turn now to the more general definition of  $MBF$ . First we note that if there exists a subset of coordinates of  $x$  which are independent of the others, then there exists a projection  $\Pi$  onto the sub-space spanned by those coordinates such that  $(\Pi A \Pi) \Pi x = \Pi b$ . In the following definition of  $MBF$  such sub-systems are solved directly, provided this can be done easily. The co-subsystem is solved recursively.

$$MBF(x_{in}, A, b, x_{out}) :$$

1. If  $A$  is diagonal, set  $x_{out} = A^{-1}b$  and stop.
2. If  $A$  includes an independent tridiagonal subsystem  $(\Pi A \Pi) \Pi x = \Pi b$ , solve it directly:  $\Pi x_{out} = (\Pi A \Pi)^{-1} \Pi b$ . If not, set  $\Pi = 0$ .
- 3.

$$\begin{aligned} y_0 & \equiv (I - \Pi)x_{in} \\ \bar{b} & \equiv (I - \Pi)b \\ y_{i+1} & = y_i - (I - \Pi)Z_i(Ay_i - \bar{b}) \quad 0 \leq i < i_1 \\ D & \equiv \text{diag}(d_1, \dots, d_N) \\ MBF & \quad (0, D^{-1}Q, D^{-1}R(Ay_{i_1} - \bar{b}), \bar{e}) \end{aligned} \quad (4)$$

$$\begin{aligned}
y_{i_1+1} &= y_{i_1} - (I - \Pi)P\bar{e} \\
y_{i+1} &= y_i - (I - \Pi)Z_i(Ay_i - \bar{b}) \quad i_1 < i \leq i_1 + i_2 \\
(I - \Pi)x_{out} &= y_{i_1+i_2+1}.
\end{aligned}$$

Trivially, one can replace action (4) by variant *a* :

$$\begin{aligned}
MBF \quad & (0, QD^{-1}, R(Ay_{i_1} - \bar{b}), e) \\
\bar{e} &= D^{-1}e
\end{aligned}$$

or variant *b* :

$$\begin{aligned}
MBF \quad & (0, D^{-1/2}QD^{-1/2}, D^{-1/2}R(Ay_{i_1} - \bar{b}), e) \\
\bar{e} &= D^{-1/2}e.
\end{aligned}$$

An iterative application of *MBF* is given by

$$\begin{aligned}
& x_0 = 0, \quad i = 0 \\
& \text{while } \|\text{residual}\| \geq \varepsilon \\
& \quad \quad \quad MBF(x_i, A, b, x_{i+1}) \\
& \quad \quad \quad i \leftarrow i + 1 \\
& \text{endwhile}
\end{aligned}$$

### 2.3 The Tridiagonal Case

Let  $I$  denote an identity operator. Suppose  $N = 2^n$  for some positive integer  $n$ , and let  $A$  be a tridiagonal  $M$ -matrix satisfying  $\text{diag}(A) = I$ . Let  $M(N)$  be the permutation matrix which reorders the variables of  $N$ -dimensional vectors such that odd numbered variables appear in a first block and even numbered variables appear in a second block. Define

$$M_0 = M(N), \quad A_0 = A.$$

Then for some bidiagonal matrices  $B_0$  and  $C_0$  we have

$$A_0 = M_0^T \begin{pmatrix} I & B_0 \\ C_0 & I \end{pmatrix} M_0 = R_{A,1}^{-1} Q_{A,1} P_{A,1}^{-1},$$

where

$$R_{A,1} = \begin{pmatrix} I & 0 \\ -C_0 & I \end{pmatrix} M_0, \quad Q_{A,1} = \begin{pmatrix} I & 0 \\ 0 & I - C_0 B_0 \end{pmatrix}, \quad P_{A,1} = M_0^T \begin{pmatrix} I & -B_0 \\ 0 & I \end{pmatrix}.$$

Note that  $Q_{A,1}$  is the Schur complement for  $A$ .

For  $i = 1, 2, \dots$ , let  $I_i$  denote an identity operator of order  $N - 2^{n-i}$ . Let  $M_i$  be the  $N \times N$  permutation matrix that reorders the coordinates  $x_i$ ,  $i = N - 2^{n-i} + 1, \dots, N$ , of an  $N$ -dimensional vector in the above manner, that is, order odd coordinates in a first block, then even coordinates in a second block. In fact,

$$M_i = \begin{pmatrix} I_i & 0 \\ 0 & M(2^{n-i}) \end{pmatrix}.$$



For  $1 \leq i < n$ , define

$$P_{A,i+1} = M_i^T \begin{pmatrix} I_i & 0 & 0 \\ 0 & I & -B_i \\ 0 & 0 & I \end{pmatrix}, \quad R_{A,i+1} = \begin{pmatrix} I_i & 0 & 0 \\ 0 & I & 0 \\ 0 & -C_i & I \end{pmatrix} M_i,$$

$$Q_{A,i+1} = R_{A,i+1} A_i P_{A,i+1} = \begin{pmatrix} I_i & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & I - C_i B_i \end{pmatrix},$$

$$D_{A,i+1} = \text{diag}(Q_{A,i+1}),$$

$$A_{i+1} = D_{A,i+1}^{-1} Q_{A,i+1} = M_{i+1}^T \begin{pmatrix} I_{i+1} & 0 & 0 \\ 0 & I & B_{i+1} \\ 0 & C_{i+1} & I \end{pmatrix} M_{i+1}, \quad (i < n-1).$$

The last equality sign implicitly defines  $B_{i+1}$  and  $C_{i+1}$ . For variants  $a$  and  $b$ , the above definition is modified to read

$$A_{i+1} = Q_{A,i+1} D_{A,i+1}^{-1}$$

and

$$A_{i+1} = D_{A,i+1}^{-1/2} Q_{A,i+1} D_{A,i+1}^{-1/2},$$

respectively.

**Lemma 1** For all variants, the matrices  $Q_i$  and  $A_i$  are tridiagonal  $M$ -matrices.

**Proof:** The lemma follows from the definition by induction on  $i$ .  $\square$

**Lemma 2** The *TBF* method, when applied with

$$Q \equiv Q_{A,1}, \quad P \equiv P_{A,1}, \quad R \equiv R_{A,1}, \quad i_1 \equiv 0, \quad i_2 \equiv 0$$

is a direct method.

**Proof:** Since

$$Q = Q_{A,1} = R_{A,1} A_0 P_{A,1} = R A P$$

the lemma follows from equation (3).  $\square$

The even numbered variables may be viewed as coarse-grid points. Then  $Q$  is a coarse grid operator,  $R$  is a fine-to-coarse restriction and  $P$  is a coarse-to-fine prolongation.

**Lemma 3** The *MBF* method applied with the operators

$$A \equiv A_{i-1}, \quad Q \equiv Q_{A,i}, \quad D \equiv D_{A,i}, \quad P \equiv P_{A,i}, \quad R \equiv R_{A,i}$$

on the  $i^{\text{th}}$  call to the *MBF* procedure, is a direct method.

**Proof:** Note that on the  $(n+1)^{\text{st}}$  call to the *MBF* procedure, the coefficient matrix  $A_n$  is diagonal, so the *MBF* procedure is a direct solve. By induction on  $i = n, \dots, 1$ , all calls to *MBF* are equivalent to calls to *TBF*, hence are direct solves.  $\square$

In fact, in the tridiagonal case the *MBF* method is equivalent to the cyclic reduction method.

Note that if the matrices  $P$  and  $R$  are defined to be the rectangular matrices

$$P_{A,i+1} = M_i^T \begin{pmatrix} -B_i \\ I \end{pmatrix}, R_{A,i+1} = \begin{pmatrix} -C_i & I \end{pmatrix} M_i,$$

$$Q_{A,i+1} = R_{A,i+1} A_i P_{A,i+1} = I - C_i B_i,$$

$$D_{A,i+1} = \text{diag}(Q_{A,i+1}),$$

$$A_{i+1} = D_{A,i+1}^{-1} Q_{A,i+1} = M_{i+1}^T \begin{pmatrix} I & B_{i+1} \\ C_{i+1} & I \end{pmatrix} M_{i+1},$$

then the algorithm will still be applicable. As a matter of fact, the only difference between this method and the former is that in the present method we are not taking advantage of the known residuals on the odd numbered variables when making the coarse grid correction. Hence, if those residuals are zero, which may happen as a result of red-black presmoothing, the present method serves as a direct method, just as the former.

## 2.4 The Separable 2-Dimensional Case

Let  $S \equiv (s_{i,j})$  and  $T \equiv (t_{i,j})$  be matrices of order  $M$  and  $N$  respectively. Define

$$S \circ T = \begin{bmatrix} s_{1,1}T & \cdot & \cdot & \cdot & \cdot & s_{1,M}T \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & s_{i,j}T & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ s_{M,1}T & \cdot & \cdot & \cdot & \cdot & s_{M,M}T \end{bmatrix}$$

Actually,  $\circ$  denotes the tensor product. Suppose  $A$  is of the form

$$A = T \circ E_{S,0} + E_{T,0} \circ S$$

where  $T$  and  $S$  are tridiagonal scaled  $M$ -matrices and  $E_{S,0}$  and  $E_{T,0}$  are diagonal matrices. For example, if

$$T = S = \text{tridiag}(-1/2, 1, -1/2), \quad E_{T,0} = E_{S,0} = I$$

then  $A$  represents a central discretization of the Poisson equation on a square with Dirichlet boundary conditions.

Suppose  $T$  and  $S$  have the same order  $N$ , which is a power of 2. As in the previous section, we define the matrices  $T_i, S_i, R_{T,i}, R_{S,i}, P_{T,i}, P_{S,i}, Q_{T,i}, Q_{S,i}, D_{T,i}$  and  $D_{S,i}$ . For any matrix  $B = (b_{i,j})_{1 \leq i,j \leq N}$  let

$$\text{rowsum}(B) \equiv \text{diag}\left(\sum_{j=1}^N b_{i,j}\right)_{1 \leq i \leq N}.$$

For  $0 \leq i < n$ , define

$$\begin{aligned}
E_T &\equiv \text{rowsum}(R_{T,i+1})E_{T,i} \cdot \text{rowsum}(P_{T,i+1}) \\
E_S &\equiv \text{rowsum}(R_{S,i+1})E_{S,i} \cdot \text{rowsum}(P_{S,i+1}) \\
P_{A,i+1} &\equiv P_{T,i+1} \circ P_{S,i+1} \\
R_{A,i+1} &\equiv R_{T,i+1} \circ R_{S,i+1} \\
D_{A,i+1} &\equiv D_{T,i+1} \circ D_{S,i+1} \\
E_{T,i+1} &\equiv D_{T,i+1}^{-1} E_T \\
E_{S,i+1} &\equiv D_{S,i+1}^{-1} E_S \\
Q_{A,i+1} &\equiv R_{T,i+1} T_i P_{T,i+1} \circ E_S + E_T \circ R_{S,i+1} S_i P_{S,i+1} \\
A_{i+1} &\equiv D_{A,i+1}^{-1} Q_{A,i+1} \\
&= T_{i+1} \circ E_{S,i+1} + E_{T,i+1} \circ S_{i+1}.
\end{aligned}$$

In the definition of *TBF*, we take

$$Q \equiv Q_{A,1}, \quad P \equiv P_{A,1}, \quad R \equiv R_{A,1}.$$

Note that if we would eliminate the word *rowsum* in the above definition, the *TBF* method would be direct, due to equation (3). Nevertheless, for smooth vectors, multiplication by a positive matrix  $B$  is well-approximated by multiplication by  $\text{rowsum}(B)$ . Since  $T_i$  is an  $M$ -matrix,  $R_{T,i+1}$  and  $P_{T,i+1}$  are positive. Consequently, if we use presmoothing and post-smoothing, i.e.  $i_1 > 0$  and  $i_2 > 0$ , then the error is smooth, so equation (2) would give a good corrector for the current approximation. Moreover, the use of the above row-sum approximation makes the system (2) much easier to solve than the original system, since it includes 4 independent subsystems:

1. A diagonal system connecting variables which are odd in both directions, i.e., variables that correspond to odd rows of both  $S$  and  $T$  (fine grid system).
2. A tridiagonal system connecting variables which are odd in the first direction and even in the second, i.e., variables that correspond to odd rows of  $T$  and even rows of  $S$  (half coarse system).
3. A tridiagonal system connecting variables which are even in the first direction and odd in the second, i.e., variables that correspond to even rows of  $T$  and odd rows of  $S$  (half coarse system).
4. A penta-diagonal system connecting variables which are even in both directions, i.e., variables that correspond to even rows of both  $S$  and  $T$  (coarse grid system).

Only the solution of the last subsystem is expensive. The *MBF* method solves this subsystem recursively by the same procedure. On the  $i^{\text{th}}$  call to *MBF* ( $0 < i \leq n$ ) the operators used are

$$A \equiv A_{i-1}, \quad Q \equiv Q_{A,i}, \quad D \equiv D_{A,i}, \quad P \equiv P_{A,i} \quad \text{and} \quad R \equiv R_{A,i}.$$

Since  $A_n$  is diagonal, the method is well-defined. The total work of *MBF* for a problem with  $N^2$  variables is

$$w(N^2) = O(N^2) + w(N^2/4) = O(N^2) + O(N^2/4) + w(N^2/16) = \dots = O(N^2)$$

Note that if we had

$$\begin{aligned}
(\text{rowsum}(R_{T,i+1}) \cdot \text{rowsum}(P_{T,i+1})) \circ I &= I \circ D_{S,i+1}^{-1} \\
I \circ (\text{rowsum}(R_{S,i+1}) \cdot \text{rowsum}(P_{S,i+1})) &= D_{T,i+1}^{-1} \circ I
\end{aligned}$$

then

$$\begin{aligned} Q_{A,i+1} &= D_{T,i+1} T_{i+1} \circ E_S + E_T \circ D_{S,i+1} S_{i+1} \\ &= T_{i+1} \circ E_{S,i} + E_{T,i} \circ S_{i+1} \end{aligned}$$

is already in the scaled form, on which the *MBF* algorithm may act recursively. Hence the scaling by  $D^{-1}$  in action (4) of Section 2.2 is not needed. Of course, these equalities cannot hold exactly, but if they hold approximately, we can avoid scaling. Especially in the non-separable case, where scaling is impossible, action (4) has to take place without scaling by  $D^{-1}$  (instead of actual scaling, we would prefer in that case to keep diagonal matrices multiplying the difference operators in each of the space directions). Hence we would like to assume the above equalities at least for all former levels, that is, at the  $i^{\text{th}}$  call to *MBF*, for all  $0 \leq j \leq i-2$ . We call that variant “noscal”.

If we use the rectangular matrices  $P$  and  $R$  of the last part of Section 2.3, we get a variant of *MBF* which we call variant *notri*. In this variant, only the last of the four subsystems described above is solved. It assumes that the other tridiagonal subsystems affect only the smoothness of the error. The row-sum operation, however, is still performed on the original triangular matrices and not on the newly defined rectangular matrices.

Instead of the operators  $A_i$  and  $Q_{A,i}$  defined above, one may use difference operators which arise from the original PDE. If the algorithm is to be automatic, all such operators have to be of the same type (i.e. central or upwind) as the original fine-grid operator. (Nevertheless, in Section 3 we will see that for some non-symmetric problems this condition has to be violated for the sake of convergence.) We use this strategy with the rectangular matrices of variant *notri*; our version is then different from classical multigrid only in the choice of restriction and prolongation operators. Note that the row-sums computed in the *MBF* algorithm are usually 4. Instead of the multiplication by these row-sums, one may divide the residual by 4 before action (4) of Section 2.2. Then one gets an algorithm which is equivalent to that of [12] for the Poisson equation, and is close to that of [5] for other problems. We denote that strategy *MGF* (Multigrid + Factorization). It should be kept in mind that when applying this strategy one must use  $2^n - 1$  grid points on the finest grid and  $2^q - 1$ ,  $1 \leq q < n$  for coarser grids in order to conserve uniformity. Here the even points, which are taken as coarse grid points, are always internal points of the original stencil. For  $2^q$  point grids, on the other hand, the last fine grid point appears as a last grid point in all grids. Hence, coarse grids are biased towards the boundary. For our method *MBF*, on the other hand, stencils of both  $2^n$  points or  $2^n - 1$  points may be used. This is critical for implementation to problems on general regions, where grid lines may contain variable numbers of grid points (see Section 4).

As mentioned above, the *MGF* method requires division of residuals by 4 before action (4) takes place. Sometimes it is better to scale the discrete operators on all grids instead of dividing the residuals by this factor. Actually, for the Poisson equation both manners are equivalent: suppose  $A_1$  has a coarse step-size  $H = 2h$ ; then normalizing  $A_1$  to have the same diagonal entries as  $A_0 = A$  amounts to multiplication of  $A_1$  by the factor 4, which is equivalent to the division of the residual in action (4) by that factor. Nevertheless, for differential equations that include derivatives of orders other than 2, this variant is not equivalent to *MGF*. We call it *MGN* (Multigrid + Normalization).

The generalization of the *MBF* method and of the other multigrid versions to nonseparable problems is straightforward. A tensor product by an  $N \times N$  diagonal matrix is to be replaced with a multiplication by an  $N^2 \times N^2$  diagonal matrix.

Another generalization of *MBF* is to non-rectangular domains. This is also straightforward, since a line containing an odd number of points may be divided into two sets, one containing odd points and the other containing even points. Then one of those sets is considered as a coarse grid, and is divided recursively in the same way. A similar strategy may be used in the other space direction.

### 3 NUMERICAL EXPERIMENTS

In this section, the *MBF* method is compared to other multigrid versions. The problems solved are of the type

$$\begin{aligned} Lu(x, y) &= f(x, y) & (x, y) \in (0, 1)^2 \\ u(x, y) &= xy & (x, y) \in \partial[0, 1]^2 \end{aligned}$$

The equations are discretized via a 3-point central difference scheme. For *MBF*, the number of grid points in each space direction is  $N = 2^n$ . For other multigrid strategies, however, the number of grid points in each space direction is  $N - 1 = 2^n - 1$ ; otherwise, coarse grids are biased towards the boundary (see Section 2.4), and the convergence is slow.

For *MBF*, we have used variant “*notri*” of Section 2.4, in which tridiagonal subsystems are not solved. The main variant, which involves the solution of those subsystems, was found to be at most as effective as “*notri*”.

The main variant of Section 2.4 was used for the hyperbolic, the non-uniform, the strongly indefinite and the discontinuous problems (the latter when applied with a red-black smoother). For other problems we have found that variant “*noscal*” described there (in which scaling of coarse-grid operators is omitted), performs equally well. Hence we have chosen to use this simpler version rather than the main variant. Indeed, it was found that for most problems its performance was very close to that of the main variant. For the hyperbolic problem, however, its performance was twice as slow.

The smoother of the error in all grids was the one provided by the *ILU*(1,1) iteration of [24] [25] [26] or the red-black (RB) iteration. This determines the operators  $Z_i$  of Section 2.2 to be the preconditioners for the *ILU*(1,1) or *RB* iteration, respectively. These smoothers were found to be superior to the Jacobi and damped-Jacobi smoothers. One presmoothing and one post-smoothing is performed. The initial guess is random. Double precision arithmetic is used.

The integers in the following tables present the number of iterations needed to reduce the  $l_2$  norm of the residual by  $10^6$ . The maximum norm of the error was also computed, and its rate of convergence was close to that of the residual.

In conjunction with the *MBF* iteration, we have used the computer code of [23] that implements the vector acceleration *RRE* that was mentioned in the introduction. The *RRE* acceleration was employed in cycling mode, by restarting it after every 10 iterations until convergence. The results of this are compared to those provided by the *MBF* iteration without acceleration denoted by *NONE*.

We have also examined the classical multigrid versions mentioned at the end of Section 2.4. This strategy is denoted by the superscript  $D$ . The number of grid points in each space direction is  $N - 1 = 2^n - 1$ . In most of the problems, the *MGF* and *MGN* versions of Section 2.4 are equivalent. Where this is not the case, we mention explicitly which of the two versions has been employed.

For comparison, we also checked the performance of a method which does not involve any multigrid strategy. This method is the Modified *ILU* method of [29] with the optimal parameter of [30], used as a preconditioner for *RRE*. It is denoted by *MILU*.

In the following tables, when a method converges very slowly we denote it by “slow”, and when a method diverges, we denote it by “div”.

#### 3.1 The Poisson Equation

In table 1 we present the results for the Poisson equation.

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>RB</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>MILU</i>
32	4	5	5	7	4	5	5	7	19
64	4	5	5	7	4	5	5	7	27
128	4	5	5	7	4	5	5	7	42

Table 1: Results for the Poisson equation

*MBF* and the classical multigrid version perform equally well for this problem.

### 3.2 Poisson Equation on a Tchebycheff-Type Grid

In table 2 we present the results for the Poisson equation, discretized via central differences on the 2-dimensional grid

$$P(j, k) \equiv \left( \frac{1 - \cos(\frac{j\pi}{N+1})}{2}, \frac{1 - \cos(\frac{k\pi}{N+1})}{2} \right) \quad 1 \leq j, k \leq N$$

The matrix operator for this scheme may be used as a preconditioner for a Tchebycheff-collocation discretization of the Poisson equation (see [31] and the references therein).

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>RB</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>MILU</i>
32	4	5	7	10	4	5	8	11	16
64	4	6	9	19	5	6	10	18	23
128	5	7	13	33	5	6	14	28	36

Table 2: The Poisson equation with non-uniform grid

The superscript <sup>D</sup> refers to the *MGF* method of the end of Section 2.4, which is in the spirit of Dendy [12]. It performs equally well as *MBF*.

### 3.3 An Anisotropic Discontinuous Equation

In table 3 we present the results for an anisotropic equation whose coefficients are discontinuous;

$$a(x)u_{xx} + a(y)u_{yy} = 0$$

Here  $a(t)$  is defined by

$$a(t) = \begin{cases} 0.01 & 0 < t \leq 0.5 \\ 1 & 0.5 < t \leq 1 \end{cases}$$

*MBF* and *MGF* perform equally well for this problem. For both methods,  $N - 1$  grid points were used in each space direction. If  $N = 2^n$  grid points are used in any of the space directions, then for coarse-grid problems the discontinuity lines are biased towards the boundary, and convergence becomes slow.

Results similar to those of table 3 were obtained for the continuous anisotropic problem

$$u_{xx} + 0.01u_{yy} = 0$$

This time, however, there was no difference in convergence rate when the number of grid points in each space direction was changed from  $N = 2^n$  to  $N - 1 = 2^n - 1$ , for both *MBF* and *MGF*.

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>RB</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>MILU</i>
32	5	6	19	slow	5	6	19	slow	23
64	7	10	26	slow	7	10	26	slow	32
128	9	13	28	slow	9	13	28	slow	48

Table 3: An anisotropic discontinuous equation

### 3.4 A Convection-Diffusion Equation with Circular Streamlines

In table 4 we present the results for the convection-diffusion equation

$$u_{xx} + u_{yy} + 150((y - 0.5)u_x - (x - 0.5)u_y) = f$$

whose characteristics are circles:

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>RB</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>MILU</i>
32	7	10	12	15	8	slow	div	div	28
64	6	10	9	12	8	slow	div	div	51
128	6	10	9	12	8	slow	div	div	94

Table 4: A convection-diffusion equation with circular streamlines

Problems of the last type are widely discussed in [6]. The approach developed there requires special treatments and is not as automatic as ours.

### 3.5 A Convection-Diffusion Equation with Radial Streamlines

In table 5 we present the results for the convection-diffusion equation

$$u_{xx} + u_{yy} + 150(xu_x + yu_y) = f$$

whose characteristics are radial lines:

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>RB</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>RB<sup>D</sup></i>	<i>MILU</i>
32	7	9	slow	div	7	9	div	div	28
64	5	6	13	12	7	8	15	15	51
128	5	6	9	10	8	9	12	15	94

Table 5: A convection-diffusion equation with radial streamlines

The <sup>D</sup> superscript denotes here the *MGF* method of the end of Section 2.4. Nevertheless, the purely automatic *MGF* version, in which all difference operators are central, diverged. To avoid that we had to use upwind difference schemes for all grids coarser than the original grid. This strategy, however, though performing almost equally well as *MBF*, suffers the disadvantage of not being automatic. Another way to overcome divergence is to use the *MGN* method of the end of Section 2.4, with the same step-size  $h$  for all grids. This strategy is non-automatic as well, and about twice as slow as the first one.

### 3.6 A Convection Equation

In table 6 we present the results for the convection equation

$$(y - 0.5)u_x - (x - 0.5)u_y = f$$

whose characteristics are circles, discretized via an upwind scheme:

	<i>RRE</i>	<i>NONE</i>	<i>RRE</i>	<i>NONE</i>
<i>N</i>	<i>ILU</i>	<i>ILU</i>	<i>ILU<sup>D</sup></i>	<i>ILU<sup>D</sup></i>
32	10	27	10	slow
64	15	49	14	slow
128	23	89	23	slow

Table 6: A convection equation with circular streamlines

The superscript <sup>*D*</sup> refers to the *MGF* method of the end of Section 2.4. Its performance is equal to that of *MBF*.

The standard *ILU* and the Modified *ILU* of [29] (on the fine grid only, without multigrid strategy) do not converge for this problem. All multigrid strategies with an *RB* smoother are very slow.

### 3.7 The Helmholtz Equation

In table 7 we present the results for the Helmholtz equation

$$u_{xx} + u_{yy} + \beta u = f$$

with  $\beta = 64$ . The *RRE* method for *MBF* was restarted in this example after every 5 iterations.

	<i>RRE</i>	<i>RRE</i>	<i>RRE</i>	<i>RRE</i>
<i>N</i>	<i>ILU</i>	<i>RB</i>	<i>ILU<sup>D</sup></i>	<i>RB<sup>D</sup></i>
32	9	14	17	16
64	9	13	17	19
128	8	15	18	20

Table 7: The Helmholtz equation

Without acceleration, all methods diverged. The *RRE* acceleration for *ILU* and *MILU* iterations (on the fine grid only, without multigrid strategy) also diverged.

The superscript <sup>*D*</sup> denotes here the *MGN* version, used with a continuation strategy; that is, use a parameter  $\beta$  smaller than that of the original PDE for grids coarser than the original grid, in such a way that the number  $h^2\beta$  is constant for all grids. Without this continuation strategy, divergence was reported. Consequently, it suffers the disadvantage of not being automatic.

This problem is of the type of problems discussed in [7]. The projection approach given there requires more work and special treatment.

For  $\beta > 64$ , the *RRE* acceleration for *MBF* seems to suffer stability problems, as the residual no longer decreases monotonically. A machine with higher precision (or Kacmarz smoother as in Section 3.8) is required. With classical multigrid, on the other hand, the acceleration is more stable: with the *ILU* smoother, it converges for  $\beta = 100$  and  $N = 64$  in 42 iterations.

Note that the Helmholtz equation and the convection-diffusion equations are better-posed as the number of grid points increase; hence the number of iterations generally decrease.



### 3.8 Helmholtz Equation with Mixed Boundary Conditions

The above experiments involve Dirichlet boundary conditions. In this sub-section we examine the Helmholtz equation with Dirichlet boundary conditions on the edges  $x = 0$ ,  $x = 1$  and  $y = 1$ , and with the mixed boundary conditions

$$\frac{\partial u}{\partial n} + \alpha u = g \quad (5)$$

on the edge  $y = 0$ .

We have repeated experiments 6.1 and 6.3 of [32], for which  $\beta = 100$ ,  $\alpha = 100i$ ,  $N = 31$  and  $\beta = 200$ ,  $\alpha = 10i$ ,  $N = 63$  respectively. On coarse grids, where the problem is very indefinite, we have used Kacmarz relaxations as a smoother. The cost of an *MBF* or *MGF* iteration was about 10 Jacobi iterations of the original problem. With *MBF*, we have converged in 10 iterations for the first problem and in 18 for the second one, which is much better than the results of [32]. With *MGF*, applied with the continuation strategy described in Section 3.7, we have converged in 23 iterations for the first problem.

### 3.9 Helmholtz Equation with Finite Elements

Finally, we have examined the Helmholtz equation

$$u_{xx} + u_{yy} + \beta u = f$$

with  $\beta = 200$  and Dirichlet boundary conditions, discretized via bilinear finite elements. The grid for those elements is not uniform; in each space direction, the domain is divided into 4 elements, and the grid points are the roots of the Legendre polynomial of degree 17 in each element. Hence the total number of grid points is  $63^2$ . This grid induces a division of the domain into squares, which serve as the bilinear finite elements. The matrix operator for this bilinear element scheme may be used as a preconditioner for a spectral element discretization of the Helmholtz equation (see [31] and the references therein). Though the coefficient matrix has nine non-zero diagonals, the operators for coarser grids have five non-zero diagonals only; they are obtained from the above finite difference approximation in the automatic or classical manners. Actually, this is a double discretization strategy. The relaxations on the finest grid are the *ILU* iteration or the four-color Gauss-Seidel iteration. On the second grid, the relaxation is *ILU* or *RB* iteration. One presmoothing and one post-smoothing are performed on those two grids. On coarser grids, since the operators are more indefinite, these relaxation methods are too divergent; hence, we use instead the Kacmarz iteration, 40 presmoothings and 40 post-smoothings on each level. Since on the third grid the number of points is  $1/16$  of that of the original grid, the total work on that grid is about five Jacobi relaxations of the original system. The cost of the whole multigrid or *MBF* procedure is about 10 such relaxations. *RRE* acceleration is restarted after every 10 multigrid or *MBF* iterations. The number of *MBF* iterations needed to reduce the residual by 6 orders of magnitude is 28 when *ILU* is used on the two finest grids and 27 when the Gauss Seidel smoother is used there. For *MGF* (with *ILU* on the two finest grids and with the continuation strategy of Section 3.7), the number of iterations needed is 52. When the residual is reduced by 6 orders of magnitude, the error is reduced by 6 orders for *MBF* and 5 orders for *MGF*.

We also examined the mixed boundary conditions case. For the mixed boundary conditions (5) on the edges  $x = 0$  and  $y = 0$ , with  $\beta = 200$ ,  $\alpha = 10i$  and  $N = 64$ , *MBF* converged in 52 iterations, each costs about as much as 7 Jacobi iterations, with *RRE* restarted after every 20 iterations. Classical *MG* methods did not converge for this problem, even with the continuation strategy of Section 3.7.

### 3.10 Helmholtz Equation with Spectral Elements

The above double discretization strategy is not limited to bilinear element schemes; it was employed successfully for the spectral element scheme of [33] as well. Again, we relax the original equation on the finest grid; then we compute the difference scheme based on the nodes of the spectral elements, and use it to generate the coarse grid operators via *MBF*. These operators are used to find the coarse grid correction. For the Helmholtz equation with the mixed boundary conditions (5) on the edges  $x = 0$  and  $y = 0$ , with the parameters  $\beta = 100$ ,  $\alpha = 100i$  and  $N = 16$  and with  $4 \times 4$  Legendre-type spectral elements, *MBF* converged in 16 iterations, each costs about as much as 5 Jacobi iterations (with *RRE* restarted after 10 iterations). For the same problem but with the parameters  $\alpha = 200$ ,  $\beta = 10i$  and  $N = 64$ , *MBF* converged in 60 iterations; each costs about as much as 2 Jacobi iterations. This rate of convergence was much better than that of the algorithm of [31], in which the spectral element scheme is preconditioned by a finite element or a finite difference scheme (even when the preconditioning system is solved by *MBF*). As a matter of fact, even for the Poisson equation our strategy was about 3 times faster than that of [31].

## 4 DISCUSSION

The *MBF* method is a version of multigrid, which is automatic in the sense that it depends on the algebraic system of equations rather than on the original PDE. Actually, it is a "black box" method for the solution of the linear system of equations. Hence it seems to be more robust than the classical multigrid method. For instance, nonsymmetric terms in the equation do not slow down the convergence, whether the characteristics are closed or open. Non-uniform grids are handled with the same efficiency, and no special treatment of the neighborhood of the boundary is needed. Moreover, when the *RRE* acceleration is applied to the method, it copes with the indefinite Helmholtz equation as well. For all these examples the rate of convergence is rather independent of the size of the problem. For anisotropic or pure advection problems, however, the rate of convergence of the *MBF* method applied with the *RRE* acceleration slightly depends on the size of the problem.

The *MBF* method is especially suited to use with the *ILU* smoother. The red-black smoother gives slightly worse results. The doubled damped Jacobi iteration as a smoother (with a damping parameter 0.5) was examined too. For all the above problems but the discontinuous-anisotropic and the hyperbolic problems, its performance was about twice slower than that of the *ILU* smoother. For those two problems, the damped Jacobi smoother was unsatisfactory.

The versions of multigrid denoted *MGF* and *MGN* perform well for problems which do not involve central first derivatives (including discontinuous and anisotropic problems). For problems which do contain central first derivatives, since the algorithm is assumed automatic, the discretization on coarse grids is of the same type as that of the fine grid, i.e. central. Hence divergence is often caused by the coarse-grids corrections. This difficulty can be handled by the special treatments of Section 3.5, but then the algorithm is no longer automatic. For the Helmholtz equation, one may overcome this difficulty by using a continuation strategy in the *MGN* version. Even though this (non-automatic) strategy is a bit slower than *MBF*, it is more stable and is applicable to more singular problems. If one uses Kacmarz relaxation on coarse grids, both *MBF* and *MGF* converge even for very indefinite problems, the *MBF* again being faster.

As opposed to the classical multigrid versions, the *MBF* is applicable whether the number of grid points in each space direction is even or odd. This indicates that it is applicable to problems defined on general regions. Given a region  $\Omega \subset \mathbb{R}^2$ , one takes as a fine grid the restriction of an infinite 2-dimensional fine grid to  $\Omega$ . For a coarser grid, one takes every other point (in both  $x$  and  $y$  space directions) in the infinite fine grid, and takes the restriction to  $\Omega$ . The other coarse grids are created in the same way. As we have seen, *MBF* is not affected by the possibility that some coarse grid points lie near  $\partial\Omega$ . The coarse grid operators are created automatically as in the above description; this can be done easily by modifying the block sizes in the coefficient matrix of the system. Thus the algorithm is easy to program.

## References

- [1] Brandt, A.: Guide to Multigrid Development, Multigrid Methods. Hackbusch, W.; and Trottenberg, U., eds.: *Lecture Notes in Mathematics* 960, Springer-Verlag, Berlin, Heidelberg 1982.
- [2] Hackbusch, W.: Multigrid Methods and Applications. Springer-Verlag, Berlin, Heidelberg, 1985.
- [3] Hackbusch, W.: The Frequency decomposition Multigrid Algorithm. In Hackbusch, W., ed.: *Robust Multigrid Methods*. Proceedings of the fourth GAMM seminar. Kiel, January 1988.
- [4] Alcouffe, R.; Brandt, A.; Dendy, J. E.; and Painter J.: The Multigrid Method for the Diffusion Equation with Strongly Discontinuous Coefficients. *SIAM J. Sci. Stat. Comp.*, vol. 2, 1981, pp. 430-454.
- [5] Kettler, R.: Analysis and Comparison of Relaxation Schemes in Robust Multigrid and Preconditioned Conjugate Gradients Methods. In Hackbusch, W.; and Trottenberg, U., eds.: *Multigrid Methods*, eds., *Lecture Notes in Mathematics* 960, Springer-Verlag, Berlin, Heidelberg 1982.
- [6] Yavneh, I.: Doctoral Thesis, Weizmann Institute of Science, Rehovot, Israel.
- [7] Brandt, A.; and Ta'asan, S.: Multigrid methods for nearly singular and slightly indefinite problems. in *Lecture Notes in Mathematics* 960, Springer-Verlag, Oct. 1985, pp. 100-122.
- [8] Brandt, A.: Algebraic Multigrid Theory: The Symmetric Case. *Appl. Math. Comp.*, vol. 19, 1986, pp. 23-56.
- [9] Brandt, A.; McCormick, S. F.; and Ruge, J.: Algebraic Multigrid (AMG) for Automatic Multigrid Solution with Application to Geodetic Computation. *SIAM J. Sci. Stat. Comp.*
- [10] Ruge, J. W.; and Stuben, K.: Algebraic Multigrid. In McCormick, S.F., ed.: *Multigrid Methods*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1987.
- [11] McCormick, S. F.: An Algebraic Interpretation of Multigrid Methods. *SIAM. J. Num. Anal.*, vol. 19, 1982, pp. 548-560.
- [12] Dendy, J. E.: Black Box Multigrid. *J. Comp. Phys.*, vol. 48, 1982, pp. 366-386.
- [13] Dendy, J. E.: Black Box Multigrid for Nonsymmetric Problems. *Appl. Math. Comp.*, vol. 13, 1983, pp. 261-283.
- [14] Dendy, J. E.: Black Box Multigrid for Periodic and Singular Problems. Los Alamos National Laboratories, LA-UR-86-3897.
- [15] Dendy, J. E.: A Multigrid Method for 3-D Problems with Discontinuous Coefficients. Los Alamos National Laboratories, LA-UR-86-1165.
- [16] Dendy, J. E.: Black Box Multigrid for Systems. *Appl. Math. Comp.*, vol. 19, 1986.
- [17] Eddy, R. P.: Extrapolating to the limit of a vector sequence. in Wang, P. C. C. ed: *Information Linkage Between Applied Mathematics and Industry*. Academic Press, N. Y., 1979, pp. 387-396.
- [18] Mesina, M.: Convergence acceleration for the iterative solution of the equations  $X = AX + f$ . *Comp. Meth. Appl. Mech. Eng.*, vol. 10, 1977, pp. 165-173.
- [19] Smith, D. A.; Ford, W. F.; Sidi, A.: Extrapolation methods for Vector Sequences. *SIAM Rev.*, vol. 29, 1987, pp. 199-233.

- [20] Sidi, A.: Convergence and Stability Properties of MPE and RRE algorithms. *SIAM J. Num. Anal.*, vol. 23, 1986, pp. 197-206.
- [21] Sidi, A.; Bridger, J.: Convergence and Stability Analysis for Some Vector Extrapolation Methods in the Presence of Defective Iteration Matrices. *J. Comp. Appl. Math.*, vol. 22, 1988, pp. 35-61.
- [22] Sidi, A.: Extrapolation vs. Projection Methods for Linear Systems of Equations. *J. Comp. Appl. Math.*, vol. 22, 1988, pp. 71-88.
- [23] Sidi, A.: Efficient Implementation of Minimal Polynomial and Reduced Rank Extrapolation Methods. *J. Comp. Appl. Math.*, vol. 36, 1991, pp. 305-337.
- [24] Meijerink, J. A.; Van der Vorst H. A.: An Iterative Solution Method for Linear Systems of Which the Coefficients Matrix is a Symmetric M-Matrix. *Math. Comp.*, vol. 31, 1977, pp. 148-162.
- [25] Meijerink, J. A.; and Van der Vorst, H. A.: Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they Occur in Practical Problems. *J. Comp. Phys.*, vol. 44, 1981, pp. 134-155.
- [26] Van der Vorst, H. A.: Iterative Solution Methods for Certain Sparse Linear Systems with a Non-symmetric Matrix Arising from PDE-problems. *J. Comp. Phys.*, vol. 49, 1982, pp. 1-19.
- [27] Van der Vorst, H. A.: Preconditioned Tchebycheff iterative solution method for large sparse linear systems with a non-symmetric matrix. *Lecture Notes in Mathematics* 968, pp. 323-333.
- [28] Van der Vorst, H. A.: High Performance Preconditioning. *SIAM J. Sci. Stat. Comp.*, vol. 10, 1989, pp. 1175-1184.
- [29] Gustafsson, S.: On modified incomplete factorization methods. *Lecture Notes in Mathematics*, 968, pp. 334-351.
- [30] Chan, T. F.: Fourier Analysis of Relaxed Incomplete Factorization Preconditioners. *SIAM J. Sci. Stat. Comp.*, vol. 12, 1991, pp. 668-680.
- [31] Quarteroni, A.; and Zampieri, E.: Finite Element Preconditioning for Legendre Spectral Collocation Approximations to Elliptic Equation and Systems. *SIAM J. Num. Anal.*, vol. 29, 1992, pp. 917-936.
- [32] Freund, R. W.: Conjugate gradients type methods for linear systems with complex symmetric coefficient matrices. *SIAM J. Sci. Stat. Comp.*, vol. 13, 1992, pp. 425-448.
- [33] Patera, A. T.: A Spectral Element Method for Fluid Dynamics; Laminar Flow in a Channel Expansion. *J. Comp. Phys.*, vol. 54, no. 3, June 1984.
- [34] Bastian, P.; and Horton, G.: Parallelization of Robust Multigrid Methods: ILU Factorization and Frequency Decomposition Method. *SIAM J. Sci. Stat. Com.*, vol. 12, 1991, pp. 1457-1470.
- [35] Varga, R.: *Matrix Iterative Analysis*. Prentice-Hall, N. J., 1962.

515-6479  
N 94-2479  
197575  
P-10

# A MULTIGRID ALGORITHM FOR THE CELL-CENTERED FINITE DIFFERENCE SCHEME\*

Richard E. Ewing and Jian Shen  
Institute for Scientific Computation  
Texas A&M University  
College Station, Texas

## SUMMARY

In this article, we discuss a non-variational  $V$ -cycle multigrid algorithm based on the cell-centered finite difference scheme for solving a second-order elliptic problem with discontinuous coefficients. Due to the poor approximation property of piecewise constant spaces and the non-variational nature of our scheme, one step of symmetric linear smoothing in our  $V$ -cycle multigrid scheme may fail to be a contraction. Again, because of the simple structure of the piecewise constant spaces, prolongation and restriction are trivial; we save significant computation time with very promising computational results.

## INTRODUCTION

In the simulation of incompressible fluid flow in porous media, we have to solve at least one second-order elliptic equation per each time step. A very important quantity is the Darcy velocity, defined by

$$\mathbf{u} = -\mathcal{K}\nabla p \quad (1)$$

where  $p$  is the pressure of the fluid and  $\mathcal{K}$  is the conductivity.  $\mathcal{K}$  can be written by  $\mathcal{K} = \frac{k}{\mu}$ , where  $k$  is a tensor representing the permeability of the medium which can be discontinuous in general, and  $\mu$  represents the viscosity of the fluid.  $\mu$  is a continuous function of both time and space variables, but may have a very sharp frontal change of values. In other words,  $\mu$  can change rapidly inside the interesting domain and the region of rapid change may move as time changes. According to the conservation law of mass balance, the Darcy velocity  $\mathbf{u}$  must be continuous along the normal direction at an element or domain boundary, no matter whether  $\mathcal{K}$  is discontinuous or not.

Now, we consider the following simple second-order elliptic equation in mixed form. Find a pair  $(p, \mathbf{u})$  such that

$$\begin{aligned} \mathbf{u} &= -\mathcal{K}\nabla p, & \text{in } \Omega = (0, 1)^2 \subset \mathbb{R}^2, \\ \nabla \cdot \mathbf{u} &= f, & \text{in } \Omega, \\ p &= 0, & \text{on } \partial\Omega, \end{aligned} \quad (2)$$

---

\*This work was supported in part by the Department of Energy under Contract No. DE-FG05-92ER25143. The authors would also like to thank Joe Pasciak for valuable discussions about this work.

where the conductivity  $\mathcal{K}(x, y) = \text{diag}(a, b)$  is positive and uniformly bounded above and below.

Because of the discontinuity of  $\mathcal{K}$ , the classical solution of  $p$  in (2) may not exist. Let  $(\cdot, \cdot)$  denote  $L^2(\Omega)$  or  $(L^2(\Omega))^2$  inner product and  $H(\text{div}; \Omega) \equiv \{\mathbf{u} \in (L^2(\Omega))^2 \mid \nabla \cdot \mathbf{u} \in L^2(\Omega)\}$ . We seek the solution pair  $(p, \mathbf{u}) \in H^1(\Omega) \times H(\text{div}; \Omega)$ , such that

$$\begin{aligned} (\mathcal{K}^{-1}\mathbf{u}, \mathbf{v}) &= (p, \nabla \mathbf{v}), \quad \forall \mathbf{v} \in H(\text{div}, \Omega), \\ (\nabla \cdot \mathbf{u}, w) &= (f, w), \quad w \in L^2(\Omega). \end{aligned} \quad (3)$$

In [5], error estimates for solving (3) by the cell-centered finite difference scheme are studied, with the following results:

$$\|P - \mathcal{P}p\|_{L^2} + \|\mathbf{U} - \pi\mathbf{u}\|_{L^2} \leq ch^s \|p\|_{1+s, \Omega \setminus \Gamma}, \quad s = 1, 2, \quad (4)$$

where  $\mathcal{P} \times \pi$  is the Raviart-Thomas projection,  $\Gamma$  are the lines of discontinuity which coincide with the grid lines, and  $(P, \mathbf{U})$  is the numerical solution of the cell-centered finite difference to approximate (3)[5]. Actually, we view the cell-centered finite difference method as a special numerical integration of the Raviart-Thomas mixed finite element method [4–6]. For  $s = 2$ , (4) is the superconvergence error estimate.

From the point of view of mass balance and accuracy, the cell-centered finite difference scheme is one of the best numerical schemes to fulfill our goal. In this article, we investigate the efficiency of the multigrid algorithm based on the cell-centered finite difference scheme introduced in [5].

## NUMERICAL SCHEME IN MULTIGRID SETTING

Let us use the Laplacian operator,  $-\Delta$ , to explain the cell-centered finite difference scheme stencil. For an interior node, the stencil for  $-\Delta$  is (a) in Figure 1. For a corner node, the stencil for  $-\Delta$  is (b) in Figure 1. For other boundary nodes, the stencil for  $-\Delta$  is (c) in Figure 1. For discontinuous conductivity, see [5] for details. Now, we consider the uniform grid only. Let  $\mathcal{M}_k$  denote the piecewise constant Raviart-Thomas rectangular pressure space defined on  $\Omega$  with mesh size  $h_k = 2^{-(k+1)}$ ,  $k = 0, 1, 2, 3, \dots, J$ . It is clear that

$$\mathcal{M}_0 \subset \mathcal{M}_1 \subset \mathcal{M}_2 \subset \dots \subset \mathcal{M}_{J-1} \subset \mathcal{M}_J \subset L^2(\Omega). \quad (5)$$

With an abuse of notation, for  $u \in \mathcal{M}_k$ ,  $u$  is either a piecewise function or a vector with its nodal values as its entries. On  $\mathcal{M}_k$ , the cell-centered finite difference approximation is to find  $P \in \mathcal{M}_k$ , such that

$$\tilde{A}_k P = F_k \equiv \mathcal{P}_k f, \quad k = 0, 1, 2, \dots, J. \quad (6)$$

Here  $\mathcal{P}_k : L^2 \rightarrow \mathcal{M}_k$  is the  $L^2$ -projection into  $\mathcal{M}_k$  defined by

$$(f, w) = (\mathcal{P}_k f, w), \quad \forall w \in \mathcal{M}_k, \quad (7)$$

and  $f$  is the load function of (2). The corresponding stencil of  $\tilde{A}_k$  is shown in Figure 1. Our goal is to find  $P \in \mathcal{M}_J$ , such that

$$\tilde{A}_J P = F_J = \mathcal{P}_J f. \quad (8)$$

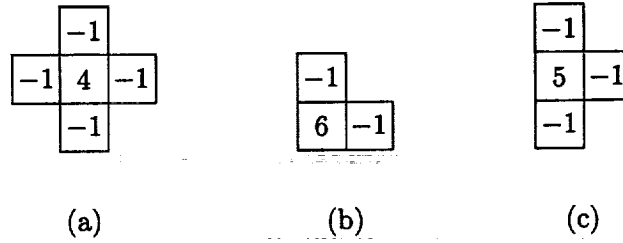


FIGURE 1. Stencils for the Laplacian operator.

The discrete  $L^2$ -inner product and associated norm on  $\mathcal{M}_k$  are denoted by

$$(u, v)_k = h_k^2 v^T u \quad \text{and} \quad \|u\|_k^2 = (u, u)_k, \quad u, v \in \mathcal{M}_k, \quad (9)$$

where  $v^T u$  is the usual algebraic inner product. Let  $A_J = \tilde{A}_J$  define an associated bilinear form  $A_J$  on  $\mathcal{M}_J$  by

$$(A_J w, \phi)_J = A_J(w, \phi), \quad \forall w, \phi \in \mathcal{M}_J. \quad (10)$$

Before we define  $A_k$  for  $0 \leq k < J$ , we first define the prolongation operator  $I_k$  and the restriction operator  $P_{k-1}^0$ . Let  $I_k : \mathcal{M}_{k-1} \rightarrow \mathcal{M}_k$ ,  $k = 1, 2, \dots, J$  be the natural imbedding from  $\mathcal{M}_{k-1}$  to  $\mathcal{M}_k$ . Thus  $P_{k-1}^0 : \mathcal{M}_k \rightarrow \mathcal{M}_{k-1}$ , the adjoint of  $I_k$  in  $(\cdot, \cdot)_k$ , is defined by

$$(P_{k-1}^0 w, \phi)_{k-1} = (w, I_k \phi)_k, \quad w \in \mathcal{M}_k, \phi \in \mathcal{M}_{k-1}. \quad (11)$$

From (9) and  $h_{k-1} = 2h_k$ , it is clear that  $P_{k-1}^0 = \frac{1}{4} I_k^T$  in matrix form. Now, we define the bilinear form  $A_{k-1}(\cdot, \cdot)$  and the matrix  $A_{k-1}$  on  $\mathcal{M}_{k-1}$  for  $k = J, J-1, \dots, 2, 1$ , by

$$2A_{k-1}(u, v) = A_k(I_k u, I_k v), \quad \forall u, v \in \mathcal{M}_{k-1}, \quad (12)$$

and the corresponding matrix relation is

$$A_{k-1} = \frac{1}{8} I_k^T A_k I_k = \frac{1}{2} P_{k-1}^0 A_k I_k. \quad (12')$$

*Remark 1.* It is shown in [5] that for piecewise smooth conductivity tensor  $\mathcal{K}$ , as long as the discontinuities coincide with the coarser grid lines

$$A_{k-1} = (1 + O(h_k^2)) \tilde{A}_{k-1}. \quad (13)$$

In (13)  $O(h_k^2) = Ch_k^2$ .  $C$  depends on the local smoothness of  $\mathcal{K}$  but is independent of the jumps. Since  $I_k$  is a simple operator, it is much easier to generate  $A_{k-1}$  by (12') than by (6) directly. Of course,  $A_k$ ,  $k = 0, 1, 2, \dots, J-1$ , are all positive definite since  $A_J$  is, and the spaces are nested. Because of (12), our multigrid algorithm can be considered as a black box solver once  $I_k$  has been defined. We mention that (12) holds for three-dimensional problems of  $-\nabla \cdot (\mathcal{K} \nabla u)$ , with (12') being changed to

$$A_{k-1} = \frac{1}{16} I_k^T A_k I_k = \frac{1}{2} P_{k-1}^0 A_k I_k.$$

We also define the adjoint of  $I_k$  in  $A_k(\cdot, \cdot)$ ,  $P_{k-1} : \mathcal{M}_k \rightarrow \mathcal{M}_{k-1}$  by

$$A_{k-1}(P_{k-1}u, v) = A_k(u, I_k v), \quad u \in \mathcal{M}_k, \quad v \in \mathcal{M}_{k-1}. \quad (14)$$

To define the smoothing process, we require linear operators  $R_k : \mathcal{M}_k \rightarrow \mathcal{M}_k$  for  $k = 1, 2, \dots, J$ . These operators may or may not be symmetric with respect to the inner product  $(\cdot, \cdot)_k$ . Let  $A_k = D_k + L_k + L_k^T$ ,  $D_k$  be the diagonal part of  $A_k$ , and  $L_k$  be the lower triangular part of  $A_k$ . The linear smoothers we have tried are the following relaxation schemes. For  $0 < \omega < 2$ ,

$$\begin{aligned} \text{(a) Gauss-Seidel: } R_k &= \left( \frac{D_k}{\omega} + L_k \right)^{-1} \text{ and } R_k^T, \\ \text{(b) Jacobi: } R_k &= \omega D_k^{-1}, \\ \text{(c) Richardson: } R_k &= \frac{\omega}{\lambda_k} I, \end{aligned} \quad (15)$$

where  $I$  is the identity operator on  $\mathcal{M}_k$  and  $\lambda_k$  is the spectral radius of  $A_k$ . We allow the relaxation parameter  $\omega$  to be different for pre-smoothing and post-smoothing processes in the following definition.

Following [1] the multigrid operator  $B_k : \mathcal{M}_k \rightarrow \mathcal{M}_k$  is defined by induction and is given as follows. The pre-smoother is denoted by  $R_k$  and the post-smoother by  $\bar{R}_k$ .

*V-Cycle Multigrid Algorithm:*

Set  $B_0 = A_0^{-1}$ . Assume that  $B_{k-1}$  has been defined and define  $B_k g$  for  $g \in \mathcal{M}_k$  as follows:

1. Set  $x^0 = 0$ .
2. Define  $x^\ell$  for  $\ell = 1, 2, \dots, m(k)$  by  $x^\ell = x^{\ell-1} + R_k(g - A_k x^{\ell-1})$ .
3. Set  $y^0 = x^{m(k)} + I_k B_{k-1} P_{k-1}^0 (g - A_k x^{m(k)})$ .
4. Define  $y^\ell$  for  $\ell = 1, 2, \dots, m(k)$  by  $y^\ell = y^{\ell-1} + \bar{R}_k(g - A_k y^{\ell-1})$ .
5. Set  $B_k g = y^{m(k)}$ .

*Remark 2.* Since equation (12) holds for all levels, this multigrid algorithm is non-variational according to [1], but the approximation property (4) is valid for each level as long as the non-variational relation (12) is satisfied. In this algorithm  $m(k)$  is a positive integer which may vary from level to level. In general this multigrid algorithm is not symmetric in  $(\cdot, \cdot)_k$  except for  $\bar{R}_k = R_k^T$ .

Setting  $K_k = I - R_k A_k$  and  $\bar{K}_k = I - \bar{R}_k A_k$ , it is straightforward to check that

$$\begin{aligned} I - B_k A_k &= \bar{K}_k^{m_2(k)} [I - I_k B_{k-1} P_{k-1}^0 A_k] K_k^{m_1(k)} \\ &= \bar{K}_k^{m_2(k)} [I - I_k B_{k-1} A_{k-1} P_{k-1}] K_k^{m_1(k)}. \end{aligned} \quad (16)$$



Equation (16) gives a fundamental recurrence relation for the multigrid operator  $B_k$ .

## COMPUTATIONAL EXPERIMENTS

We have tested the multigrid algorithm described in Section 2. We use a power method to compute the largest and the smallest eigenvalues of  $B_J A_J$ .

The linear smoothers we have tried are the following. Let  $m$  be a positive integer.  $m(k) = m$  for all  $k$ ,

$$S_1(m) : R_k = \frac{1}{\lambda_k} I, \quad \bar{R}_k = R_k,$$

$$S_2(m) : R_k = \frac{1}{\lambda_k} I, \quad \bar{R}_k = 2R_k, \quad \text{where } \lambda_k \text{ is the largest eigenvalue of } A_k,$$

$$S_3(m) : R_k = (D_k + L_k)^{-1}, \quad \bar{R}_k = R_k^T,$$

$$S_4(m) : R_k = 1.35 D_k^{-1}, \quad \bar{R}_k = \frac{1}{2} R_k,$$

$$S_5(m) : R_k = \left( \frac{D_k}{1.35} + L_k \right)^{-1}, \quad \bar{R}_k = \left( \frac{D_k}{0.675} + L_k^T \right)^{-1},$$

$$S_6(m) : R_k = \left( \frac{2}{3} D_k + L_k \right)^{-1}, \quad \bar{R}_k = (2D_k + L_k^T)^{-1}.$$

Note that only  $S_1(m)$  and  $S_3(m)$  make  $B_J A_J A_J(\cdot, \cdot)$  symmetric. The rest are neither symmetric nor  $A_J(\cdot, \cdot)$  symmetric. We also have tried nonlinear smoothers, conjugate gradient, and diagonally preconditioned conjugate gradient algorithms. We shall use  $N(m)$  to represent our nonlinear multigrid by diagonally preconditioned conjugate gradient smoothers. The reason we choose different relaxation numbers comes from the suggestion [3] for an algebraic multigrid algorithm, and from our computational experiments.

We list our test results in Tables 1–9 at the end of this paper for the following problems:

Ex. 1. Poisson problem:  $\mathcal{K} \equiv 1$  in (2).

Ex. 2. Isotropic problem with nearly singular piecewise smooth conductivity:

$$\mathcal{K} = \left[ 0.001 + 11.1 \left( 1 + \cos(3.561\pi x) \sin(3.001\pi y) \right) \right] q,$$

$$q = \begin{cases} 10^{-4}, & \text{if } x \geq \frac{1}{2} \text{ and } y \geq \frac{1}{2}, \\ 1, & \text{otherwise.} \end{cases}$$

Ex. 3. Same kind of problems as Ex. 2:

$$\mathcal{K} = \left[ 0.001 + 45.1 \left( 1 + \cos(9.431\pi x) \sin(3.001\pi y) \right) \right] q,$$

$$q = \begin{cases} 10^4, & \text{if } x \geq \frac{1}{2} \text{ and } y \geq \frac{1}{2}, \\ 1, & \text{otherwise.} \end{cases}$$

Ex. 4. Anisotropic problem with smooth conductivity:

$$\mathcal{K} = \text{diag}(a, b),$$

$$a = 0.001 + 45.1 \left( 1 + \cos(9.431\pi x) \sin(9.431\pi y) \right),$$

$$b = 0.001 + 45.1 \left( 1 + \sin(9.431\pi x) \cos(9.431\pi y) \right).$$

Note that all the solutions of our examples have the superconvergence results proved in [5], i.e., satisfying (4) with  $s = 2$ .

In Tables 1 and 6, for example, the second row of Table 1 means  $J + 1 = 3$  level multigrid with  $h_J = \frac{1}{8}$ ,  $\lambda_m, S_1(1)$  means  $\lambda_m = \min \lambda(B_J A_J)$  by  $S_1(1)$  smoothers, and  $\lambda_M, S_1(1)$  means  $\lambda_M = \max \lambda(B_J A_J)$  by  $S_1(1)$  smoothers. From Table 1, we can see that even when  $I - B_J A_J$  fails to be a reducer,  $B_J$  may still be a good preconditioner. In Tables 5–7, it is interesting to see the relations of the number of  $V$ -cycles ( $\#V$ ), average contraction numbers (avc) and the time spent on the machine (cpu in seconds) when solving a fixed problem on a fixed grid by using different multilevels. In Tables 3–5, and 7–9, avc is defined by

$$\text{avc} = \frac{1}{n} \sum_{j=1}^n \frac{\|r_j\|_J^2}{\|r_{j-1}\|_J^2},$$

where  $n = \#V$  is the total number of  $V$ -cycles and  $\|r_j\|_J$  is the discrete  $L^2$ -norm of the residual after the  $j$ th  $V$ -cycle. The stop tolerance for all the iterative algorithms is  $\|r_n\|_J^2 \leq \epsilon = 10^{-14}$ . Our coarsest grid solver is a diagonal preconditioned conjugate gradient solver with tolerance  $\epsilon_0 = 10^{-19}$ . In Tables 7–9, “cg” means the standard conjugate gradient algorithm, its corresponding “ $\#V$ ” means the total iteration steps, when  $\|r_n\|_J^2 \leq \epsilon = 10^{-14}$ , and “bpcg” means the incomplete factorization preconditioned conjugate gradient algorithm [2].

## REFERENCES

1. Bramble, J. H., Pasciak, J. E., and Xu, J.: The Analysis of Multigrid Algorithm with Nonnested Spaces or Noninherited Quadratic Forms. *Math. Comp.*, vol. 56, 1991, pp. 1–34.
2. Concus, P., Golub, G. H., and Meurant, G.: Block Preconditioning for the Conjugate Gradient Method. *SIAM J. Sci. Stat. Comput.*, vol. 6, 1985, pp. 220–252.
3. McCormick, S. F., ed.: *Multigrid Methods*. SIAM, Philadelphia, Pennsylvania, 1987.

4. Russell, T.F. and Wheeler, M.F.: Finite Element and Finite Difference Methods for Continuous Flows in Porous Media. *The Mathematics of Reservoir Simulation* (R. E. Ewing, ed.). SIAM, Philadelphia, Pennsylvania, 1983.
5. Shen, J.: *Mixed Finite Element Methods: Analysis and Computational Aspects*. Ph.D. Thesis, University of Wyoming, 1992.
6. Weiser, A. and Wheeler, M.F.: On the Convergence of Block-Centered Finite Differences for Elliptic Problems. *SIAM J. Numer. Anal.*, vol. 25, 1988, pp. 351–375.

Table 1. For Ex. 1

GRID	$J$	$\lambda_m, S_1(1)$	$\lambda_M, S_1(1)$	$\lambda_m, S_1(2)$	$\lambda_M, S_1(2)$
$4^2$	1	0.548	1.351	0.788	1.134
$8^2$	2	0.446	1.804	0.704	1.297
$16^2$	3	0.397	2.394	0.663	1.470
$32^2$	4	0.367	3.128	0.639	1.633
$64^2$	5	0.345	4.023	0.623	1.783
$128^2$	6	0.325	5.106	0.609	1.924
$256^2$	7	0.299	6.417	0.592	2.059

Table 2. For Ex. 1

GRID	$J$	$\lambda_m, S_3(1)$	$\lambda_M, S_3(1)$	$\lambda_m, S_3(2)$	$\lambda_M, S_3(2)$
$4^2$	1	0.858	1.142	0.971	1.037
$8^2$	2	0.812	1.239	0.960	1.062
$16^2$	3	0.794	1.344	0.954	1.089
$32^2$	4	0.785	1.445	0.951	1.112
$64^2$	5	0.784	1.535	0.950	1.131
$128^2$	6	0.784	1.614	0.949	1.146
$256^2$	7	0.783	1.685	0.949	1.159

Table 3. For Ex. 1 by  $B_J(S_2(1))$ 

GRID		$J = 1$	$J = 2$	$J = 3$	$J = 4$	$J = 5$	$J = 6$	$J = 7$	$J = 8$
$64^2$	#V	23	34	45	48	50	50		
	avc	0.121	0.243	0.349	0.374	0.389	0.389		
	cpu	4.4	2.1	1.7	1.7	1.7	1.7		
$128^2$	#V	24	38	52	62	69	71	71	
	avc	0.126	0.266	0.384	0.468	0.489	0.500	0.500	
	cpu	35	11.5	7.1	7.0	7.0	7.0	7.0	
$256^2$	#V	26	40	57	75	92	97	99	99
	avc	0.129	0.27	0.405	0.502	0.572	0.584	0.586	0.586
	cpu	248.0	75.2	35.2	34.3	35.3	35.1	35.1	35.2

Table 4. For Ex. 1 by  $B_J(S_3(1))$ 

GRID		$J = 1$	$J = 2$	$J = 3$	$J = 4$	$J = 5$	$J = 6$	$J = 7$	$J = 8$
$64^2$	#V	16	22	28	33	34			
	avc	0.050	0.118	0.185	0.256	0.256			
	cpu	4.0	1.7	1.5	1.5	1.5			
$128^2$	#V	17	24	31	38	43	46	46	
	avc	0.053	0.129	0.204	0.275	0.325	0.345	0.345	
	cpu	33.0	8.5	7.0	6.0	6.1	6.4	6.4	
$256^2$	#V	18	26	34	42	51	58	61	61
	avc	0.054	0.136	0.219	0.296	0.367	0.417	0.436	0.436
	cpu	252.0	61.0	27.0	24.0	28.0	31.0	32.0	32.0

Table 5. For Ex. 1 by  $B_J(S_3(2))$ 

GRID		$J = 1$	$J = 2$	$J = 3$	$J = 4$	$J = 5$	$J = 6$	$J = 7$	$J = 8$
$64^2$	#V	9	10	11	11	11	11		
	avc	0.0055	0.0092	0.011	0.012	0.0121	0.0121		
	cpu	3.3	2.0	1.5	1.0	1.0	1.0		
$128^2$	#V	10	11	12	12	12	12	12	
	avc	0.0066	0.011	0.0136	0.015	0.0155	0.0156	0.0156	
	cpu	17.0	5.3	3.8	3.0	3.0	3.2	3.2	
$256^2$	#V	10	12	12	13	13	13	13	13
	avc	0.0067	0.015	0.015	0.018	0.019	0.0191	0.0191	0.0191
	cpu	152.0	34.0	17.0	16.0	15.0	15.3	15.3	15.3

Table 6. For Ex. 2

GRID	$J$	$\lambda_m, S_3(1)$	$\lambda_M, S_3(1)$
$4^2$	1	0.772	1.090
$8^2$	2	0.687	1.208
$16^2$	3	0.718	1.329
$32^2$	4	0.737	1.442
$64^2$	5	0.751	1.541
$128^2$	6	0.759	1.626
$256^2$	7	0.762	1.699

Table 7. For Ex. 3

GRID	$J$		$N(1)$	$S_3(1)$	$S_5(1)$	$S_4(1)$	cg	$\ r_0\ _J^2$
$64^2$	5	#V	25	33	25	34	17,445	$1.4 \times 10^6$
		avc	0.164	0.255	0.157	0.276		
		cpu	2.3	0.6	0.3	0.6	143.0	
$128^2$	6	#V	29	45	29	35	55,647	$1 \times 10^7$
		avc	0.195	0.35	0.202	0.258		
		cpu	12.5	4.5	3.5	4.5	1,835.0	
$256^2$	7	#V	31	61	33	38	142,610	$8.2 \times 10^7$
		avc	0.213	0.449	0.270	0.071		
		cpu	53.5	27.5	15.5	19.5	17,003.0	

Table 8. For Ex. 3

GRID	J		$N(2)$	$S_3(2)$	$S_5(2)$	$S_6(2)$	$S_4(2)$	bpcg
$64^2$	5	#V	12	11	11	15	17	26
		avc	0.028	0.016	0.011	0.047	0.071	
		cpu	2.3	1.0	1.0	1.0	1.0	1.2
$128^2$	6	#V	14	12	11	17	17	41
		avc	0.034	0.020	0.012	0.053	0.061	
		cpu	9.0	1.8	1.6	2.5	3.5	5.5
$256^2$	7	#V	14	13	13	18	19	63
		avc	0.035	0.023	0.017	0.056	0.071	
		cpu	36.5	11.5	11.5	14.5	17.5	33.5

Table 9. For Ex. 4

GRID	J		$N(3)$	$S_6(2)$	$S_4(3)$	bpcg	cg	$\ r_0\ _J^2$
$64^2$	5	#V	13	18	21	27	313	$1.2 \times 10^{10}$
		avc	0.012	0.042	0.084			
		cpu	32.0	0.5	1.5	1.3	2.3	
$128^2$	6	#V	16	19	31	40	651	$9.1 \times 10^{10}$
		avc	0.031	0.048	0.181			
		cpu	13.0	2.5	12.0	5.5	23.0	
$256^2$	7	#V	21	25	57	62	1,329	$7.2 \times 10^{11}$
		avc	0.07	0.091	0.374			
		cpu	68.0	18.0	64.0	34.0	161.0	

516-34  
197576  
N94-21480<sup>12</sup>

# A SEMI-LAGRANGIAN APPROACH TO THE SHALLOW WATER EQUATIONS

J. R. Bates

National Aeronautics and Space Administration,  
Goddard Laboratory for Atmospheres,  
Greenbelt, MD, 20771

Stephen F. McCormick and John Ruge  
Computational Math Group, Campus Box 170,  
University of Colorado at Denver,  
PO Box 173363, Denver, CO, 80217-3364

David S. Sholl  
Program in Applied Mathematics, Campus Box 526,  
University of Colorado at Boulder  
Boulder, CO, 80309-0526

Irad Yavneh  
Oceanography and GTP,  
National Center for Atmospheric Research,  
Boulder, CO, 80307-3000

## Abstract

We present a formulation of the shallow water equations that emphasizes the conservation of potential vorticity. A locally conservative semi-Lagrangian time-stepping scheme is developed, which leads to a

system of three coupled PDE's to be solved at each time level. We describe a smoothing analysis of these equations, on which an effective multigrid solver is constructed. Some results from applying this solver to the static version of these equations are presented.

## 1 Formulation of the Shallow Water Equations

The shallow water equations provide a two-dimensional prototype of the equations needed for three-dimensional simulations of atmospheric motions [1] [2]. They are useful for testing the viability of new numerical schemes for atmospheric simulation because they share many of the properties with, but lack the full complexity of, a full three-dimensional system. The shallow water equations can be written as

$$\frac{du}{dt} = -\phi_x + fv, \quad (1)$$

$$\frac{dv}{dt} = -\phi_y - fu, \quad (2)$$

$$\frac{d\phi}{dt} = -\phi D, \quad (3)$$

where  $u$  and  $v$  are the velocity components of the wind,  $D = u_x + v_y$  is the divergence of the velocity,  $f$  is the Coriolis parameter, and  $\phi$  is the geopotential height, assumed to be a positive function. The derivatives are material derivatives, that is,

$$\frac{d}{dt} = u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} + \frac{\partial}{\partial t}. \quad (4)$$

A considerable amount of effort has gone into designing numerical methods that will solve these equations (see for example the references cited in [1]). The purpose of this paper is to study a multigrid scheme applied to a form of these equations that is of special physical interest.

There are many possible formulations of the shallow water equations. We will derive a different formulation from the one above that has certain physical and numerical advantages. To this end, we define vorticity by

$$\zeta = v_x - u_y. \quad (5)$$



Then, subtracting the  $y$ -derivative of Eq. 2 from the  $x$ -derivative of Eq. 1 gives

$$\frac{d}{dt}(\zeta + f) = -(\zeta + f)D. \quad (6)$$

Solving for  $D$  in Eq. 3 and substituting it into Eq. 6 yields

$$\frac{d}{dt}\left[\frac{\zeta + f}{\phi}\right] = 0. \quad (7)$$

Eq. 7 is important in practice because it clearly asserts that the physical quantity  $(\zeta + f)/\phi$ , called potential vorticity, is conserved in time along any Lagrangian trajectory.

Now adding the  $x$ -derivative of Eq. 1 and the  $y$ -derivative of Eq. 2 gives

$$\frac{dD}{dt} = -\nabla^2\phi - \nabla \cdot (f\mathbf{k} \times \mathbf{V}) - N, \quad (8)$$

where  $\mathbf{k} = (0, 0, 1)$ ,  $\mathbf{V} = (u, v, 0)$ , and  $N = (u_x)^2 + (v_y)^2 + 2v_xu_y$ . It is not hard to see that Eqs. 3, 7, and 8 are equivalent to the original formulation of the shallow water equations (Eqs. 1-3), but they are not yet in the form we wish to consider.

From the point of view of a multigrid solver, we will see that it is convenient to rewrite these equations in terms of the geopotential,  $\phi$ , the stream function,  $\psi$ , and the velocity potential,  $\chi$ . The latter two variables satisfy

$$\mathbf{V} = \mathbf{k} \times \nabla\psi + \nabla\chi, \quad (9)$$

$$\zeta = \nabla^2\psi, \quad (10)$$

$$D = \nabla^2\chi. \quad (11)$$

Using these variables, we arrive at the form of the shallow water equations used in this paper:

$$\frac{d}{dt}\left[\frac{\nabla^2\psi + f}{\phi}\right] = 0. \quad (12)$$

$$\frac{d\nabla^2\chi}{dt} = -\nabla^2\phi - \nabla \cdot (f\mathbf{k} \times \nabla\chi - f\nabla\psi) - N, \quad (13)$$

$$\frac{d\phi}{dt} = -\phi\nabla^2\chi, \quad (14)$$

where  $N = (\psi_{xy} - \chi_{xx})^2 + (\psi_{xy} + \chi_{yy})^2 - 2(\psi_{yy} - \chi_{xy})(\psi_{xx} + \chi_{xy})$ .

These equations have several attractive properties. As already noted, they emphasize the conservation of potential vorticity along Lagrangian trajectories. Furthermore, we shall show in Section 2 that, when a semi-Lagrangian approach is taken for the time derivatives, all of the variables appear in potential form in the resulting equations. This means that a simple vertex centered grid is sufficient to discretize the problem spatially; a staggered grid is not needed. This fact should be particularly useful when the problem is posed on a spherical domain. Finally, we shall see in Section 3 that these equations are well suited for multigrid solution.

An ideal domain for simulating atmospheric motions is a sphere. However, a spherical coordinate system introduces many difficulties that may confuse the task of developing an efficient solver for the equations at hand. Thus, as a first step in determining the feasibility of applying multigrid methods to our formulation of the shallow water equations, we have chosen to solve the system on a cylindrical domain. Specifically, we consider a domain that is periodic in the  $x$  direction with length  $d$  and includes  $y$  in the range  $[0, L]$ . We set  $\chi = \psi = 0$  and  $\phi = \phi_0$  at the  $y$  boundary, where  $\phi_0$  is a given constant. We assume that the Coriolis parameter may be written as

$$f = f_0 + \beta y, \quad (15)$$

with  $f_0$  and  $\beta$  constants. This model allows us to determine the effectiveness of multigrid methods for these equations without the complications of constructing a full three-dimensional global atmospheric model.

## 2 A Semi-Lagrangian Time Stepping Scheme

Eqs. 12-14 are written in a Lagrangian reference frame in which the evolution of the fluid is observed along the paths of imaginary fluid particles. There are some obvious disadvantages of evolving a set of particles along Lagrangian trajectories numerically. In particular, a grid that is initially uniform will in general become very irregular, often leading to a degradation of global accuracy. As a compromise, semi-Lagrangian methods have been developed to produce numerical methods that preserve the advantages of regular grids while simultaneously taking advantage of the Lagrangian form of the equations. There is an extensive body of literature describing these

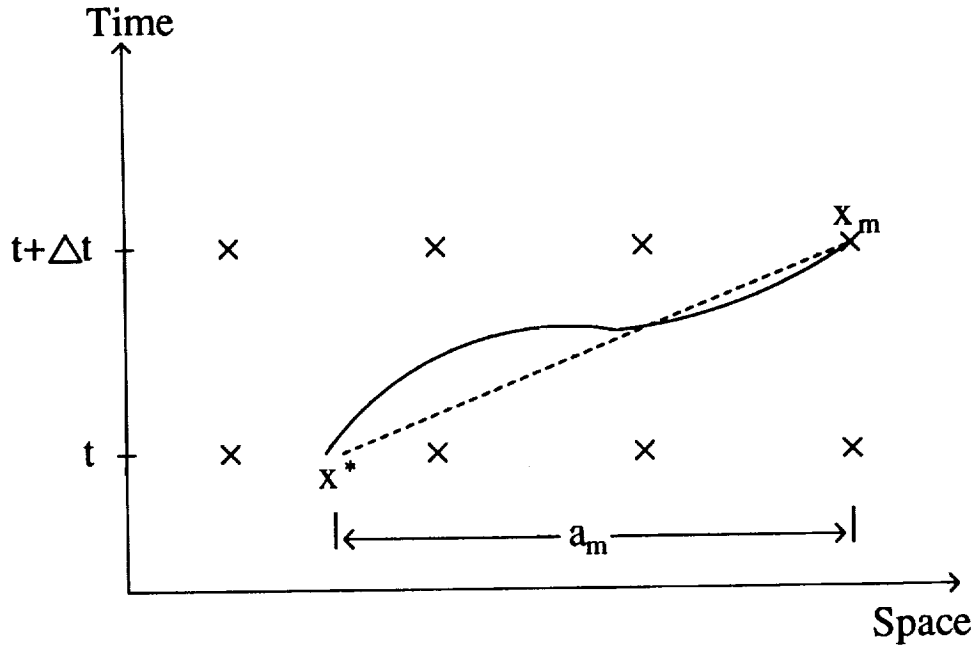


Figure 1: A schematic diagram showing the main quantities used in the calculation of the departure points for the semi-Lagrangian time-stepping scheme. The exact trajectory is represented by a solid line and the approximate trajectory with a dashed line.

methods. In particular, [1] provides an excellent review of the application of semi-Lagrangian methods to meteorological problems. This reference describes in detail a semi-Lagrangian scheme for the integration of Eqs. 1-3. The scheme we describe in this section is an adaptation of this scheme to our reformulation of the shallow water equations, and the reader is urged to consult [1] for more detail.

The fundamental idea of a semi-Lagrangian scheme is to impose a regular grid at the new time level, and to backtrack the fluid trajectories to the previous time level. At the old time level, the quantities that are needed are evaluated by interpolation from their known values on a regular grid. In general, as is the case in our problem, the velocity field at the new time step is unknown, so the critical problem in this idea is the computation of the trajectory departure points.

A schematic representation of the quantities involved in computing the

departure points is shown in Fig. 1. The displacement between a grid point on the new time level,  $\mathbf{x}_m(t)$ , and the departure point of the trajectory leading to this point on the previous time level,  $\mathbf{x}_m^*(t - \Delta t)$ , is denoted by  $\mathbf{a}_m$ . If the velocity field is considered to be constant from  $t - \Delta t$  to  $t$ , then  $\mathbf{a}_m$  satisfies the equation

$$\mathbf{a}_m = \Delta t \mathbf{V}(\mathbf{x}_m - \frac{\mathbf{a}_m}{2}, t - \frac{\Delta t}{2}). \quad (16)$$

The velocity at time  $t - \Delta t/2$  may be defined by extrapolation from the two previous time levels by

$$\mathbf{V}(\mathbf{x}, t - \frac{\Delta t}{2}) = \frac{3}{2} \mathbf{V}(\mathbf{x}, t - \Delta t) - \frac{1}{2} \mathbf{V}(\mathbf{x}, t - 2\Delta t) + \mathcal{O}(\Delta t^2). \quad (17)$$

Eqs. 16 and 17 give an implicit equation for  $\mathbf{a}_m$  in terms of the known velocity field at two previous time levels, and we may consider an iterative method for determining the correct  $\mathbf{a}_m$ . Assuming that a suitable approximation is made, then  $\mathbf{x}_m - \mathbf{a}_m/2$  would not generally lie on a grid point, so the velocities at this point must be obtained by interpolation. It has been shown [4] [5] [6] that for problems of this type it is sufficient to use linear interpolation to define the quantities in Eq. 17. It is also known [7] that successive iteration for the solution of Eq. 16 converges provided

$$\Delta t \leq \frac{1}{\max[|u_x|, |u_y|, |v_x|, |v_y|]}. \quad (18)$$

Once the  $\mathbf{a}_m$  are known, the departure point values of the variables in our equations are defined as illustrated by

$$\phi_m^*(t - \Delta t) = \phi(\mathbf{x}_m - \mathbf{a}_m, t - \Delta t). \quad (19)$$

Again, these values must be interpolated from known values at the grid points. It has been found [4] [5] [6] that it is advantageous to do this using cubic interpolation. A material time derivative may then be discretized by

$$\frac{d\phi}{dt} = \frac{1}{\Delta t} [\phi(t) - \phi^*(t - \Delta t)], \quad (20)$$

and nonderivative quantities can be represented by the simple average

$$\phi = \frac{1}{2} [\phi(t) + \phi^*(t - \Delta t)]. \quad (21)$$

Using this discretization, our formulation of the shallow water equations may be manipulated to show that the equations that determine the solution of the system at a new time level are

$$\nabla^2 \psi^+ + f^+ - f_1 \phi^+ = 0, \quad (22)$$

$$\nabla^2 \chi^+ + \tau[\nabla^2 \phi^+ + \beta \chi_x^+ - \beta \psi_y^+ - f \nabla^2 \psi^+] = f_2, \quad (23)$$

$$\phi^+[1 + \tau \nabla^2 \chi^+] = f_3, \quad (24)$$

where

$$f_1 = \frac{\nabla^2 \psi^* + f^*}{\phi^*}, \quad (25)$$

$$f_2 = \nabla \cdot [\mathbf{V}^* - \tau(f \mathbf{k} \times \mathbf{V}^* + \nabla \phi^*)], \quad (26)$$

$$f_3 = \phi^*(1 - \tau \nabla^2 \chi^*), \quad (27)$$

and  $\tau = \Delta t/2$ . The starred quantities are evaluated at the trajectory departure points at the previous time level, and the superscript  $+$  refers to quantities defined on a regular spatial grid at the new time level. We refer to Eqs. 22-24 as the static equations. The superscript  $+$  will be omitted in what follows.

The numerical algorithm needed to integrate our form of the shallow water equations splits naturally into two pieces. The first task is to compute the departure point quantities needed to define  $f_1$ ,  $f_2$ , and  $f_3$ . This is done in the manner outlined above, using information from two previous time levels. The velocity field at any time level may be obtained from  $\chi$  and  $\psi$  using  $u = -\psi_y + \chi_x$  and  $v = \psi_x + \chi_y$ . Once the departure point quantities are known, the second task is to solve the static equations. As we shall demonstrate below, it is possible to construct an efficient multigrid solver for these equations. Note that nowhere in this method is it necessary to solve Eqs. 10 and 11 for  $\chi$  and  $\psi$  in terms of  $u$  and  $v$ .

### 3 Coupling Analysis of the Static Equations

The coupling between the equations in any system of equations plays a pivotal role in the behavior of the system. In particular, when discretized systems of PDE's are to be solved by multigrid, the coupling of the equations

must determine the character of the relaxation schemes that are to be applied. Fortunately, a straightforward method for analyzing the coupling of a system and its relation to constructing a multigrid solver is available [8]. In this section we apply this method to the static equations derived above. Throughout the section, we use the definitions and notation of [8], to which the reader is referred for an understanding of the technique we are about to use.

The linearized static equations are given in brief as follows:

$$\begin{pmatrix} \nabla^2 & -f_1 & 0 \\ -f\tau\nabla^2 - \tau\beta\partial_y & \tau\nabla^2 & \nabla^2 + \tau\beta\partial_x \\ 0 & 1 + \tau\nabla^2\chi & \tau\phi\nabla^2 \end{pmatrix} \begin{pmatrix} \psi \\ \phi \\ \chi \end{pmatrix} = \begin{pmatrix} -f \\ f_2 \\ f_3 \end{pmatrix}. \quad (28)$$

In constructing this system, we have associated variables with equations in the natural way; that is,  $\psi$ ,  $\phi$ , and  $\chi$  are associated with Eqs. 22, 23, and 24, respectively.

The order array and weight array for this system are

$$Q = \begin{bmatrix} 2 & 0 & N \\ 2 & 2 & 2 \\ N & 0 & 2 \end{bmatrix}, \quad (29)$$

and

$$W = \begin{bmatrix} N & 2 & N \\ 0 & N & 0 \\ N & 2 & N \end{bmatrix}, \quad (30)$$

respectively.

To account for finite mesh size effects, we need the scaled coefficient array

$$C = \begin{bmatrix} 1 & -f_1 & N \\ -f & 1 & \tau^{-1} \\ N & \frac{1+\tau\nabla^2\chi}{\tau\phi} & 1 \end{bmatrix}. \quad (31)$$

The computation of these arrays is straightforward. The method of [8] is almost automatic, and the arrays are included here explicitly only for completeness. From these arrays, the coupling graph may be constructed, as shown in Fig. 2.

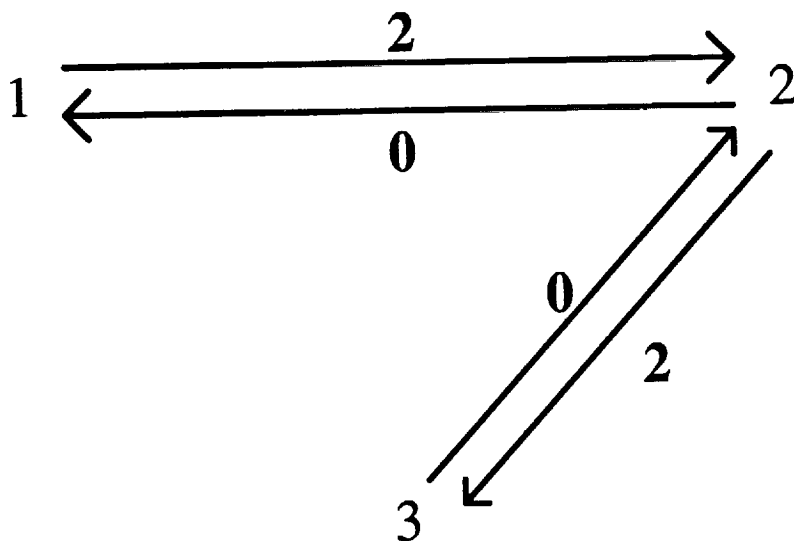


Figure 2: The coupling graph for the static equations. The finite mesh-size coupling coefficients are  $f_1$  ( $1 \rightarrow 2$ ),  $f$  ( $2 \rightarrow 1$ ),  $1/\tau$  ( $2 \rightarrow 3$ ), and  $[1 + \tau \nabla^2 \chi]/\tau \phi$  ( $3 \rightarrow 2$ ).

We may conclude immediately from the coupling graph that Eq. 22 is weakly coupled to Eq. 23 when

$$f_1 f h^2 \ll 1, \quad (32)$$

and that Eqs. 23 and 24 are weakly coupled when

$$(1 + \tau \nabla^2 \chi) \frac{h^2}{\tau^2 \phi} \ll 1. \quad (33)$$

This implies that if both of these conditions are satisfied, then each equation may be relaxed separately, as though the system were fully decoupled.

We now need to estimate the quantities in these coupling conditions using a physically realistic solution of a slightly different version of the shallow water equations. The equations we are dealing with assume that the surface of the fluid is free. To fix the surface profile of the fluid (the so-called 'rigid-lid' condition), we set  $d\phi/dt = 0$  in Eq. 1. It can then be shown by direct substitution that the following is an exact form for the resulting Rossby-Haurwitz wave solution:

$$u = U - A l \cos ly \sin k(x - ct), \quad (34)$$

$f$	$f_0 = 1 \times 10^{-4} s^{-1}, \beta = 1.57 \times 10^{-11} m^{-1} s^{-1}$	$\tau$	$500s$
$d$	$1 \times 10^7 m$	$f_1$	$\simeq 10^{-8}$
$L$	$5 \times 10^6 m$	$f_2$	$\simeq 10^{-7}$
$\phi_0$	$1 \times 10^4 m$	$f_3$	$\simeq 10^4$

Table 1: Some typical physical parameters for the shallow water equations.

$$v = Ak \sin ly \cos k(x - ct), \quad (35)$$

$$\begin{aligned} \phi = & \phi_0 - f_0 U y - \frac{1}{2} \beta U y^2 + A \sin k(x - ct) [f \sin ly - (c - U) l \cos ly] + \\ & \frac{1}{4} A^2 [l^2 \cos 2k(x - ct) + k^2 \cos 2ly], \end{aligned} \quad (36)$$

where  $A$  and  $U$  are constants,  $c = U - (\beta^2/(k^2 + l^2))$  is the Rossby-Haurwitz phase speed,  $k = 2\pi m/L$  for integer  $m$ , and  $l = n\pi/d$  for integer  $n$ . Waves of this type are the dominant feature of large scale weather motions. This solution satisfies different boundary conditions from the problem we are treating, but it is nevertheless useful for estimating the size of the parameters in our system. It can be shown for this solution that

$$\nabla^2 \chi = 0 \quad (37)$$

and

$$\nabla^2 \psi = -A(k^2 + l^2) \sin ly \sin k(x - ct). \quad (38)$$

Some typical numerical values of the parameters in the coupling conditions are shown in Table 1. A Rossby-Haurwitz wave with  $n = m = 1$ ,  $A = 3 \times 10^7 m^2 s^{-1}$ , and  $U = 20 m s^{-1}$ , together with standard physical constants, was used to derive the data in this table. From these values it can be seen that Eq. 32 is satisfied, but Eq. 33 is certainly violated on intermediate and coarse grids. In terms of constructing a good smoother for the system, this means that Eq. 22 can be relaxed as though it were decoupled from the system, but the two remaining equations must be dealt with together, at least on coarse grids. In practise, it is easiest to use the same smoother on all grids to start with.

To deal with Eqs. 22 and 23 together, collective relaxation is used. For linear equations, this means that, when the equations are relaxed at a point,



corrections are made to all the variables associated with the equations such that the residuals of the equations become zero at that point. This may be done by replacing  $\phi$  and  $\chi$  with  $\phi + \delta_\phi$  and  $\chi + \delta_\chi$ , respectively, in Eqs. 23 and 24 at a single point and the differential operators with their discretized counterparts and solving for the corrections  $\delta_\phi$  and  $\delta_\chi$ . Because Eq. 24 is nonlinear, a term proportional to  $\delta_\phi \delta_\chi$  appears. We neglect this term and solve the resulting linear system directly. This method is equivalent to taking a single Newton step for these equations.

## 4 Preliminary Numerical Results

A preliminary code has been implemented that applies the multigrid method just described to the static equations. Eq. 22 was relaxed by red-black Gauss-Seidel iteration, and Eqs. 23 and 24 were relaxed collectively as described above in a lexicographic ordering. The equations are nonlinear, so the Full Approximation Scheme (FAS) [9] was used for the coarse-to-fine corrections. Full weighting was used for the fine to coarse grid restrictions, and linear interpolation for the coarse to fine grid transfers. Note that the grid transfers are straightforward because all the variables are defined on the same vertex centered grid. The standard five-point discretization was used for the Laplacian operator. Similarly, other derivatives were discretized using the usual finite difference formulae. At the time of writing, a semi-Lagrangian time-stepping scheme had been implemented, but the two codes had not been fully combined.

In order to test the convergence of the multigrid scheme, we set the forcing functions in the static equations to a variety of functional forms. The magnitude of these functions was indicated by the Rossby-Haurwitz wave solution introduced in the previous section. When the problem was solved on a  $64 \times 32$  grid with a V(1,1) cycle, the convergence rates for the  $L^2$  norm of the residuals were 0.22, 0.25, and 0.27 for Eqs. 22-24, respectively. When a V(2,1) cycle was used, the rates were 0.15, 0.13, and 0.14. In each of these cases, a single relaxation sweep consisted of relaxing Eq. 22 once followed by relaxing Eqs. 23 and 24 collectively once.

These results suggest that multigrid may be an efficient way of solving these equations. Clearly, there are many possible variants on the scheme described above. For instance, the coupling analysis suggests that it may be

fruitful to relax Eqs. 23 and 24 independently on fine grids and switch to collective relaxation only on coarser grids.

## References

- [1] Staniforth, J. and Côté, J.: *Semi-Lagrangian Integration Schemes for Atmospheric Models - A Review*, Monthly Weather Review **119**, pp. 2206-2223 (1991).
- [2] Pedlosky, J. : *Geophysical Fluid Dynamics*, 2nd ed., Springer-Verlag (1987).
- [3] McDonald, A. and Bates, J.R.: *Semi-Lagrangian Integration of a Grid-point Shallow Water Model On A Sphere*, Monthly Weather Review **117**, pp. 131-137 (1989).
- [4] Staniforth, J. and Pudykiewicz, J.: *Reply to comments on and addenda to 'Some properties and comparative performances of the semi-Lagrangian method of Robert in the solution of the advection-diffusion equation'*, Atmos. Ocean **23**, pp. 195-200 (1985).
- [5] Temperton, C. and Staniforth, J.: *An Efficient Two-Time-Level Semi-Lagrangian Semi-Implicit Integration Scheme*, Quart. J. Roy. Meteor. Soc. **113**, pp. 1025-1039 (1987).
- [6] Bates, J.R., Semazzi, F.H.M., Higgins, R.W. and Barros, S.R.M. : *Integration of the Shallow Water Equations On The Sphere Using A Vector Semi-Lagrangian Scheme with a Multigrid Solver*, Mon. Wea. Rev. **118**, pp. 1615-1627 (1990).
- [7] Pudykiewicz, J., Benoit, R. and Staniforth, A. : *Preliminary Results From A Partial LRTAP Model Based On An Existing Meteorological Forecast Model*, Atmos. Ocean. **23**, pp. 267-303 (1985).
- [8] Yavneh, I.: *A Method For Devising Efficient Multigrid Smoothers for Complicated PDE Systems*, SIAM J. Sci. Comput., In Press
- [9] Brandt, I.: *1984 Multigrid Guide with Applications to Fluid Dynamics*, Monograph, GMD-Studie 85, GMD-FIT, Postfach 1240, D-5205, St. Augustin 1, West Germany, 1985

# MULTIGRID SOLUTION OF THE NAVIER-STOKES EQUATIONS ON HIGHLY STRETCHED GRIDS WITH DEFECT CORRECTION

Peter M. Sockol  
Internal Fluid Mechanics Division  
NASA Lewis Research Center  
Cleveland, OH 44135, U.S.A.

N 947-231481

197577

P. 15

## SUMMARY

Relaxation-based multigrid solvers for the steady incompressible Navier-Stokes equations are examined to determine their computational speed and robustness. Four relaxation methods with a common discretization have been used as smoothers in a single tailored multigrid procedure. The equations are discretized on a staggered grid with first order upwind used for convection in the relaxation process on all grids and defect correction to second order central on the fine grid introduced once per multigrid cycle. A fixed W(1,1) cycle with full weighting of residuals is used in the FAS multigrid process. The resulting solvers have been applied to three 2D flow problems, over a range of Reynolds numbers, on both uniform and highly stretched grids. In all cases the  $L_2$  norm of the velocity changes is reduced to  $10^{-6}$  in a few 10's of fine grid sweeps. The results from this study are used to draw conclusions on the strengths and weaknesses of the individual relaxation schemes as well as those of the overall multigrid procedure when used as a solver on highly stretched grids.

## 1. INTRODUCTION

In recent years there has been considerable progress in the development of multigrid solvers for the steady incompressible Navier-Stokes equations. The multigrid process and its application to fluid dynamics has been well described by Brandt<sup>1</sup>. Ghia *et al.*<sup>2</sup> used the streamfunction vorticity formulation with the coupled strongly implicit scheme of Rubin and Khosla<sup>3</sup> as a smoothing operator and an accommodative multigrid cycle. Defect correction was used to increase the accuracy of the convection terms. Vanka<sup>4</sup> employed a locally coupled Gauss-Seidel smoother for the primitive variable formulation together with an accommodative cycle. Demuren<sup>5</sup> extended Vanka's smoother to one in which local corrections were coupled to neighboring pressure corrections and solved the resulting equations by both a strongly implicit technique and an alternating direction line Gauss-Seidel scheme. Thompson and Ferziger<sup>6</sup> used Vanka's smoother as well as a fully coupled alternating direction line Gauss-Seidel extension and an accommodative cycle. This study also introduced defect correction together with local adaptive grid refinement. Sivaloganathan and Shaw<sup>7</sup> used the SIMPLE pressure-correction scheme of Patankar and Spalding<sup>8</sup> as a smoother for the primitive variable formulation. The smoothing analysis given in Shaw and Sivaloganathan<sup>9</sup> indicates that a fixed V-cycle was used in the multigrid process. Dick<sup>10</sup> developed a partially flux-split discretization for the primitive variable

formulation and used a coupled red-black smoother and a fixed W-cycle. Finally, a few solvers have used boundary-fitted curvilinear coordinates with primitive variables. Joshi and Vanka<sup>11</sup> extended Vanka's coupled Gauss-Seidel relaxation technique to this system. Rayner<sup>12</sup> and Shyy *et al.*<sup>13</sup> developed variants to the SIMPLE pressure correction method for use as smoothers with the latter applicable to all speeds. The last three references all employed a fixed V-cycle.

In most of the above efforts a single relaxation scheme has been used as a smoothing operator in a chosen multigrid cycle and applied to one or more problems in order to demonstrate the characteristics of the flow solver. This doesn't provide much guidance in the choice of smoother or multigrid cycle for the developer of a solver for a particular application. Furthermore, among the above works only Brandt<sup>1</sup> and Thompson and Ferziger<sup>6</sup> have addressed the need for highly refined grids in local regions, which is present in most flow problems. The adaptive use of several levels of uniform local subgrids<sup>6</sup> is attractive in the multigrid context since it adds extra points only where they are needed. A more conventional approach employs stretched grids which may make it easier to resolve thin regions of steep gradients such as boundary layers adjacent to solid surfaces. This raises the question, however, as to whether fast multigrid performance can be maintained on these grids.

The present work considers the primitive variable formulation of the steady incompressible Navier-Stokes equations in Cartesian coordinates. Four relaxation methods with a common discretization have been used as smoothers and embedded in a single tailored multigrid procedure. The equations are discretized on a staggered grid with first order upwind used for convection in the relaxation process on all grids and defect correction to second order central on the fine grid introduced once per multigrid cycle. The resulting solvers have been applied to three two-dimensional problems over a range of Reynolds numbers on both uniform and highly stretched grids. The results from this study are used to draw conclusions on the strengths and weaknesses of the individual relaxation schemes as well as those of the overall multigrid procedure when used as a solver on highly stretched grids. The results from an earlier study using first order hybrid differencing will be presented elsewhere<sup>14</sup> in somewhat greater detail.

## 2. DISCRETE FORMULATION

The steady incompressible Navier-Stokes equations in non-dimensional form are written as

$$\frac{\partial uu}{\partial x} + \frac{\partial uv}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re} \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right), \quad (1)$$

$$\frac{\partial uv}{\partial x} + \frac{\partial vv}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re} \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right), \quad (2)$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \quad (3)$$

where  $u$  and  $v$  are the  $x$  and  $y$  velocity components,  $p$  is the pressure, and  $Re$  is the Reynolds number.

These equations are discretized on a staggered grid using a finite volume approach:

$$-R_{i,j}^u \equiv L^u u_{i,j} + dy_j \Delta_x p_{i,j} = 0, \quad (4)$$

$$-R_{i,j}^v \equiv L^v v_{i,j} + dx_i \Delta_y p_{i,j} = 0, \quad (5)$$

$$-R_{i,j}^c \equiv dy_j \nabla_x u_{i,j} + dx_i \nabla_y v_{i,j} = 0, \quad (6)$$

where  $\Delta_x$ ,  $\nabla_x$ ,  $\Delta_y$ ,  $\nabla_y$ , are forward and backward differences in  $x$  and  $y$ , respectively,  $dx_i = x_i - x_{i-1}$ ,  $dy_j = y_j - y_{j-1}$ , and

$$L^u u_{i,j} = a_c^u u_{i,j} - a_w^u u_{i-1,j} - a_e^u u_{i+1,j} - a_s^u u_{i,j-1} - a_n^u u_{i,j+1}, \quad (7)$$

$$L^v v_{i,j} = a_c^v v_{i,j} - a_w^v v_{i-1,j} - a_e^v v_{i+1,j} - a_s^v v_{i,j-1} - a_n^v v_{i,j+1}, \quad (8)$$

When these expressions require points outside the domain, such as  $L^u u_{i,j}$  adjacent to a horizontal boundary, these points are transferred to the boundary by quadratic extrapolation. A linear extrapolation is employed at an outflow boundary where  $p_{i,j}$  is specified.

The coefficients in Eqs. (7) and (8) are obtained by utilizing first order upwinding for convection and second order central differencing for diffusion. The difference between first order upwind and second order central convection discretizations on the finest grid is added as a defect correction source term in a manner similar to that of Thompson and Ferziger<sup>6</sup>. Prior to each sweep through the grid a single set of coefficients,  $a^p$ , is obtained for equations centered on the  $p_{i,j}$  locations and held constant during the sweep. The coefficients  $a^u$  and  $a^v$  are obtained by averaging. Thus

$$(a_c^u)_{i,j} = [(a_c^p)_{i,j} + (a_c^p)_{i+1,j}]/2, \quad (a_c^v)_{i,j} = [(a_c^p)_{i,j} + (a_c^p)_{i,j+1}]/2.$$

For the convective terms this is equivalent to obtaining the cell face velocities by averaging. For the viscous terms this introduces an error on a stretched grid that is of the same order as the truncation error. In the immediate vicinity of a reentrant corner this practice must be modified to ensure that the convective velocity normal to the wall is set to zero.

### 3. RELAXATION METHODS

Each of the relaxation methods employed as a multigrid smoother in this work is adapted from or similar to a known technique from the literature, and hence the descriptions of the schemes will be brief. The methods are written in a common block-tridiagonal form for the corrections along a horizontal line:

$$-A_i \Delta V_{i-1} + B_i \Delta V_i - C_i \Delta V_{i+1} = D_i, \quad (9)$$

where  $\Delta V_i$  is the vector of local corrections,  $A_i$ ,  $B_i$ ,  $C_i$  are square matrices, and  $D_i$  is the vector of local residuals. By appropriate choices of the square matrices, Eq.(9) can be used to describe both point or explicit schemes and semi or fully implicit schemes. This equation is now particularized for each of the methods.

The first method, here labeled Block Gauss-Seidel (BGS), is a locally coupled explicit scheme introduced by Vanka<sup>4</sup>. Four discrete momentum equations and one continuity equation are solved for a set of local corrections. In this case

$$\begin{aligned}\Delta \mathbf{V}_i &= (\Delta u_{i-1,j}, \Delta u_{i,j}, \Delta v_{i,j-1}, \Delta v_{i,j}, \Delta p_{i,j})^T, \\ \mathbf{D}_i &= (R_{i-1,j}^u, R_{i,j}^u, R_{i,j-1}^v, R_{i,j}^v, R_{i,j}^c)^T,\end{aligned}\quad (10)$$

$\mathbf{B}_i$  is a  $5 \times 5$  matrix,

$$\mathbf{B}_i = \begin{pmatrix} (a_c^u)_{i-1,j} & 0 & 0 & 0 & dy_j \\ 0 & (a_c^u)_{i,j} & 0 & 0 & -dy_j \\ 0 & 0 & (a_c^v)_{i,j-1} & 0 & dx_i \\ 0 & 0 & 0 & (a_c^v)_{i,j} & -dx_i \\ -dy_j & dy_j & -dx_i & dx_i & 0 \end{pmatrix}, \quad (11)$$

and  $\mathbf{A}_i = \mathbf{C}_i = 0$ . Elimination of the  $\Delta u$ 's and  $\Delta v$ 's gives a simple expression for  $\Delta p_{i,j}$  and back substitution then gives the local  $\Delta u$ 's and  $\Delta v$ 's. In a single sweep through the grid, each momentum equation is updated twice and each continuity equation once.

The second method, labeled Pressure-linked Line Block Gauss-Seidel (PLBGS), is a locally coupled semi-implicit scheme which is similar to the line relaxation scheme of Demuren<sup>5</sup>. This case is a simple extension of BGS:

$$\begin{aligned}\Delta \mathbf{V}_i &= (\Delta u_{i,j}, \Delta v_{i,j-1}, \Delta v_{i,j}, \Delta p_{i,j})^T, \\ \mathbf{D}_i &= (R_{i,j}^u, R_{i,j-1}^v, R_{i,j}^v, R_{i,j}^c)^T,\end{aligned}\quad (12)$$

$\mathbf{B}_i$  is a  $4 \times 4$  matrix obtained by eliminating the top row and left column from Eq.(11), and  $\mathbf{A}_i = \mathbf{C}_i = 0$  except for the lower left and upper right corner elements, respectively. Elimination of the  $\Delta u$ 's and  $\Delta v$ 's gives a scalar tridiagonal equation for the  $\Delta p$ 's along the horizontal line and back substitution then gives the  $\Delta u$ 's and  $\Delta v$ 's along the line. During a single sweep in the  $+y$  direction, each  $u$ -momentum equation is updated once, each  $v$ -momentum equation twice, and each continuity equation once. The fewer momentum updates and the efficiency of the scalar tridiagonal inversion gives a scheme that costs 15% less per sweep than BGS. In general both  $x$  and  $y$  sweeps are combined in an alternating pattern to form an effective relaxation technique.

The third method, labeled Line Block Gauss-Seidel (LBGS), is a locally coupled, fully implicit scheme, which is apparently very similar to the coupled alternating line approach of Thompson and Ferziger<sup>6</sup>. The vectors  $\Delta \mathbf{V}_i$  and  $\mathbf{D}_i$  and the matrix  $\mathbf{B}_i$  are the same as for PLBGS, while  $\mathbf{A}_i$  and  $\mathbf{C}_i$  are  $4 \times 4$  matrices having diagonal plus the lower left and upper right corner elements, respectively. The number of equation updates and sweeping patterns are the same as for PLBGS. In this case algebraic elimination in the block-tridiagonal inversion gives a scheme that costs only 15% more per sweep than BGS.

The final method is the Semi-Implicit Pressure-Correction scheme (SIMPLE) introduced by Patankar and Spalding<sup>8</sup>. In this case

$$\begin{aligned}\Delta \mathbf{V}_i &= (\Delta u_{i,j}, \Delta v_{i,j})^T, \\ \mathbf{D}_i &= (R_{i,j}^u, R_{i,j}^v)^T,\end{aligned}\quad (13)$$

where  $A_i, B_i, C_i$  are diagonal  $2 \times 2$  matrices. The pressure is obtained from an elliptic equation derived by substituting reduced forms of the discrete momentum equations for coupled velocity and pressure corrections into continuity. For this work one SIMPLE iteration consists of a single scalar line Gauss-Seidel sweep for each momentum equation with the pressure fixed. This is followed by four alternating direction line Gauss-Seidel sweeps of the elliptic pressure-correction equation. Taking more than one sweep through the momentum equations before correcting the pressure *invariably* resulted in partial decoupling of the velocity components and slower convergence. Each of these combined SIMPLE iterations costs about 30% more than one sweep of BGS.

For each of these relaxation techniques some degree of underrelaxation is required to obtain convergence. In the present work this is implemented through direct modification of the momentum equations. For BGS, LBGS, and SIMPLE, the diagonal velocity coefficients,  $a_c^u$  and  $a_c^v$ , in the matrix  $B_i$  are divided by a factor  $r_{mom}$  where  $0 < r_{mom} < 1$ . For PLBGS the residuals,  $R^u$  and  $R^v$ , are multiplied by  $r_{mom}$ . In addition for SIMPLE the pressure corrections and the corresponding velocity corrections required to satisfy continuity are unrelaxed.

Finally, we note that considerable improvement can be obtained with each of the above methods by employing a symmetric sweeping pattern. Thus for BGS each lexicographic sweep is followed by one in the reverse direction. For PLBGS, LBGS, and SIMPLE a four sweep symmetric alternating line pattern is used, i.e. relaxation is performed sequentially in the  $+x, +y, -y, -x$  directions. These techniques result in an approximately 25% improvement in convergence rates.

#### 4. MULTIGRID IMPLEMENTATION

Local relaxation methods, such as those of the previous section, are in general much more efficient at reducing short wavelength error components on a given grid than those of longer wavelength. Multigrid seeks to overcome this problem by transferring the longwave components of the solution to a sequence of coarser grids where relaxation is more effective and much cheaper. Since the FAS-FMG technique used in this work has been well documented in the literature<sup>1,2,4-7</sup>, it will not be described here. The focus will instead be on the current implementation and in particular on those aspects which are important for achieving a fast, robust Navier-Stokes solver.

In the present work the coarse grids are created by "standard coarsening," i.e., every second grid point in both  $x$  and  $y$  is deleted from one grid to the next coarser grid. The fine-to-coarse restriction operator  $\hat{I}_j^c$  for unknowns employs cell-face averaging for the velocities,

$$u_{i,j}^c = (u_{i,j-1} dy_{j-1} + u_{i,j} dy_j) / dy_j^c, \quad v_{i,j}^c = (v_{i-1,j} dx_{i-1} + v_{i,j} dx_i) / dx_i^c, \quad (14)$$

and full-weighting for the pressures,

$$p_{i,j}^c = (p_{i-1,j-1} dx_{i-1} dy_{j-1} + p_{i-1,j} dx_{i-1} dy_j + p_{i,j-1} dx_i dy_{j-1} + p_{i,j} dx_i dy_j) / (dx_i^c dy_j^c), \quad (15)$$

where  $( )^c$  represents a coarse-grid value. The restriction operator  $I_f^c$  for residuals uses full-weighting, in which all the fine-grid contributions to a coarse-grid cell are accounted for:

$$\begin{aligned}(I_f^c R^u)_{i,j} &= R_{i,j-1}^u + R_{i,j}^u + \frac{1}{2}(R_{i-1,j-1}^u + R_{i-1,j}^u + R_{i+1,j-1}^u + R_{i+1,j}^u), \\ (I_f^c R^v)_{i,j} &= R_{i-1,j}^v + R_{i,j}^v + \frac{1}{2}(R_{i-1,j-1}^v + R_{i,j-1}^v + R_{i-1,j+1}^v + R_{i,j+1}^v),\end{aligned}\tag{16}$$

where  $R^u$  and  $R^v$ , given by Eqs.(4) and (5), are already area-weighted. With the cell-face averaging given by Eqs.(14) and full-weighting similar to Eq.(15) for  $R^c$ , as defined by Eq.(6), the coarse-grid source term vanishes for the continuity equation.

In many of the previous works<sup>4,5,7,13</sup> cell-face averaging was also used in the restriction of  $R^u$  and  $R^v$ . For uniform grids this has little effect on the multigrid convergence rate. For the highly stretched grids employed in this work this proved to be ineffective. In some cases convergence slowed by a factor of three or four. In others little or no benefit was gained from the multigrid process.

The coarse-to-fine prolongation operator  $I_c^f$  for corrections employs bilinear interpolation in computational space where the grid spacing is taken to be uniform. For fine grid points adjacent to boundaries, a zero normal gradient is assumed for pressures. The overall convergence has proven to be insensitive to the details of this approximation. The same operator with one modification is also used to interpolate "converged" results to obtain initial values on a fine grid in the FMG process. The velocity component parallel to an adjacent wall is obtained by bilinear extrapolation from the interior since the boundary layer is poorly resolved on the coarse grid.

The multigrid solvers in this work have been coded to permit fixed V-cycles and W-cycles. During the course of this effort it was found that for the difficult cases with high Reynolds numbers or highly stretched grids a W(1,1) cycle was the most effective strategy in terms of robustness and computational cost. Hence, all results presented in this paper were performed using this cycle. The defect correction source term, discussed earlier, is updated once per cycle on the finest grid. Accommodative cycles<sup>1,2,4-6</sup>, which decide on whether or not to restrict to a coarser grid based on the ratio of errors from two successive sweeps, proved to be too costly since the second sweep on each visit to a grid contributed little to the overall convergence of the method.

The symmetric sweeping pattern described in the previous section has been interleaved with the multigrid process. A sweep counter is established for every grid level, and on each visit to that level the next direction in the sweep pattern for that grid is performed. This proved to be sufficient to give all the convergence benefits of the sweeping symmetry. Finally, it should be noted that varying the momentum relaxation factor  $r_{mom}$  from grid to grid during the cycle provided considerable performance enhancement for the BGS, PLBGS, and LBGS solvers. No benefit, however, was observed when this was tried with the SIMPLE-based solver.

## 5. CONVERGENCE CRITERIA

The various convergence criteria used in this work are all based on an  $L_2$  norm of the dynamic velocity changes occurring during a sweep through the grid. This would seem



to be a more appropriate form for a system of coupled equations than one based on a combination of the residuals of the different equations. The pressures have been excluded since they are only determined to within an arbitrary constant. Introduce the definition

$$\epsilon^k = \sqrt{\sum_{i,j=1}^{n_x^k, n_y^k} [(\Delta u_{i,j}^k)^2 + (\Delta v_{i,j}^k)^2] / (2n_x^k n_y^k)}, \quad (17)$$

where  $n_x^k$  and  $n_y^k$  are the number of cells on grid  $k$  in  $x$  and  $y$  respectively, and  $\Delta u_{i,j}^k$ ,  $\Delta v_{i,j}^k$  are the dynamic velocity changes obtained during a sweep on grid  $k$ . Then for a sequence of coarse-to-fine grids,  $k = 1$  to  $m$ , the overall convergence criterion on grid  $m$  is taken as

$$\epsilon^m < 10^{-6}. \quad (18)$$

In most cases at convergence given by Eq.(18) the value of  $\max(\Delta u, \Delta v)$  is approximately  $10^{-5}$ . For intermediate grids in the FAS-FMG process, convergence before interpolating to the next finer grid is taken as

$$\epsilon^k < 10^{-3}, \quad (19)$$

and for the coarsest grid,  $k = 1$ , "solution" is given by

$$\epsilon^1 < \epsilon^k / 10, \quad (20)$$

where now  $\epsilon^k$  is the most recent error on the current finest grid.

Finally, it is noted that all computations in this work were performed on an Amdahl 5980 in scalar mode. All CPU times reported in the next sections are for this machine.

## 6. COMPUTATIONAL RESULTS

Three problems have been chosen to test the performance of the multigrid solvers under different conditions: flow in a driven cavity, developing flow in a straight channel, and flow over an open cavity.

### Driven Cavity Flow

The driven cavity is the prototypical recirculating flow and has long been used as a standard test problem for Navier-Stokes solvers. The second-order streamfunction-vorticity results of Ghia *et al.*<sup>2</sup> are generally accepted as the standard. Flow is set up in a square cavity with three stationary walls and a top lid that moves to the right with constant speed ( $u = 1$ ). Profiles of  $u$  on the vertical centerline computed on a uniform  $256 \times 256$  grid for  $Re = 1000$  and  $5000$  are compared with the standard results<sup>2</sup> in Figure 1. The present defect correction results agree with the standard to within plotting accuracy.

The first set of results for this flow is for a uniform grid with  $Re$  varying from 100 to 5000. Table I compares the uniform grid results for each solver on a  $256 \times 256$  grid in terms of cpu times, number of fine grid sweeps, and total work units for each case where a work unit is the cpu time required for one fine grid sweep of the particular smoother. Here

$r_{mom}^{fg}$  is the fine grid relaxation factor for the solver. As the Reynolds number increases the table indicates a significant advantage for BGS and PLBGS over LBGS and SIMPLE due to faster convergence and less cost per sweep.

The second set of results is obtained for  $Re = 1000$  on a grid with hyperbolic tangent stretching in  $x$  and  $y$  and the maximum mesh aspect ratio (AR) varying from 1 to 40. Table II compares the stretched grid results for each solver on a  $256 \times 256$  grid. As AR is increased to large values BGS is seen to have a significant advantage over the other three methods in both number of sweeps and cpu time. The use of highly stretched grids produces a strong asymmetry in the momentum equation coupling coefficients in regions of high mesh aspect ratio and this was expected to adversely affect the smoothing properties of an explicit scheme<sup>1</sup> such as BGS. The alternating direction semi-implicit and fully implicit schemes were introduced to see if they would give more robust performance for these cases. This proved not to be true for the Navier-Stokes solvers used in this study.

### Developing Channel Flow

The second test problem is the deceptively simple one of developing flow in a straight channel one unit high by four units long. Uniform velocities ( $u = 1$ ,  $v = 0$ ) are specified at the entrance and a constant pressure ( $p = 0$ ) is set at the exit. Note, for incompressible flow, the common exit condition,  $\partial u / \partial x = 0$ , implies  $\partial v / \partial y = \partial p / \partial y = 0$ . Profiles of  $u$  vs  $y$  along the channel for  $Re = 1000$  are shown in Figure 2. For this and higher Reynolds numbers the flow is far from fully developed at the exit. This flow has velocities strongly aligned with the  $x$  direction over much of the domain and the  $u$  momentum equation becomes increasingly decoupled in  $y$  away from the walls as  $Re$  is increased. This situation is known to cause problems for multigrid solvers (see e.g. Brandt<sup>1</sup> and Mulder<sup>15</sup>) and thus was chosen as a fitting test case for this study.

The first set of results is for a uniform grid with  $Re$  again varying from 100 to 5000. The uniform grid results for each solver on a  $256 \times 64$  grid are compared in Table III. It is evident that the multigrid performance of all solvers degrades more rapidly with increasing  $Re$  than was the case for the driven cavity. The relatively poor performance of SIMPLE is probably due to the partial decoupling between  $u$  and  $v$  at high  $Re$  which was observed during the iterative process. Note, however, that all methods still converged in under 100 fine grid sweeps even at the highest Reynolds numbers.

The second set of results for this flow is for hyperbolic tangent stretching in  $y$  only, again with AR varying from 1 to 40 and  $Re = 1000$ . Stretched grid results for each solver on a  $256 \times 64$  grid are compared in Table IV. As AR is increased to large values, it is evident that LBGS has a major advantage over the other smoothers in both fine grid sweeps and cpu time. This case of strong alignment on a stretched grid is the only one in which an implicit scheme (LBGS) has a substantial advantage over the explicit BGS.

### Open Cavity Flow

The final test problem combines the driven cavity and developing channel flows and adds the complication of a strong corner singularity. The domain consists of a channel one unit high by two units long on top of an open square cavity one unit on a side located at

the left boundary of the channel. Uniform flow ( $u = 1, v = 0$ ) enters the channel at the left and the flow exits at the right ( $p = 0$ ). Streamfunction and vorticity contours for  $Re = 1000$  are shown in Figure 3. Note the lack of separation and the strong concentration of vorticity contours at the downstream corner.

As before the first set of results is for a uniform grid with  $Re$  varying from 100 to 5000. The uniform grid results for each solver on a  $128 \times 128 + 256 \times 128$  grid are compared in Table V. The results for both fine grid sweeps and cpu time show that BGS, PLBGS, and LBGS remain competitive as Reynolds number is increased but SIMPLE suffers a substantial penalty.

The second set of results for this flow is for hyperbolic tangent stretching in both  $x$  and  $y$ , in each of three square regions, with AR varying from 1 to 40 and  $Re = 1000$ . The stretched grid results for each solver on a  $128 \times 128 + 256 \times 128$  grid are compared in Table VI. Here it is evident that BGS has a significant advantage in fine grid sweeps and cpu time as AR increases. It should also be noted that PLBGS and LBGS appeared to be more sensitive to the presence of the corner singularity and to the choice of  $r_{mom}$  for the set of grids used in the multigrid process. However no detailed study of this effect was performed.

## 7. CONCLUSIONS

From the above results, it is evident that a proper combination of tailored multigrid elements can yield a fast robust solver for the steady incompressible Navier-Stokes equations even on highly stretched grids. In particular, for fine-to-coarse restriction of residuals, the use of full weighting is important on stretched grids. For coarse-to-fine prolongation of corrections, on the other hand, bilinear interpolation works well and is insensitive to the details of the boundary treatment. And finally a fixed  $W(1,1)$  multigrid cycle appears to offer a good mix of robustness and computational efficiency.

For recirculating flows such as the driven cavity, all four smoothers are effective and competitive. On uniform grids BGS and PLBGS offer a significant advantage over LBGS and SIMPLE, primarily due to less cost per sweep. On stretched grids BGS and SIMPLE show superior multigrid performance, but BGS is substantially cheaper per sweep.

For strongly aligned flows such as that in a developing channel, all four solvers degrade more rapidly with increasing Reynolds number than for recirculating flows with SIMPLE falling off much more rapidly than the others, but they all still converge in under 100 fine grid sweeps. On highly stretched grids, however, LBGS offers a major advantage in both multigrid performance and net cpu time over the other three smoothers. This is the only case in which an implicit scheme is distinctly superior to the explicit BGS.

For mixed recirculating/aligned flows such as the open cavity, all four smoothers are effective. On uniform grids, SIMPLE again degrades much more rapidly than the others with increasing Reynolds number. On stretched grids BGS offers a small advantage in multigrid performance, but this becomes significant when net cpu time is considered. It is also notable that BGS is less sensitive than the other smoothers to the corner singularity in this flow.

On balance BGS offers the best mix of robustness and computational speed for all three classes of flows. The semi-implicit schemes PLBGS and SIMPLE offer little or no advantage and in general are less robust. The fully implicit LBGS is superior only for the case of highly aligned flows on stretched grids. The pressure correction scheme SIMPLE is in general more costly than the other three and degrades much more rapidly than the others with increasing Reynolds number. Finally, we note that for a general multigrid solver set up using domain decomposition, it might be highly effective to use BGS over most domains but retain the option to use LBGS in strongly aligned domains.

## REFERENCES

1. A. Brandt, 'Multigrid Techniques: 1984 Guide With Applications to Fluid Dynamics', von Karman Institute, *Lecture Series 1984-04*, 1984.
2. U. Ghia, K. N. Ghia and C. T. Shin, 'High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method', *J. Comput. Phys.*, **48**, 387-411 (1982).
3. S. G. Rubin and P. K. Khosla, 'Navier-Stokes Calculations with a Coupled Strongly Implicit Method—I', *Comput. Fluids*, **9**, 163-180 (1981).
4. S. P. Vanka 'Block-Implicit Multigrid Solution of the Navier-Stokes Equations in Primitive Variables', *J. Comput. Phys.*, **65**, 138-158 (1986).
5. A. O. Demuren 'Application of Multi-Grid Methods for Solving the Navier-Stokes Equations', *Proc. Inst. Mech. Engrs., Part C.*, **203**, 255-265 (1989).
6. M. C. Thompson and J. H. Ferziger 'An Adaptive Multigrid Technique for the Incompressible Navier-Stokes Equations', *J. Comput. Phys.*, **82**, 94-121 (1989).
7. S. Sivaloganathan and G. J. Shaw 'A Multigrid Method for Recirculating Flows', *Int. J. Numerical Methods Fluids*, **8**, 417-440 (1988).
8. S. V. Patankar and D. B. Spalding 'A Calculation Procedure for Heat and Mass Transfer in 3-D Parabolic Flows', *Int. J. Heat Mass Transfer*, **15**, 1787-1806 (1972).
9. G. J. Shaw and S. Sivaloganathan 'On the Smoothing Properties of the SIMPLE Pressure-Correction Algorithm', *Int. J. Numerical Methods Fluids*, **8**, 441-461 (1988).
10. E. Dick 'A Multigrid Method for Steady Incompressible Navier-Stokes Equations Based on Partial Flux Splitting', *Int. J. Numerical Methods Fluids*, **9**, 113-120 (1989).
11. D. S. Joshi and S. P. Vanka 'Multigrid Calculation Procedure for Internal Flows in Complex Geometries', *Numer. Heat Transfer, Part B.*, **20**, 61-80 (1991).
12. D. Rayner 'Multigrid Flow Solutions in Complex Two-Dimensional Geometries', *Int. J. Numerical Methods Fluids*, **13**, 507-518 (1991).
13. W. Shyy, M-H. Chen and C-S Sun 'A Pressure-Based FMG/FAS Algorithm for Flow at All Speeds', AIAA 92-0548, 30th Aerospace Sciences Meeting & Exhibit (1992).
14. P. M. Sockol 'Multigrid Solution of the Navier-Stokes Equations on Highly Stretched Grids', to be published in *Int. J. Numerical Methods Fluids*.
15. W. A. Mulder 'A New Multigrid Approach to Convection Problems', *J. Comput. Phys.*, **83**, 303-323 (1989).

Table I. Driven Cavity Convergence  
Uniform  $256 \times 256$  grid,  $AR = 1$

Scheme ( $r_{mom}^{fg}$ )	Re				
	100	400	1000	3200	5000
	(cpu seconds/fg. sweeps/work units)				
BGS (0.6)	140.2	165.4	197.6	350.7	483.4
	9	10	12	22	30
	22.1	26.0	31.0	55.6	76.0
PLBGS (0.7)	147.3	154.6	207.1	348.1	517.4
	10	11	14	24	36
	26.6	27.6	37.2	62.6	93.3
LBGS (0.7)	180.6	183.4	219.0	435.4	642.8
	10	10	12	24	36
	25.1	25.4	30.5	60.5	89.1
SIMPLE (0.7)	257.3	256.7	307.2	554.4	806.5
	12	12	14	26	38
	27.2	27.4	32.8	59.0	85.9

Table II. Driven Cavity Convergence  
Stretched  $256 \times 256$  grid,  $Re = 1000$

Scheme ( $r_{mom}^{fg}$ )	AR				
	1	5	10	20	40
	(cpu seconds/fg. sweeps/work units)				
BGS (0.6)	197.8	168.6	168.6	199.9	231.3
	12	10	10	12	14
	31.0	26.4	26.4	31.2	36.1
PLBGS (0.5)	260.5	205.7	211.0	268.0	324.6
	18	14	15	19	23
	47.1	36.9	37.9	47.9	57.9
LBGS (0.9)	220.3	220.5	290.2	395.0	604.2
	12	12	16	22	34
	30.5	30.1	39.5	53.7	81.9
SIMPLE (0.7)	311.8	276.2	304.9	305.4	344.8
	14	13	14	14	16
	32.7	28.4	31.9	31.8	36.0

Table III. Developing Channel Convergence  
Uniform  $256 \times 64$  grid,  $AR = 1$

Scheme ( $r_{mom}^{fg}$ )	Re				
	100	400	1000	3200	5000
	(cpu seconds/fg. sweeps/work units)				
BGS (0.7)	53.0	64.5	82.4	152.3	203.4
	12	14	18	34	46
	30.1	36.4	46.9	87.0	116.2
PLBGS (0.8)	49.6	80.0	95.2	159.9	218.0
	12	20	24	40	56
	32.5	52.2	62.3	105.2	142.5
LBGS (0.8)	59.0	78.5	78.3	136.1	173.5
	12	16	16	28	36
	30.7	40.5	40.7	70.7	89.4
SIMPLE (0.7)	83.6	124.0	168.9	324.4	418.7
	16	24	32	64	84
	39.9	58.8	79.9	154.3	199.6

Table IV. Developing Channel Convergence  
Stretched  $256 \times 64$  grid,  $Re = 1000$

Scheme ( $r_{mom}^{fg}$ )	AR				
	1	5	10	20	40
	(cpu seconds/fg. sweeps/work units)				
BGS (0.7)	81.9	139.9	202.1	251.7	268.2
	18	32	46	58	62
	46.9	79.4	113.8	141.1	150.8
PLBGS (0.7)	110.2	117.3	193.8	196.3	230.4
	28	30	50	50	59
	72.4	75.5	124.5	127.2	147.5
LBGS (0.85)	78.4	87.8	105.4	123.9	142.1
	16	18	22	26	30
	40.6	44.7	54.0	63.4	73.3
SIMPLE (0.7)	168.2	142.0	162.0	201.1	261.5
	32	28	32	40	52
	79.9	67.6	76.9	95.1	123.6

Table V. Open Cavity Convergence  
Uniform  $128 \times 128 + 256 \times 128$  grid,  $AR = 1$

Scheme ( $r_{mom}^{fg}$ )	Re				
	100	400	1000	3200	5000
	(cpu seconds/fg. sweeps/work units)				
BGS (0.7)	173.5	203.1	230.7	381.5	573.3
	12	14	16	26	40
	29.7	34.6	39.7	65.2	98.7
PLBGS (0.7)	186.2	237.7	262.2	465.6	616.6
	15	19	20	36	48
	36.4	46.4	51.7	91.2	120.9
LBGS (0.8)	166.5	221.1	256.9	439.4	619.5
	11	14	16	28	40
	25.7	34.4	40.2	68.7	97.5
SIMPLE (0.7)	243.9	319.5	419.6	705.6	1004.2
	14	18	24	40	58
	32.5	42.6	56.1	93.5	133.6

Table VI. Open Cavity Convergence  
Stretched  $128 \times 128 + 256 \times 128$  grid,  $Re = 1000$

Scheme ( $r_{mom}^{fg}$ )	AR				
	1	5	10	20	40
	(cpu seconds/fg. sweeps/work units)				
BGS (0.7)	230.0	175.8	176.7	206.0	233.6
	16	12	12	14	16
	39.6	30.1	30.2	35.2	40.0
PLBGS (0.5)	367.1	263.4	238.3	245.2	298.3
	28	20	18	19	23
	70.6	50.8	46.1	47.2	57.1
LBGS (0.95)	224.4	219.8	222.2	283.0	343.0
	14	14	14	18	22
	35.6	34.5	34.6	44.1	53.5
SIMPLE (0.7)	423.0	289.3	280.8	282.1	379.6
	24	16	16	16	22
	55.9	38.0	37.2	37.2	50.5

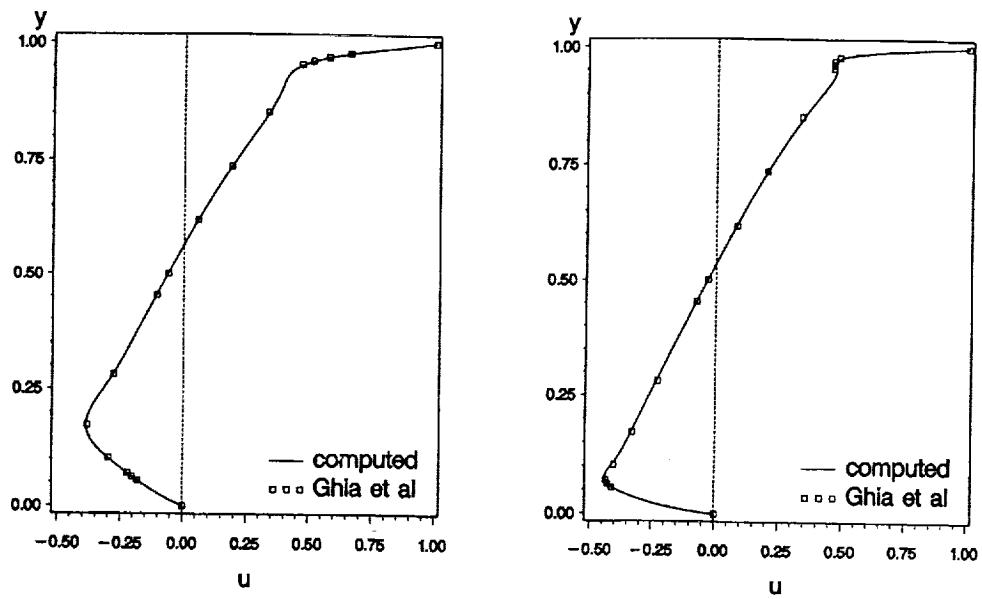


Figure 1. Driven cavity u-velocities on vertical centerline computed on a uniform  $256 \times 256$  grid for  $Re = 1000$  (left) and  $5000$  (right)

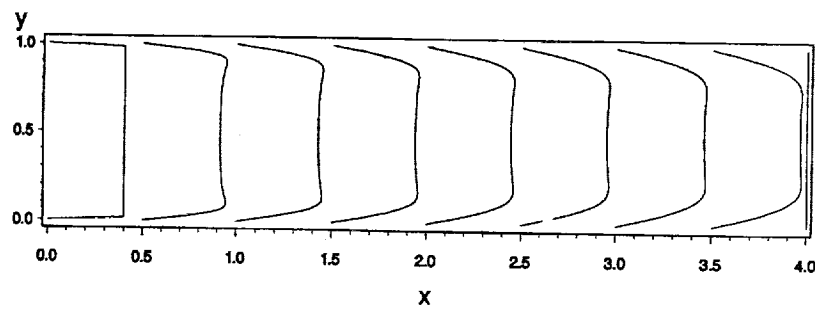


Figure 2. Developing channel u-velocity profiles for  $Re = 1000$



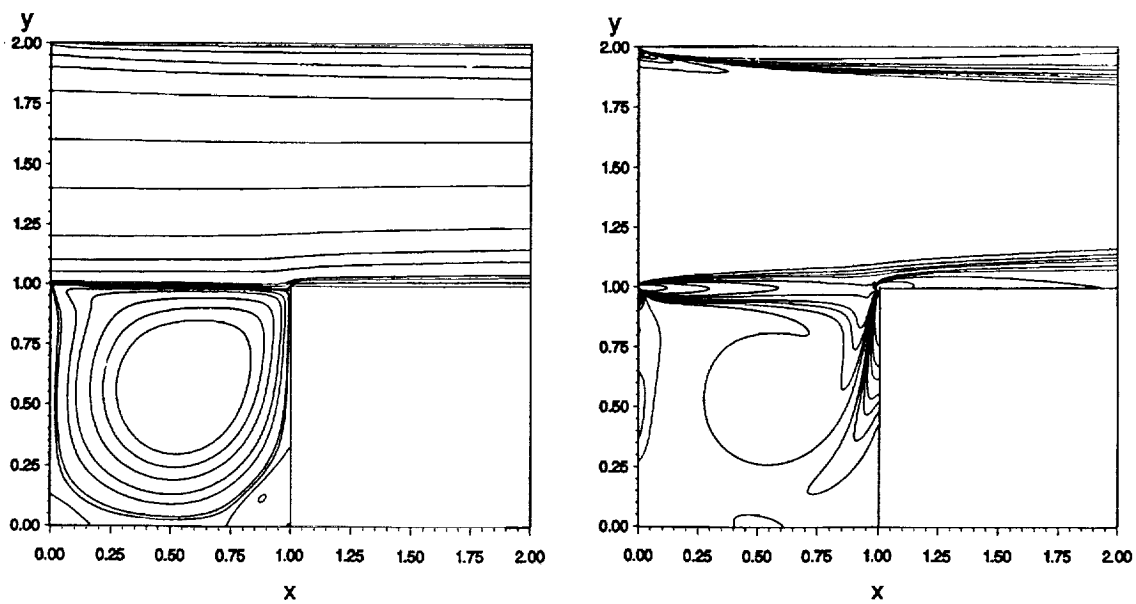


Figure 3. Open cavity stream function (left) and vorticity (right) contours for  $Re = 1000$



# THE MULTIGRID PRECONDITIONED CONJUGATE GRADIENT METHOD

Osamu Tatebe  
Department of Information Science  
University of Tokyo  
Tokyo, JAPAN

N 94-21482  
518-64

197578

p. 14

## SUMMARY

A multigrid preconditioned conjugate gradient method (MGCG method), which uses the multigrid method as a preconditioner of the PCG method, is proposed. The multigrid method has inherent high parallelism and improves convergence of long wavelength components, which is important in iterative methods. By using this method as a preconditioner of the PCG method, an efficient method with high parallelism and fast convergence is obtained. First, it is considered a necessary condition of the multigrid preconditioner in order to satisfy requirements of a preconditioner of the PCG method. Next numerical experiments show a behavior of the MGCG method and that the MGCG method is superior to both the ICCG method and the multigrid method in point of fast convergence and high parallelism. This fast convergence is understood in terms of the eigenvalue analysis of the preconditioned matrix. From this observation of the multigrid preconditioner, it is realized that the MGCG method converges in very few iterations and the multigrid preconditioner is a desirable preconditioner of the conjugate gradient method.

## 1 INTRODUCTION

The typical numerical methods of a king-size system of linear equations, after discretization of the partial differential equations, are the preconditioned conjugate gradient method (PCG method) and the multigrid method [12]. The conjugate gradient method is valued in that it suits to parallel computing and even ill-conditioned problems can be easily solved with the help of a good preconditioning.

This paper considers an efficient preconditioner and proposes a multigrid preconditioned conjugate gradient method (MGCG method) which is the conjugate gradient method with the multigrid method as a preconditioner. The combination of the multigrid method and the conjugate gradient method was already considered. Kettler and Meijerink [7] and Kettler [8] treated the multigrid method as a preconditioner of the conjugate gradient method. However this paper formulates the MGCG method more generally than these and requirements of the multigrid preconditioner are studied. On the other hand, Bank and Douglas [2] treated the conjugate gradient method as a relaxation method of the multigrid method. Braess [3] considered these two combinations and reported that the conjugate gradient method with a multigrid preconditioning is effective for elasticity problems.

We study requirements of the valid multigrid preconditioner and evaluate this preconditioner by some numerical experiments and eigenvalue analysis. Especially, eigenvalue analysis is a more direct and more reasonable criterion than convergence rate, since the number of iterations of the conjugate gradient method until convergence depends on the eigenvalues' distribution of the preconditioned matrix. In Sections 2 and 3, the preconditioned conjugate gradient method and the multigrid method which are the basis of this paper are briefly explained. Section 4 discusses the requirements of the valid two-grid preconditioner for the conjugate gradient method; then in Section 5, it is extended to the requirements of the multigrid preconditioner. In Section 7, numerical experiments show that the MGCG method converges with very few iterations even for ill-conditioned problems. In Section [8], eigenvalue analysis is performed, and it is realized why the MGCG method can easily solve the problem that the ordinary multigrid method itself does not converge rapidly. When the multigrid method is used as a preconditioner of the conjugate gradient method, it becomes quite an effective and desirable preconditioner of the conjugate gradient method.

## 2 THE PRECONDITIONED CONJUGATE GRADIENT METHOD

If a real  $n \times n$  matrix  $A$  is symmetric and positive definite, the solution of a linear system  $Ax = f$  is equivalent to minimization of the quadratic function

$$Q(x) = \frac{1}{2}x^T Ax - f^T x. \quad (1)$$

The conjugate gradient method is one of the minimization methods and uses  $A$ -conjugate vectors as direction vectors which are generated sequentially. Theoretically this method has the striking property that the number of steps until convergence is at most  $n$  steps. This method can be adapted successfully to the parallel and vector computation, since one CG iteration requires only one product of the matrix and the vector, two inner products, tree linked triads, two scalar divides and one scalar compare operation.

Next the preconditioned conjugate gradient method is explained. Let  $U$  be a nonsingular matrix and define  $\tilde{A} = UAU^T$ ; then solve  $\tilde{A}\tilde{x} = \tilde{f}$  using plain conjugate gradient method. Let  $x^0$  be an initial approximate vector; then an initial residual  $r^0$  is  $r^0 = f - Ax^0$ . Let  $M = U^T U$ ,  $\tilde{r}^0 = Mr^0$  and an initial direction vector  $p^0 = \tilde{r}^0$ . The PCG algorithm is described by Program 1.

The matrix  $M$  is a precondition matrix and this paper focuses on this computation. A new proposal is the PCG method exploiting the multigrid method as a preconditioner.

On the other hand, the matrix  $M$  should satisfy some conditions: symmetric and positive definite. Therefore if the matrix of the multigrid method is symmetric and positive definite, it is reasonable to use the multigrid method as a preconditioner of the CG method. In Sections 4 and 5, the conditions of the multigrid preconditioner in order to satisfy the requirements of a preconditioner of the conjugate gradient method are investigated.

```

i = 0;
while ( !convergence ) {
     $\alpha_i = (\tilde{r}_i, r_i) / (p_i, A p_i);$ 
     $x_{i+1} = x_i + \alpha_i p_i;$ 
     $r_{i+1} = r_i - \alpha_i A p_i;$ 
    convergence test;
     $\tilde{r}_{i+1} = M r_{i+1};$  // preconditioning
     $\beta_i = (\tilde{r}_{i+1}, r_{i+1}) / (\tilde{r}_i, r_i);$ 
     $p_{i+1} = \tilde{r}_{i+1} + \beta_i p_i;$ 
    i++;
}

```

Program 1. The PCG iteration

### 3 THE MULTIGRID METHOD

In the iterative methods, the frequency components of the residual are reduced most rapidly on the grid corresponding to them. The multigrid method makes good use of this characteristic and exploits a lot of grids to converge as rapid as possible.

These grids are leveled and numbered from the coarsest grid. This number is called the *level number*. If the multigrid method is applied to the solver of linear equations, the residual is reduced moving it from grid to grid. The basic element of the multigrid method is the *defect correction principle*. The defect correction scheme consists of three processes: *pre-smoothing* process, *coarse grid correction* and *post-smoothing* process. In the smoothing process, various methods, such as ILU, ADI and zebra relaxation, are proposed. One purpose of this research is, however, formation of an efficient method with high parallelism. Thus an iterative method with high parallelism, such as the damped Jacobi method or a multi-color symmetric SOR method (SSOR method), is used as the smoothing method.

An operation of transferring a vector on a finer grid to a vector on a coarser grid is called *restriction*, and an opposite operator is called *prolongation*. A matrix presenting the operation of restriction is written  $r$  in this paper, and prolongation is  $p$ .

In the following section, the equation of grid level  $i$  is described as

$$L_i x_i = f_i$$

and restriction is defined by adjoint of prolongation. That is,

$$r = b p^T,$$

where  $b$ , a scalar constant, is satisfied.

## 4 THE TWO-GRID PRECONDITIONER

This section and the next section examine whether the multigrid method suits a preconditioner of the PCG method. First it is shown that two kinds of two-grid methods, one with pre-smoothing and no post-smoothing and the other with both pre-smoothing and post-smoothing, satisfy the conditions of a preconditioner: the matrix of the two-grid method is symmetric and positive definite. Next it is shown that V-cycle and W-cycle multigrid methods also hold.

A linear equation,  $L_i x_i = f_i$ , is concerned. If  $R$  is a matrix of a relaxant calculation and  $u$  is an approximate vector, one two-grid iteration can be shown by matrix form in Table 1.

$  \begin{aligned}  u &= H^m u + R f && // \text{ pre-smoothing} \\  d &= r (L_i u - f) && // \text{ coarse grid correction} \\  v &= L_{i-1}^{-1} d \\  u &= u - p v \\  u &= H^m u + R f && // \text{ post-smoothing}  \end{aligned}  $
---

Table 1. The two-grid iteration

In this paper the relaxant calculation is an iterative method with high parallelism, and the matrix  $R$  is defined as follows. Let  $L_i$  be an  $n \times n$  nonsingular, symmetric matrix and be split as

$$L_i = P - Q, \quad (2)$$

where  $P$  is a nonsingular matrix and the symmetric part of  $P + Q$  is positive definite. For example, in the case of the point Jacobi method,  $P$  is a diagonal matrix containing diagonal elements of  $L_i$ . Then the  $i$ 'th approximate vector  $u^i$  is updated such as

$$u^{i+1} = P^{-1} Q u^i + P^{-1} f. \quad (3)$$

If an initial approximate vector  $u^0$  is zero vector and  $m$  iterations are done,  $R$  is equal to

$$R = \sum_{i=0}^{m-1} H^i P^{-1}, \quad \text{with } H = P^{-1} Q. \quad (4)$$

$H$  is called an *iterative matrix*.

### 4.1 The two-grid preconditioner with pre-smoothing only

First consider a no post-smoothing case. The matrix of one iteration of Table 1 equals

$$\begin{aligned}
 M &= (I - p L_{i-1}^{-1} r L_i) R + p L_{i-1}^{-1} r \\
 &= R + p L_{i-1}^{-1} r (I - L_i R).
 \end{aligned} \quad (5)$$

Then the following theorem holds.

**Theorem 1** The matrix  $L_{l-1}^{-1}$  is symmetric and positive definite, and  $N = I - L_l R$ . If the matrix  $N$  and  $P$  are symmetric, the matrix  $M$  of Eq. (5) is symmetric in the  $N$ -energy inner product. If the matrix  $N$  is symmetric and nonsingular, the matrix  $P$  is symmetric and  $m$  is even; then the matrix  $M$  is positive definite in the  $N$ -energy inner product, provided that  $N$ -energy inner product  $(x, y)_N = (x, Ny)$ .

*Proof.* Since  $N$  is symmetric,  $(I - L_l R)^T = I - L_l R$ . Therefore

$$I - R^T L_l = I - L_l R.$$

Since  $P$  is symmetric, the matrix  $R$  is also symmetric. Then

$$R L_l = L_l R. \quad (6)$$

And

$$\begin{aligned} (x, My)_N &= x^T N R y + x^T N p L_{l-1}^{-1} r (I - L_l R) y \\ &= x^T (I - L_l R) R y + x^T (I - L_l R) p L_{l-1}^{-1} r (I - L_l R) y. \end{aligned} \quad (7)$$

Besides

$$\begin{aligned} (Mx, y)_N &= x^T M^T N y \\ &= x^T R N y + x^T (I - L_l R) p L_{l-1}^{-1} r N y \\ &= x^T (I - L_l R) R y + x^T (I - L_l R) p L_{l-1}^{-1} r (I - L_l R) y \quad (\text{because of Eq. (6)}) \\ &= (x, My)_N. \end{aligned} \quad (8)$$

Therefore the matrix  $M$  is symmetric in the  $N$ -energy inner product.

Next, it is shown that the matrix  $M$  is the positive definite in the  $N$ -energy inner product. It is equal to  $(x, Mx)_N > 0$ . Then

$$\begin{aligned} N &= I - L_l R \\ &= I - R L_l \\ &= I - \sum_{i=0}^{m-1} (P^{-1} Q)^i P^{-1} (P - Q) \\ &= (P^{-1} Q)^m \\ &= H^m. \end{aligned}$$

Thus

$$\begin{aligned} NM &= (I - L_l R) \{R + p L_{l-1}^{-1} r (I - L_l R)\} \\ &= H^m R + H^m p L_{l-1}^{-1} r H^m. \end{aligned}$$

Since  $P$  is symmetric and nonsingular and  $L_l$  is symmetric and positive definite, then  $H = P^{-1} Q = I - P^{-1} L_l$  has real eigenvalues. Hence if  $m$  is even,  $H^m$  is positive definite. If  $P + Q$  is positive definite and  $m$  is even, then  $R$  is positive definite (see [11]). Therefore  $H^m R$  is positive

definite. Since  $H^m$  is symmetric and  $pL_{l-1}^{-1}r$  is semi-positive definite,  $H^m pL_{l-1}^{-1}r H^m$  is semi-positive definite. Thus  $NM$  is positive definite.  $\square$

The iterative method which holds the assumption of Theorem 1 is the damped Jacobi method. From this theorem, the two-grid preconditioner with the damped Jacobi method as a relaxant calculation fills the conditions of the preconditioner of the CG method which uses the  $N$ -energy inner product instead of the usual inner product.

#### 4.2 The two-grid preconditioner with both pre-smoothing and post-smoothing

Next consider the two-grid iteration with both pre-smoothing and post-smoothing. Suppose the pre-smoothing and the post-smoothing are the same method. Then the matrix of one two-grid iteration in Table 1 equals

$$\begin{aligned} M &= H^m \{ (I - pL_{l-1}^{-1}rL_l)R + pL_{l-1}^{-1}r \} + R \\ &= H^m R + R + H^m pL_{l-1}^{-1}r (I - L_l R). \end{aligned} \quad (9)$$

However since  $P$  and  $Q$  are symmetric,

$$I - L_l R = (QP^{-1})^m = (H^T)^m.$$

Therefore the matrix  $M$  of Eq. (9) is rewritten as

$$M = H^m R + R + H^m pL_{l-1}^{-1}r (H^T)^m. \quad (10)$$

Then the following theorem is satisfied.

**Theorem 2** *The matrix  $L_{l-1}^{-1}$  is symmetric and positive definite. If the matrix  $P$  is symmetric, the matrix  $M$  of Eq. (10) is symmetric and positive definite.*

*Proof.* Since the matrix  $P$  is symmetric, the matrix  $R$  is also symmetric. Thus

$$M^T = R(H^T)^m + R + H^m pL_{l-1}^{-1}r (H^T)^m.$$

Now

$$\begin{aligned} H^m R &= H^m \sum_{i=0}^{m-1} H^i P^{-1}. \\ R(H^T)^m &= \sum_{i=0}^{m-1} H^i P^{-1} (H^T)^m. \end{aligned}$$

Moreover since  $P$  is symmetric and  $H = P^{-1}Q$ , then  $P^{-1}H^T = HP^{-1}$ . Therefore

$$H^m R = R(H^T)^m.$$



After all, the matrix  $M$  is symmetric. Next show that the matrix  $M$  is positive definite.

$$\begin{aligned} M &= H^m R + R + H^m p L_{l-1}^{-1} r (H^T)^m \\ &= \sum_{i=0}^{2m-1} H^i P^{-1} + H^m p L_{l-1}^{-1} r (H^T)^m. \end{aligned} \quad (11)$$

Since the first term of right hand expression  $\sum_{i=0}^{2m-1} H^i P^{-1}$  of Eq. (11) is the matrix after  $2m$  times iteration, it is positive definite if  $P + Q$  is positive definite. Since  $L_{l-1}^{-1}$  is positive definite,  $H^m p L_{l-1}^{-1} r (H^T)^m$  is semi-positive definite. Therefore  $M$  is positive definite.  $\square$

The iterative methods which hold the assumption of Theorem 2 are the damped Jacobi method, Red-Black Symmetric Gauss-Seidel method (RB-SGS method), multi-color SSOR method, ADI method and so on. From this theorem, the two-grid preconditioner with one of these iterative methods as a relaxant calculation fulfills the conditions of the preconditioner of the CG method.

## 5 THE MULTIGRID PRECONDITIONER

In the previous section the possibility of two kinds of two-grid preconditioners is considered. In the following, only the latter two-grid preconditioner, with both pre-smoothing and post-smoothing, is discussed. However the same discussion can be applied to the former two-grid preconditioner. In this section, extension to the multigrid preconditioner is argued. The following theorem holds.

**Theorem 3** *If assumptions of Theorem 1 and 2 are satisfied, all  $MG(m, n)$  methods ( $m, n \geq 1$ ) satisfy conditions of a preconditioner of the CG method, where  $m$  is a multigrid cycle and  $n$  is the number of iterations of the smoothing method.*

*Proof.* The matrix  $M_l$  of the V-cycle multigrid method can be defined as

$$\begin{aligned} M_0 &= L_0^{-1} \text{ or } R_0 \\ M_i &= H^m R_i + R_i + H^m p M_{i-1} r (H^T)^m. \quad (i \geq 1) \end{aligned}$$

$M_0$  is symmetric and positive definite. If  $M_i$  is symmetric and positive definite,  $M_{i+1}$  is also symmetric and positive definite because of Theorem 2. By mathematical induction, every  $M_i$  ( $i \geq 0$ ) is symmetric and positive definite. Therefore the V-cycle multigrid method can be used as a preconditioner.

Next the W-cycle multigrid method is considered. If the matrix  $N_l^{(n)}$  is the multigrid method with  $n$  recursive calls of the multigrid method on level number  $l-1$  as the solution on the coarse grid,  $N_l^{(n)}$  is defined as

$$\begin{aligned} N_0^{(n)} &= L_0^{-1} \text{ or } R_0 \\ N_i^{(n)} &= \sum_{i=0}^{n-1} H_{\text{mg}}^i \{ H^m R_i + R_i + H^m p N_{i-1}^{(n)} r (H^T)^m \}, \quad (i \geq 1) \end{aligned}$$

where  $H_{mg}^i = H^{2m} - H^m p N_{i-1}^{(n)} r L_i H^m$ .  $N_0^{(n)}$  is symmetric and positive definite. If  $N_{i-1}^{(n)}$  is symmetric and positive definite,  $H^m R_i + R_i + H^m p N_{i-1}^{(n)} r (H^T)^m$  is symmetric and positive definite by Theorem 2. Thus  $N_i^{(n)}$  is also symmetric. And because of  $\rho(H_{mg}) < 1$  by [?],  $N_i^{(n)}$  is positive definite. The W-cycle multigrid method is the case of  $n = 2$ . Therefore the W-cycle multigrid method and all MG( $n, m$ ) ( $m, n \geq 1$ ) satisfy the conditions of the preconditioner.  $\square$

## 6 THE MGCG METHOD

In the previous section, the multigrid preconditioner which is valid for a preconditioner of the CG method is considered. When only pre-smoothing is performed, the multigrid preconditioner with an even number of iterations of the damped Jacobi smoothing can become a preconditioner of the conjugate gradient method with the  $N$ -energy inner product instead of the usual inner product. When both pre-smoothing and post-smoothing are performed, the multigrid preconditioner with RB-SSOR smoothing, ADI method and so on, fulfills the requirements of a preconditioner of the conjugate gradient method. Thus the multigrid preconditioned conjugate gradient method (MGCG method) is mathematically valid. There are several variations of this preconditioner. If  $m$  is a cycle of the multigrid method,  $l$  is a relaxant method,  $n$  is the number of iterations of the relaxant method and  $g$  is the number of grids, the MGCG method is specified as MGCG( $l, m, n, g$ ). But  $g$  is an optional parameter and if this parameter is omitted, all available grids are used. For example, MGCG(RB, 1, 2) is the MGCG method of the V-cycle multigrid preconditioner with two iterations of the Red-Black SSOR smoothing.

## 7 NUMERICAL EXPERIMENTS

### 7.1 Problems

A two-dimensional Poisson equation with Dirichlet boundary condition:

$$-\nabla(k\nabla u) = f \quad \text{in } \Omega = [0, 1] \times [0, 1]$$

$$\text{with } u = g \quad \text{on } \partial\Omega,$$

where  $k$  is a real function, is considered. The equation is defined by a diffusion constant  $k$ , a source term  $f$  and a boundary condition  $g$ . Numerical experiments are performed in the following two conditions.

**Problem 1** Diffusion constant is uniform and source term is equal to 0. Boundary condition is  $g = 0$  except  $y = 1$  and  $g = 3x(1 - x)$  on  $y = 1$ .

**Problem 2** Diffusion constant and source term are depicted by Figs. 1 and 2. Boundary condition  $g$  is always equal to 0.

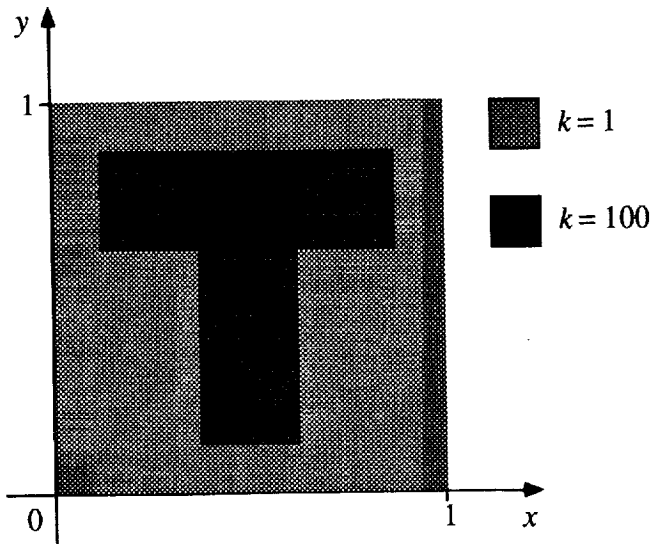


Figure 1. Diffusion constant of problem 2

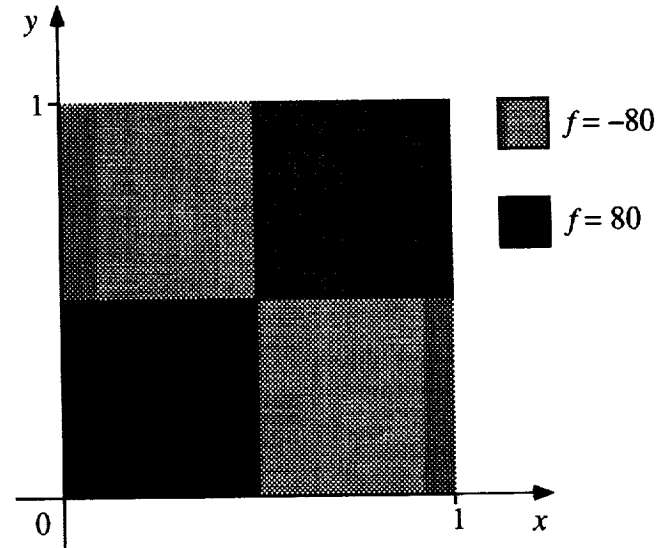


Figure 2. Source term of problem 2

Problem 1 is a simple case, and the multigrid method is expected to converge efficiently. The multigrid preconditioner is also expected to be efficient. Problem 2 has a non-uniform diffusion constant and the area with a large diffusion constant looks like a letter 'T'; therefore it has a rich distribution of eigenvalues of the problem matrix, which is investigated in the next section. Moreover since a source term is complex, it does not happen that specific iterative methods, such as ICCG method and MICCG method, accidentally converge very rapidly.

These problems are discretized to three kinds of meshes:  $64 \times 64$ ,  $128 \times 128$  and  $256 \times 256$ , by the finite element method. These coefficient matrices become symmetric, positive definite and block tridiagonal.

## 7.2 Solutions

In numerical experiments, three methods: the MGCG(RB, 1, 2) method, the ICCG(1, 2) method and the MG(1, 2) method, are compared. The ICCG(1, 2) method is the PCG method with the incomplete Cholesky decomposition having an additional one line to the original problem sparse matrix. The MG(1, 2) method is the identical method to the multigrid preconditioner of the MGCG(RB, 1, 2) method.

Numerical experiments are performed on the HP9000/720 and the program is written by C++ with original vector and matrix classes.

### 7.3 Convergence of the MGCG method

size	MGCG( $RB, 1, 2$ )		MGCG( $RB, 1, 4$ )		ICCG( $1, 2$ )		MG( $1, 2$ )	
	# of iter.	time(sec.)	iter.	time	iter.	time	iter.	time
$63^2$	5	0.56	4	0.61	38	1.19	7	0.65
$127^2$	5	3.16	5	4.58	72	10.88	7	4.05
$255^2$	5	15.8	5	23.7	134	89.5	7	20.2

(HP9000/720; C++)

Table 2. Problem 1

size	MGCG( $RB, 1, 2$ )		MGCG( $RB, 1, 4$ )		ICCG( $1, 2$ )		MG( $1, 2$ )	
	# of iter.	time(sec.)	iter.	time	iter.	time	iter.	time
$63^2$	9	0.98	8	1.19	53	1.65	150	13.4
$127^2$	9	5.54	8	7.21	103	15.49	135	75.3
$255^2$	9	27.8	8	37.4	200	133.0	122	341.5

(HP9000/720; C++)

Table 3. Problem 2

Tables 2 and 3 are results of these numerical experiments. The number of iterations and the time of each method until convergence are measured. The number of iterations of the MGCG method and the ICCG method is that of CG iterations and the number of iterations of the multigrid method is that of V-cycle iterations. From results of the two problems, the following points are notable:

- The MGCG method converges with very few iterations.
- The number of iterations of the MGCG method does not increase when a mesh size is larger.
- Even for complex problems, such as problem 2, the MGCG method converges fast.

The first item is discussed by an eigenvalue analysis in the next section. From the second item, the MGCG method is advantageous over the ICCG method even as large as the mesh size is. It is a principle of the multigrid method that the number of iterations does not depend upon the mesh size. If the problem is simple such as problem 1, the multigrid method converges very fast; however, in complex problems, such as problem 2, it converges very slowly. To avoid this, the multigrid method should have the stronger relaxation method, but the stronger relaxation method has poor parallelism. Moreover in problem 2, it is considered that the locking effect [?] has occurred. From the third item, the MGCG method is also superior to the multigrid method as a result of stably fast convergence and high parallelism.

## 8 EIGENVALUE ANALYSIS

In order to study the efficiency of the multigrid preconditioner, the eigenvalue distribution of a coefficient matrix after preconditioning is examined. The number of iterations of the conjugate gradient method until convergence depends upon an initial vector, a distribution of eigenvalues of a coefficient matrix and a right-hand term, but due to a good initial vector and a simple right-hand term, the conjugate gradient method happens to converge fast unreasonably, so the eigenvalue distribution is investigated. The problem is the same problem in Section 7 and the area is discretized to the mesh of  $16 \times 16$  by the finite element method. The condition number of this coefficient matrix is 5282.6.

A matrix after the multigrid preconditioning is calculated as follows. The matrix  $M$  of Eq. (5) or (10) is Cholesky decomposed as  $M = U^T U$ , then eigenvalues of the matrix  $U L_1 U^T$  is investigated. On the other hand the matrix using the ICCG method is calculated as follows. The matrix  $L_1$  is incomplete Cholesky decomposed as  $L_1 = S^T S - T$ , and the general eigenvalue problem  $L_1 \mathbf{x} = \lambda S^T S \mathbf{x}$  is solved in order to examine eigenvalues after preconditioning.

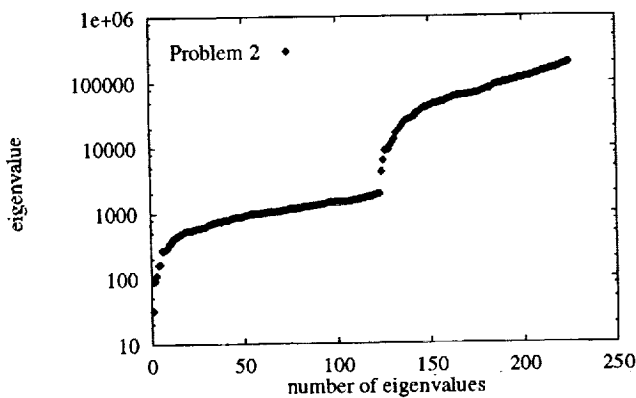


Figure 3. Eigenvalue distribution of a problem matrix

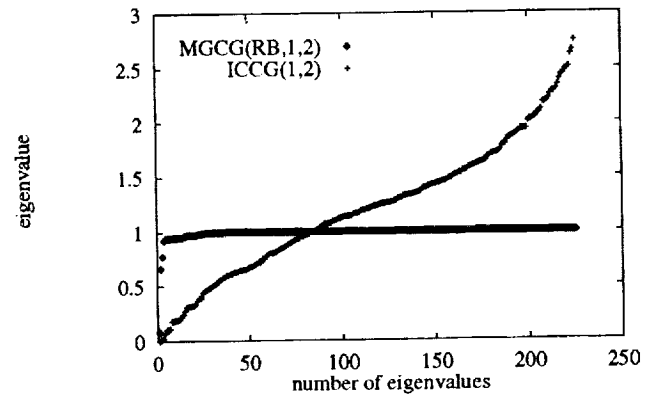


Figure 4. Eigenvalue distribution after preconditioning

The eigenvalue distribution of the problem matrix is shown in Fig. ???. The horizontal  $x$  axis is the order of the eigenvalues and the vertical  $y$  axis values are the eigenvalues. This vertical axis is in a log scale. The eigenvalue distribution of the matrix after preconditioning is shown in Fig. ???. This vertical axis is in a linear scale. In order to compare, preconditioning is carried out in both the multigrid method and the incomplete Cholesky decomposition.

The eigenvalue distribution of the multigrid preconditioner is effective for the conjugate gradient method as the following points:

1. Almost all eigenvalues are clustered around 1 and a few eigenvalues are scattered between 1 and 0.
2. The smallest eigenvalue is larger than the ICCG method.
3. Condition number is decreased.

The first item is no problem for the conjugate gradient method. All of these characteristics are desirable to accelerate the convergence of the conjugate gradient method. In problem 1, there are no scattered eigenvalues. So the multigrid method converges efficiently, however in problem 2, the scattered eigenvalues prevent the ordinary multigrid method from converging rapidly. Therefore using the multigrid method as a preconditioner of the conjugate gradient method is quite important.

## 9 CONCLUSION

This paper investigates the conjugate gradient method with a multigrid preconditioner (MGCG method). Necessary conditions of a preconditioning matrix of the conjugate gradient method are symmetric and positive definite. First two kinds of two-grid preconditioners are considered and conditions of both preconditioners are given in order to satisfy necessary conditions of a preconditioner. Secondly extension to the multigrid preconditioner is carried out and conditions for a valid multigrid preconditioner are also given. Thirdly numerical experiments are performed and the MGCG method has faster convergence and a more effective method than both the ICCG method and the multigrid method. Finally eigenvalue analysis is performed in order to verify the effect of the multigrid preconditioner. It concludes that the multigrid preconditioner is an excellent preconditioner and it improves the number of the CG iterations remarkably. Consequently the MGCG method has the following properties:

- The number of iterations does not increase even when a mesh is finer.
- Even in the case that the problem is ill-conditioned, the MGCG method is effective.
- The distribution of the eigenvalues of the matrix after preconditioning is suited to the conjugate gradient method.
- The MGCG method has high parallelism.

The multigrid method roughly solves any problems, since almost all eigenvalues of Section ?? are clustered around the unity, but a few scattered eigenvalues prevent fast convergence. The conjugate gradient method hides the defect of the multigrid method. Therefore the MGCG method becomes an efficient method. Parallelization of the MGCG method and implementation on the multicomputers are beyond the scope of this paper, so this facility is no more mentioned. However since the MGCG method has high parallelism and fast convergence, this method is a very promising method as the solution of a large-scaled sparse, symmetric and positive definite matrix.

## ACKNOWLEDGMENT

I appreciate the valuable comments of Prof. Y. Oyanagi at the University of Tokyo and Prof. K. Murata at Kanagawa University. I would like to thank Prof. M. Mori and Prof. M. Sugihara at the University of Tokyo for useful comments and discussions.

## REFERENCES

- [1] Adams, L. M., *Iterative Algorithms for Large Sparse Linear Systems on Parallel Computers*. PhD thesis, University of Virginia, November 1982.
- [2] Bank, R. E. and C. C. Douglas, "Sharp Estimates for Multigrid Rates of Convergence with General Smoothing and Acceleration," *SIAM J. Numer. Anal.*, vol. 22, no. 4, pp. 617–633, August 1985.
- [3] Braess, D., "On the Combination of the Multigrid Method and Conjugate Gradients," in *Multigrid Methods II* (W. Hackbusch and U. Trottenberg, eds.), vol. 1228 of *Lecture Notes in Mathematics*, pp. 52–64, Springer-Verlag, 1986.
- [4] Hackbusch, W., "Convergence of Multi-Grid Iterations Applied to Difference Equations," *Mathematics of Computation*, vol. 34, no. 150, pp. 425–440, 1980.
- [5] Hackbusch, W., "Multi-grid Convergence Theory," in *Multigrid Methods* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 177–219, Springer-Verlag, 1982.
- [6] Hackbusch, W., *"Multi-Grid Methods and Applications"*. Springer-Verlag, 1985.
- [7] Kettler, R. and J. A. Meijerink, "A multigrid method and a combined multigrid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains," Tech. Rep. 604, Shell Oil Company, 1981.
- [8] Kettler, R., "Analysis and Comparison of Relaxation Schemes in Robust Multigrid and Preconditioned Conjugate Gradient Methods," in *Multigrid Methods* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 502–534, Springer-Verlag, 1982.
- [9] Maitre, J. F. and F. Musy, "Multigrid Methods: Convergence Theory in a Variational Framework," *SIAM J. Numer. Anal.*, vol. 21, pp. 657–671, 1984.
- [10] Moji, J. J., *"Parallel Algorithms and Matrix Computation"*. Oxford University Press, 1988.
- [11] Ortega, J. M., *"Introduction to Parallel and Vector Solution of Linear Systems"*. Plenum Press, 1988.
- [12] Stüben, K. and U. Trottenberg, "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications," in *Multigrid Methods* (W. Hackbusch and U. Trottenberg, eds.), vol. 960 of *Lecture Notes in Mathematics*, pp. 1–176, Springer-Verlag, 1982.

- [13] Wesseling, P., "Theoretical and practical aspects of a multigrid method," *SIAM Journal on Scientific and Statistical Computing*, vol. 3, pp. 387–407, 1982.



# MAPPING ROBUST PARALLEL MULTIGRID ALGORITHMS TO SCALABLE MEMORY ARCHITECTURES<sup>1</sup>

Andrea Overman  
NASA Langley Research Center  
Hampton, VA 23681-0001

John Van Rosendale  
ICASE  
NASA Langley Research Center  
Hampton, VA 23681-0001

N94-21483

519-61

197579

p. 13

## SUMMARY

The convergence rate of standard multigrid algorithms degenerates on problems with stretched grids or anisotropic operators. The usual cure for this is the use of line or plane relaxation. However, multigrid algorithms based on line and plane relaxation have limited and awkward parallelism and are quite difficult to map effectively to highly parallel architectures. Newer multigrid algorithms that overcome anisotropy through the use of multiple coarse grids rather than line relaxation are better suited to massively parallel architectures because they require only simple point-relaxation smoothers.

In this paper, we look at the parallel implementation of a V-cycle multiple semicoarsened grid (MSG) algorithm on distributed-memory architectures such as the Intel iPSC/860 and Paragon computers. The MSG algorithms provide two levels of parallelism: parallelism within the relaxation or interpolation on each grid and across the grids on each multigrid level. Both levels of parallelism must be exploited to map these algorithms effectively to parallel architectures. This paper describes a mapping of an MSG algorithm to distributed-memory architectures that demonstrates how both levels of parallelism can be exploited. The result is a robust and effective multigrid algorithm for distributed-memory machines.

---

<sup>1</sup>This research was supported by the National Aeronautics and Space Administration under NASA contract nos. NAS1-19480 and NAS1-18605 while the second author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

## INTRODUCTION

The convergence rate of standard multigrid algorithms degenerates on problems that have anisotropic discrete operators. Such operators arise when the continuous operator is anisotropic or when the discretization is based on highly stretched grids. Although a number of effective cures exist for this difficulty, the best sequential algorithms (based on line or plane relaxation) do not appear to be viable on emerging, massively parallel architectures. Thus, newer algorithms, which achieve robustness through the use of multiple coarse grids rather than line or plane relaxation and require only point-relaxation smoothers, are an attractive alternative.

The problems with line- and plane-relaxation algorithms on parallel architectures have only recently become apparent. Although the tridiagonal systems involved can be solved in parallel by substructured elimination, for example, this approach approximately doubles their computational cost. In addition, a more subtle difficulty exists. The fastest robust sequential algorithms combine line- and plane-relaxation algorithms with semicoarsening. Unfortunately, this means that the size of the line and plane solutions required on coarse grids is the same as on the fine grid. For example, an  $n^2$ -point grid in two dimensions with a parallel tridiagonal solver and  $O(n^2)$  processors gives a theoretical upper bound on parallel efficiency of only  $O(1/\log^2 n)$ . Thus, the fact that parallel implementations of such algorithms have proven problematic is not surprising (refs. 1,2,3).

An alternate approach to robustness, based on using multiple grids on every coarse multigrid level, is newer and relatively untried. Through the use of appropriate coarse grids, one can obtain point-relaxation algorithms as robust as line- and plane-relaxation algorithms (refs. 4,5,6,7). However, because of the large number of coarse grids required, these algorithms are not quite competitive with line- and plane-relaxation algorithms on sequential machines. On parallel architectures, the opposite is true (refs. 5,8,9) because the increased parallelism due to the multiple coarse grids is an attractive bonus. In particular, Douglas' method is robust and can be mapped effectively to parallel architectures (ref. 5); Horton (ref. 9) has looked recently at the mapping of Hackbusch's Frequency Decomposition method (ref. 6) to parallel architectures.

In this paper, we study the mapping of the multiple semicoarsened grid (MSG) algorithm, a variant of Mulder's multiple coarse-grid algorithm (ref. 10), to highly parallel architectures. The MSG algorithm (ref. 7) is relatively robust and at the same time provides ample parallelism for current parallel architectures. We take as our model problem the symmetric, positive-definite Helmholtz equation

$$a u_{xx} + b u_{yy} + c u_{zz} - d u = f$$

with  $a, b, c, d \geq 0$  and focus on the mapping issues involved in implementing this algorithm on distributed-memory architectures such as the Intel iPSC/860 and Paragon.

This paper is organized as follows. We begin with a description of the MSG algorithm in the next section, which is followed by a discussion of observed convergence rates. Our parallel implementation is then described. We present the experimental results, and, finally, conclusions are given.

## ALGORITHM DESIGN

We first need to describe the MSG algorithm. For notational simplicity, assume that the domain of the model problem is the unit square in two dimensions and that this problem is to be solved on an  $n \times n$  uniform grid as

$$\Omega_h = \{(ih, jh) \mid i = 0, 1, \dots, n; j = 0, 1, \dots, n\}$$

with  $h = 1/n$ . Define the coarser grids  $\Omega^{l,m}$ , which are obtained by successive semicoarsening of  $\Omega_h$   $l$  times in the  $x$ -direction and  $m$  times in the  $y$ -direction. Thus,  $\Omega^{l,m}$  has  $(n+1)/2^l$  grid points in the  $x$ -direction and  $(n+1)/2^m$  grid points in the  $y$ -direction.

Notice that the notation does not distinguish between a grid obtained by semicoarsening first in the  $y$ -direction and then in the  $x$ -direction and a grid obtained by semicoarsening first in the  $x$ -direction and then in the  $y$ -direction. Either path leads to a grid of the same shape and size. As shown by Mulder (ref. 10), such equivalent grids must be combined in order to construct reasonable algorithms in three or more dimensions.

Figure 1 shows the interrelations between the various grids for a two-dimensional problem with an  $8 \times 8$  fine grid. With coarse grids combined as in this diagram, for a  $16 \times 16$  problem one would have only 16 grids altogether; without combining, the full binary tree of grids would contain 69 grids.

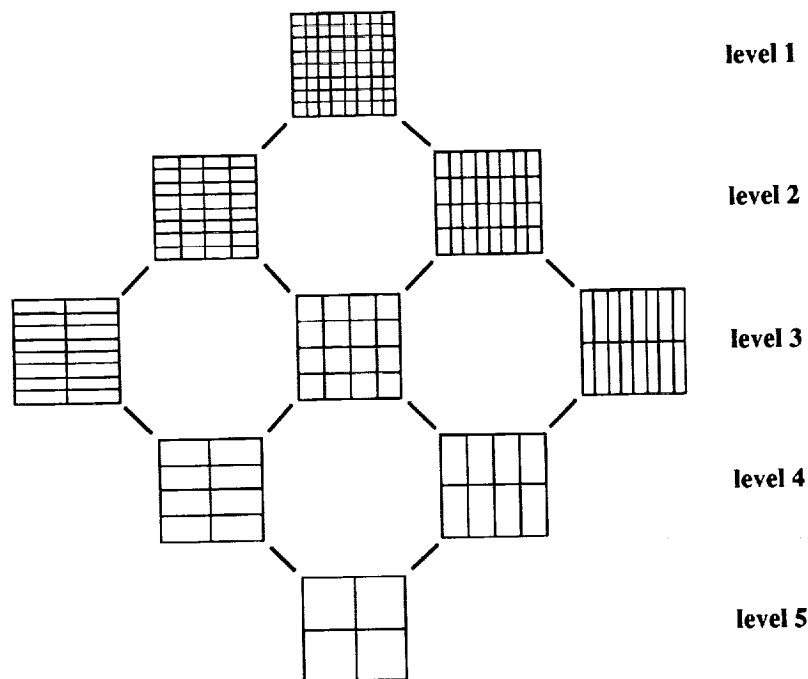


Figure 1. Semicoarsening of an  $8 \times 8$  grid.

Given this family of grids, one can construct a V-cycle correction scheme analogous to the standard full-coarsening multigrid algorithm. One-dimensional linear interpolation provides a

natural prolongation operator; its adjoint gives the "full weighting" restriction operator. These choices, together with any reasonable smoother, yield a multigrid algorithm. However, the resulting algorithm is not robust.

The problem with this simple correction scheme is explained. If the prolongation is scaled so that the full correction is obtained from the modes that are oscillatory in  $x$  but not  $y$  and conversely, then the result is double the required correction of the smoothest components that belong to both coarse grids, and divergence results. On the other hand, if the prolongation is scaled to get the proper correction of the smooth components, then some of the oscillatory components are undercorrected, and robustness is lost.

The resolution of this problem is to filter either the residuals that are being restricted or the corrections that are being prolonged to achieve a convergent V-cycle for the model problem

$$a u_{xx} + b u_{yy} = f$$

where the convergence rate is independent of  $a, b > 0$ . This filtering can be performed in several ways.

Let  $v^{l,m}$  denote the correction on grid  $\Omega^{l,m}$ . Also let  $R_x$  and  $R_y$  denote restriction in the  $x$ - and  $y$ -directions, and, similarly, let  $P_x$  and  $P_y$  denote prolongations. The first effective solution to this problem was given by Mulder (ref. 10). Mulder forms the fine-grid correction

$$P_x v^{1,0} + P_x R_x P_y v^{0,1}$$

given solutions  $v^{0,1}$  and  $v^{1,0}$  on the second level and similar solutions for coarser levels. One can think of the operator  $P_x R_x$  here as a high-pass filter that filters out the excess correction for the smooth modes common to both coarse grids.

In recent work, Naik and Van Rosendale have been looking at the analogous scheme with the correction

$$(1 + 1/2 P_y R_y) P_x v^{1,0} + (1 + 1/2 P_x R_x) P_y v^{0,1}$$

which can be thought of as a symmetric version of Mulder's scheme. A V-cycle proof for one variant of this scheme appears to be possible.

A third way of making the correction is to compute a scalar-valued function  $\alpha(x, y)$ , which depends on the strength of the discrete differential operator in each coordinate direction. Then, with a properly chosen  $\alpha$ , one uses the correction

$$\alpha(x, y) P_x v^{1,0} + [1 - \alpha(x, y)] P_y v^{0,1}$$

A V-cycle convergence proof for this scheme, at least for constant coefficient problems, was given in ref. 7. This reference also provides details on the computation of  $\alpha(x, y)$ .

On sequential machines, any of these schemes is effective and robust. Mulder's scheme and its symmetrized version eliminate the necessity of choosing  $\alpha$ ; the extra work involved in their interpolations is trivial. However, because the communication required for interpolation is awkward

and expensive on parallel architectures, we used the alpha-switch algorithm here, which reduces the complexity of the interpolations. It is as robust as the alternatives and simpler to implement.

Generalization of this alpha-switch algorithm to the three dimensions is straightforward. Instead of simply computing  $\alpha(x, y)$ , one computes  $\alpha(x, y, z)$  and  $\beta(x, y, z)$  and then uses the three weights

$$\alpha(x, y, z) \quad \beta(x, y, z) \quad 1 - \alpha(x, y, z) - \beta(x, y, z)$$

From the point of view of parallel architectures, computation of the switching factors  $\alpha$  and  $\beta$  is analogous to a Jacobi sweep, which needs to be done only once at the beginning of the computation.

## OBSERVED CONVERGENCE RATES

Experimentally, the MSG algorithm converges extremely well for the model problem

$$a u_{xx} + b u_{yy} + c u_{zz} - d u = f$$

where the convergence rate is independent of  $a, b, c, d \geq 0$  and uniform mesh size. Alternatively, MSG can be used for stretched grids, as shown in Table 1. The results given are observed convergence rates for Poisson's equation with Dirichlet boundary conditions and a random initial guess. Slow variation in the coefficients  $a, b, c$  or in mesh spacing have a similar impact on convergence. The Helmholtz term  $d \geq 0$  can improve convergence on coarse grids, but is largely irrelevant. All of the above information applies only to problems with smooth coefficients. Special algorithms are required for problems with severe coefficient jumps (refs. 11,3). The discretization used throughout our experiments was a symmetric seven-point finite-difference stencil, with the smoothing done by three red-black successive over-relaxation (SOR) sweeps on every grid.

The problem with this algorithm on sequential machines is the large number of grids required and the resulting high cost per V-cycle. With the usual coarsening by a factor of 2 (as shown in Table 1), the total storage for all grids in three dimensions is eight times that of the finest grid. Thus, the work per V-cycle is also eight times the work on the finest grid, which does not include the cost of the interpolations.

A more attractive sequential algorithm can be made by changing the coarsening factor. In any semicoarsening algorithm, one has fewer Fourier modes to reduce than in full-coarsening algorithms; thus, one can afford to coarsen the grids faster.

If we use coarsening by a factor of 4, for example<sup>2</sup>, then the total storage becomes

$$(1 + 1/4 + 1/16 + \dots)^3 = 64/27$$

times that on the finest grid. Thus, the total work is about  $2\frac{1}{3}$  times that on the finest grid.

---

<sup>2</sup>The red-black SOR smoother used yields poor convergence rates for odd coarsening factors. Thus, the reasonable choices for the coarsening factor are 2 and 4 because either 6 or 8 would make the space of "oscillatory" functions (which must be effectively reduced by the smoother) too large.

Table 1. Convergence Rates of MSG on Various Grids With Factor-of-2 Coarsening

	$8 \times 8 \times 8$	$16 \times 16 \times 16$	$32 \times 32 \times 32$
Uniform Grids			
$dx = 1000, dy = dz = 1$	0.04	0.06	0.07
$dx = 10, dy = dz = 1$	0.04	0.06	0.08
$dx = 0.1, dy = dz = 1$	0.02	0.05	0.07
$dx = 0.001, dy = dz = 1$	0.03	0.07	0.08
Chebyshev Grids			
Chebyshev in $x$	0.04	0.06	0.11
Chebyshev in $x, y$	0.04	0.04	0.12
Chebyshev in $x, y, z$	0.03	0.04	0.15

Table 2 gives the observed convergence rates for the same problems as in Table 1; however, factor-of-4 coarsening was used. Although the convergence rates in Table 2 are poorer than in Table 1, the reduced computational cost per V-cycle more than compensates for this. Three V-cycles of the algorithm can be accomplished with factor-of-4 coarsening for less than the cost of one V-cycle with a factor-of-2 coarsening. With the  $32^3$  grid, because  $0.3^3 = 0.027$ , the three V-cycles with a factor-of-4 coarsening are more effective than one V-cycle with a factor-of-2 coarsening.

Massively parallel architectures that have hundreds or thousands of processors might change these considerations and increase the effectiveness of the algorithm with a factor-of-2 coarsening because it provides more parallelism on coarse grids. However, because the algorithm with a factor-of-4 coarsening seemed to provide ample parallelism and the memory per processor is limited on the Intel iPSC/860, we used a factor-of-4 coarsening in our code.

In addition to the use of a factor-of-2 coarsening, the parallelism can be further increased by use of concurrent iteration on all grid levels (refs. 12,13). This form of MSG is particularly attractive on SIMD machines, where the mapping strategies needed for the V-cycle algorithm are prohibitively complex. In joint research with J. Dendy, this alternative is currently being explored for problems with severe coefficient jumps. However, while the concurrent iteration version of MSG maps very nicely to SIMD machines (ref. 7), its convergence rate is in the range of 0.5–0.6, even with a factor-of-2 coarsening. Thus, one trades numerical performance for massive parallelism.

Table 2. Convergence Rates of MSG on Various Grids With Factor-of-4 Coarsening

	$8 \times 8 \times 8$	$16 \times 16 \times 16$	$32 \times 32 \times 32$
Uniform Grids			
$dx = 1000, dy = dz = 1$	0.21	0.20	0.23
$dx = 10, dy = dz = 1$	0.21	0.20	0.24
$dx = 0.1, dy = dz = 1$	0.11	0.13	0.18
$dx = 0.001, dy = dz = 1$	0.11	0.15	0.14
Chebyshev Grids			
Chebyshev in $x$	0.19	0.18	0.26
Chebyshev in $x, y$	0.11	0.14	0.25
Chebyshev in $x, y, z$	0.05	0.19	0.26

## MAPPING MSG TO SCALABLE ARCHITECTURES

The V-cycle MSG algorithm achieves fast convergence and contains substantial parallelism, although exploitation of this parallelism is fairly awkward. This awkwardness is in contrast to the standard (full-coarsening) multigrid, where parallel implementation is straightforward. For the MSG case, we designed a program to compute efficient mappings of the algorithm to a distributed-memory architecture. The computed mappings were then implemented with the PARTI<sup>3</sup> runtime primitives developed at ICASE (refs. 14,15). Although this implementation was complex, without PARTI or analogous tools, implementation would have been prohibitively difficult. In this section, we describe our implementation strategy.

### Load Balancing

Two basic issues must be addressed in mapping the V-cycle MSG algorithm to distributed-memory architectures: processors must be assigned to the grids on each level and each grid must be partitioned across the processors assigned to it. Because a large number of possible mapping strategies exist, we made two major simplifying choices. First, we chose to map each multigrid level independently of the mapping of all other levels. Second, if the number of processors was greater than the number of grids on a level, we chose to assign each processor to, at most, one

<sup>3</sup>PARTI is an acronym for Parallel Automated Runtime Toolkit at ICASE.

grid on that level.

The first assumption is justified by the observation that the smoothing iteration is more frequent and more computationally intensive than the interpolation, so that the achievement of a good mapping during the smoothing step is crucial to performance. Also, any mapping that achieves an approximate load balance during the smoothing step is bound to induce a large amount of communication during interpolation. One reason for this is that the number of grids on each level almost always differs from the number on neighboring levels; thus, no mapping exists that simultaneously minimizes communication and achieves load balance.

The second assumption that each processor is assigned to no more than one grid on every level was taken to minimize communication, although it does induce some load imbalance. For example, suppose one has three grids on a level to be split over eight processors. Then each grid would ideally receive 2.66 processors. However, such a mapping is complex and clearly increases communication. Instead, one grid would be assigned to two processors, and the other two grids to three each.

In the current implementation, we did not split processors across grids. Instead, we carefully determined those grids that should get fewer and those that should get more processors to achieve approximate load balance without splitting processors across grids. In general, long thin grids (grids with one array dimension much smaller than the others) induce less communication when split over multiple processors than fat grids (grids with all array dimensions about equal). Thus, one maximizes load balance by assigning excess processors to the fattest grids.

Given these preliminaries, our load balancing algorithm follows. By assuming one has  $p$  processors and more processors than grids on all multigrid levels, the algorithm for distributing processors to grids is

```
Assign  $p$  processors to the finest grid
For  $level := 2$  to  $max\_level$  {
     $ngrids := number\_of\_grids(level)$ 
    assign  $\lfloor p/ngrids \rfloor$  processors to each grid
     $p\_excess := p - ngrids \lfloor p/ngrids \rfloor$ 
    assign one more processor to each of the  $p\_excess$  fattest grids
}
```

We call this the *maximally distributed* strategy.

This algorithm gives a distribution of processors to grids. Afterwards, one still has to partition each grid across the processors. To do this, we blocked the finest grid across processors in all three directions; coarser grids were blocked in one direction. One reason for this choice is that coarser grids often have an odd or prime number of processors, so that partitioning in more than one direction can be quite awkward. In all cases, the direction in which the coarser grids were blocked was chosen to minimize interprocessor communication.

In an alternate implementation referred to as the *aligned* strategy, all coarse grids were aligned to the finest grid, which requires each coarse grid to be partitioned among the full set of processors.



Although this strategy will eliminate communication during the interpolation, it leads to increased communication within a single grid and may quickly lead to idle processors. In the future, a strategy that uses a combination of the two described above may be implemented. In this *hybrid* implementation, coarse grids would be aligned in the first few levels; on lower levels, individual grids would be assigned to only a subset of processors.

### PARTI Implementation

As stated, the MSG algorithm was implemented in parallel with the multiblock PARTI routines. The multiblock library was designed to support block-structured aerodynamics codes in which one uses multiple, logically rectangular grid blocks to resolve complex aerodynamic geometries (ref. 16). Because the structure of such codes is fairly similar to that of MSG, we found that the same routines could be effectively used to implement this algorithm.

The PARTI library for block-structured codes allows multiple grid blocks to be processed in parallel and carries out the necessary communication required to move information among the grids. In our parallel implementation that maps coarse grids to subsets of processors, an individual "decomposition" is defined for the fine grid and for each coarser grid. In order to have all processors active on the finest grid, the fine-grid decomposition is embedded into the entire processor space. Then, for each subsequent level, the coarse-grid decompositions are embedded into an approximately equal portion of the processor space, as described in the last section. The single coarse grid on the coarsest level contains few points so it is mapped to one physical processor.

Our parallel version reads a file that holds the grid mapping and distribution information. A subroutine was created to use this mapping information along with the appropriate PARTI routines to set up the problem. As in most multigrid codes, the sequential code uses several large arrays to hold the residual, solution, and right-hand-side data for all grids on all levels. Individual grid sizes and starting index locations into the large arrays are computed and passed as parameters to subroutines. This strategy was maintained in the parallel version; however, the sizes and starting locations were modified to reflect the parallelism and the additional space required for holding boundary data for those grids distributed over more than one processor.

While PARTI aims to require minimal changes to the sequential source program, our parallel implementation was 20 to 25 percent larger than the original sequential program, and some subroutines required an extensive rewrite. Emerging FORTRAN dialects, like High Performance FORTRAN, FORTRAN D, and Vienna FORTRAN, may soon ease this programming burden. However, the current versions of these languages are not expressive enough to allow mapping strategies as complex as those described in this paper. The improvement of such languages, and of software tools like PARTI, is an area of active research at ICASE and elsewhere. The present situation, in which the effective mapping of an algorithm to a parallel architecture is an arduous task of many months, is clearly unacceptable.

## EXPERIMENTAL RESULTS

We recently implemented this algorithm and the mapping strategy on a 32-node Intel iPSC/860 and will soon migrate this program to a 64-node Intel Paragon and possibly a CM-5. The current results are preliminary, but are sufficiently encouraging to suggest the relative efficacy of this class of algorithms. For a problem with  $16^3$  mesh cells, the achieved efficiencies are given in Table 3.

Table 3. Efficiency of Problem With  $16^3$ -Point Grid on iPSC/860

Processors	1	2	4	8	16
Efficiency	1.0	.83	.66	.42	.25

Table 4. MSG Performance on the Intel iPSC/860

Size	Nodes	Total Time (secs)	V-cycle Time, (secs)	
			First V-cycle	Subsequent V-cycles
$16^3$	1	6.96	3.07	1.22
	2	4.21	1.70	.804
	4	2.63	1.05	.508
	8	2.07	.925	.373
	16	1.71	.793	.302
$32^3$	4	22.6	11.6	3.55
	8	13.5	7.15	2.03
	16	8.39	4.59	1.23
	32	5.27	2.61	.867
$64^3$	16	49.5	28.8	6.63
	32	24.1	12.1	3.87

These efficiencies were computed relative to the parallel implementation run on one node. A large amount of overhead can be incurred with the runtime software. For the  $16^3$  problem, the parallel code run on one processor takes approximately four times longer than the sequential code that contains no PARTI calls. For larger problems, the overhead should become less significant.

Another issue here is the choice of stencil. With the 7-point stencils used, the communication/computation ratio is four times greater than for 27-point stencils, and our efficiencies are correspondingly lower. However, the PARTI library does not currently update the corner ghost points needed for the 27-point stencils, so we were restricted to the use of 7-point stencils. This restriction will be changed in the next release of the library.

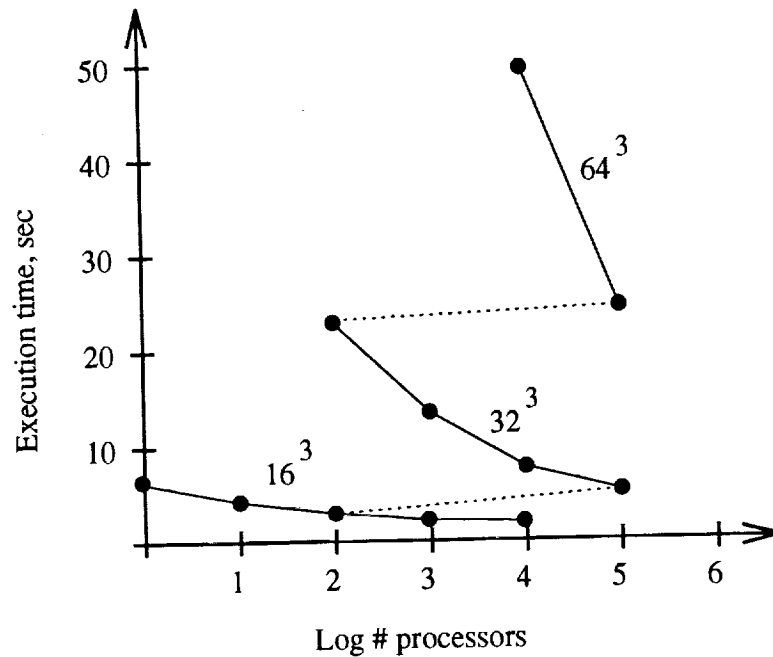


Figure 2. Execution time versus number of processors.

Table 4 shows performance results for several problem sizes. The table contains the overall program timings, along with the timings for each V-cycle. The results show the extra time required in the first V-cycle for setting up the communication schedules. These schedules are saved and, therefore, do not need to be recomputed on subsequent iterations.

Figure 2 expands on the data in Table 4. The graph shows that the  $32^3$  problem run on 4 nodes requires approximately the same amount of time as the  $64^3$  problem run on 32 nodes. This result is to be expected because the  $64^3$  problem has about eight times as much work. In Figure 2, a horizontal connecting line between the two cases (the dashed line on the graph) would indicate the achievement of perfect memory-bounded speedup (ref. 17); however, because of various overheads, this line slopes slightly.

The number of cases plotted here was constrained by current limitations of the PARTI library. For example, we were unable to obtain any timings on the machine that used more than 32 processors. Also, because of the large amount of memory consumed by the PARTI communication library, the user memory available on each processor decreased. These problems should be resolved in future releases of the PARTI library. The multiblock library is in a preliminary stage. We expect that further optimizations will improve the performance of block-structured codes with the multiblock library. The performance effects of some optimizations made to the PARTI primitives used in unstructured codes are described in ref. 18.

#### Alternate Mapping Strategies

We have also experimented with the *aligned* mapping strategy that was described briefly in

the previous section. With this strategy, the cost of the first V-cycle is much lower than in the *maximally distributed* strategy because the communication that occurs in the interpolation is easier to analyze. However, subsequent V-cycles are more expensive than in the maximally distributed strategy. This difference seems to be due both to the increased communication within each grid (because each grid is subdivided more finely) and to the sequentialization of all grids on every level. As a result, the aligned strategy is less effective than the maximally distributed strategy, even though it reduces interprocessor communication during the interpolation.<sup>4</sup> In future work, we plan to study various hybrid strategies like those proposed in ref. 9 that combine the advantages of both the aligned and maximally distributed strategies.

## CONCLUSIONS

We have examined the parallel implementation of a multigrid algorithm based on multiple coarse grids. Such multigrid algorithms have a fast convergence that is independent of grid stretching and can be effectively mapped to highly parallel architectures. We have developed a strategy for mapping such algorithms to parallel machines and have given preliminary results on the effectiveness of this strategy in mapping MSG to the Intel iPSC/860. The PARTI library is being ported to the Intel Paragon; we plan to try our algorithms on this larger machine in the near future.

## ACKNOWLEDGMENTS

The authors wish to thank Alan Sussman for making the library available to us while it has been under development and for frequent consultations on the use of the multiblock PARTI routines. We also wish to acknowledge discussions on parallel multigrid issues with Joe Dendy, Naomi Naik, and Graham Horton.

## REFERENCES

1. Dendy, J. E., Jr.; Ida, M. P.; and Rutledge J. M.: A Semicoarsening Multigrid Algorithm for SIMD Machines. *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 6, Nov. 1992, pp. 1460-1469.
2. Overman, A.; and Van Rosendale, J.: Mapping Implicit Spectral Methods to Distributed Memory Architectures. ICASE Report 91-52, June 1991.
3. Smith, R. A.; and Weiser, A.: Semicoarsening Multigrid on a Hypercube. *SIAM J. Sci. Stat. Comput.*, vol. 13, no. 6, Nov. 1992, pp. 1314-1329.

<sup>4</sup>A problem also exists with using the PARTI library for the aligned strategy. Currently, PARTI does not handle cases in which the decomposition of a coarse grid across processors results in a processor that receives no mesh points, a case that frequently arises with this strategy. Future versions of the PARTI library may eliminate this restriction.

4. Ta'asan, S.; and Brandt, A.: Multigrid Solutions to Quasi-Elliptic Schemes. In *Progress in Supercomputing in Computational Fluid Dynamics*. E. Murman and S. Abarbanel, eds., Proceedings of the U.S.-Israel Workshop, 1984, pp. 235-255.
5. Douglas, C. C.: A Tupleware Approach to Domain Decomposition Methods. *Appl. Numer. Math.*, 8, 1991, pp. 353-373.
6. Hackbusch, W.: The Frequency Decomposition Multigrid Method, Part I: Application to Anisotropic Equations. *Numer. Math.*, 56, 1989, pp. 229-245.
7. Naik N.; and Van Rosendale, J.: The Improved Robustness of Multigrid Elliptic Solvers Based on Multiple Semicoarsened Grids. *SIAM J. Numer. Anal.*, vol. 30, no. 1, Feb. 1993, pp. 215-229.
8. Frederickson, P.; and McBryan, O.: Parallel Superconvergent Multigrid. In *Multigrid Methods: Theory, Applications, and Supercomputing*. S. F. McCormick, ed., Marcel Dekker, New York, 1988, pp. 195-210.
9. Bastian, P.; and G. Horton, G.: Parallelization of Robust Multigrid Methods: ILU Factorization and Frequency Decomposition Method. *SIAM J. Sci. Stat. Comput.*, vol. 12, no. 6, Nov. 1991, pp. 1457-1470.
10. Mulder, W.: A New Multigrid Approach to Convection Problems. *J. Comp. Phys.*, vol. 83, 1989, pp. 303-329.
11. Alcouffe, R. E.; Brandt, A.; Dendy, J. E. Jr.; and Painter, J. W.: The Multi-Grid Method for the Diffusion Equation with Strongly Discontinuous Coefficients. *SIAM J. Sci. Stat. Comput.*, vol. 2, 1981, pp. 430-454.
12. Gannon, D.; and Van Rosendale, J.: Highly Parallel Multigrid Solvers for Elliptic PDE's: An Experimental Analysis. ICASE Report 82-36, 1982.
13. Gannon, D.; and Van Rosendale, J.: On the Structure of Parallelism in a Highly Concurrent PDE Solver. *J. Parallel and Distributed Comp.*, vol. 3, 1986, pp. 106-135.
14. Sussman, A.; Saltz, J.; Das, R.; Gupta, S.; Mavriplis, D.; Ponnusamy, R.; and Crowley, K.: PARTI Primitives for Unstructured and Block Structured Problems. *Computing Systems in Engineering*, vol. 3, no. 1-4, 1992, pp. 73-86.
15. Chase, C.; Crowley, K.; Saltz, J.; and Reeves, A.: Parallelization of Irregularly Coupled Regular Meshes. ICASE Report 92-1, Jan. 1992.
16. Vatsa, V.; Sanetrik, M.; and Parlette, E.: Development of a Flexible and Efficient Multigrid-Based Multiblock Flow Solver. AIAA Paper 93-0677, Jan. 1993.
17. Gustafson, J.; Montry, G.; and Benner, R.: Development of Parallel Methods for a 1024-Processor Hypercube. *SIAM J. Sci. Stat. Comput.*, vol. 9, 1988.
18. Das, R.; Mavriplis, D. J.; Saltz, J.; and Gupta, S.: The Design and Implementation of a Parallel Unstructured Euler Solver Using Software Primitives. AIAA Paper 92-0562, Jan. 1992.



## Unstructured Multigrid Through Agglomeration

V. Venkatakrishnan  
Computer Sciences Corporation  
M.S. T045-1, NAS Applied Research Branch  
NASA Ames Research Center, Moffett Field, CA 94035  
D. J. Mavriplis  
ICASE  
M.S. 132C  
NASA Langley Research Center, Hampton, VA 23681  
M.J. Berger  
Courant Institute of Mathematical Sciences  
New York University, New York, NY 10012

520-64  
197580  
P-14

## Abstract

In this work the compressible Euler equations are solved using finite volume techniques on unstructured grids. The spatial discretization employs a central difference approximation augmented by dissipative terms. Temporal discretization is done using a multistage Runge-Kutta scheme. A multigrid technique is used to accelerate convergence to steady state. The coarse grids are derived directly from the given fine grid through agglomeration of the control volumes. This agglomeration is accomplished by using a greedy-type algorithm and is done in such a way that the load, which is proportional to the number of edges, goes down by nearly a factor of 4 when moving from a fine to a coarse grid. The agglomeration algorithm has been implemented and the grids have been tested in a multigrid code. An area-weighted restriction is applied when moving from fine to coarse grids while a trivial injection is used for prolongation. Across a range of geometries and flows, it is shown that the agglomeration multigrid scheme compares very favorably with an unstructured multigrid algorithm that makes use of independent coarse meshes, both in terms of convergence and elapsed times.

## 1 Introduction

Multigrid techniques have been successfully used in computational aerodynamics for over a decade [1, 2]. The main advantage of the multigrid method when solving steady flows is the enhanced convergence while requiring little additional storage. In addition, multigrid can be used in conjunction with any convergent base scheme, with adequate care exercised in constructing proper restriction and prolongation operators between the grids. Perhaps the biggest advantage of multigrid is the fact that it deals directly with the nonlinear problem without requiring an elaborate linearization and the attendant storage required to store the matrix that arises from the linearization. Thus, multigrid techniques have enabled the practical solution of complex aerodynamic flows using millions of grid points.

The initial efforts in multigrid were directed towards the solution of flows on structured grids where coarse grids can easily be derived from a given fine grid. Typically, this is done by omitting alternate grid lines in each dimension. These ideas have been extended to triangular

grids in two dimensions and to tetrahedral meshes in three dimensions [3, 4, 5, 6]. In previous work by the second author, a sequence of unnested triangular grids of varying coarseness is constructed [3]. Piecewise linear interpolation operators are derived during a preprocessing step by using efficient search procedures. The residuals are restricted to coarse grids in a conservative manner. It has been shown that such a scheme can consistently obtain convergence rates comparable to those obtained with existing structured grid multigrid methods. For complex geometries, especially in three dimensions, however, constructing coarse grids that faithfully represent the complex geometries can become a difficult proposition. Thus, it is often desirable to derive the coarse grids directly from a given fine grid.

The agglomeration multigrid strategy has been investigated by Lallemand et al. [7] and Smith [8]. Lallemand et al. use a base scheme where the variables are stored at the vertices of the triangular mesh, whereas Smith uses a scheme that stores the variables at the centers of triangles. In the present work, a vertex-based scheme is employed. Two dimensional triangular grids contain twice as many cells as vertices (neglecting boundary effects), and three dimensional tetrahedral meshes contain 5 to 6 times more cells than vertices. Thus, on a given grid, a vertex scheme incurs substantially less computational overhead than a cell-based scheme. Increased accuracy can be expected from a cell-based scheme, since this involves the solution of a larger number of unknowns. However, the increase in accuracy does not appear to justify the additional computational overheads, particularly in three dimensions.

The main idea behind the agglomeration strategy of Lallemand et al. [7] is to agglomerate the control volumes for the vertices using heuristics. The centroidal dual, composed of segments of the median of the triangulation, is a collection of the control volumes over which the Euler equations in integral form are solved. On simple geometries, Lallemand et al. were able to show that the agglomerated multigrid technique performed as well as the multigrid technique which makes use of unnested coarse grids. However, the convergence rates, especially for the second order accurate version of the scheme, appeared to degrade somewhat. Furthermore, the validation of such a strategy for more complicated geometries and much finer grids, as well as the incorporation of viscous terms for the Navier-Stokes equations, remains to be demonstrated. The work of Smith [8] constitutes the basis of a commercially available computational fluid dynamics code, and as such has been applied to a number of complex geometries [9]. However, consistently competitive multigrid convergence rates have yet to be demonstrated.

In the present work, the agglomeration multigrid strategy is explored further. The issues involved in a proper agglomeration and the implications for the choice of the restriction and prolongation operators are addressed. Finally, flows over non-simple two-dimensional geometries are solved with the agglomeration multigrid strategy. This approach is compared with the unstructured multigrid algorithm of Mavriplis [3] which makes use of unnested coarse grids. Convergence rates as well as CPU times on a Cray Y-MP/1 are compared using both methods.



## 2 Governing equations and discretization

The Euler equations in integral form for a control volume  $\Omega$  with boundary  $\partial\Omega$  read

$$\frac{d}{dt} \int_{\Omega} u \, dv + \oint_{\partial\Omega} F(u, n) \, dS = 0. \quad (1)$$

Here  $u$  is the solution vector comprised of the conservative variables: density, the two components of momentum, and total energy. The vector  $F(u, n)$  represents the inviscid flux vector for a surface with normal vector  $n$ . Equation (1) states that the time rate of change of the variables inside the control volume is the negative of the net flux of the variables through the boundaries of the control volume. This net flux through the control volume boundary is termed the *residual*. In the present scheme the variables are stored at the vertices of a triangular mesh. The control volumes are non-overlapping polygons which surround the vertices of the mesh. They form the *dual* of the mesh, which is composed of segments of medians. Associated with each edge of the original mesh is a (segmented) dual edge. The contour integrals in Equation (1) are replaced by discrete path integrals over the edges of the control volume. Figure 1 shows a triangulation for a four-element airfoil and Figure 2 shows the centroidal dual. Each cell in Figure 2 represents a control volume. The path integrals are computed by using the trapezoidal rule. This can be shown to be equivalent to using a piecewise linear finite-element discretization. For dissipative terms, a blend of Laplacian and biharmonic operators is employed, the Laplacian term acting only in the vicinity of shocks. A multi-stage Runge-Kutta scheme is used to advance the solution in time. In addition, local time stepping, enthalpy damping and residual averaging are used to accelerate convergence. The principle behind the multigrid algorithm is that the errors associated with the high frequencies are annihilated by the carefully chosen smoother (the multi-stage Runge-Kutta scheme) while the errors associated with the low frequencies are annihilated on the coarser grids where these frequencies manifest themselves as high frequencies. In previous work [3], as well as in the present work, only the Laplacian dissipative term (with constant coefficient) is used on the coarse grids. Thus the fine grid solution itself is second order accurate, while the solver is only first order accurate on the coarse grids.

## 3 Details of agglomeration

The agglomeration (referred to also as coarsening) algorithm is a variation on the one used by Lallemand et al. [7] and is given below:

1. Pick a starting vertex on the surface of one of the airfoils.
2. Agglomerate control volumes associated with its neighboring vertices which are not already agglomerated.
3. Define a front as comprised of the exterior faces of the agglomerated control volumes. Place the exposed edges in a queue.
4. Pick the new starting vertex as the unprocessed vertex incident to a new starting edge which is chosen from the following choices given by order of priority:

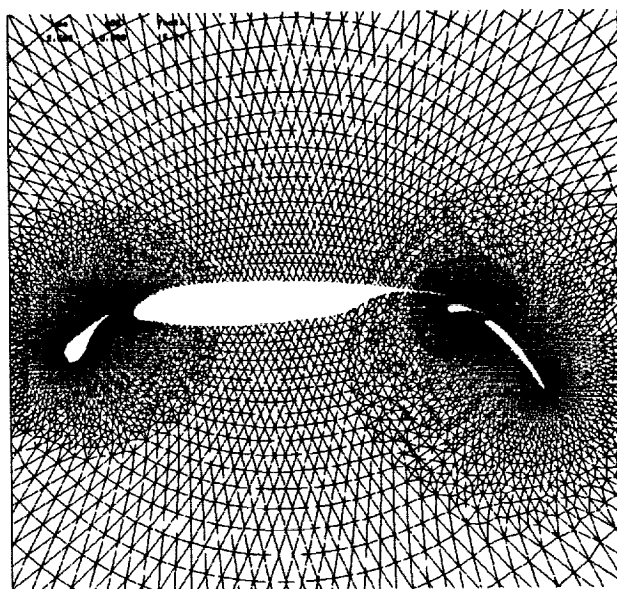


Figure 1: Grid about a four-element airfoil.

- An edge on the front that is on the solid wall.
- An edge on the solid wall.
- An edge on the front that is on the far field boundary.
- An edge on the far field boundary.
- The first edge in the queue.

5. Go to Step 2 until the control volumes for all vertices have been agglomerated.

There are many other ways of choosing the starting vertex in Step 4 of the algorithm, but we have found the above strategy to be the best. The efficiency of the agglomeration technique can be characterized by a histogram of the number of fine grid cells comprising each coarse grid cell. Ideally, each coarse grid cell will be made up of exactly four fine grid cells. The various strategies can be characterized by how close they come to this ideal case. One variation is to pick the starting edge randomly from the edges currently on the front. Figure 3 shows a plot of the number of coarse grid cells as a function of the number of fine grid cells comprising them, with our agglomeration algorithm described above, and with the variation. It is clear that our agglomeration algorithm is superior to the variant. The number of coarse grid cells having exactly one fine cell (singletons) is also much smaller with our algorithm compared to the variant. We have also investigated another variation where the starting vertex in Step 4 is randomly picked from the field and this turns out to be much worse. It is possible to identify the singleton cells and agglomerate them with the neighboring cells, but this has not been done.

The procedure outlined above is applied recursively to create coarser grids. Figure 4 shows an example of the agglomerated coarse grid. The boundaries between the control volumes on the coarse grids are composed of the edges of the fine grid control volumes. We have observed that the number of such edges only goes down by a factor of 2 when going from a fine to a coarse grid. Since the computational load is proportional to the number of edges,

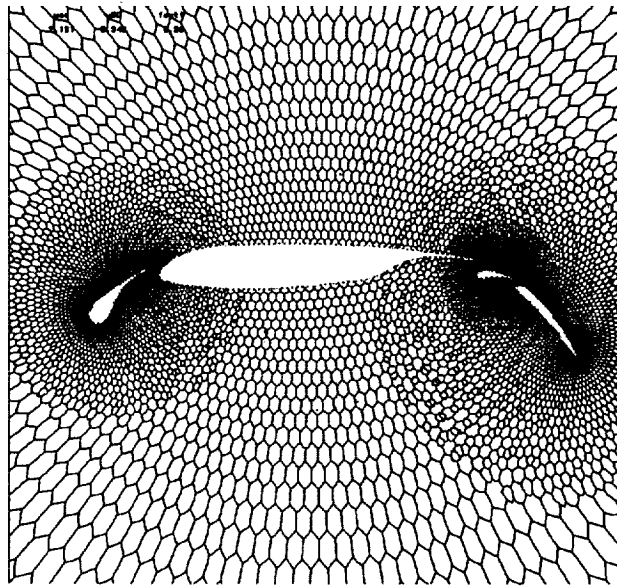


Figure 2: Centroidal dual for the triangulation of Figure 1.

this is unacceptable in the context of multigrid. However, if we recognize that the multiple edges separating two control volumes can be replaced by a single edge connecting the end points, then the number of edges does go down by a factor of 4. Since only a first order discretization is used on the coarse grids, there is no approximation involved in this step. If a flux function that involved the geometry in a nonlinear fashion were used, such as the Roe's approximate Riemann solver, this is still a very good approximation. It may also be seen from Figure 4 that once this approximation is made, the degree of a node in this graph is still 3 i.e., each node in the interior has precisely three edges emanating from it. Thus the agglomerated grid implies a triangulation of the vertices of a dual graph of the coarse grid. Trying to reconstruct the triangulation is not a good idea, since this may result in a graph with intersecting edges (non planar graph), which leads to non-valid triangulations. If a valid triangulation could always be constructed, it would be possible to use the coarse grid triangulation for constructing piecewise linear operators for prolongation and restriction akin to the non-nested multiple grid scheme [3]. In practice, we have often found the implied coarse grid triangulations to be invalid and therefore the coarse grids are only defined in terms of control volumes. This has some important implications for the multigrid algorithm discussed below.

Since the fine grid control volumes comprising a coarse grid control volume are known, the restriction is similar to that used for structured grids. The residuals are simply summed from the fine grid cells and the variables are interpolated in an area-weighted manner. For the prolongation operator, we use a simple injection (a piecewise constant interpolation). This is an unfortunate but unavoidable consequence of using the agglomeration strategy. A piecewise linear prolongation operator implies a triangulation, the avoiding of which is the main motivation for the agglomeration. However, additional smoothing steps may be employed to minimize the adverse impact of the injection. This is achieved by applying an averaging procedure to the injected corrections. In an explicit scheme, solution updates are directly proportional to the computed residuals. Thus, by analogy, for the multigrid scheme, correc-

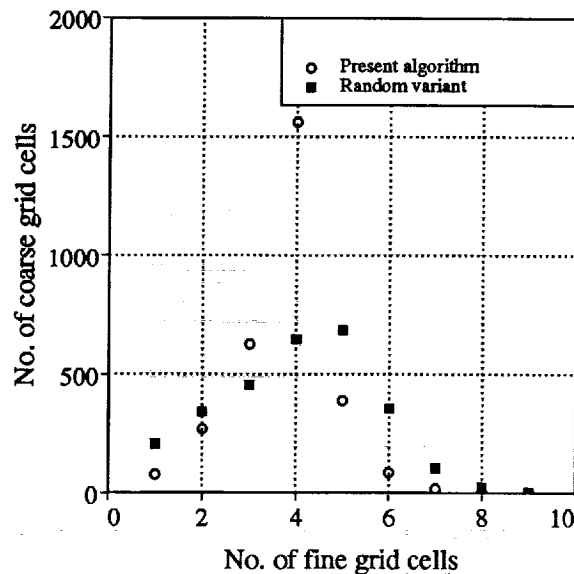


Figure 3: No. of coarse grid cells as a function of the fine grid cells they contain.

tions may be smoothed by a procedure previously developed for implicit residual smoothing [3]. The implicit equations for the smoothed corrections are solved using two iterations of a Jacobi scheme after the prolongation at each grid level.

The agglomeration step is done as a preprocessing operation on a workstation. It is very efficient and employs hashing to combine the multiple fine grid control volume edges separating two coarse grid cells into one edge. The time taken to derive 5 coarse grids on a Silicon Graphics work station model 4D/25 (20 MHz clock) for the grid shown in Figure 1 with 11340 vertices is 83 seconds.

## 4 Results and discussion

Results are presented for two inviscid flow calculations and the performance of the agglomerated multigrid algorithm is compared with that of the non-nested multiple grid multigrid algorithm of [3]. The first flow considered is flow over an NACA0012 airfoil at a freestream Mach number of 0.8 and angle of attack of  $1.25^\circ$ . The dual to the fine grid having 4224 vertices is shown in Figure 5. The sequence of unnested grids (not shown) for use with the non-nested multigrid algorithm contains 1088, 288 and 80 vertices, respectively. The agglomerated grids are shown in Figure 6. These grids have 1088, 288 and 80 vertices (regions) as well. Figure 7 shows the convergence histories obtained with the non-nested and agglomeration multigrid algorithms. Both the multigrid strategies employ W-cycles. The convergence histories show that the multigrid algorithm slightly outperforms the agglomeration algorithm. The CPU times required for 100 iterations on the Cray Y-MP/1 are 25 and 24 seconds, respectively. Thus the two schemes perform equally well.

The next case considered is flow over a four-element airfoil. The freestream Mach number is 0.2 and the angle of attack is  $5^\circ$ . The fine grid has 11340 vertices and is shown in Figure 1. The coarse grids for use with the non-nested multigrid algorithm (not shown) contain 2942 and 727 vertices. The two agglomerated grids are shown in Figure 8. These grids contain

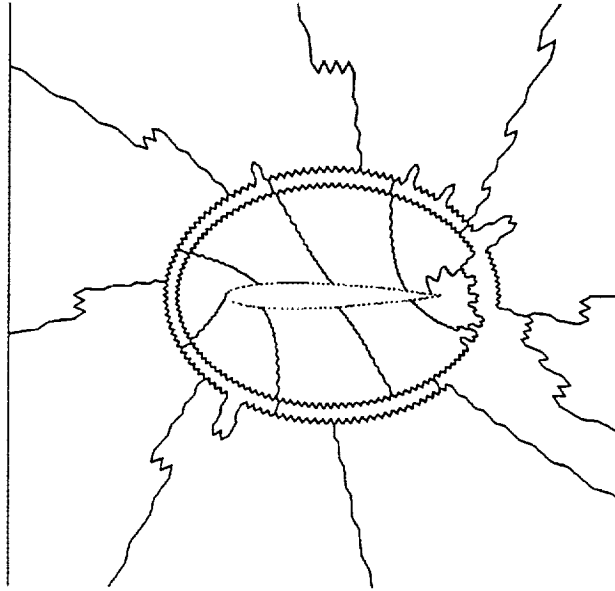


Figure 4: An example of an agglomerated coarse grid.

3027 and 822 vertices (regions), respectively. The convergence histories of the non-nested and agglomeration multigrid algorithms are shown in Figure 9. The convergence histories are comparable but the convergence is slightly better with the agglomerated multigrid strategy. This is a bit surprising since the original multigrid algorithm employs a piecewise linear prolongation operator. A possible explanation is that the agglomeration algorithm creates better coarse grids than those employed in the non-nested algorithm. The CPU times required on the Cray Y-MP are 59 and 58 seconds with the original and the agglomerated multigrid, respectively, using three grids.

Perhaps the biggest advantage of the agglomeration algorithm lies in its ability to generate very coarse grids without any user intervention. Such extremely coarse grids should be beneficial in multigrid. Figure 10 shows two coarser grids for the four element airfoil case. These grids contain 63 and 22 vertices, respectively. With these grids it is now possible to use a 6 level agglomeration multigrid strategy. However, because these coarse grids are rather nonuniform, it is imperative that the first order coarse grid operator be a strictly positive scheme (i.e. one can no longer rely on assumptions of grid smoothness as conditions for stability). With the original first order operator in place, which is composed of a central difference plus a dissipative flux, it is difficult to guarantee the positivity of the scheme for arbitrary grids. In fact, the scheme has been found to be unstable on some of the very coarse and distorted agglomerated meshes. However, if the flux is replaced by a truly first order upwind flux, given for example by Roe's flux difference splitting [10], a stable scheme can be recovered for these coarse agglomerated grids. Thus, for each of the coarse grids obtained by agglomeration, a check of the convergence properties of the coarse grid operator at the desired flow conditions is carried out if problems are experienced with the multigrid. This step ensures that the coarse grid operators are convergent and that the problems with the multigrid, if any, come from the inter-grid communication. Figure 11 shows the convergence history with the 6 grid level agglomerated multigrid scheme. Also shown is the convergence with the 3 grid agglomeration multigrid scheme. In this particular case, Roe's upwind flux is

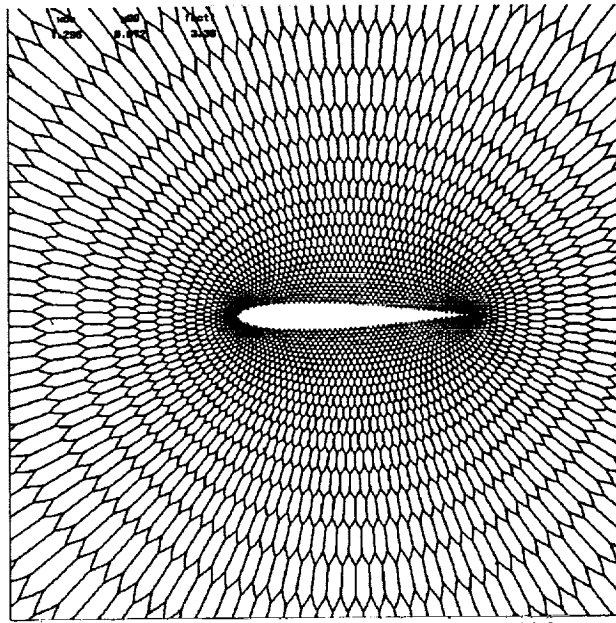


Figure 5: Dual to the fine grid having 4420 vertices.

used on the two coarsest grids, where central differencing proved unreliable. The time taken for the 6 grid agglomeration multigrid is 86 seconds. Thus the improved convergence rate is not entirely reflected in terms of the required computational resources. This is attributed to the increased time required by the Roe's upwind scheme, which involves a substantial number of floating point operations. This case serves to demonstrate the importance of the stability of each of the individual coarse grid operators in the multigrid scheme. Although first order upwinding has been employed on the distorted coarse meshes for demonstration purposes, it should be possible to construct stable central difference operators on such meshes.

## 5 Conclusions

It has been shown that the agglomeration multigrid strategy can be made to approximate the efficiency of the unstructured multigrid algorithm using independent, non-nested coarse meshes, in terms of both convergence rates and CPU times. It is further shown that arbitrarily coarse grids can be obtained with the agglomeration technique, although care must be taken to ensure that the coarse grid operator is convergent on these grids. Agglomeration has direct applications to three dimensions, where it may be difficult to derive coarse grids that conform to the geometry. In future work, alternate methods of generating coarse grids will be investigated. These may include the creation of maximal independent sets to create the coarse grid seed points and using these seed points to agglomerate the fine grid cells around them. A maximal independent set is a subset of the graph containing only vertices that are distance 2 apart in the original graph. Since coarsening algorithms can be viewed as partitioning strategies, there also exists a possible interplay between agglomerated multigrid techniques and distributed memory parallel implementations of the algorithm, which should be further investigated. Finally, the implementation of the viscous terms for Navier-Stokes flows on arbitrary polygonal control volumes must be carried out for this type of strategy to be applicable to viscous flows.

## 6 Acknowledgements

The first author thanks the NAS Applied Research Branch at NASA Ames Research Center for funding this project under contract NAS 2-12961. The second author's work was supported under NASA contract NAS1-19480. The third author was partially supported by AFOSR 91-0063, by DOE grant DE-FG02-88ER25053, by a PYI, NSF ASC-8858101 and by the Research Institute for Advanced Computer Science (RIACS) under Cooperative Agreement NCC2-387 between the National Aeronautics and Space Administration (NASA) and the Universities Space Research Association (USRA).

## References

- [1] R.H. Ni. A Multiple Grid Scheme for Solving the Euler Equations Proc *AIAA Journal*, Vol 20, pp. 1565-1571, 1982.
- [2] A. Jameson. Solution of the Euler Equations by a Multigrid Method. In *Applied Math. and Comp.*, Vol. 13, pp. 327-356, 1983.
- [3] D. J. Mavriplis and A. Jameson. Multigrid Solution of the Two-Dimensional Euler Equations on Unstructured Triangular Meshes. In *AIAA Journal*, Vol 26, No. 7, July 1988, pp. 824-831.
- [4] D. J. Mavriplis. Three Dimensional Multigrid for the Euler Equations. Proc *AIAA 10th Comp. Fluid Dynamics Conf.*, Honolulu, Hawaii, June 1991, pp. 239-248.
- [5] M. P. Leclercq. Resolution des Equations d'Euler par des Methodes Multigrilles Conditions aux Limites en Regime Hypersonique. *Ph.D Thesis, Applied Math, Universite de Saint-Etienne*, April, 1990.
- [6] J. Peraire, J. Peiro, and K. Morgan. A 3D Finite-Element Multigrid Solver for the Euler Equations. *AIAA paper 92-0449*, January 1992.
- [7] M. Lallemand, H. Steve, and A. Dervieux. Unstructured Multigridding by Volume Agglomeration: Current Status. In *Computers and Fluids*, Vol. 21, No. 3, pp. 397-433, 1992.
- [8] W. A. Smith. Multigrid Solution of Transonic Flow on Unstructured Grids. Proc *Recent Advances and Applications in Computational Fluid Dynamics, Proceedings of the ASME Winter Annual Meeting*, Ed. O. Baysal, November, 1990.
- [9] G. Spragle, W. A. Smith, and Y. Yadlin. Application of an Unstructured Flow Solver to Planes, Trains and Automobiles. *AIAA Paper 93-0889*, January 1993.
- [10] P. L. Roe. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. In *Journal of Computational Physics*, Vol. 43, pp. 357-372, 1981.

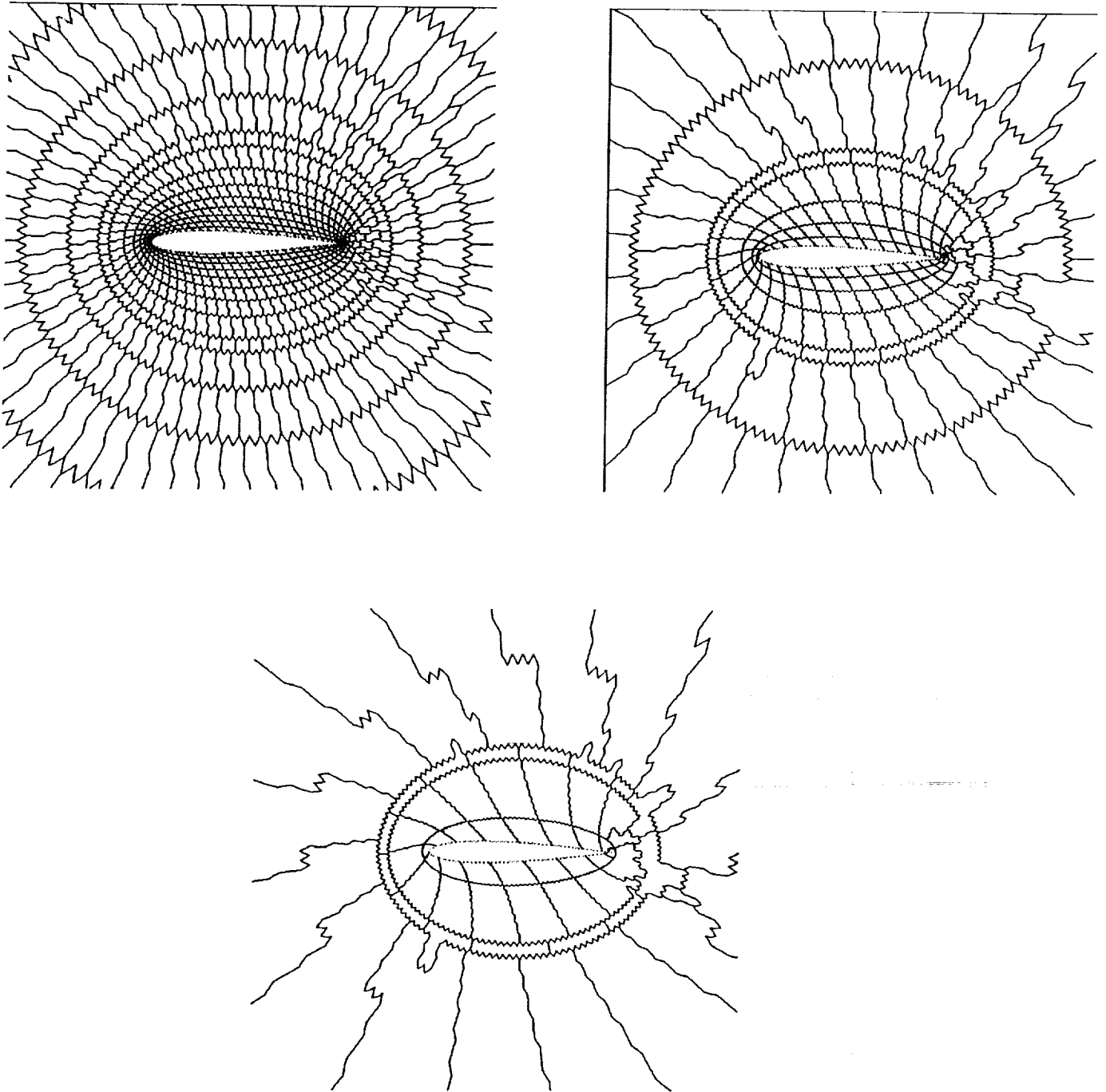


Figure 6: Three agglomerated coarse grids for the NACA0012 test case.



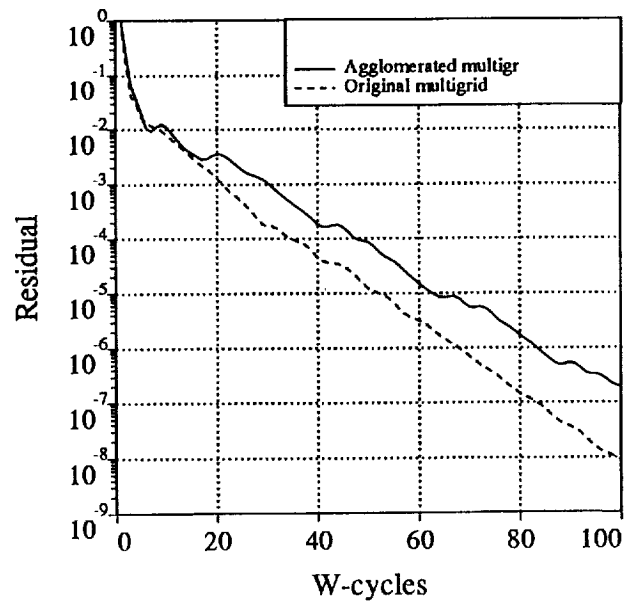


Figure 7: Convergence histories with the agglomerated and original multigrid.

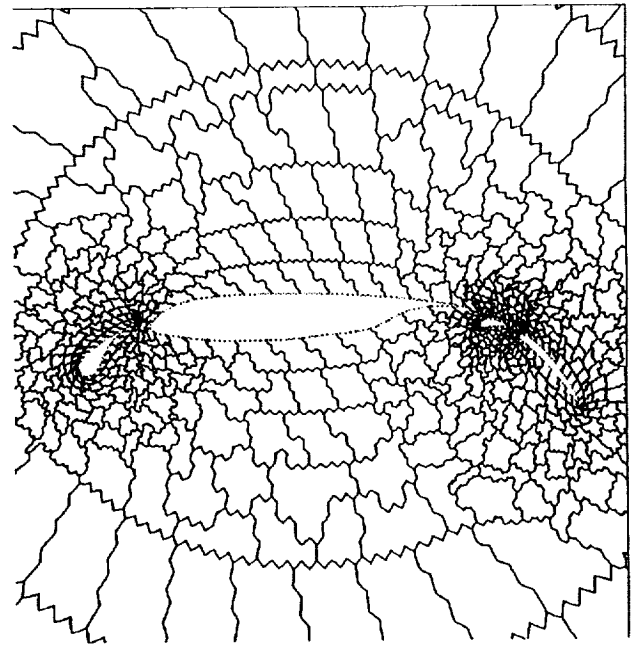
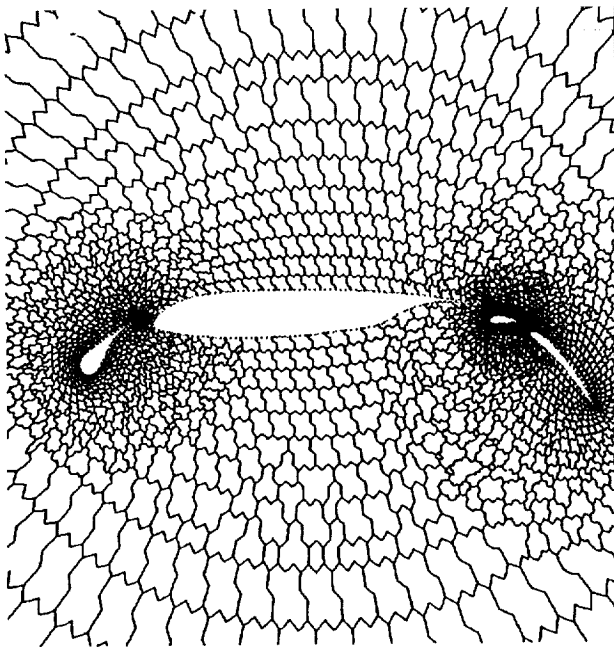


Figure 8: Two agglomerated coarse grids for the four-element test case.

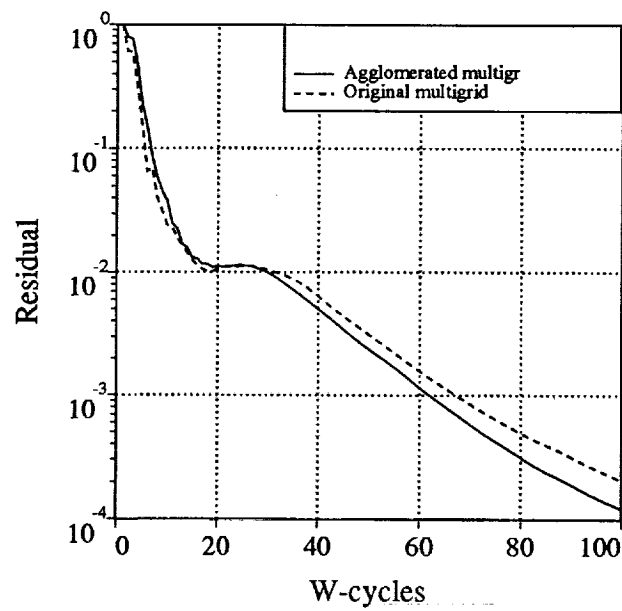


Figure 9: Convergence histories with the agglomerated and original multigrid.

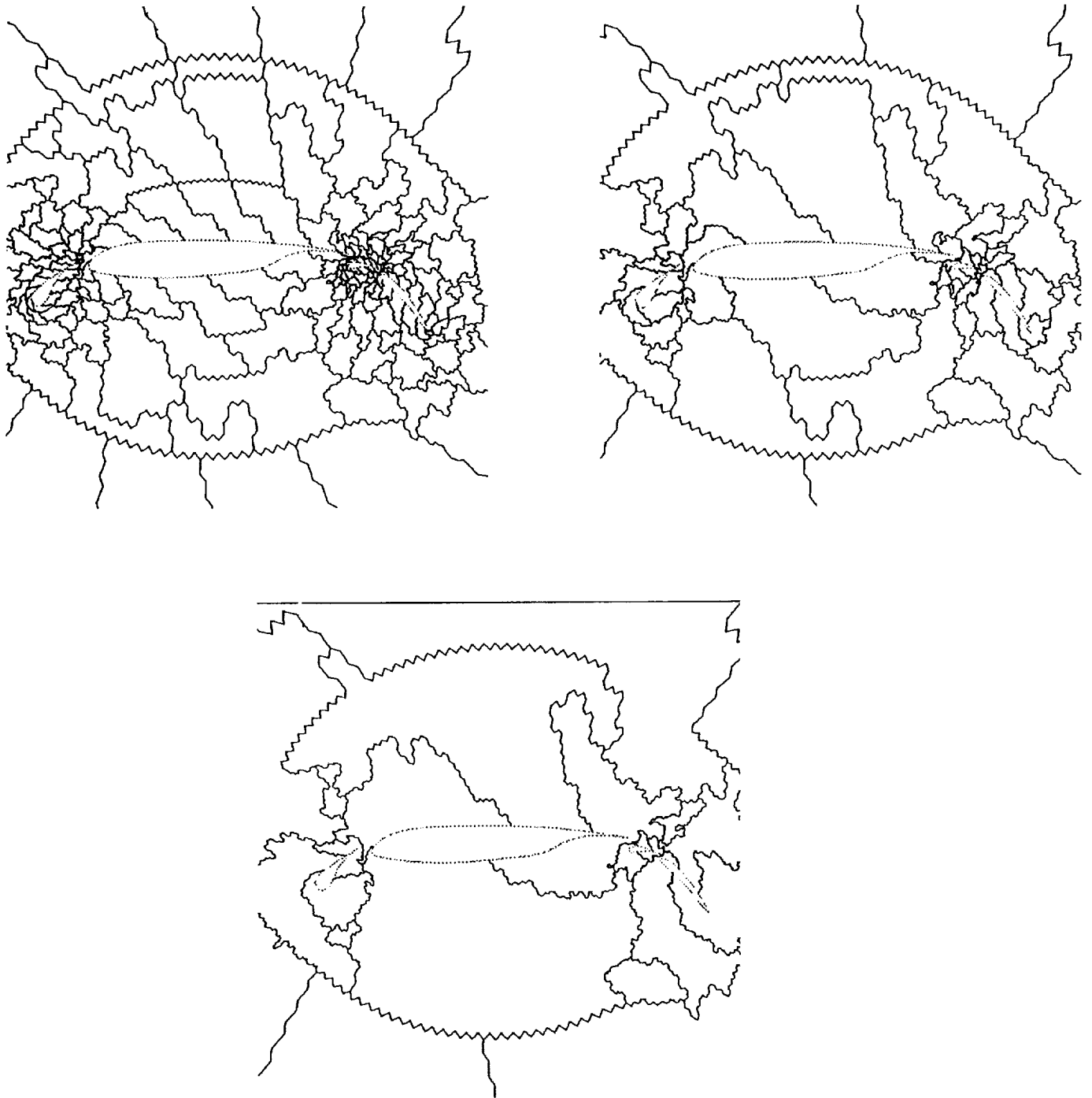


Figure 10: Three coarser grids for the four-element test case.

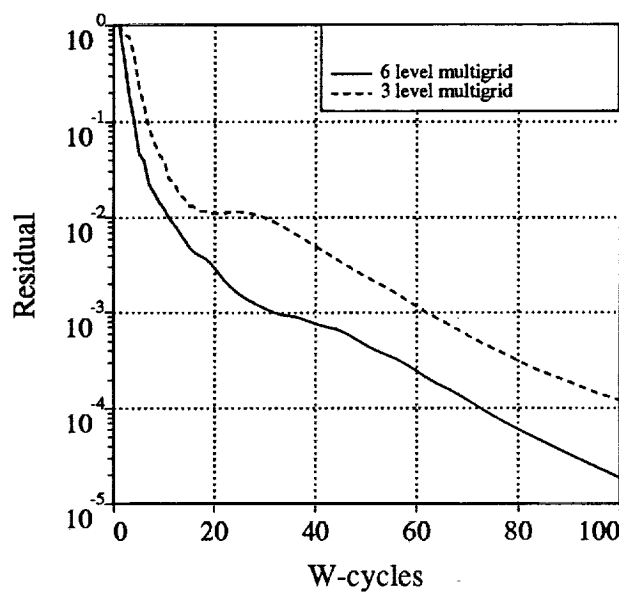


Figure 11: Convergence histories with the 6-level and 3-level agglomerated multigrid algorithms.

## MULTIGRID PROPERTIES OF UPWIND-BIASED DATA RECONSTRUCTIONS

Gary P. Warren and Thomas W. Roberts  
NASA Langley Research Center  
Hampton, VA

521-34  
197581  
p-15

## SUMMARY

The multigrid properties of two data reconstruction methods used for achieving second-order spatial accuracy when solving the two-dimensional Euler equations are examined. The data reconstruction methods are used with an implicit upwind algorithm which uses linearized backward-Euler time-differencing. The solution of the resulting linear system is performed by an iterative procedure. In the present study only regular quadrilateral grids are considered, so a red-black Gauss-Seidel iteration is used. Although the Jacobian is approximated by first-order upwind extrapolation, two alternative data reconstruction techniques for the flux integral that yield higher-order spatial accuracy at steady state are examined. The first method, probably most popular for structured quadrilateral grids, is based on estimating the cell gradients using one-dimensional reconstruction along curvilinear coordinates. The second method is based on Green's theorem. Analysis and numerical results for the two-dimensional Euler equations show that data reconstruction based on Green's theorem has superior multigrid properties as compared to the one-dimensional data reconstruction method.

## INTRODUCTION

Multigrid methods have become a popular tool for obtaining steady solutions of the Euler or Navier-Stokes equations. Although true multigrid performance is difficult to obtain, there is no doubt that multigrid methods can significantly decrease the computer time necessary for convergence. However, the gain in performance from a single grid algorithm is directly related to the type of smoothing operator used on each level. Although explicit methods may be simple to program and have a relatively small number of operation counts, the unconditional stability that implicit methods offer tends to greatly overcome their disadvantages. In addition, explicit time advancement methods generally do not exhibit good smoothing properties when used with higher-order upwind data reconstruction techniques for a system of equations.

In addition to the time advancement technique, the method of flux evaluation plays an important role in algorithm efficiency. One commonly used way to achieve higher order accuracy is to reconstruct the data on cell faces appropriately using the cell centered data. For grids which consist of logically rectangular cells, the most popular approach is to use simple one-dimensional curve fitting methods such as used by Anderson *et al.* [1]. The one-dimensional data reconstruction methods have been used with great success in two and three-dimensional CFD codes which use grids consisting of logically rectangular cells.

General fluid dynamics problems may require generating grids around complex shapes for which it is difficult to generate a single grid consisting of logically rectangular cells. Using multiple-block

grids to model complex geometries has been implemented with success using multigrid algorithms [2][3]. Another approach for generating grids around complex geometries is to use triangular elements. On unstructured triangular grids, however, data reconstruction methods based on Green's theorem are more prevalent since this does not require interpolation along a coordinate direction.

In reference [4] the authors presented a single grid stability analysis and numerical experiments of several different data reconstruction methods. In this paper, we extend this work to show the effect of the data reconstruction on multigrid performance. The Full-Approximation Scheme (FAS) multigrid method has been incorporated into a quadrilateral-based unstructured grid Euler solver using the implicit time marching method of reference [5].

## GOVERNING EQUATIONS

The governing equations are the time-dependent Euler equations, which express the conservation of mass, momentum, and energy for an inviscid gas. The equations are given by

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{1}{A} \oint_{\Omega} \bar{\mathbf{F}} \cdot \hat{\mathbf{n}} d\Omega = 0 \quad (1)$$

where  $A$  is the area of the cell that is bounded by the contour  $\Omega$  with the outward-pointing unit normal  $\hat{\mathbf{n}}$ . The state vector  $\mathbf{Q}$  and the flux vectors  $\bar{\mathbf{F}}$  are given as

$$\mathbf{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix}, \quad \bar{\mathbf{F}} \cdot \hat{\mathbf{n}} = \begin{bmatrix} \rho U \\ \rho U u + p \hat{\mathbf{n}}_x \\ \rho U v + p \hat{\mathbf{n}}_y \\ (e + p)U \end{bmatrix} \quad (2)$$

where  $\rho$  is the density,  $u$  and  $v$  are the  $x$  and  $y$  components of the velocity,  $e$  is the energy per unit volume,  $p$  is the pressure, and  $U$  is the velocity in the direction of the outward pointing normal to the cell

$$U = \hat{\mathbf{n}}_x u + \hat{\mathbf{n}}_y v \quad (3)$$

The equations are closed with the equation of state for a perfect gas

$$p = (\gamma - 1) \left[ e - \rho(u^2 + v^2)/2 \right] \quad (4)$$

where  $\gamma$  is the ratio of specific heats.

## TIME ADVANCEMENT ALGORITHM

The method used for accelerating the solution to steady state is the Full Approximation Scheme (FAS) multigrid method. The technique used for smoothing the errors on each grid level is based on the scheme described in reference [5] applied to a grid of quadrilateral cells. The method is an

implicit upwind algorithm that uses linearized backward-Euler time differencing. The cell-averaged solution vector  $\mathbf{Q}$  is updated at each time level  $n$  with the equations

$$\mathbf{L}^n \Delta \mathbf{Q}^n = -\mathbf{R}(\mathbf{Q}^n) \quad (5)$$

$$\mathbf{Q}^{n+1} = \mathbf{Q}^n + \Delta \mathbf{Q}^n \quad (6)$$

The operator  $\mathbf{R}(\mathbf{Q}^n)$  is the discrete approximation to the flux integral in equation (1) at time level  $n$ . The fluxes are evaluated with Van Leer flux-vector splitting [6] and are second-order accurate if a linear data reconstruction method is used. The operator  $\mathbf{L}^n$  is written as

$$\mathbf{L}^n = \frac{A}{\Delta t} \mathbf{I} + \frac{\partial \mathbf{R}^n}{\partial \mathbf{Q}^n} \quad (7)$$

To minimize the bandwidth and maintain block-diagonal dominance of the matrix  $\mathbf{L}^n$ , the Jacobian  $\partial \mathbf{R}^n / \partial \mathbf{Q}^n$  is approximated by first-order upwind differencing rather than by exactly linearizing the second-order right-hand side of equation (5). The steady-state solution remains second-order accurate. The solution of the linear system (5) is performed by an iterative procedure. In the present study, subiterations are performed using red-black Gauss-Seidel where the flux-Jacobians in equation (7) are frozen at the current time level. It is recognized that the linear system must be solved adequately to gain the full benefits of an implicit formulation. However, the scope of this work is to analyze the effects of various data reconstructions to compute the right-hand side of equation (5). The stability and smoothing analysis presented later assumes the linear system is solved exactly at each time step.

## UPWIND STENCILS

All of the reconstruction stencils used for the right-hand side of equation (5) in this study are based on MUSCL-type differencing [6]. In this approach, the flux vector  $\hat{\mathbf{F}}$  is split into two components

$$\vec{\mathbf{F}} \cdot \hat{\mathbf{n}} = \hat{\mathbf{F}}(\mathbf{Q}^+, \mathbf{Q}^-) = \hat{\mathbf{F}}^-(\mathbf{Q}^+) + \hat{\mathbf{F}}^+(\mathbf{Q}^-) \quad (8)$$

where

$$\mathbf{Q}_{\text{face}}^\pm = \mathbf{Q}_{\text{cell}}^\pm + \Theta^\pm(\mathbf{Q}) \quad (9)$$

The values of  $\mathbf{Q}$  are determined on each side of a cell face by using an interpolation operator  $\Theta$ , and reconstructing the cell-centered data on each face as shown in figure 1. Upwind fluxes are computed from the two face values with Van Leer flux-vector splitting [6]. The stencils that are considered differ in the interpolation operator  $\Theta$ .

One of the most common methods of data reconstruction for upwind structured flow solvers is to interpolate the data to the cell face using only the cells along the curvilinear coordinate direction which is perpendicular to the face [1]. Using the cell numbering shown in figure 2, a family of schemes is given by

$$Q_{\text{face}}^- = Q_2 + \frac{1}{4}[(1-\kappa)\Delta_- + (1+\kappa)\Delta_+]Q_2 \quad (10)$$

$$Q_{\text{face}}^+ = Q_3 - \frac{1}{4}[(1+\kappa)\Delta_- + (1-\kappa)\Delta_+]Q_3 \quad (11)$$

where

$$\Delta_+ Q_i = Q_{i+1} - Q_i \quad (12)$$

$$\Delta_- Q_i = Q_i - Q_{i-1} \quad (13)$$

These formulas assume the grid has been transformed from physical  $(x, y)$  space to computational  $(\xi, \eta)$  space where the grid spacing  $(\delta\xi, \delta\eta)$  is unity. Using this family of schemes as the interpolation operator results in the flux integration in a cell depending on a total of 9 cells for  $-1 \leq \kappa < 1$  as shown in figure 3.

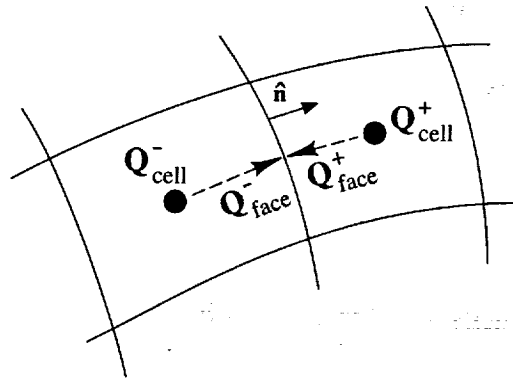


Figure 1. Data reconstruction for upwind fluxes

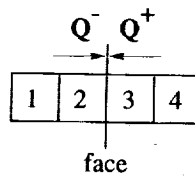


Figure 2. Cell Numbering for  $\kappa$  Methods

■ - cell being updated

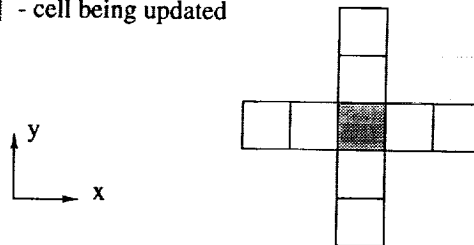


Figure 3. 9-point stencil



We can examine the relation between the discrete equations (10) - (13) and equation (8) by expanding the terms of the equations in a Taylor's series. Examining the interpolation of  $Q^-$  along a  $\xi$  coordinate line, a Taylor's series expansion about cell 2 is written as

$$Q_{\text{face}}^- = Q_2 + \Delta\xi \left. \frac{\partial Q}{\partial \xi} \right|_{\xi=\xi_2} + \frac{(\Delta\xi)^2}{2} \left. \frac{\partial^2 Q}{\partial \xi^2} \right|_{\xi=\xi_2} + \dots \quad (14)$$

If  $\kappa = 0$ , a central difference across cell 2 is used to calculate the gradient so that

$$\Theta^-(Q) = \Delta\xi \left. \frac{\partial Q}{\partial \xi} \right|_{\xi=\xi_2} \approx \left( \frac{1}{2} \right) \left( \frac{Q_3 - Q_1}{2} \right) \quad (15)$$

For  $\kappa = -1$ , the gradient is approximated using only one-sided information

$$\Theta^-(Q) = \Delta\xi \left. \frac{\partial Q}{\partial \xi} \right|_{\xi=\xi_2} \approx \frac{1}{2} (Q_2 - Q_1) \quad (16)$$

Although not considered in this study, if  $\kappa = 1/3$ , the first and second derivatives of equation (14) are estimated with central differences which yield a spatially third-order accurate steady-state solution in one dimension.

The other stencil used in constructing the data on the face is based on Green's theorem. This was used for triangular grids by Barth and Jespersen [7] and Frink [8]. This method of data reconstruction was also used by Anderson [5] on triangular grids in conjunction with the implicit scheme shown here. The interpolation operator is evaluated in physical ( $x, y$ ) space and is written as

$$\Theta^\pm(Q) = (\nabla Q \cdot \mathbf{r})^\pm \quad (17)$$

where  $\nabla Q$  is the average gradient in the cell and is evaluated using Green's theorem.

$$\begin{aligned} \frac{\partial Q}{\partial x} &= \frac{1}{A} \oint_{\Omega} (Q) \hat{n}_x d\Omega \\ \frac{\partial Q}{\partial y} &= \frac{1}{A} \oint_{\Omega} (Q) \hat{n}_y d\Omega \end{aligned} \quad (18)$$

To evaluate this numerically, inverse-distance weighting is used to transfer the cell-averaged data to the nodes [8].

$$Q_{\text{node}} = \frac{\sum_{i=1}^4 \frac{Q_{\text{cell}_i}}{r_i}}{\sum_{i=1}^4 \frac{1}{r_i}} \quad (19)$$

where  $r_i$  is the distance from the  $i$ -th cell center to the node. This reduces to simple averaging for uniform grids. Next, the trapezoidal rule is used to integrate around the cell. The  $x$ -component is given by

$$(\nabla Q)_x \approx \frac{1}{A_{\text{cell}}} \sum_{i=1}^4 \left( \frac{Q_{\text{node}_1} + Q_{\text{node}_2}}{2} \right) \Delta s_{\text{face}_i} \hat{n}_x \quad (20)$$

Here,  $A$  is the area of the cell,  $\text{node}_1$  and  $\text{node}_2$  define  $\text{face}_i$ ,  $\Delta s$  is the length of  $\text{face}_i$ , and  $\hat{n}_x$  is the  $x$ -component of the outward pointing unit normal. The data on the cell faces is then determined using (17) where the position vector,  $\mathbf{r}$ , is computed from the cell center to the face center. Using Green's theorem and the trapezoidal rule results in a stencil of 21 cells for the flux integration. The complete procedure for determining  $Q$  values on the cell faces is shown in figure 4.

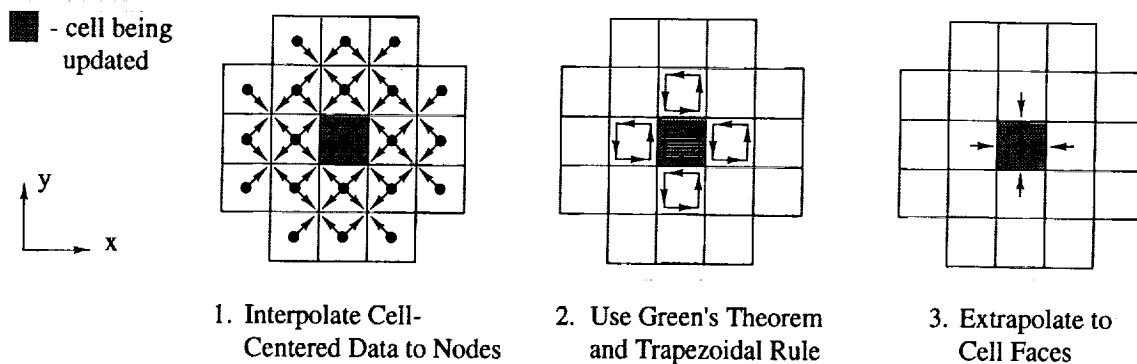


Figure 4. Data Reconstruction Using Green's Theorem

## TRUNCATION ERROR

A truncation error analysis for the 9-point stencils using  $\kappa = 0$  and  $\kappa = -1$  as well as the 21-point stencil has been shown in reference [4] and is summarized here for completeness. The truncation error of each of the three stencils is examined by considering the semi-discrete approximation to a scalar advection equation with non-negative coefficients  $a$  and  $b$ .

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} + b \frac{\partial u}{\partial y} = 0 \quad (21)$$

This linear equation is a simplified model of the two-dimensional Euler equations.

Leaving the equation continuous in time, the spatial derivatives are approximated by each stencil and expanded in a Taylor series about the point being updated. The 9-point stencil with  $\kappa = -1$  leads to the following equation:

$$\begin{aligned}\frac{\partial u}{\partial t} = & -a \frac{\partial u}{\partial x} - b \frac{\partial u}{\partial y} + \frac{1}{3} \left( a \delta x^2 \frac{\partial^3 u}{\partial x^3} + b \delta y^2 \frac{\partial^3 u}{\partial y^3} \right) \\ & - \frac{1}{4} \left( a \delta x^3 \frac{\partial^4 u}{\partial x^4} + b \delta y^3 \frac{\partial^4 u}{\partial y^4} \right) + h.o.t.\end{aligned}\quad (22)$$

where  $\delta x$  and  $\delta y$  are the grid spacing in the  $x$  and  $y$  directions, respectively. The difference approximation is second order in the grid spacing with a dispersive leading truncation error term. The approximation is also dissipative, as can be seen from the fourth-derivative term of the truncation error. For an advection velocity that is aligned with the grid ( $a$  or  $b = 0$ ), the dissipative term reduces to a fourth derivative in the flow direction.

For the 9-point stencil with  $\kappa = 0$  we get the following equation:

$$\begin{aligned}\frac{\partial u}{\partial t} = & -a \frac{\partial u}{\partial x} - b \frac{\partial u}{\partial y} + \frac{1}{12} \left( a \delta x^2 \frac{\partial^3 u}{\partial x^3} + b \delta y^2 \frac{\partial^3 u}{\partial y^3} \right) \\ & - \frac{1}{8} \left( a \delta x^3 \frac{\partial^4 u}{\partial x^4} + b \delta y^3 \frac{\partial^4 u}{\partial y^4} \right) + h.o.t.\end{aligned}\quad (23)$$

This equation differs from equation (22) in the magnitude of the coefficients of the dispersive and dissipative terms. We expect this difference formula to be less dissipative than the fully-upwind stencil.

A Taylor series expansion of the 21-point node-averaged stencil for the scalar advection equation gives the following:

$$\begin{aligned}\frac{\partial u}{\partial t} = & -a \frac{\partial u}{\partial x} - b \frac{\partial u}{\partial y} + \frac{1}{12} \left( a \delta x^2 \frac{\partial^3 u}{\partial x^3} + b \delta y^2 \frac{\partial^3 u}{\partial y^3} \right) \\ & - \frac{1}{8} \left( a \delta x \frac{\partial^2}{\partial x^2} + b \delta y \frac{\partial^2}{\partial y^2} \right) \left( \delta x^2 \frac{\partial^2 u}{\partial x^2} + \delta y^2 \frac{\partial^2 u}{\partial y^2} \right) + h.o.t.\end{aligned}\quad (24)$$

This equations looks remarkably similar to equation (23), as the coefficients of the dispersive and dissipative terms are identical. However, the dissipative term of the 21-point stencil contains cross derivatives and looks similar to a biharmonic term. Note that even for a grid-aligned advection velocity the cross-derivative term does not vanish. We expect that this difference stencil, although of the same formal accuracy as the 9-point stencil, will be more dissipative.

## STABILITY ANALYSIS

The basic stability properties of the upwind stencils considered here were examined in reference [4]. A Von Neumann analysis is used to examine the stability and convergence properties of the 9-point  $\kappa = 0$  and  $\kappa = -1$  stencils and the 21-point stencil. For each of the stencils, the equations are discretized according to equations (5) to (7). The operator  $\mathbf{L}^n$  is obtained by first-order interpolation in all cases, and the right-hand side  $\mathbf{R}(\mathbf{Q}^n)$  is obtained with the three second-order stencils.

Although the Von Neumann analysis is commonly applied to the scalar advection equation, we examine the stability of the system obtained by linearizing the Euler equations about a constant state, similar to the work in reference [5]. Applying a Fourier transform in space to the solution vector  $\mathbf{Q}^n$  gives the equation

$$\mathbf{Q}^n = z^n \hat{\mathbf{Q}}_0 \exp(i\phi) \exp(i\psi) \quad (25)$$

where  $\phi = \pi x / \delta x$ ,  $\psi = \pi y / \delta y$  are the Fourier modes in the  $x$  and  $y$  directions, respectively, and  $z$  is the amplification factor. Substitution of this expression into (5) yields the following equation

$$\hat{\mathbf{L}}\{(z-1)\hat{\mathbf{Q}}_0\} = -\hat{\mathbf{R}}\{\hat{\mathbf{Q}}_0\} \quad (26)$$

where  $\hat{\mathbf{L}}$  and  $\hat{\mathbf{R}}$  are the Fourier symbols of the left- and right-hand-side operators for the constant-coefficient problem. Equations (25) and (26) lead to a generalized eigenvalue problem for  $z$ . By rearranging terms, we define the amplification matrix

$$\hat{\mathbf{G}} = \mathbf{I} - \hat{\mathbf{L}}^{-1} \hat{\mathbf{R}} \quad (27)$$

and  $z$  is an eigenvalue of  $\hat{\mathbf{G}}$ . The amplification matrix is  $4 \times 4$  and complex; a necessary condition for stability is that the magnitude of the eigenvalues of  $\hat{\mathbf{G}}$  are less than one for all  $\phi$  and  $\psi$ . We will refer to the amplification factor for a given mode as the magnitude of the largest eigenvalue for that mode. The matrix  $\hat{\mathbf{G}}$  depends upon four parameters: the Mach number; the flow direction; the CFL number, defined here as  $c \delta t / \delta x$ , where  $c$  is the speed of sound; and the cell aspect ratio,  $\delta y / \delta x$ .

The eigenvalue problem was solved numerically for a series of Fourier modes  $\phi$  and  $\psi$  in the range  $[-\pi, \pi]$ . Below we show the amplification factors for a Mach number of 0.8, flow aligned with the grid in the  $x$ -direction, a CFL number of 100, and a cell aspect-ratio of 1. These results are typical of the stability properties of the implicit scheme at other Mach numbers.

Shown in figure 5 are the amplification factors for the 9-point stencil with  $\kappa = -1$  and  $\kappa = 0$  for a CFL of 100. This CFL number represents the asymptotic behavior for the three stencils considered here as shown in reference [4]. Note that the fully-upwind scheme ( $\kappa = -1$ ) has very poor damping of the short-wavelength modes. As  $\text{CFL} \rightarrow \infty$  the amplification factor of the  $\phi = \pm\pi$  mode asymptotically approaches 1. Although unconditionally stable, the scheme is a very poor smoother for an FAS multigrid scheme using high CFL numbers. On the other hand, the upwind-biased stencil ( $\kappa = 0$ ) leads to a scheme with excellent smoothing properties. All the Fourier modes are very well damped; in particular, the checkerboard and sawtooth modes have an amplification factor that tends to 0 with increasing CFL numbers. This scheme appears to be a very good multigrid smoother.

By using the 21-point stencil to discretize the steady-state operator we get even better stability properties, as is seen in figure 6. All the high-frequency modes are damped extremely well; the amplification factor for  $\phi, \psi = \pm\pi$  has an asymptote of 0, making this operator an excellent choice as a multigrid smoother.

Considering the 9-point,  $\kappa = 0$  stencil and the 21-point stencil in the case where the flow is skew to the grid, we get the results shown in figure 7. In both cases the damping of the short wavelengths is

essentially unchanged. The damping of the long wavelengths is worse, however, and the deterioration is somewhat more noticeable for the 9-point stencil, particularly for the intermediate wavelengths. The 21-point stencil retains its excellent stability properties over a larger range of wavelengths.

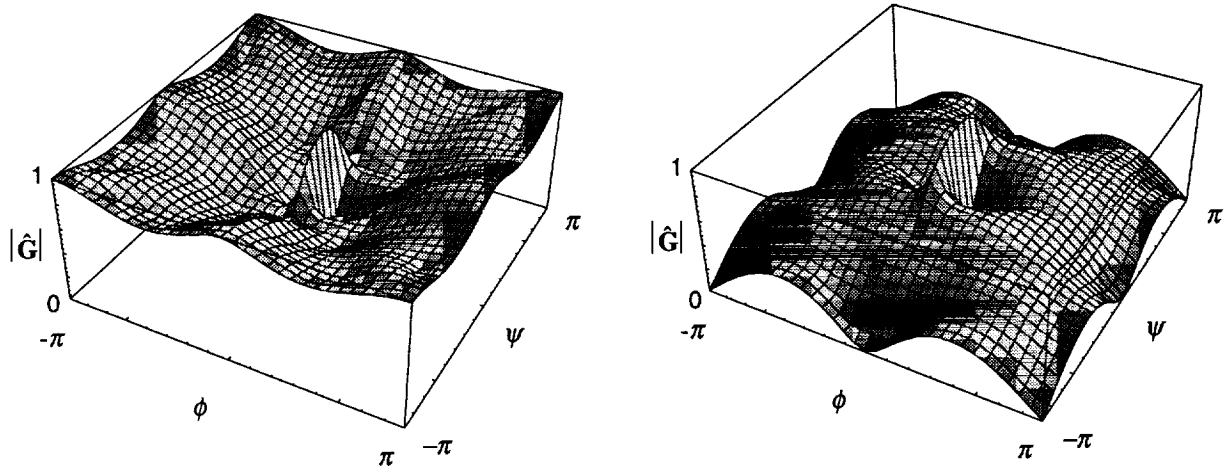


Figure 5. Amplification factors for 9-point stencil, Mach = 0.8,  $\alpha = 0$ , CFL = 100:  $\kappa = -1$  (left) and  $\kappa = 0$  (right)

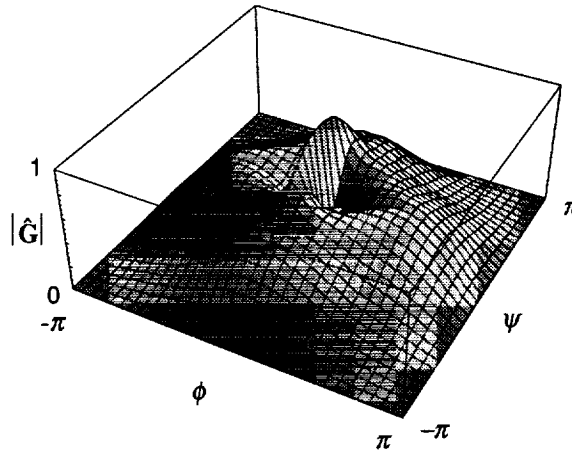


Figure 6. Amplification factor for 21-point stencil, Mach = 0.8,  $\alpha = 0$ , CFL = 100

Shown in figure 8 are the smoothing factors, defined as the maximum of the amplification factor over the range  $\pi/2 < |\phi|, |\psi| < \pi$ , and average amplification factors for the 9-point stencil over a range of CFL numbers from 1 to 1024 and  $\kappa$  from -1 to 1. The Mach number and flow angle are 0.8 and 45 degrees, respectively. These plots clearly show that the  $\kappa = 0$  stencil has the best smoothing properties for the 9-point stencil.

A comparison of the smoothing and amplification factors for the 21-point and the 9-point,  $\kappa = 0$  stencils is shown in figures 9 and 10. Shown in figure 9 are the smoothing and average amplification factors for flow aligned with the grid. Note that for CFL numbers up to about 16, the smoothing factors are identical. The asymptotic smoothing factors are slightly different: 0.524 and 0.563 for the 21-point and 9-point stencils, respectively. In contrast to the smoothing factors, the average amplification factor is about 50% lower for the 21-point stencil compared to the 9-point stencil. In figure 10 plots of

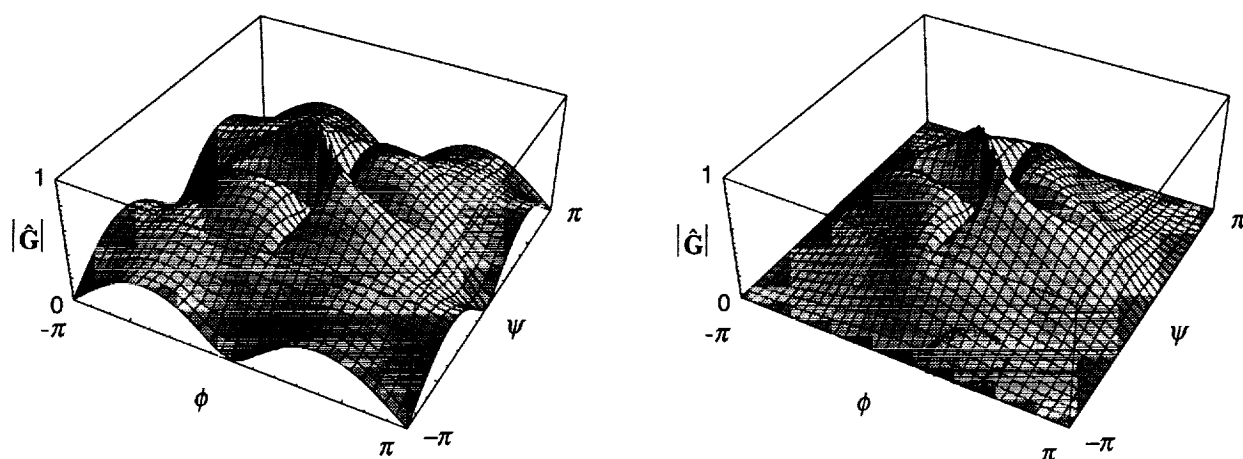


Figure 7. Flow at 45 degrees to the grid: 9-point stencil, Mach = 0.8,  $\alpha = 45$  degrees, CFL = 100:  $\kappa = 0$  (left) and 21-point stencil (right)

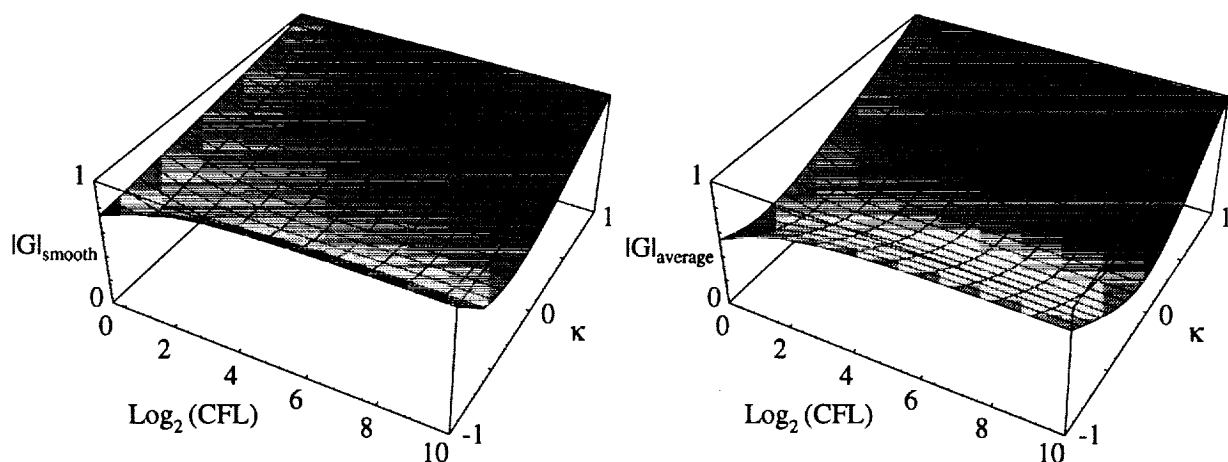


Figure 8. Smoothing factors and average amplification factors for  $\kappa$  methods

the smoothing factor and average eigenvalues are shown for both stencils for flow at 45 degrees to the grid. The average amplification factors are virtually unchanged, but there is some difference in the smoothing factors. The asymptotic values of the smoothing factors have deteriorated, increasing to 0.554 and 0.628 for the 21-point and 9-point stencils, respectively. The 21-point stencil's smoothing factor is less sensitive to the flow angle than that of the 9-point,  $\kappa = 0$  stencil.

The effect of grid aspect ratio on the 21-point and 9-point  $\kappa = 0$  stencil is shown in figure 11. Note that there is a large degradation in the smoothing properties for the 9-point  $\kappa = 0$  stencil when using high aspect ratio cells such as those in a viscous calculation near a solid wall or wake region. The 21-point stencil, however, is generally not affected by the cell aspect ratio. This insensitivity of the smoothing factor as the flow angle and grid aspect ratio changes means that we expect that it will result in more uniform multigrid performance than the 9-point,  $\kappa = 0$  stencil, over a variety of flow conditions and grid topologies.

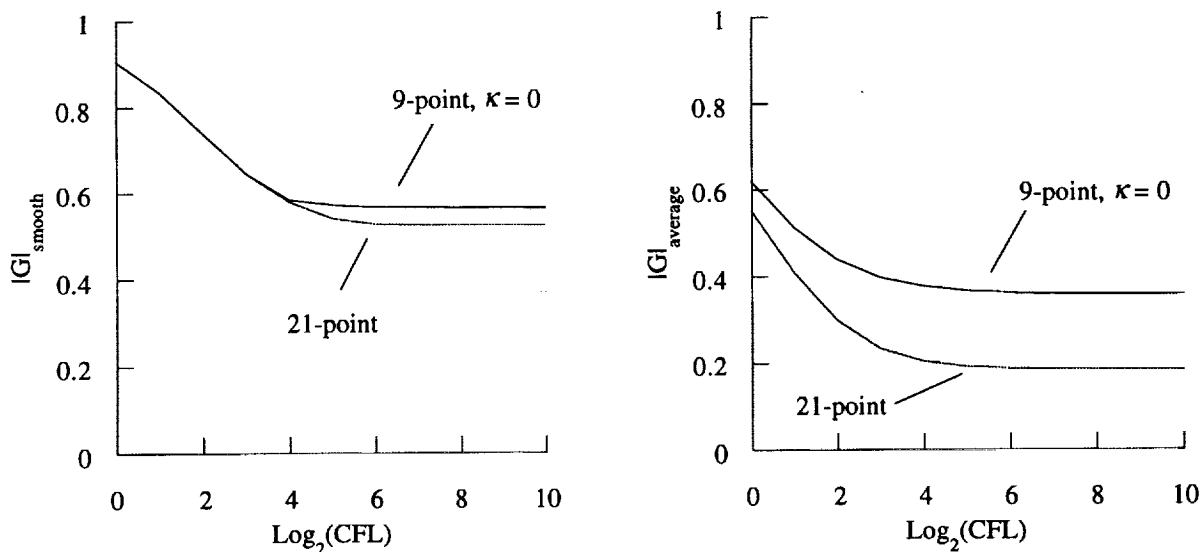


Figure 9. Smoothing factors and average amplification factors for the 21-point stencil and the 9-point stencil,  $\kappa = 0$ , for a flow angle of 0 degrees

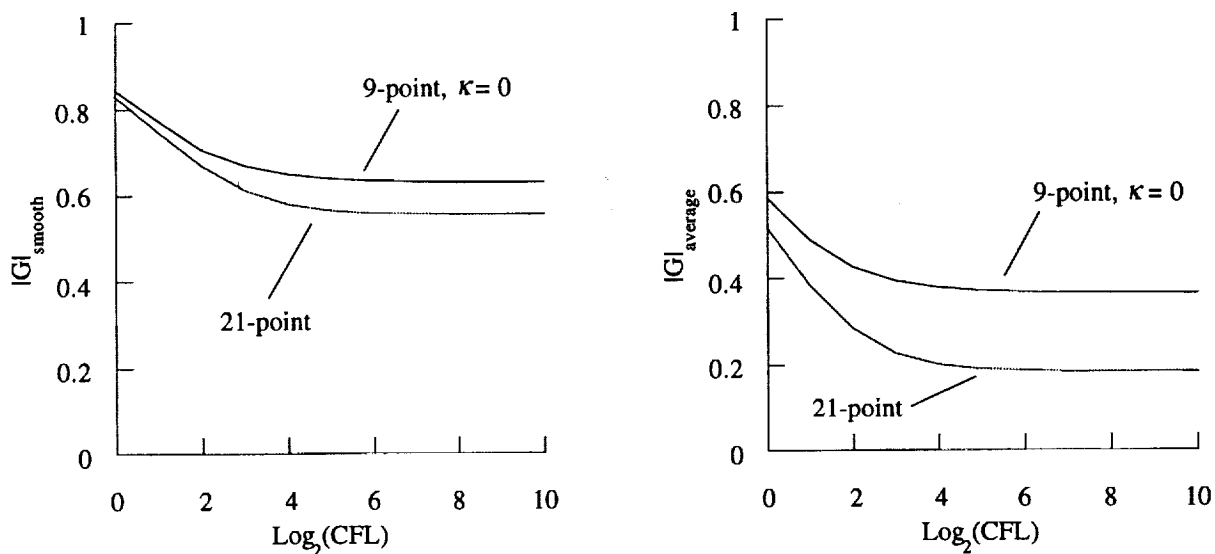


Figure 10. Smoothing factors and average amplification factors for the 21-point stencil and the 9-point stencil,  $\kappa = 0$ , for a flow angle of 45 degrees

## EULER RESULTS

Results for the two-dimensional Euler equations are now presented. Two test cases are used in this study. The first case is the subsonic flow in a channel with a  $3\% \sin^2 x$  bump. This case was chosen because the flow is nearly grid aligned in every cell. The channel length is three times the channel height and the length of the bump is equal to the channel height. A freestream Mach number of 0.3 is used. The grid used in this study consists of 157 points along the wall and 49 points normal to the

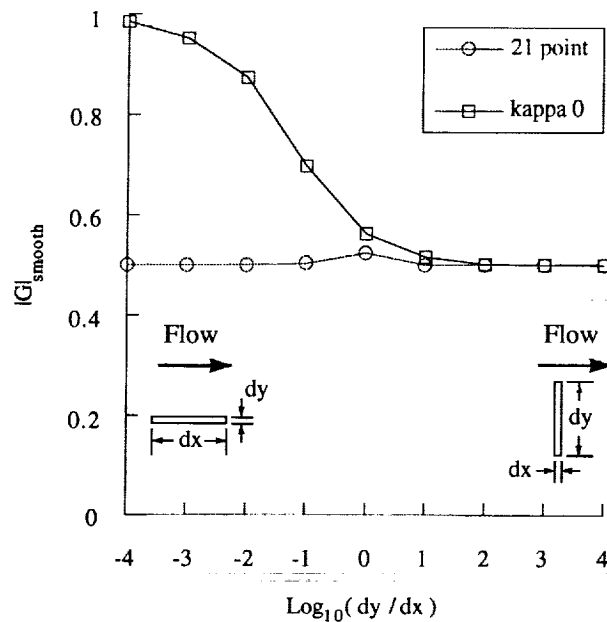


Figure 11. Effect of Grid Aspect Ratio on Smoothing Factor for Flow Aligned with Grid

wall and is shown in figure 12. The density contours for the converged solution using the 21-point stencil are also shown in figure 12. All of the cases utilize a 3-level V-cycle using 15 subiterations to solve the linear system at each level. One smoothing iteration is performed on each level except the coarsest grid where 3 smoothing steps are performed.

Convergence histories for this case using the 9-point stencil with  $\kappa = -1$  are shown in figure 14a. As the CFL increases, the convergence rate improves up to a CFL of about 10 after which the convergence degrades, eventually becoming unstable. As discussed above, when the CFL is increased, high frequency error modes approach neutral stability. The analysis, however, assumes the linear system is solved exactly at each time step which is generally not the case with only 15 subiterations. Therefore, the scheme may require a prohibitive number of subiterations to remain stable at high CFL numbers.

The convergence histories for the 9-point stencil with  $\kappa = 0$  are shown in figure 14b. Unlike the 9-point stencil with  $\kappa = -1$ , this stencil produces very good convergence rates as the CFL is increased. Note that there is little decrease in the spectral radius after a CFL of 100. This is consistent with the analysis shown in figure 10. The convergence histories for the 21-point stencil are shown in figure

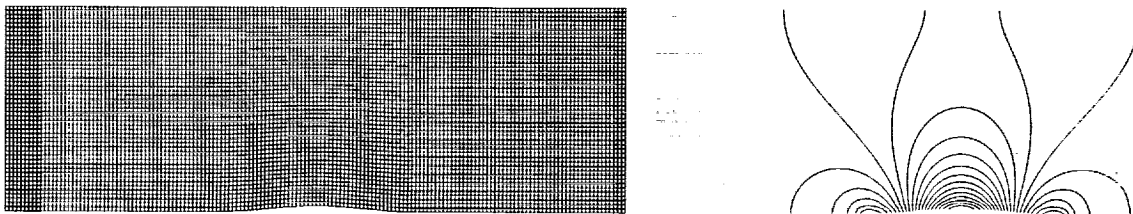


Figure 12. 3%  $\text{Sin}^2(x)$  bump grid and contours



14c and are very similar to the 9-point,  $\kappa = 0$  stencil. For this test case, in which the flow is aligned with the grid, both stencils have very good convergence properties.

To examine the behavior of the schemes with higher aspect ratio cells and when the flow is not aligned with the grid, a second test case is considered which is a NACA 0012 airfoil in a Mach = 0.8 freestream at 0 degrees angle-of-attack. The calculations were performed on a 65x25 c-grid which is shown in figure 13 along with the converged density contours obtained with the 21-point stencil. All cases were run using a 3-level V-cycle and 20 subiterations to solve the linear system.

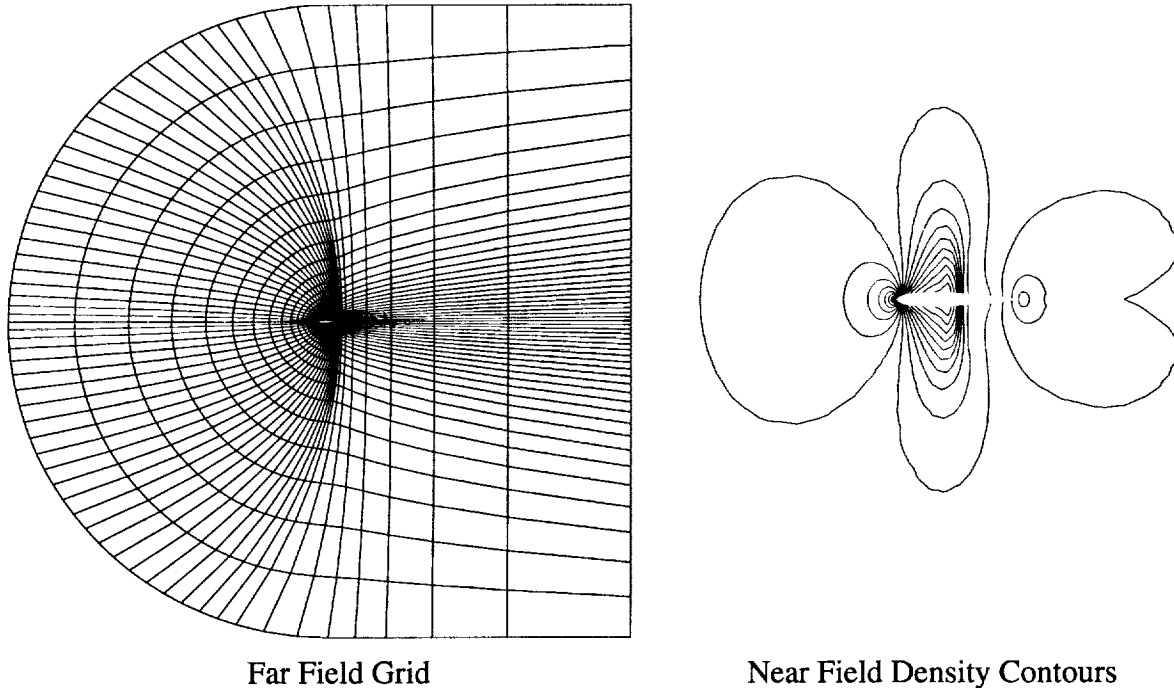
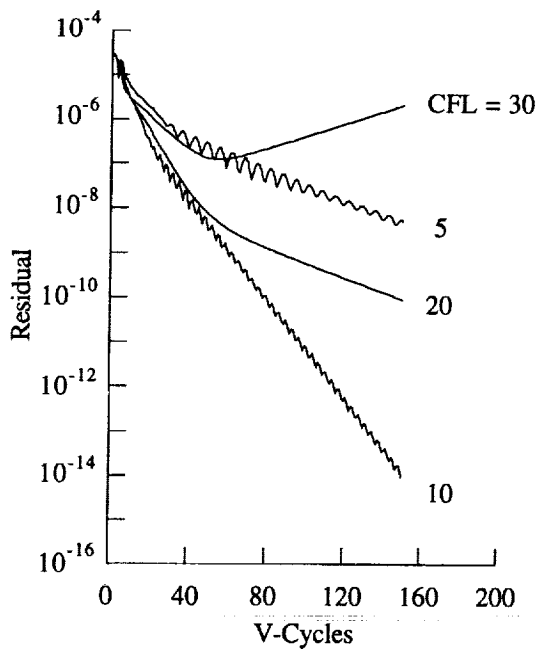


Figure 13. NACA 0012, Mach = 0.8,  $\alpha = 0^\circ$  grid and contours

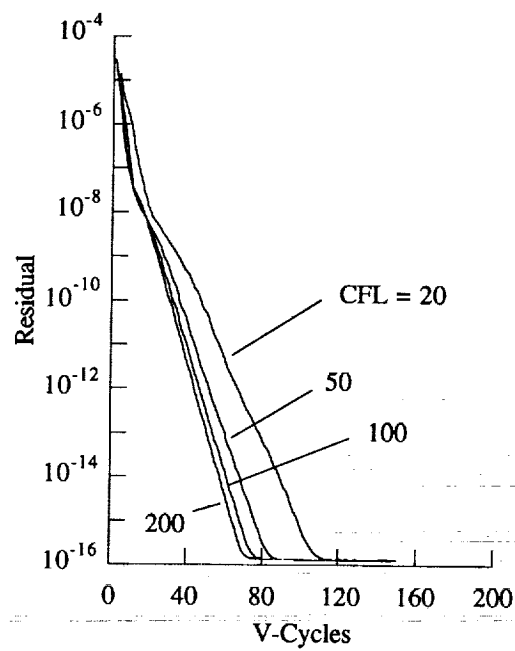
The convergence histories for both the 21-point stencil and 9-point stencil with  $\kappa = 0$  are shown in figure 14d. Only the  $\kappa = 0$  value is used because of the poor convergence properties of the  $\kappa = -1$  stencil. As shown, the 21-point stencil converges significantly faster than the 9-point  $\kappa = 0$  stencil. In particular, note that the number of multigrid cycles to reach a residual of  $10^{-16}$  using the 21-point stencil is about the same as for the channel flow. By contrast, the 9-point,  $\kappa = 0$  stencil shows a marked deterioration in performance compared to the channel flow case. These results are consistent with the analysis for flow angularity and cell aspect ratio effect presented above.

## DISCUSSION

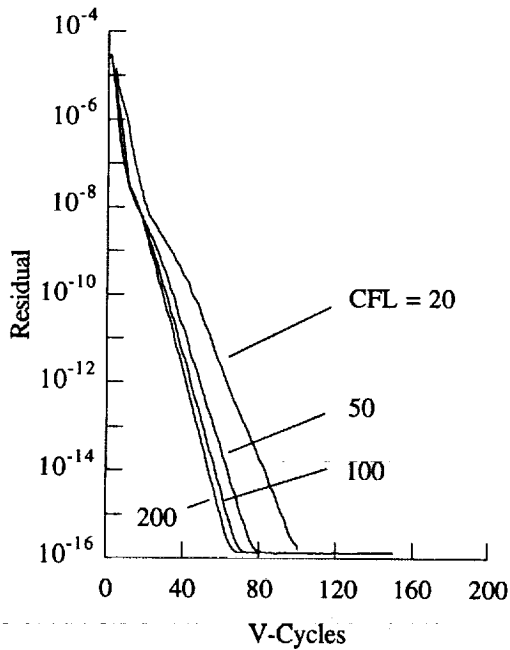
The analysis and computations presented indicate that the choice of data reconstruction for upwind methods can have a substantial effect on the multigrid performance for a given time advancement scheme. In particular, the popular 9-point,  $\kappa = -1$  stencil exhibits very poor multigrid conver-



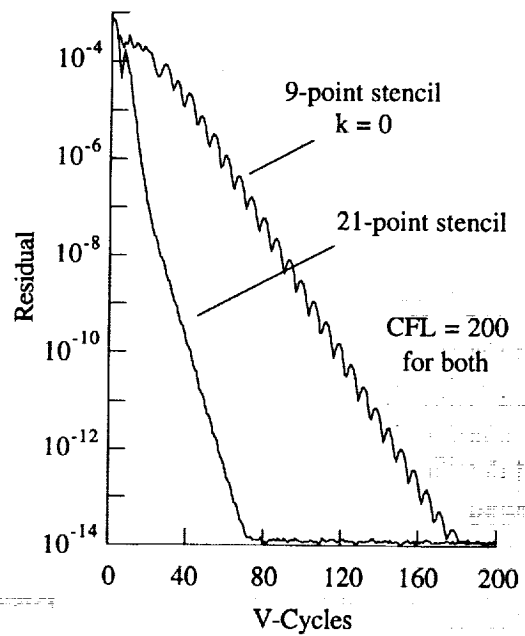
a)  $\text{Sin}^2x$  bump, 9-point,  $\kappa = -1$  stencil



b)  $\text{Sin}^2x$  bump, 9-point,  $\kappa = 0$  stencil



c)  $\text{Sin}^2x$  bump, 21-point stencil



d) Transonic airfoil, 21-point and 9-point,  $\kappa = 0$  stencil

Figure 14. Residual Histories

gence for high CFL numbers. The 9-point,  $\kappa = 0$  stencil has much better smoothing properties but still has difficulty damping the high frequency waves if the flow is not aligned with the grid. By using an interpolation operator based on Green's theorem, excellent smoothing properties are obtained for high CFL numbers regardless of the flow angularity as shown in figures 9 and 10. This has been shown through analysis and confirmed through numerical experiments.

## REFERENCES

1. Anderson, W. K., Thomas, J. L., and Van Leer, B., "A Comparison of Finite Volume Flux Vector Splitting for the Euler Equations," *AIAA J.*, Vol. 24, No. 9, Sept. 1986, pp. 1453-1460.
2. Vatsa, V. N., Sanetrik, M. D., and Parlette, E. B., "Development of a Flexible and Efficient Multigrid-Based Multiblock Flow Solver," AIAA Paper 93-0677, January 1993.
3. Ghaffari, F., Luckring, J. M., Thomas, J. L., and Bates, B. L., "Navier-Stokes Solutions About the F/A-18 Forebody-Leading-Edge Extension Configuration," *Journal of Aircraft*, Vol. 27, pp. 737-748, 1990.
4. Roberts, T. W. and Warren, G. P., "Analysis of Implicit Second-Order Upwind-Biased Stencils," AIAA Paper 93-3379, June 1993.
5. Anderson, W. K., "Grid Generation and Flow Solution Method for Euler Equations on Unstructured Grids," NASA Technical Memorandum 4295, April 1992.
6. Van Leer, B., "Flux Vector Splitting for the Euler Equations," *Lecture Notes in Physics*, Vol. 170, pp. 501-512, 1982.
7. Barth, T. J., and Jespersen, D. C., "The Design and Application of Upwind Schemes on Unstructured Meshes," AIAA Paper 89-0366, January 1989.
8. Frink, N. T., "Upwind Scheme for Solving the Euler Equations on Unstructured Tetrahedral Meshes," *AIAA J.*, Vol. 30, No. 1, Jan. 1992, pp. 70-77.
9. Mulder, W. A., and Van Leer, B., "Experiments with Implicit Upwind Methods for the Euler Equations," *J. Comput. Phys.*, Vol. 59, No. 2, June 1985, pp. 232-246.



ON THE PREDICTION OF MULTIGRID EFFICIENCY  
THROUGH LOCAL MODE ANALYSIS

S22-64

R.V. Wilson  
Department of Mechanical Engineering and Mechanics  
Old Dominion University  
Norfolk, Virginia

197682

p. 11

## ABSTRACT

A single grid local mode analysis is used to predict the smoothing properties of numerical schemes for solving the Navier-Stokes equations with factorization based on Stone's Strongly Implicit Method. Four difference approximations for the convection terms are considered, namely, hybrid, central, second-order upwind, and third-order upwind. Smoothing factors from the analysis are compared with practical convergence factors in a multigrid method for flow over a backward facing step and it is found that the local mode analysis correctly predicts the effects of Reynolds number and higher-order schemes.

## 1 INTRODUCTION

The successful use of multigrid methods to accelerate convergence rates is dependent on the ability of the numerical algorithm to dampen high frequency error components since these components cannot be resolved on coarser grids. High frequency components have short coupling ranges; therefore, their smoothing is a localized process meaning that only one isolated computational stencil need be analyzed and the effect of boundaries can be neglected. This is the approach of local mode analysis for the prediction of smoothing properties which was first introduced by Brandt [1] for various partial differential equations and numerical algorithms. Shaw and Sivaloganathan [2] extended this analysis to the SIMPLE pressure correction algorithm using alternating direction implicit (ADI) relaxation for the solution of the algebraic system of equations for varying Reynolds numbers and under-relaxation factors. Convection terms were approximated using a hybrid of first-order upwind and second-order central differencing.

The present paper uses local mode analysis to predict the smoothing properties of numerical algorithms for calculation of two-dimensional recirculating flows using higher-order difference schemes for convection terms introduced via deferred correction and Stone's Strongly Implicit Method for factorization of the resulting system of algebraic equations. Reynolds number and higher-order convection approximation effects are addressed and compared to multigrid results for laminar flow over a backward facing step.

## 2 THEORETICAL ANALYSIS

### 2.1 Governing equations

The equations governing steady, two-dimensional, incompressible flow can be written as:

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial}{\partial y}(\rho v) = 0 \quad (1)$$

$$\frac{\partial}{\partial x}(\rho u^2) + \frac{\partial}{\partial y}(\rho uv) = -\frac{\partial p}{\partial x} + \mu \left( \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) \quad (2)$$

$$\frac{\partial}{\partial x}(\rho uv) + \frac{\partial}{\partial y}(\rho v^2) = -\frac{\partial p}{\partial y} + \mu \left( \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) \quad (3)$$

where  $u$  and  $v$  are the velocity components in the  $x$  and  $y$  directions, respectively,  $p$  is the pressure,  $\mu$  is the absolute viscosity, and  $\rho$  is the density. Equations (1) - (3) represent the conservation of mass and momentum in the  $x$  and  $y$  directions, respectively.

The solution sequence is a predictor-corrector method which follows the SIMPLE algorithm of Patankar and Spalding [3]. Factorization of the system of equations is based on Stone's Strongly Implicit Method. The flow geometry and boundary conditions are shown in Figure 1.

The governing equations are discretized by integrating over a set of three staggered control volumes and the locations of variables are shown in Figure 2. The central control volume is for the pressure. Equation (2) can be discretized by integrating over the left-shifted control volume for the  $u$  component of velocity. This leads to:

$$a_p u_p = a_E u_E + a_W u_W + a_N u_N + a_S u_S + a_{EE} u_{EE} + a_{WW} u_{WW} + a_{NN} u_{NN} + a_{SS} u_{SS} - \frac{(p_P - p_W)}{h_x} + \left( \frac{\mu_o}{h_x h_y} \right) (v_N - v_{NW} + v_W - v_P) + \frac{\mu_o}{h_x^2} (u_E - 2u_P + u_W) \quad (4)$$

Equation (3) is discretized by integrating over the bottom-shifted control volume for the  $v$  component of velocity:

$$a_p v_p = a_E v_E + a_W v_W + a_N v_N + a_S v_S + a_{EE} v_{EE} + a_{WW} v_{WW} + a_{NN} v_{NN} + a_{SS} v_{SS} - \frac{(p_P - p_S)}{h_y} + \left( \frac{\mu_o}{h_x h_y} \right) (u_E - u_{SE} + u_S - u_P) + \frac{\mu_o}{h_y^2} (v_N - 2v_P + v_S) \quad (5)$$

where the  $a_i$  coefficients contain convection and diffusion terms, the subscripts of  $u$ ,  $v$ , and  $p$  refer to the location of the variables (see figure 2),  $h_x$  and  $h_y$  are the grid spacing in the  $x$  and  $y$  directions, respectively, and  $\mu_o$  is the absolute viscosity of the fluid, assumed constant.

### 2.2 Approximation of convection terms

The  $a_i$  coefficients in equations (4) and (5) are dependent on the approximation used for the convec-

tion terms. The present analysis investigates the hybrid, central, second-order upwind (2nd OU), and third-order upwind (3rd OU) approximation schemes.

The coefficients for the hybrid scheme have the following form:

$$a_E = \max \left\{ 0, \frac{\mu_o}{h_x^2} - \frac{|\rho u_o|}{2h_x} \right\} + \max \left\{ -\frac{\rho u_o}{h_x}, 0 \right\} \quad (6)$$

$$a_W = \max \left\{ 0, \frac{\mu_o}{h_x^2} - \frac{|\rho u_o|}{2h_x} \right\} + \max \left\{ \frac{\rho u_o}{h_x}, 0 \right\} \quad (7)$$

$$a_N = \max \left\{ 0, \frac{\mu_o}{h_y^2} - \frac{|\rho v_o|}{2h_y} \right\} + \max \left\{ -\frac{\rho v_o}{h_y}, 0 \right\} \quad (8)$$

$$a_S = \max \left\{ 0, \frac{\mu_o}{h_y^2} - \frac{|\rho v_o|}{2h_y} \right\} + \max \left\{ \frac{\rho v_o}{h_y}, 0 \right\} \quad (9)$$

$$a_{EE} = a_{WW} = a_{NN} = a_{SS} = 0 \quad (10)$$

$$a_P = \sum a_i \quad (11)$$

where the sum for  $a_P$  is taken over the  $a$  coefficients,  $u_o$  and  $v_o$  are the frozen velocity components due to the linearization of equations (2) and (3), the  $\max\{a,b\}$  operator selects the maximum of the arguments  $a$  and  $b$ , and  $||$  represents the absolute value.

The coefficients for the higher-order schemes have the following general form:

$$a_E = \frac{\mu_o}{h_x^2} - \frac{\rho u_o}{2h_x} + f_1 \left[ \max \left\{ \frac{\rho u_o}{h_x}, 0 \right\} + \max \left\{ -\frac{\rho u_o}{h_x}, 0 \right\} \right] + f_2 \max \left\{ -\frac{\rho u_o}{h_x}, 0 \right\} \quad (12)$$

$$a_W = \frac{\mu_o}{h_x^2} + \frac{\rho u_o}{2h_x} + f_1 \left[ \max \left\{ \frac{\rho u_o}{h_x}, 0 \right\} + \max \left\{ -\frac{\rho u_o}{h_x}, 0 \right\} \right] + f_2 \max \left\{ \frac{\rho u_o}{h_x}, 0 \right\} \quad (13)$$

$$a_N = \frac{\mu_o}{h_y^2} - \frac{\rho v_o}{2h_y} + f_1 \left[ \max \left\{ \frac{\rho v_o}{h_y}, 0 \right\} + \max \left\{ -\frac{\rho v_o}{h_y}, 0 \right\} \right] + f_2 \max \left\{ -\frac{\rho v_o}{h_y}, 0 \right\} \quad (14)$$

$$a_S = \frac{\mu_o}{h_y^2} + \frac{\rho v_o}{2h_y} + f_1 \left[ \max \left\{ \frac{\rho v_o}{h_y}, 0 \right\} + \max \left\{ -\frac{\rho v_o}{h_y}, 0 \right\} \right] + f_2 \max \left\{ \frac{\rho v_o}{h_y}, 0 \right\} \quad (15)$$

$$a_{EE} = -f_1 \max \left\{ -\frac{\rho u_o}{h_x}, 0 \right\} \quad (16)$$

$$a_{WW} = -f_1 \max \left\{ \frac{\rho u_o}{h_x}, 0 \right\} \quad (17)$$

$$a_{NN} = -f_1 \max \left\{ -\frac{\rho v_o}{h_y}, 0 \right\} \quad (18)$$

$$a_{SS} = -f_1 \max \left\{ \frac{\rho v_o}{h_y}, 0 \right\} \quad (19)$$

$$a_P = \sum a_i \quad (20)$$

The values of  $f_1$  and  $f_2$  depend on the higher-order method to be used. For central differencing,  $f_1 = f_2 = 0$ , for 2nd OU differencing,  $f_1 = 1/2$  and  $f_2 = 1$ , and for 3rd OU differencing,  $f_1 = 1/8$  and  $f_2 = 1/4$ .

### 2.3 Local mode analysis

The Strongly Implicit Method (SIP) by Stone [4] solves the algebraic equations shown in equations (4) and (5) in a fully implicit manner. All variables are treated as unknowns, as opposed to line-relaxation methods which consider only lines of constant  $x$  or  $y$  as unknowns while sweeping through the computational domain. In the local mode analysis outlined below, the variables that are updated at the end of a relaxation sweep will be denoted by a dot over the variable, such as  $\dot{u}_p$ , while those from the beginning of the relaxation sweep or those unchanged by the current relaxation sweep will be written as above, such as  $u_p$ . Higher-order approximations for the convection terms are introduced via deferred correction (see Khosla and Rubin [5]). In this procedure, the  $a_i$  coefficients are calculated initially using equations (6) - (11) for the hybrid scheme. As the solution proceeds, the higher-order scheme is slowly introduced via corrections to the source terms. At the end when the solution is fully converged, the coefficients are effectively those of the higher-order scheme outlined in equations (12) - (20). The base hybrid coefficients will be denoted by an additional subscript  $h$ , such as  $a_p|_h$ , while the higher-order coefficients will not have an additional subscript and will be written as above, such as  $a_p$ .

The relaxation of equation (4) (the discretized  $x$  momentum equation), with under-relaxation and deferred correction can be written as:

$$\begin{aligned} a_p|_h \dot{u}_p &= a_p|_h u_p + r_m (a_E|_h \dot{u}_E + a_W|_h \dot{u}_W + a_N|_h \dot{u}_N + a_S|_h \dot{u}_S - a_p|_h u_p) \\ &+ r_m \left\{ \frac{-(p_p - p_w)}{h_x} + \left( \frac{\mu_o}{h_x h_y} \right) (v_N - v_{NW} + v_W - v_p) + \frac{\mu_o}{h_x^2} (\dot{u}_E - 2\dot{u}_p + \dot{u}_W) \right\} \\ &+ r_l r_m [ (a_E - a_E|_h) (u_E - u_p) + (a_W - a_W|_h) (u_W - u_p) + (a_N - a_N|_h) (u_N - u_p) + (a_S - a_S|_h) (u_S - u_p) \\ &+ a_{EE} (u_{EE} - u_p) + a_{WW} (u_{WW} - u_p) + a_{NN} (u_{NN} - u_p) + a_{SS} (u_{SS} - u_p) ] \end{aligned} \quad (21)$$

where  $r_m$  is the under-relaxation factor for the  $x$  and  $y$  momentum equations,  $r_l$  is the relaxation factor for introducing higher-order coefficients and is set to unity for the analysis. The exact solution for  $U$ ,  $V$ , and  $P$  also satisfies equation (21). If an equation written with the exact solution for  $U$ ,  $V$ , and  $P$  is subtracted from equation (21), which is written in terms of the approximate solution,  $u$ ,  $v$ , and  $p$ , the error in the solution can be introduced. The error has components defined as;  $\epsilon^u = U - u$ ,  $\epsilon^v = V - v$ , and  $\epsilon^p = P - p$ .

Equation (21) written in terms of the error becomes:

$$\begin{aligned} a_p|_h \epsilon_p^u &= a_p|_h \epsilon_p^u + r_m (a_E|_h \epsilon_E^u + a_W|_h \epsilon_W^u + a_N|_h \epsilon_N^u + a_S|_h \epsilon_S^u - a_p|_h \epsilon_p^u) \\ &+ r_m \left\{ \frac{-(\epsilon_p^p - \epsilon_w^p)}{h_x} + \left( \frac{\mu_o}{h_x h_y} \right) (\epsilon_N^v - \epsilon_{NW}^v + \epsilon_W^v - \epsilon_p^v) + \frac{\mu_o}{h_x^2} (\epsilon_E^u - 2\epsilon_p^u + \epsilon_W^u) \right\} \\ &+ r_l r_m [ (a_E - a_E|_h) (\epsilon_E^u - \epsilon_p^u) + (a_W - a_W|_h) (\epsilon_W^u - \epsilon_p^u) + (a_N - a_N|_h) (\epsilon_N^u - \epsilon_p^u) + (a_S - a_S|_h) (\epsilon_S^u - \epsilon_p^u) \\ &+ a_{EE} (\epsilon_{EE}^u - \epsilon_p^u) + a_{WW} (\epsilon_{WW}^u - \epsilon_p^u) + a_{NN} (\epsilon_{NN}^u - \epsilon_p^u) + a_{SS} (\epsilon_{SS}^u - \epsilon_p^u) ] \end{aligned} \quad (22)$$



Since the continuous governing equations (1) - (3) have been linearized during the discretization, a single Fourier component of the error can be considered as:

$$\begin{aligned}\varepsilon_p^u &= \alpha_\theta^u e^{i\left[\theta_1 \frac{x}{h_x} + \theta_2 \frac{y}{h_y}\right]} \\ \varepsilon_w^u &= \alpha_\theta^u e^{i\left[\theta_1 \frac{(x-h_x)}{h_x} + \theta_2 \frac{y}{h_y}\right]} \\ \varepsilon_s^u &= \alpha_\theta^u e^{i\left[\theta_1 \frac{x}{h_x} + \theta_2 \frac{(y-h_y)}{h_y}\right]}\end{aligned}\quad (23)$$

where  $i = \sqrt{-1}$ ,  $\theta_1$  and  $\theta_2$  are the components of the phase angle vector,  $\alpha_\theta$ , which is the error amplitude of the single Fourier mode  $\theta_1$ ,  $\theta_2$ . Similar expressions exist for other grid points to the east and north and for the variables  $v$  and  $p$ . Substituting the single Fourier modes into equation (22) and dividing through by  $e^{i[\theta_1 x/h_x + \theta_2 y/h_y]}$ , equation (22) becomes:

$$\begin{aligned}\alpha_\theta^u \left\{ a_{p|_h} - r_m \left( a_{E|_h} e^{i\theta_1} + a_{W|_h} e^{-i\theta_1} + a_{N|_h} e^{i\theta_2} + a_{S|_h} e^{-i\theta_2} - \frac{4\mu_o s_1^2}{h_x^2} \right) \right\} = \\ \alpha_\theta^u \{ a_{p|_h} (1 - r_m) + r_1 r_m [ (a_E - a_{E|_h}) (e^{i\theta_1} - 1) + (a_W - a_{W|_h}) (e^{-i\theta_1} - 1) + (a_N - a_{N|_h}) (e^{i\theta_2} - 1) \\ + (a_S - a_{S|_h}) (e^{-i\theta_2} - 1) + a_{EE} (e^{2i\theta_1} - 1) + a_{WW} (e^{-2i\theta_1} - 1) + a_{NN} (e^{2i\theta_2} - 1) + a_{SS} (e^{-2i\theta_2} - 1) ] \\ - \frac{4r_m \mu_o s_1 s_2}{h_x h_y} \alpha_\theta^v - \frac{2r_m s_1 i}{h_x} \alpha_\theta^p \}\end{aligned}\quad (24)$$

where:  $s_1 = \sin(\theta_1/2)$   
 $s_2 = \sin(\theta_2/2)$

Equation (24) can be written in a more compact form, if the following variables are defined:

$$\begin{aligned}\eta^u &= a_{p|_h} - r_m \left( a_{E|_h} e^{i\theta_1} + a_{W|_h} e^{-i\theta_1} + a_{N|_h} e^{i\theta_2} + a_{S|_h} e^{-i\theta_2} - \frac{4\mu_o s_1^2}{h_x^2} \right) \\ v &= a_{p|_h} (1 - r_m) + r_1 r_m [ (a_E - a_{E|_h}) (e^{i\theta_1} - 1) + (a_W - a_{W|_h}) (e^{-i\theta_1} - 1) + (a_N - a_{N|_h}) (e^{i\theta_2} - 1) \\ &\quad + (a_S - a_{S|_h}) (e^{-i\theta_2} - 1) + a_{EE} (e^{2i\theta_1} - 1) + a_{WW} (e^{-2i\theta_1} - 1) + a_{NN} (e^{2i\theta_2} - 1) + a_{SS} (e^{-2i\theta_2} - 1) ] \\ \phi &= \frac{4r_m \mu_o s_1 s_2}{h_x h_y} \\ \zeta^u &= \frac{2r_m s_1 i}{h_x}\end{aligned}$$

Equation (24) can then be written as:

$$\alpha_\theta^u = \frac{1}{\eta^u} (v \alpha_\theta^u - \phi \alpha_\theta^v - \zeta^u \alpha_\theta^p) \quad (25)$$

Following the same procedure for equation (5) (the discretized y momentum equation) yields:

$$\dot{\alpha}_{\theta}^v = \frac{1}{\eta^v} (v\alpha_{\theta}^v - \phi\dot{\alpha}_{\theta}^u - \zeta^v\alpha_{\theta}^p) \quad (26)$$

where:

$$\eta^v = a_P|_h - r_m \left( a_E|_h e^{i\theta_1} + a_W|_h e^{-i\theta_1} + a_N|_h e^{i\theta_2} + a_S|_h e^{-i\theta_2} - \frac{4\mu_o s_2^2}{h_y^2} \right)$$

$$\zeta^v = \frac{2r_m s_2 i}{h_y}$$

Equations (25) and (26) can be combined to give the amplification matrix  $A_I$  for the complete operation. This yields:

$$\begin{bmatrix} \dot{\alpha}_{\theta}^u \\ \dot{\alpha}_{\theta}^v \\ \dot{\alpha}_{\theta}^p \end{bmatrix} = \begin{bmatrix} \left\{ \frac{v}{\eta^u} \right\} & \left\{ \frac{-\phi}{\eta^u} \right\} & \left\{ \frac{-\zeta^u}{\eta^u} \right\} \\ \left\{ \frac{-\phi v}{\eta^u \eta^v} \right\} & \left\{ \frac{\phi^2}{\eta^u \eta^v} + \frac{v}{\eta^v} \right\} & \left\{ \frac{\phi \zeta^u}{\eta^u \eta^v} - \frac{\zeta^v}{\eta^v} \right\} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_{\theta}^u \\ \alpha_{\theta}^v \\ \alpha_{\theta}^p \end{bmatrix}$$

In compact form:

$$\dot{\alpha}_{\theta} = A_I \alpha_{\theta} \quad (27)$$

Equation (27) yields the amplification matrix defining how the amplitude of the Fourier mode, with phase angles  $\theta_1$  and  $\theta_2$ , is amplified during relaxation of the  $x$  and  $y$  momentum equations.

The SIMPLE pressure correction of Patankar and Spalding [3] follows the relaxation of the  $x$  and  $y$  momentum equations. A single dot over a variable will denote a value at the completion of the relaxation of the  $x$  and  $y$  momentum equations. A double dot over a variable will denote a value at the completion of the pressure correction. The variables  $u$ ,  $v$ , and  $p$  are corrected following Shaw and Sivaloganathan [2]:

$$\ddot{u}_p = \dot{u}_p - \frac{r_{uv}}{a_p^u h_x} (\delta p_p - \delta p_w) \quad (28)$$

$$\ddot{v}_p = \dot{v}_p - \frac{r_{uv}}{a_p^v h_y} (\delta p_p - \delta p_s) \quad (29)$$

$$\ddot{p}_p = \dot{p}_p + r_p \delta p_p \quad (30)$$

where  $r_{uv}$  is the relaxation factor for correcting  $u$  and  $v$  velocities and  $r_p$  is the relaxation factor for updating pressure. The value  $\delta p$  is a pressure increment such that the velocity field  $\dot{u}$  and  $\dot{v}$  will satisfy conservation of mass. It is obtained by discretizing equation (3) and substituting for the velocities and corrections given in equations (28) - (30). This yields an equation for the pressure correction:

$$a_p^p \delta p_p = a_p^N \delta p_N + a_p^S \delta p_S + a_p^E \delta p_E + a_p^W \delta p_W - \frac{1}{h_x} (\dot{u}_E - \dot{u}_P) - \frac{1}{h_y} (\dot{v}_N - \dot{v}_P) \quad (31)$$

where:  $\alpha_N^p = \frac{1}{a_p^v h_y^2} = \alpha_s^p$

$$\alpha_E^p = \frac{1}{a_p^u h_x^2} = \alpha_w^p$$

$$\alpha_P^p = \alpha_N^p + \alpha_s^p + \alpha_E^p + \alpha_w^p$$

Equations (28) - (31) can be written in compact form as:

$$\ddot{u}_p = \dot{u}_p - \frac{r_{uv}}{a_p^u} \delta_h^x \delta p_h \quad (32)$$

$$\ddot{v}_p = \dot{v}_p - \frac{r_{uv}}{a_p^v} \delta_h^y \delta p_h \quad (33)$$

$$\ddot{p}_h = \dot{p}_h + r_p \delta p_h \quad (34)$$

$$P_h \delta p_h = \delta_h^x \dot{u}_h + \delta_h^y \dot{v}_h \quad (35)$$

If equation (35) is solved for  $\delta p_h$  and used in equation (32)-(34):

$$\ddot{u}_h = \dot{u}_h - \frac{r_{uv}}{a_p^u} \delta_h^x P_h^{-1} (\delta_h^x \dot{u}_h + \delta_h^y \dot{v}_h) \quad (36)$$

$$\ddot{v}_h = \dot{v}_h - \frac{r_{uv}}{a_p^v} \delta_h^y P_h^{-1} (\delta_h^x \dot{u}_h + \delta_h^y \dot{v}_h) \quad (37)$$

$$\ddot{p}_h = \dot{p}_h + r_p P_h^{-1} (\delta_h^x \dot{u}_h + \delta_h^y \dot{v}_h) \quad (38)$$

This assumes that the pressure correction equation has been solved exactly. As before, the error is introduced by writing equations (36)-(38) using the exact solution and then subtracting the result from equations (36)-(38) respectively. The errors become:

$$\ddot{\epsilon}_h^u = \dot{\epsilon}_h^u - \frac{r_{uv}}{a_p^u} \delta_h^x P_h^{-1} (\delta_h^x \dot{\epsilon}_h^u + \delta_h^y \dot{\epsilon}_h^v) \quad (39)$$

$$\ddot{\epsilon}_h^v = \dot{\epsilon}_h^v - \frac{r_{uv}}{a_p^v} \delta_h^y P_h^{-1} (\delta_h^x \dot{\epsilon}_h^u + \delta_h^y \dot{\epsilon}_h^v) \quad (40)$$

$$\ddot{\epsilon}_h^p = \dot{\epsilon}_h^p + r_p P_h^{-1} (\delta_h^x \dot{\epsilon}_h^u + \delta_h^y \dot{\epsilon}_h^v) \quad (41)$$

The Fourier components of the error can be substituted as before to give the  $A_2$  amplification matrix which governs the amplification of errors during the pressure correction phase:

$$\begin{bmatrix} \ddot{\alpha}_\theta^u \\ \ddot{\alpha}_\theta^v \\ \ddot{\alpha}_\theta^p \end{bmatrix} = \begin{bmatrix} \left\{ 1 + \frac{4r_{uv}s_1^2}{a_p^u P_h h_x^2} \right\} & \left\{ \frac{4r_{uv}s_1 s_2}{a_p^u P_h h_x h_y} \right\} \\ \left\{ \frac{4r_{uv}s_1 s_2}{a_p^v P_h h_x h_y} \right\} & \left\{ 1 + \frac{4r_{uv}s_2^2}{a_p^v P_h h_y^2} \right\} \\ \left\{ \frac{2r_p s_1}{P_h h_x} \right\} & \left\{ \frac{2r_p s_2}{P_h h_y} \right\} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \begin{bmatrix} \dot{\alpha}_\theta^u \\ \dot{\alpha}_\theta^v \\ \dot{\alpha}_\theta^p \end{bmatrix}$$

In compact form:

$$\ddot{\alpha}_\theta = A_2 \dot{\alpha}_\theta \quad (42)$$

where:  $\hat{P}_h = a_E^p e^{i\theta_1} + a_W^p e^{-i\theta_1} + a_N^p e^{i\theta_2} + a_S^p e^{-i\theta_2} - a_P^p$

Equation (42) gives the amplification matrix defining how the amplitude of the Fourier mode with phase angles  $\theta_1$  and  $\theta_2$  is amplified during the pressure correction phase of the algorithm. The amplification matrix for relaxation of the  $x$  and  $y$  momentum equations and the pressure correction is obtained by combining equation (27) and (42) as:

$$\ddot{\alpha}_\theta = A_2 A_1 \alpha_\theta = A \alpha_\theta \quad (43)$$

## 2.4 Smoothing factor

The smoothing factor is a measure of the worst reduction of the high frequency error components for one complete relaxation sweep. It is calculated as the largest eigenvalue of the amplification matrix  $A$  given by equation (43) for the Fourier modes  $\theta_1$  and  $\theta_2$  in the high frequency range defined as:  $\pi/2 \leq |\theta_1| \leq \pi$  and  $\pi/2 \leq |\theta_2| \leq \pi$ .

## 3 RESULTS

To test the predictive capability of the LMA presented in Section 2, flow over a backward facing step (BFS) was computed using a multigrid code based on the FAS-FMG (full approximation storage - full multi-grid) algorithm proposed by Brandt [1]. Higher-order schemes were introduced through deferred correction only on the finest of three grids with constant grid spacing in the  $x$  and  $y$  directions. The grid sizes from coarsest to finest grid are,  $n_x \times n_y = 66 \times 18$ ,  $130 \times 34$ , and  $258 \times 66$ , where  $n_x$  is the number of grid points in the  $x$  direction and  $n_y$  is the number of grid points in the  $y$  direction. Smoothing properties on the two coarser grids are identical for the four schemes since hybrid coefficients were used on these grids. Local mode analysis was used to estimate the smoothing factor on the finest grid and this result was compared to the number of work units to reach convergence for the multigrid result. The work units (WU) and convergence factor (CF) are indicators of the smoothing properties of the algorithm and numerical scheme. The work units for a two-dimensional problem with grid refinement in the  $x$  and  $y$  directions are defined as:

$$WU = \sum_{i=1}^N \tau_i 2^{2(i-N)} \quad (44)$$

where  $\tau_i$  is the number of iterations on the  $i^{th}$  grid at convergence,  $i = 1$  for the coarsest grid, and  $i = N$  for the finest grid. The convergence factor is defined as:

$$CF = (r_f/r_i)^{1/\Delta WU} \quad (45)$$

where  $r_i$  is the initial norm of the residuals of the  $x$  momentum,  $y$  momentum, and pressure correction equation on the fine grid,  $r_f$  is the norm of the residuals at convergence on the fine grid, and  $\Delta WU$  is the

change in work units on the fine grid.

The BFS flow was solved for Reynolds numbers (based on the upstream channel height) of 100, 250, and 400 using the hybrid, central, 2nd OU, and 3rd OU schemes. The work units for convergence and convergence factors are shown in Table 1. The smoothing factors were calculated for conditions identical to the finest grid of the multigrid results (the same grid spacing, relaxation factors, density, etc.). The only parameter varied was the frozen velocity components needed to calculate the coefficients in equations (6) - (20) of the various schemes. The maximum velocity components provide an upper bound for the smoothing factor which will dominate the smoothing properties since it was found that as the cell-Reynolds number (Reynolds number with the length scale based on the grid spacing) approaches zero, corresponding to regions where the velocity approaches zero, the smoothing factor also decreases. An estimate of the maximum velocity components for the BFS flow is  $u_o = 1.5$  at  $y = 0.25$  at the inlet, and  $v_o = 0.15$  near recirculation regions. Three principal flow directions are considered with velocity components given by:  $u_o, v_o = (0, 0.15), (1.5, 0.15),$  and  $(1.5, 0)$ . The SIP method exhibits symmetry about the  $x$  and  $y$  axis so that other flow direction results can be obtained from the three principal flow direction results. For example, the smoothing factors for  $u_o, v_o = (-1.5, 0.15), (-1.5, -0.15),$  and  $(1.5, -0.15)$  are equal to the smoothing factor for  $u_o, v_o = (1.5, 0.15)$ . The smoothing factor is then defined as the largest eigenvalue of the amplification matrix  $A$ , defined by equation (43), for the three flow directions while restricting the phase angles to the high frequency range. Results from the three principal flow directions show that the flow direction  $u_o, v_o = (1.5, 0.15)$  produced the largest eigenvalue for all Reynolds numbers, and thus the smoothing factor was based on this flow direction. The computed smoothing factors are shown in Table 2.

The results of Table 2 show that as the cell-Reynolds number in the LMA is increased, the smoothing factor also increases for the four schemes. More work units will be required to smooth the high frequency error components. The results in Table 1 show that the work units increase and the convergence factor deteriorates as the Reynolds number increases. For the  $Re = 100$  results, the LMA predicts that the smoothing properties of the hybrid, central, and 3rd OU will be virtually identical while that of the 2nd OU will be slightly worse. The multigrid results confirm this prediction. For the  $Re = 250$  and 400 results, the LMA predicts that the hybrid difference scheme will have the best smoothing properties while the central difference scheme will have the worst, and the 2nd OU and 3rd OU difference schemes should be similar with the 3rd OU difference scheme slightly better. The multigrid results confirm these predictions with the exception being that the 2nd OU difference scheme results converged in slightly less number of work units when compared to the 3rd OU difference scheme. Their convergence factors are similar.

Table I: Work Units/Convergence Factors of Multigrid Results

Difference Scheme	$Re = 100$	$Re = 250$	$Re = 400$
Hybrid	59/0.868	137/0.930	321/0.981
Central	58/0.873	166/0.964	569/0.993
2nd OU	63/0.891	148/0.952	421/0.988
3rd OU	58/0.875	152/0.956	428/0.988

Table II: Smoothing Factors from Local Mode Analysis

Difference Scheme	$Re = 100$	$Re = 250$	$Re = 400$
Hybrid	0.902	0.924	0.931
Central	0.910	0.950	0.968
2nd OU	0.931	0.946	0.950
3rd OU	0.899	0.926	0.935
$Re_{\Delta x}/Re_{\Delta y}$	11.72/0.47	29.30/1.18	46.88/1.88

## CONCLUSION

Local mode analysis was performed using four schemes for the approximation of convection terms: hybrid, central, second-order upwind, and third-order upwind, over a range of cell-Reynolds numbers. The smoothing factors from this analysis were compared with actual multigrid results for flow over a backward facing step to test the predictive capability of local mode analysis. It was found that this analysis is useful in predicting the smoothing properties of the four schemes along with the effect of flow Reynolds number. This analysis could be extended to predict optimum relaxation factors, grid aspect ratios, and other solution algorithms.

## ACKNOWLEDGEMENTS

The author would like to thank Dr. A.O. Demuren for his helpful comments. Computations were performed on the Cray computers at NASA Langley Research Center, Hampton, Virginia.

## REFERENCES

1. Brandt, A.: Multi-level Adaptive Solutions to Boundary-Value Problems, *Math. Comput.*, vol. 31, no. 138, 1977, pp. 330-390.
2. Shaw, G.J.; and Sivaloganathan, S.: On the Smoothing Properties of the SIMPLE Pressure-Correction Algorithm, *Int. J. Numer. Methods Fluids*, vol 8, 1988, pp. 441-461.
3. Patankar, S.V.; and Spalding, D.B.: A Novel Finite Difference Formulation for Differential Expressions Involving both First and Second Derivatives. *Int. J. Numer. Methods Eng.*, vol 4, 1972, pp. 551-559.
4. Stone, H.L.; Iterative Solution of Implicit Approximation of Multidimensional Partial Differential Equations. *SIAM J. on Num. Analysis*, vol. 5, 1968, pp.530-558.
5. Khosla, P.K.; and Rubin, S.G.; A Diagonally Dominant Second-Order Accurate Implicit Scheme. *Computers and Fluids*, vol 2, 1974, pp.207-209.

# FIGURES

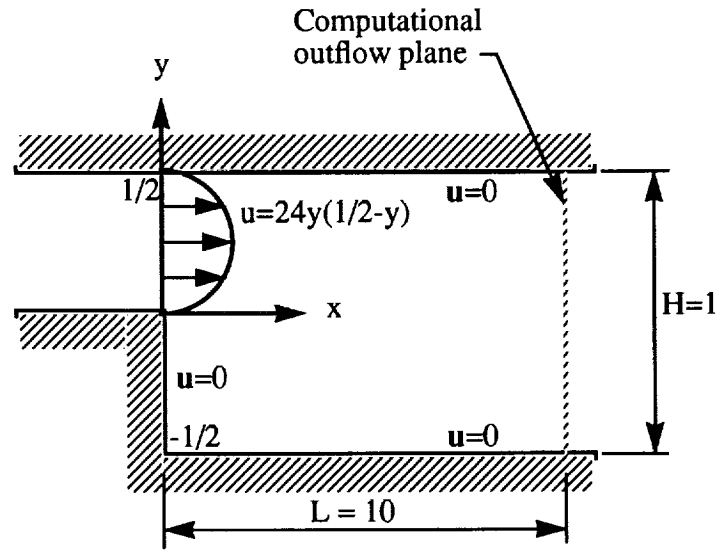


Figure 1. Geometry and boundary conditions for flow over a backward facing step.

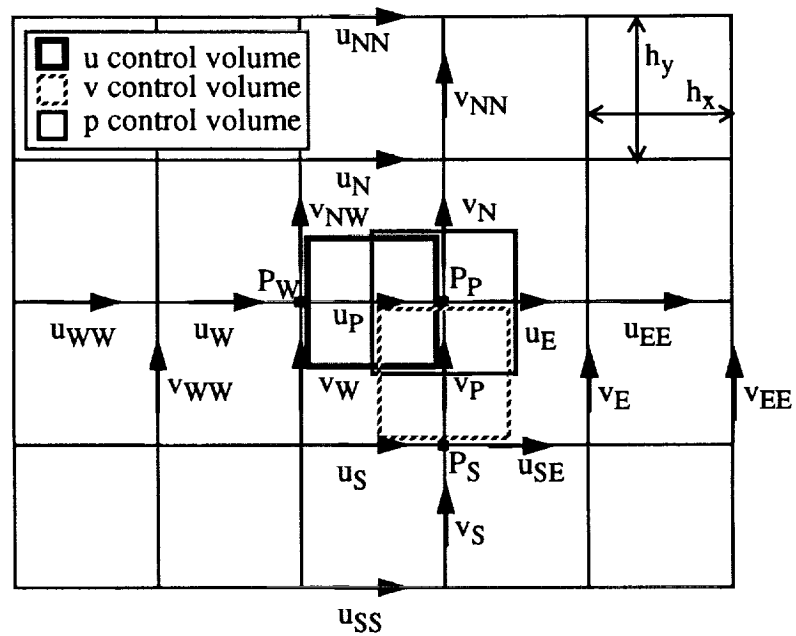


Figure 2. Location of variables for staggered grid.





NUMERICAL STUDY OF A MULTIGRID METHOD  
WITH FOUR SMOOTHING METHODS  
FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS  
IN GENERAL COORDINATES

N 94-21487

523-24

197583

p. 18

S. Zeng and P. Wesseling  
Faculty of Technical Mathematics and Informatics  
Delft University of Technology  
P.O. Box 5031, 2600 GA Delft, The Netherlands

SUMMARY

The performance of a linear multigrid method using four smoothing methods, called SCGS, CLGS, SILU and CILU, is investigated for the incompressible Navier-Stokes equations in general coordinates, in association with Galerkin coarse grid approximation. Robustness and efficiency are measured and compared by application to test problems. The numerical results show that CILU is the most robust, SILU the least, with CLGS and SCGS in between. CLGS is the best in efficiency, SCGS and CILU follow, and SILU is the worst.

INTRODUCTION

Robustness and efficiency of a multigrid method are strongly influenced by the smoother used. Because there are so many factors influencing robustness and efficiency, it is hard to say in general which method is the most appropriate choice for certain applications. In this paper, we study four smoothing methods for the incompressible Navier-Stokes equations in general coordinates, namely the SCGS (Symmetrical Coupled Gauß-Seidel [18]), CLGS (Collective Line Gauß-Seidel, adapted from SCAL [16]), SILU (Scalar ILU, or TILU in [23]) and CILU (Collective ILU [30]), respectively, which are used in a linear multigrid method. Galerkin coarse grid approximation (GCA) is used. An elementary introduction to GCA can be found in [21]. Application to the Navier-Stokes equations is discussed in [29] and [31].

The multigrid method using the above four smoothers solves the velocity and the pressure simultaneously (collectively). Decoupled solution is also used in practice, solving the velocity and the pressure separately. A comparison is given in [1] of multigrid methods using coupled solution with SCGS and CLSOR (Coupled Line Successive Over Relaxation) smoothing and multigrid methods using decoupled solution. Comparisons are presented in [13] and [14] for multigrid methods using the SCGS method and methods using the uncoupled MGPC method (Multigrid Pressure Correction) and the SPC (SIMPLE Pressure Correction) smoothing methods by means of local Fourier analysis as well as numerical experiments. It is stated in [17] that it is advantageous to use the coupled approach. However, both coupled and decoupled solution methods are widely used in practice.

The comparisons mentioned above are made for nonlinear multigrid methods in which the coarse grid operators are computed by using discretization coarse grid approximation (DCA). The relative merits of DCA and GCA are discussed in [21]. A nonlinear multigrid using DCA for the applications discussed in the present paper is presented in [8], [9], [10]. Here we apply GCA. Our main reason is that discretization of the Navier-Stokes equations in general coordinates on a staggered grid is a complicated affair, and GCA enables us to completely separate discretization and multigrid solution. In this paper, attention is focussed on smoothing.

## EQUATIONS AND DISCRETIZATION

The incompressible Navier-Stokes equations in tensor formulation in general curvilinear coordinates are given as follows:

$$\frac{\partial U^\alpha}{\partial t} + U^\beta U_{,\beta}^\alpha + g^{\alpha\beta} p_{,\beta} - Re^{-1} (g^{\beta\gamma} U_{,\beta}^\alpha + g^{\alpha\beta} U_{,\beta}^\gamma)_{,\gamma} = B^\alpha, \quad (1)$$

$$U_{,\alpha}^\alpha = 0. \quad (2)$$

Here  $U^\alpha$ ,  $\alpha = 1, 2, \dots, d$ , are the contravariant velocity components with  $d$  the number of space dimensions,  $p$  is the pressure,  $t$  is the time,  $B^\alpha$  is the contravariant component of the body force, and  $g^{\alpha\beta}$  is the metric tensor. About tensor notation, see [2] for more details.  $U_{,\beta}^\alpha$  is the contravariant derivative. Readers not familiar with tensor analysis can understand what is going on by assuming that Cartesian coordinates are used, and interpreting  $U_{,\beta}^\alpha$  as  $\partial U^\alpha / \partial x^\beta$ .  $U^\alpha$  and  $B^\alpha$  are defined by  $U^\alpha = \mathbf{a}^\alpha \cdot \mathbf{u}$ ,  $B^\alpha = \mathbf{a}^\alpha \cdot \mathbf{b}$ , where  $\mathbf{u}$  and  $\mathbf{b}$  are the physical velocity vector and the body force, respectively, and  $\mathbf{a}^\alpha$  is the contravariant base vector of the general coordinate system. Let  $\mathbf{x} = (x^1, x^2, \dots, x^d)$  be a Cartesian coordinate system and  $\xi = (\xi^1, \xi^2, \dots, \xi^d)$  be a general coordinate system. Then the contravariant base vector  $\mathbf{a}^\alpha$  is defined as  $\mathbf{a}^\alpha = \text{grad}(\xi^\alpha)$ , and the metric tensor  $g^{\alpha\beta}$  is defined by  $g^{\alpha\beta} = \mathbf{a}^\alpha \cdot \mathbf{a}^\beta$ . It is found that to achieve better accuracy the variable  $V^\alpha = \sqrt{g} U^\alpha$  should be used instead of  $U^\alpha$  ([7], [12], [22]), with  $\sqrt{g}$  the Jacobian of the transformation  $\mathbf{x} \mapsto \xi$ :  $\sqrt{g} = |\partial x^\alpha / \partial \xi^\beta|$ .

A finite volume discretization of equations (1) and (2) is presented in [7], [12], [22] on staggered grids in general coordinates. From now on we concentrate on two dimensions. Cells may be indexed by a two-tuple of integers  $i = (i_1, i_2) \in \mathcal{G}$ ,  $\mathcal{G} = \{1, 2, \dots, I\} \times \{1, 2, \dots, J\}$ , with  $I$  and  $J$  the number of cells in the  $\xi^1$ - and the  $\xi^2$ -direction. The index system for discrete variables is defined as follows. The  $V^1$  variable at the center of the left face, the  $V^2$  variable at the center of the lower face and the  $p$  variable at the center of a cell have the same index as the cell. Cells can be numbered in many ways. But unless indicated otherwise, we use the lexicographic order. Variables can also be numbered in different ways, for example, blockwise ordering. We use blockwise ordering for representation of equations; orderings used in the smoothers may be different and are specified together with the smoothers. In blockwise ordering,  $V^1$ ,  $V^2$  and  $p$  are ordered separately:  $(\dots, V_k^1, V_{k+1}^1, \dots, V_k^2, V_{k+1}^2, \dots, p_k, p_{k+1}, \dots)$ . Let  $\mathbf{V} = (V^1, V^2)$ ,  $\mathbf{B} = (B^1, B^2)$  and  $\mathbf{p}$  represent the discrete velocity, the body force and the pressure grid functions, respectively. The discretization results in the following discrete system:

$$\begin{aligned} \frac{1}{\Delta t} \mathbf{V}^{n+1} + \theta \mathbf{Q}'(\mathbf{V}^{n+1}) + \theta \mathbf{G} \mathbf{p}^{n+1} &= \mathbf{f}^{v(n+1)}, \\ \mathbf{D} \mathbf{V}^{n+1} &= \mathbf{f}^c(n+1) \end{aligned} \quad (3)$$

with

$$\begin{aligned} \mathbf{f}^{v(n+1)} &= \theta \mathbf{B}^{n+1} + (1 - \theta) \mathbf{B}^n + \frac{1}{\Delta t} \mathbf{V}^n - (1 - \theta) \mathbf{Q}'(\mathbf{V}^n) - (1 - \theta) \mathbf{G} \mathbf{p}^n, \\ \mathbf{f}^{c(n+1)} &= 0. \end{aligned} \quad (4)$$

The superscript  $n$  denotes the time level. The parameter  $\theta$  is in  $[0,1]$ . The backward Euler method is obtained by setting  $\theta = 1$ , which is the method used in our numerical experiments. Note that (3) is nonlinear, and is to be solved by a linear multigrid method. Therefore it should be linearized outside multigrid iterations at each time level. Linearization with Newton's method gives

$$(\mathbf{U}^\alpha \mathbf{U}^\beta)^{n+1} = (\mathbf{U}^\alpha)^{n+1} (\mathbf{U}^\beta)^n + (\mathbf{U}^\alpha)^n (\mathbf{U}^\beta)^{n+1} - (\mathbf{U}^\alpha \mathbf{U}^\beta)^n. \quad (5)$$

So we have

$$\mathbf{Q}'(\mathbf{V}^{n+1}) = \mathbf{Q}_1 \mathbf{V}^{n+1} + \mathbf{Q}_2(\mathbf{V}^n) \quad (6)$$

with both  $\mathbf{Q}_1$  and  $\mathbf{Q}_2$  evaluated at time level  $n$ . Note that  $\mathbf{Q}_1$  is linear. As a consequence, using blockwise ordering, the linear system to be solved at each time level can be written as

$$\mathbf{K} \mathbf{x} = \mathbf{f}, \quad (7)$$

where

$$\mathbf{K} = \begin{pmatrix} \mathbf{Q} & \theta \mathbf{G} \\ \mathbf{D} & 0 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} \mathbf{V}^{n+1} \\ \mathbf{p}^{n+1} \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} \mathbf{f}^{v(n+1)} \\ \mathbf{f}^{c(n+1)} \end{pmatrix} \quad (8)$$

with

$$\mathbf{Q} = \frac{1}{\Delta t} + \theta \mathbf{Q}_1, \quad \mathbf{f}^{v(n+1)} = \mathbf{f}^{v'(n+1)} - \theta \mathbf{Q}_2(\mathbf{V}^n). \quad (9)$$

A stationary solution is reached if

$$\mathbf{K}_s \mathbf{x} = \mathbf{f}_s \quad (10)$$

is satisfied, where

$$\mathbf{K}_s = \begin{pmatrix} \mathbf{Q}' & \mathbf{G} \\ \mathbf{D} & 0 \end{pmatrix}, \quad \mathbf{f}_s = \begin{pmatrix} \mathbf{B} \\ \mathbf{f}^c \end{pmatrix}. \quad (11)$$

## THE SMOOTHING METHODS

In this section, the four smoothing methods to be used, SCGS, CLGS, SILU and CILU, are described briefly. SCGS is of collective point Gauß-Seidel type. It is a well-known fact that Gauß-Seidel smoothing is not robust when cells in physical space are stretched, which occurs often in general boundary fitted coordinates. Line smoothers are better than point smoothers in handling such problems. Based on the idea of SCGS, a line version called SCAL is presented in [16]. Successful applications of the SCGS and the SCAL methods to problems in Cartesian coordinates can be found in, for instance, [4], [15], [16], [18]. Satisfactory results are also reported for problems in general coordinates ([8], [9], [10], [11]). The results show that SCAL seems to be more attractive than SCGS. Good smoothers may also be derived by employing ILU factorization. For a survey of ILU smoothers, see [20]. Two versions of ILU smoothers, called SILU and CILU, for the incompressible Navier-Stokes equations are presented in [23] and [30], respectively.

## The SCGS Method

The SCGS method updates variables cell by cell in a smoothing sweep, first in lexicographical order and then in backward lexicographical order. The five variables, say for cell  $i \in \mathcal{G}$ ,  $V_i^1, V_{i+e_1}^1, V_i^2, V_{i+e_2}^2, p_i$ , which are located at the centers of the four cell faces and the center of cell  $i$ , are updated simultaneously, with  $e_1 = (1, 0)$ ,  $e_2 = (0, 1)$ . For convenience, we introduce  $e_0 = (0, 0)$ . Let the array  $y$  contain the above five variables, and let the local system for the correction  $\delta y$  of  $y$  be given by

$$A\delta y = c, \quad (12)$$

with  $c^T = (c_i^{v1}, c_{i+e_1}^{v1}, c_i^{v2}, c_{i+e_2}^{v2}, c_i^p)$  and  $A$  a  $5 \times 5$  matrix. The local system is formulated as follows. Equation (8) can be written, in more detail, as

$$K = \begin{pmatrix} Q^{11} & Q^{12} & \theta G^1 \\ Q^{21} & Q^{22} & \theta G^2 \\ D^1 & D^2 & 0 \end{pmatrix}, \quad x = \begin{pmatrix} V^1 \\ V^2 \\ p \end{pmatrix}, \quad f = \begin{pmatrix} f^{v1} \\ f^{v2} \\ f^c \end{pmatrix}. \quad (13)$$

$c$  contains the residuals of the five equations corresponding to the five variables and is computed by

$$\begin{aligned} c_i^{v1} &= (f^{v1} - Q^{11}V^1 - Q^{12}V^2 - G^1p)_i, & c_{i+e_1}^{v1} &= (f^{v1} - Q^{11}V^1 - Q^{12}V^2 - G^1p)_{i+e_1}, \\ c_i^{v2} &= (f^{v2} - Q^{21}V^1 - Q^{22}V^2 - G^2p)_i, & c_{i+e_2}^{v2} &= (f^{v2} - Q^{21}V^1 - Q^{22}V^2 - G^2p)_{i+e_2}, \\ c_i^p &= (f^c - D^1V^1 - D^2V^2)_i. \end{aligned} \quad (14)$$

Using stencil notation ([21]),  $A$  can be written as

$$A = \begin{pmatrix} Q^{11}(i, e_0) & & & & G^1(i, e_0) \\ & Q^{11}(i + e_1, e_0) & & & G^1(i + e_1, -e_1) \\ & & Q^{22}(i, e_0) & & G^2(i, e_0) \\ & & & Q^{22}(i + e_2, e_0) & G^2(i + e_2, -e_2) \\ D^1(i, e_0) & D^1(i, e_1) & D^2(i, e_0) & D^2(i, e_2) & 0 \end{pmatrix}. \quad (15)$$

Equation (15) is solved analytically. The correction  $\delta y$  is added immediately to  $y$ :

$$y := y + \omega \delta y, \quad (16)$$

where  $\omega$  is an underrelaxation factor.

## The CLGS Method

The CLGS method is in fact the same as the SCAL method proposed in [16], except that a smoothing sweep is composed of line Gauß-Seidel in CLGS instead of alternating zebra in SCAL. So CLGS updates variables line by line successively. Let the vector  $y$  accommodate the variables for a whole horizontal  $i_2$ -line of cells:

$$y^T = (\cdots, V_i^1, V_i^2, V_{i+e_2}^2, p_i, V_{i+e_1}^1, V_{i+e_1}^2, V_{i+e_1+e_2}^2, p_{i+e_1}, V_{i+2e_1}^1, V_{i+2e_1}^2, V_{i+2e_1+e_2}^2, p_{i+2e_1}, \cdots), \quad i = (i_1, i_2) \in \mathcal{G}. \quad (17)$$

Updating  $\mathbf{y}$  gives horizontal line Gauß-Seidel smoothing. Similarly, if the line is taken vertical for a fixed  $i_1$ , we have the following arrangement of variables:

$$\mathbf{y}^T = (\dots, V_i^2, V_i^1, V_{i+e_1}^1, p_i, V_{i+e_2}^2, V_{i+e_2}^1, V_{i+e_2+e_1}^1, p_{i+e_2}, V_{i+2e_2}^2, V_{i+2e_2}^1, V_{i+2e_2+e_1}^1, p_{i+2e_2}, \dots), \quad i = (i_1, i_2) \in \mathcal{G}, \quad (18)$$

which gives vertical line Gauß-Seidel smoothing. We will use forward horizontal line smoothing, unless indicated otherwise. Other types of line smoothings can be constructed easily by changing the sequence of visiting lines. Let the equation system for the correction  $\delta\mathbf{y}$  of  $\mathbf{y}$  be denoted again by equation (12), which is readily derived from equation (7), with  $\mathbf{c}$  the residuals of the equations corresponding to the variables in  $\mathbf{y}$ . For a line, for example with  $i_2$  fixed, the variables having the same  $i_1$  are grouped together to form a 4-vector  $(V_i^1, V_i^2, V_{i+e_2}^2, p_i)$ . This collective arrangement of variables results in a  $4 \times 4$  block matrix representation of the matrix  $\mathbf{A}$ , which has non-zero elements ( $4 \times 4$  matrices) at positions  $(i_1, i_1 \pm 1, i_1 - 2)$  in the  $i_1$ -th row of  $\mathbf{A}$ . Solution of equation (12) can be carried out easily by using block LU factorization, which needs no further discussion. Updating is performed by (16). Because variables are collectively updated and line Gauß-Seidel relaxation is employed, this method is called CLGS.

### The SILU Method

The SILU method is constructed as follows. Because  $\mathbf{K}$  in (7) is indefinite, it is hard to find a regular splitting ([19])

$$\mathbf{K} = \mathbf{M} - \mathbf{N} \quad (19)$$

such that the classical iteration

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{M}^{-1}(\mathbf{K}\mathbf{x}^i - \mathbf{b}) \quad (20)$$

converges. Therefore, an  $\mathbf{r}$ -transformation  $\bar{\mathbf{K}}$  is used ([23], [24], [25], [26]), and a regular splitting

$$\mathbf{K}\bar{\mathbf{K}} = \mathbf{M} - \mathbf{N} \quad (21)$$

is easier to find. Equation (21) corresponds to the following splitting of  $\mathbf{K}$ :

$$\mathbf{K} = \mathbf{M}\bar{\mathbf{K}}^{-1} - \mathbf{N}\bar{\mathbf{K}}^{-1}. \quad (22)$$

So with underrelaxation, the iteration (20) is revised as

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \omega\bar{\mathbf{K}}\mathbf{M}^{-1}(\mathbf{K}\mathbf{x}^i - \mathbf{b}). \quad (23)$$

The matrix  $\bar{\mathbf{K}}$  chosen and the product  $\mathbf{K}\bar{\mathbf{K}}$  are given by

$$\bar{\mathbf{K}} = \begin{pmatrix} \mathbf{I} & -\mathbf{Q}^{-1}\mathbf{G}\mathbf{E}^{-1}\mathbf{F} \\ 0 & \mathbf{E}^{-1}\mathbf{F} \end{pmatrix}, \quad \mathbf{K}\bar{\mathbf{K}} = \begin{pmatrix} \mathbf{Q} & 0 \\ \mathbf{D} & -\mathbf{F} \end{pmatrix} \quad (24)$$

with  $\mathbf{E} = \mathbf{D}\mathbf{Q}^{-1}\mathbf{G}$  and  $\mathbf{F} = \mathbf{D}\mathbf{G}$ . Since  $\bar{\mathbf{K}}$  involves the computation of  $\mathbf{Q}^{-1}$  and  $\mathbf{E}^{-1}$ , which is not practical, the following approximation  $\tilde{\mathbf{K}}$  of  $\bar{\mathbf{K}}$  is applied:

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{I} & -\mathbf{G} \\ 0 & \tilde{\mathbf{F}}^{-1}\mathbf{D}\tilde{\mathbf{Q}}^{-1}\mathbf{G} \end{pmatrix}, \quad (25)$$

where  $\tilde{\mathbf{F}} = \text{diag}(\mathbf{D}\mathbf{G})$  and  $\tilde{\mathbf{Q}} = \text{diag}(\mathbf{Q})$ . Hence we use:

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \omega \tilde{\mathbf{K}} \mathbf{M}^{-1} (\mathbf{K} \mathbf{x}^i - \mathbf{b}). \quad (26)$$

Note that the approximation  $\tilde{\mathbf{K}}$  of  $\mathbf{K}$  is different from that used in [23], which is

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{I} & -\mathbf{G} \\ 0 & \mathbf{D}\mathbf{G} \end{pmatrix}, \quad (27)$$

because we found with (27) the multigrid method does not work. The ILU factorization of  $\mathbf{K}\tilde{\mathbf{K}}$  uses a nine-point ILU factorization, in which the ordering of variables is nested, that is,  $(\dots, V_k^1, V_k^2, p_k, V_{k+1}^1, V_{k+1}^2, p_{k+1}, \dots)$ . So equation (21) can be rewritten as

$$\mathbf{K}\tilde{\mathbf{K}} = (\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}) - \mathbf{N} \quad (28)$$

with  $\mathbf{L}$  and  $\mathbf{U}$  strictly lower and upper triangular matrices and  $\mathbf{D}$  a diagonal matrix. For the  $k$ -th row of the matrix  $\mathbf{M}$ , the non-zero pattern  $G$  for incomplete factorization is chosen to be a nine-point pattern  $G = (k, k \pm 3, k \pm 3I, k \pm 3I \pm 3, k \pm I \mp 3)$  with  $I$  the number of cells in the  $\xi^1$ -direction, and the elements of  $\mathbf{M}$  in  $G$  are chosen to be equal to the corresponding elements of  $\mathbf{K}\tilde{\mathbf{K}}$ . In this paper, this method is referred to as SILU because it works with scalar elements of matrices and to distinguish it from CILU, which works with block elements (here  $3 \times 3$  matrices) and is explained now.

### The CILU Method

CILU differs from SILU in two aspects: the choice of r-transformation and a collective treatment of unknowns. The r-transformation  $\tilde{\mathbf{K}}$  and the corresponding  $\mathbf{K}\tilde{\mathbf{K}}$  are given by

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{I} & -\mathbf{Q}^{-1}\mathbf{C} \\ 0 & \zeta\mathbf{I} \end{pmatrix}, \quad \mathbf{K}\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{Q} & (\zeta - 1)\mathbf{G} \\ \mathbf{D} & -\mathbf{D}\mathbf{Q}^{-1}\mathbf{G} \end{pmatrix}. \quad (29)$$

Note that a parameter  $\zeta$  is introduced. It is observed that  $\zeta$  sometimes has significant effect on convergence (cf. [30]), but here for simplicity it is fixed at 2, which is found to be a good compromise for different problems. Obviously,  $\tilde{\mathbf{K}}$  and  $\mathbf{K}\tilde{\mathbf{K}}$  both should be approximated since the computation of  $\mathbf{Q}^{-1}$  is impracticable. They are approximated by:

$$\tilde{\mathbf{K}} = \begin{pmatrix} \mathbf{I} & -\tilde{\mathbf{Q}}^{-1}\mathbf{G} \\ 0 & \zeta\mathbf{I} \end{pmatrix}, \quad \widetilde{\mathbf{K}\tilde{\mathbf{K}}} = \begin{pmatrix} \mathbf{Q} & (\zeta - 1)\mathbf{G} \\ \mathbf{D} & -\mathbf{D}\tilde{\mathbf{Q}}^{-1}\mathbf{G} \end{pmatrix}, \quad (30)$$

respectively.  $\widetilde{\mathbf{K}\tilde{\mathbf{K}}}$  is approximately factorized as follows:

$$\widetilde{\mathbf{K}\tilde{\mathbf{K}}} = \mathbf{M} - \mathbf{N} = (\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U}) - \mathbf{N}. \quad (31)$$

Similar to CLGS, variables are grouped together. For cell  $i$ , three variables having the same cell index are grouped in a 3-vector  $(V_i^1, V_i^2, p_i)$ . Of course, this corresponds to nested ordering. This collective treatment of variables leads to a  $3 \times 3$  block matrix representation of  $\widetilde{\mathbf{K}\tilde{\mathbf{K}}}$ . The ILU factorization works with the  $3 \times 3$  blocks as elements. Because of the collective treatment, we call the resulting ILU method CILU. In a typical row, for example row  $k$ ,  $\widetilde{\mathbf{K}\tilde{\mathbf{K}}}$  has non-zero elements ( $3 \times 3$  matrices) at positions  $(k, k \pm 1, k \pm I, k \pm I \pm 1, k \pm I \mp 1, k - 2, k + I - 2, k - 2I, k - 2I + 1)$ . We choose the following non-zero pattern  $G = (k, k \pm 1, k \pm I, k \pm I \pm 1, k \pm I \mp 1)$  for the approximate factorization.

## THE LINEAR MULTIGRID ALGORITHM

The linear multigrid algorithm solves the linearized equation system (7) at each time step. The F-cycle will be used. The number of pre-smoothings and the number of post-smoothings are both 1. The coarsest grid will be as coarse as possible; the coarsest grid is  $2 \times 2$  in our cases. A direct solver is applied on the coarsest grid. The transfer operators for prolongation may be different in the computation of coarse grid matrices by means of Galerkin coarse grid approximation and in the computation of coarse grid correction. For the computation of coarse grid matrices, the prolongation operators for the velocities in the momentum equations are bilinear interpolation, but the hybrid interpolation [28] is used in the continuity equation in order to preserve the structure of the matrix on every grid. The prolongation operator for the pressure is a piecewise constant interpolation. For completeness, we describe the hybrid interpolation here. Cell-centered coarsening is used, taking unions of four fine grid cells to form a coarse grid cell, as illustrated in figure 1. The correspondence between the numbering of

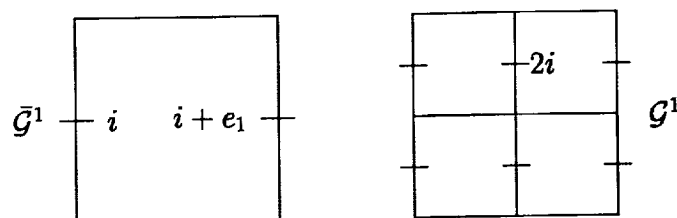


Figure 1. A cell of  $\bar{\mathcal{G}}^1$  and the corresponding four cells of  $\mathcal{G}^1$ ; the grid points are indicated by —.

the variables  $\bar{V}^1 \subset \bar{\mathbf{U}} : \bar{\mathcal{G}}^1 \mapsto \mathcal{R}$  on the coarse grid  $\bar{\mathcal{G}}^1$  and of  $V^1 \subset \mathbf{U} : \mathcal{G}^1 \mapsto \mathcal{R}$  on the fine grid  $\mathcal{G}^1$  is also presented in this figure; coarse grid quantities are indicated by an overbar. The hybrid interpolation  $\mathbf{P}^1 : \bar{\mathbf{U}}^1 \mapsto \mathbf{U}^1$  is constructed by using linear interpolation in the  $\xi^1$ -direction but zeroth order interpolation in the  $\xi^2$ -direction:

$$[\mathbf{P}^{1*}] = \frac{1}{2} \begin{bmatrix} we & 2 & \underline{we} \\ we & 2 & we \end{bmatrix}. \quad (32)$$

where  $\mathbf{P}^{1*}$  is the adjoint of  $\mathbf{P}^1$  (cf. [21] for this way of specifying a prolongation). Here  $w = 0$  when the “west” point refers to a point outside domain and  $w = 1$  elsewhere, and similarly for  $e$  relative to “east” points. The underlined element indicates that the corresponding point has index  $2i$  on the fine grid, if the operator is applied to point  $i$  on the coarse grid. The hybrid interpolation  $\mathbf{P}^2$  for  $V^2$  is constructed similarly. Coarse grid correction is computed by using bilinear interpolation for the velocities and piecewise constant interpolation for the pressure. The restriction operators use the adjoint of the hybrid interpolation for the momentum equations and that of the piecewise constant interpolation for the continuity equation. More details about the choice of transfer operators are given in [28] and [31], and an efficient computation of Galerkin coarse grid approximation is presented in [29] and [31] for systems of equations.

Reduction factors are used as measure of the performance of multigrid. The average and the asymptotic reduction factor will be presented. Let  $r = \|\mathbf{r}\|$ , with  $\mathbf{r} = \mathbf{f} - \mathbf{K}\mathbf{x}$  the residual of equation (7) and the norm  $\|\mathbf{r}\|$  defined by

$$\|\mathbf{r}\| = \left( \frac{1}{M} \sum_{m=1}^M \left( \sum_{j \in \mathcal{G}^m} (\mathbf{f}^m - \mathbf{K}^m \mathbf{x})_j^2 / N_g^m \right) \right)^{\frac{1}{2}}, \quad (33)$$

with  $M$  the number of partial differential equations and  $N_g^m$  the number of grid points in  $\mathcal{G}^m$ . At each time step we have a linearized equation system which is solved by a number of linear multigrid iterations. Let  $r_0$  and  $r_n$  denote respectively the residual norm before and after  $n$  cycles of multigrid iterations on the finest grid. The average reduction factor  $\bar{\rho}$  is defined by

$$\bar{\rho} = (r_n/r_0)^{\frac{1}{n}}. \quad (34)$$

The reduction factor  $\rho_i$  at the  $i$ -th iteration is defined by

$$\rho_i = r_i/r_{i-1}. \quad (35)$$

If a limit of  $\rho_i$  exists, then it is the asymptotic reduction factor. Define  $r_s = \|\mathbf{r}_s\|$ , with  $\mathbf{r}_s = \mathbf{f}_s - \mathbf{K}_s \mathbf{x}$  the residual of equation (10). A steady state is approximately obtained if

$$r_s^t/r_s^0 \leq \epsilon \ll 1 \quad (36)$$

is satisfied, where  $r_s^0$  is  $r_s$  at time 0 and  $r_s^t$  is  $r_s$  at time  $t$ . The values of  $\epsilon$  are reported in figures 3–8.

From the results of the following experiments, we choose the most robust method and undertake a further test, which aims at finding a proper choice of prolongation operators for the formulation of coarse grid operators. So the prolongation operators for the velocity in the momentum equations now use the hybrid interpolation for the velocities in the continuity equation. This specification of prolongation operators violates the well-known accuracy condition ([6]) for transfer operators. In [31], it is found that with such specification the multigrid method still works fine. The conclusion is that bilinear prolongation is better for low Reynolds number cases, whereas hybrid interpolation is better for high Reynolds number cases. With application to various test problems, which are described later, we perform some further experiments and try to select the best choice.

## NUMERICAL EXPERIMENTS AND RESULTS

Three test problems are chosen, which are the square driven cavity problem, the skewed driven cavity problem and the L-shaped driven cavity problem, as illustrated in figure 2. These impose various difficulties. For brevity, we refer to the square driven cavity problem as problem 1, the skewed driven cavity problem as problem 2, and the L-shaped driven cavity as problem 3. In problem 1, the grid is uniform Cartesian. This gives the simplest discretization, because stretched mesh cells and mixed derivatives do not occur. In problem 2, the grid is still uniform but the grid lines are not orthogonal, so mixed derivatives occur. Giving rise to more difficulties, problem 3 has a stretched non-uniform non-orthogonal grid. For each test problem, two Reynolds numbers are considered,  $Re = 1$  and



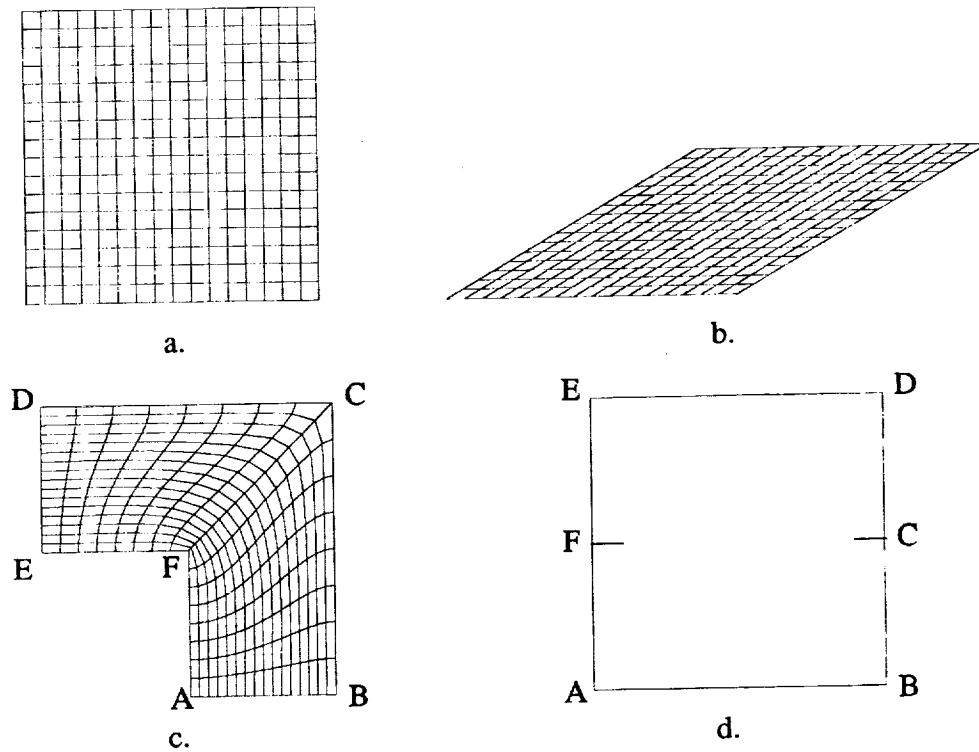


Figure 2: The three test problems and corresponding grids: a. The square driven cavity problem; b. the skewed driven cavity problem; c. the L-shaped driven cavity problem; d. the computational domain of the L-shaped driven cavity problem

$Re = 1000$ . The two cases represent viscosity-dominated flows and (mostly) convection-dominated flows. Benchmark solutions for  $Re = 1000$  are provided in [3], [5], [11], respectively, for problems 1–3. All computations are carried out on an HP-730 computer.

Prior to the measurement of reduction factors a linear system should be specified. It is natural to use equation (7) at steady state (more precisely, almost steady state). For  $Re = 1$ , 20 time steps with  $\Delta t = 1$  are carried out to give the matrix and the right-hand side at the 'steady state', with each time step accompanied by one multigrid iteration. Only one multigrid iteration is used because we do not want to compute the real time history and so it is not necessary to solve the linear system at each time step very accurately. For  $Re = 1000$ , the number of time steps is changed to 100 with  $\Delta t = 0.2$ . The smoother used in the computations for the 'steady states' is CILU, with the underrelaxation parameter  $\omega$  fixed at 0.7. A smaller time step is needed for larger Reynolds numbers to increase the main diagonal because the discretization uses central differencing, which results in bad smoothing for  $Re$  and  $\Delta t$  being too large. Figures 3–8 present the streamlines of the test problems. They match well with the corresponding results in [3], [5], [11].

In order to determine the best performance of each smoother, the underrelaxation parameter is sampled at an interval 0.1 to find a good value. Tables 1–3 give the reduction factors for the multigrid methods using different smoothers on the  $128 \times 128$  grids corresponding to the best values of the underrelaxation factor  $\omega$ . If machine accuracy is not reached, the reduction factors for the last 5

iterations are given, otherwise the reduction factors for the last 5 successive iterations before machine accuracy is reached. The maximum number of grid levels  $l_f$  is also given; exceeding this causes the algorithm to fail.

From these tables, we deduce the following observations.

- The SCGS smoother works for all test cases. However, it is clearly very problem-sensitive. The underrelaxation parameter changes significantly as the problems and the Reynolds number change. For problem 3 and  $Re = 1000$  it is very slow. For problem 2, overrelaxation has to be employed instead of underrelaxation. The number of grid levels must be reduced in problem 3, i.e., the coarse grids cannot be very coarse in order to obtain smoothing.
- The CLGS seems to work better than the SCGS smoother, because the underrelaxation parameter does not vary so much and usually the reduction factors are smaller. The one exceptional case is problem 1, where the number of grids has to be reduced by 1, even for  $Re = 1$ . What is more surprising in this case is that for  $Re = 1000$  convergence cannot be achieved with forward horizontal line smoothing. But using forward vertical line smoothing and strong damping, we recover convergence, which is, however, worse than that of the SCGS smoother. To improve the performance for this case, perhaps symmetrical line smoothing should be used. So further tested are symmetrical horizontal line smoothing (SHCLGS), symmetrical vertical line smoothing (SVCLGS) and symmetrical alternating line smoothing (SACLGS). It is found that SHCLGS and SACLGS both do not show any improvement, because the horizontal sweeps destroy smoothing seriously; SVCLGS gives some improvement, giving the best average reduction  $\bar{\rho} = 0.5610$  for  $\omega = 0.2$ .
- With the SILU smoother, the underrelaxation parameter is less sensitive to change of problem and Reynolds number than with SCGS and CLGS, but the reduction factors are usually larger than those of SCGS and CLGS. The number of grid levels cannot exceed 4 or 5, otherwise the method does not work due to loss of smoothing on coarser grids. The well-known dependence of ILU smoothers on grid point ordering plays a role in problem 3. SILU is here found to be a bad smoother with lexicographic grid point ordering. The results presented have been obtained with a backward ordering, starting from corner D (cf. figure 2d) and moving first down and then to the left.
- The CILU smoother is not problem-sensitive. Very good convergence is obtained for all test problems. It is possible to fix the underrelaxation parameter at one value, which here is found to be 0.8. The dependence on the grid point ordering is pronounced for problem 3, for which the backward ordering described for SILU was used.

According to the above observations, we can arrange the four smoothers in the following order from the best to worst: CILU, CLGS, SCGS, SILU. Of course, this conclusion is not general, because discretization and transfer operators both certainly affect the overall performance of an algorithm.

Apart from robustness, efficiency should also be taken into account. Table 4 gives the CPU time in seconds per cycle ( $t_c$ ) for the smoothers. The most robust smoother CILU takes twice as much time per cycle as the other three smoothers. The efficiency of two multigrid methods using two smoothers

(referred to as method 1 and method 2) may be compared as follows. Let the average reduction factor of method 1 be  $\bar{\rho}_1$  and that of method 2 be  $\bar{\rho}_2$ , and let the CPU time per multigrid cycle be  $t_{c1}$  and  $t_{c2}$ , respectively. For a required accuracy, for example a reduction  $\epsilon$  of the initial residual norm, method 1 takes  $t_{c1} \ln \epsilon / \ln \bar{\rho}_1$  CPU time and method 2 takes  $t_{c2} \ln \epsilon / \ln \bar{\rho}_2$  CPU time. Define the efficiency factor  $E_f$  of method 1 with respect to method 2 by

$$E_f = \frac{t_{c2} \ln \bar{\rho}_1}{t_{c1} \ln \bar{\rho}_2}. \quad (37)$$

So if  $E_f > 1$ , then method 1 is more efficient; if  $E_f < 1$  then method 2 is more efficient. For comparisons among more than 2 methods, one of them is used as a standard, in place of method 2. Using  $\bar{\rho}$  and  $t_c$  given in tables 1–4 and taking CILU as the standard for the comparison, table 5 presents  $E_f$  in all the test cases. Bigger numbers mean higher efficiency. Apparently, The SCGS smoother and the CLGS smoother are mostly more, but not very much, efficient than the CILU smoother; the SILU smoother is mostly less efficient. Because SCGS and CLGS can be easily altered to parallelizable versions by using black-white or zebra ordering, one may argue that SCGS and CLGS have more parallelization potential than CILU, and higher efficiency can be obtained. But this may be true only in two dimensions.

Now with CILU, we investigate convergence of the multigrid method using the hybrid interpolation instead of bilinear interpolation for the velocities in the momentum equations in the formulation of coarse grid operators. The results are given in table 6 in terms of the reduction factors for the best values of  $\omega$ . Clearly, the method works much better for  $Re = 1000$  than for  $Re = 1$ . Using the hybrid prolongation for  $Re = 1000$  the method performs equally as well as the method using the bilinear prolongation. It is easy to see that for low Reynolds number cases bilinear prolongation is better, but this is not so clear for high Reynolds number cases. We found that for high Reynolds numbers there are some cases in which bilinear prolongation does not work but the hybrid prolongation still works well. Therefore it is safer to use the hybrid prolongation for high Reynolds numbers. One may conclude again that the hybrid prolongation is more suitable for high Reynolds numbers and bilinear prolongation is more suitable for low Reynolds numbers.

## CONCLUSIONS

The performance of the multigrid method using SCGS, CLGS, SILU and CILU smoothers are studied for the incompressible Navier-Stokes equations in general coordinates. Galerkin coarse grid approximation is used in the computation of coarse grid matrices. Both robustness and efficiency of the methods are investigated and measured in terms of reduction factors and efficiency factors. The results show that the most robust smoother is CILU; CLGS and SCGS follow. SILU is the worst. For efficiency, the order from the best to the worst is CLGS, SCGS, CILU and SILU. Although CILU is somewhat less efficient than CLGS and SCGS and it has less parallelization potential in two dimensions, it may be more promising in three dimensions because it is much more robust than all the others and parallelization can also be established among planes.

For prolongation operators in the computation of coarse grid operators, the hybrid interpolation is a more appropriate choice for high Reynolds numbers, whereas bilinear interpolation is a more appropriate choice for low Reynolds numbers.

## REFERENCES

1. Arakawa, Ch.; Demuren A.O.; Rodi, W.; and Schönung, B.: *Application of Multigrid Methods for the Coupled and Decoupled Solution of the Incompressible Navier-Stokes Equations*. In M. Deville (ed.): Proc. 7th GAMM Conf. on Numer. Methods in Fluid Mech., Louvain-la-Neuve, Sept. 9–11, 1987. Notes on Numer. Fluid Mech., vol. 20, pp.1–8, Vieweg, Braunschweig, 1988.
2. Aris, R.: *Vectors, Tensor and the Basic Equations of Fluid Mechanics*. Prentice-Hall, Englewood Cliffs, NJ, 1962.
3. Demirdžić, I.; Lilek, Ů.; and Perić, M.: *Fluid Flow and Heat Transfer Test Problems for Non-Orthogonal Grids: Bench-Mark Solutions*. Int. J. Numer. Methods in Fluids, vol. 15, pp.329–354, 1992.
4. Gaskell, P.H.; and Wright, N.G.: *A Multigrid Algorithm for the Investigation of Thermal Recirculating Fluid Flow Problems*. In R.W. Lewis, K. Morgan and W.G. Habashi (eds.): Proc. 5th Int. Conf. on Numer. Methods for Thermal Problems, Montreal, Canada, June 29–July 3, 1987. Numerical Methods in Thermal Problems, vol. 5, pp.1194–1215, Pineridge, Swansea, 1987.
5. Ghia, U.; Ghia, K.N.; and Shin C.T.: *High-Re Solutions for Incompressible Flow Using the Navier-Stokes Equations and a Multigrid Method*. J. Comp. Phys., vol. 48, pp.387–411, 1982.
6. Hackbusch, W.: *Multi-Grid Methods and Applications*. Springer, Berlin, 1985.
7. Mynett, A.E.; Wesseling, P.; Segal, A.; and Kassels, C.G.M.: *The ISNaS Incompressible Navier-Stokes Solver: Invariant Discretization*. Applied Scientific Research, vol. 48, pp.175–191, 1991.
8. Oosterlee, C.W.; and Wesseling, P.: *A Multigrid Method for an Invariant Formulation of the Incompressible Navier-Stokes Equations in General Coordinates*. Comm. Appl. Numer. Methods., vol. 8, pp.721–734, 1992.
9. Oosterlee, C.W.; and Wesseling, P.: *A Multigrid Method for a Discretization of the Incompressible Navier-Stokes Equations in General Coordinates*. In J.B. Vos, A. Rizzi, and I.L. Ryhming (eds.): Proc. 9th GAMM Conf. on Numer. Methods in Fluid Mech. Notes on Num. Fluid Mech., vol. 35, pp.99–106, Vieweg, Braunschweig, 1992.
10. Oosterlee, C.W.; and Wesseling, P.: *A Robust Multigrid Method for a Discretization of the Incompressible Navier-Stokes Equations in General Coordinates*. In Ch. Hirsch, J. Périaux and W. Kordulla (eds.): Computational Fluid Dynamics'92. Proc. First European Computational Fluid Dynamics Conference, pp.101–107, Elsevier, Amsterdam, 1992.

11. Oosterlee, C.W.; Wesseling, P.; Segal, A.; and Brakkee, E.: *Benchmark Solutions of the Incompressible Navier-Stokes Equations in General Coordinates on Staggered Grids*. Report 92-67, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands, 1992. To appear in *Int. J. Numer. Methods in Fluids*.
12. Segal, A.; Wesseling, P.; van Kan, J.; Oosterlee, C.W.; and Kassels, C.G.M.: *Invariant Discretization of the Incompressible Navier-Stokes Equations in Boundary Fitted Co-ordinates*. *Int. J. Numer. Methods in Fluids*, vol. 15, pp.411–426, 1992.
13. Sivaloganathan, S.: *The Use of Local Mode Analysis in the Design and Comparison of Multigrid Methods*. *Comp. Phys. Comm.*, vol. 65, pp.246–252, 1991.
14. Sivaloganathan, S.; Shaw, G.J.; Shah, T.M.; and Mayers, D.F.: *A Comparison of Multigrid Methods for the Incompressible Navier-Stokes Equations*. In K.W. Morton and M.J. Baines (eds.): *Numerical Methods for Fluid Dynamics III*, pp.410–417. Oxford University Press, 1988.
15. Thompson, C.P.; Leaf, G.K.; and Vanka, S.P.: *Application of a Multigrid Method to a Buoyancy-Induced Flow Problem*. In S.F. McCormick (ed.): *Multigrid Methods. Lecture Notes in Pure and Applied Math.*, vol. 110, pp.605–629, Marcel Dekker, New York, 1988.
16. Thompson, M.C.; and Ferziger, J.H.: *An Adaptive Multigrid Technique for the Incompressible Navier-Stokes Equations*. *J. Comp. Phys.*, vol. 82, pp.94–121, 1989.
17. Vanka, S.P.: *Block-Implicit Calculation of Steady Turbulent Recirculating Flows*. *Int. J. Heat Mass Transfer*, vol. 28, pp.2093–2193, 1985.
18. Vanka, S.P.: *Block-Implicit Solution of Navier-Stokes Equations in Primitive Variables*. *J. Comp. Phys.*, vol. 65, pp.138–158, 1986.
19. Varga, R.S., *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1962.
20. Wesseling, P.: *A Survey of Fourier Smoothing Analysis Results*. In W. Hackbusch and U. Tottenberg (eds.): *Multigrid Methods III. Proc. 3rd European Conf. on Multigrid Methods*, Bonn, Oct.1–4, 1990. *International Series of Numerical Mathematics*, vol. 98, pp.105–127, Birkhäuser, 1991.
21. Wesseling, P.: *An Introduction to Multigrid Methods*. John Wiley & Sons, Chichester, 1992.
22. Wesseling, P.; Segal, A.; van Kan, J.; Oosterlee, C.W.; and Kassels, C.G.M.: *Finite Volume Discretization of the Incompressible Navier-Stokes Equations in General Coordinates on Staggered Grids*. *Comp. Fluid Dyn. J.*, vol. 1, pp.27–33, 1992.
23. Wittum, G.: *Multi-Grid Methods for Stokes and Navier-Stokes Equations—Transforming Smoothers: Algorithms and Numerical Results*. *Numer. Math.*, vol. 54, pp.27–33, 1989.
24. Wittum, G.: *On the Convergence of Multi-Grid Methods with Transforming Smoothers—Theory with Applications to the Navier-Stokes Equations*. *Numer. Math.*, vol. 57, pp.15–38, 1990.

25. Wittum, G.: *The Use of Fast Solvers in Computational Fluid Dynamics*. In P. Wesseling (ed.): Proc. 8th GAMM-Conf. on Numer. Methods in Fluid Mech., Sept. 27–29, 1989, Delft, The Netherlands. Notes on Fluid Numer. Mech., vol. 29, pp.547–581, Vieweg, Braunschweig, 1990.
26. Wittum, G.: *R-Transforming Smoothers for the Incompressible Navier-Stokes Equations*. In W. Hackbusch (ed.): Proc. 5-th GAMM-Seminar, Jan. 20–22, 1989, Kiel, Germany. Notes on Numer. Fluid Mech., vol. 30, pp.153–162, Vieweg, Braunschweig, 1990.
27. De Zeeuw, P.M.: *Matrix-Dependent Prolongations and Restrictions in a Block Multigrid Method Solver*. J. Comput. Appl. Math. vol. 3, pp.1–27, 1990.
28. Zeng, S.; and Wesseling, P.: *Galerkin Coarse Grid Approximation for the Incompressible Navier-Stokes Equations in General Coordinates*. Report 92-35, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands, 1992.
29. Zeng, S.; and Wesseling, P.: *An Efficient Algorithm for the Computation of Galerkin Coarse Grid Approximation for the Incompressible Navier-Stokes Equations in General Coordinates*. Report 92-40, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands, 1992.
30. Zeng, S.; and Wesseling, P.: *An ILU Smoother for the Incompressible Navier-Stokes Equations in General Coordinates*. Report 92-91, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands, 1992.
31. Zeng, S.; and Wesseling, P.: *Multigrid Solution of the Incompressible Navier-Stokes Equations in General Coordinates*. To appear in SIAM J. of Num. Anal.

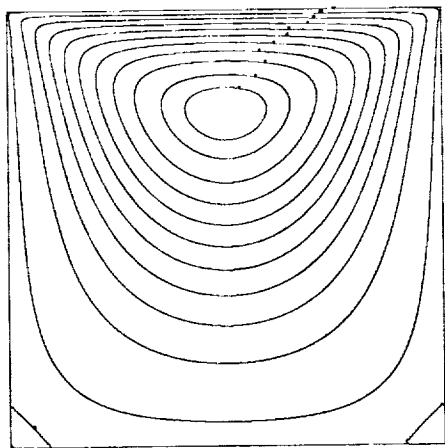


Figure 3: Streamlines for problem 1,  
 $Re = 1, r_s^t/r_s^0 < 4.581 \times 10^{-11}$

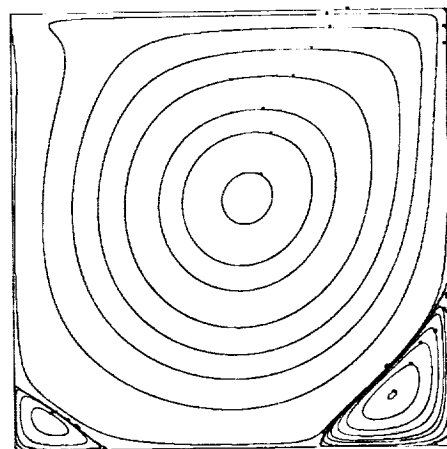


Figure 4: Streamlines for problem 1,  
 $Re = 1000, r_s^t/r_s^0 < 1.804 \times 10^{-3}$

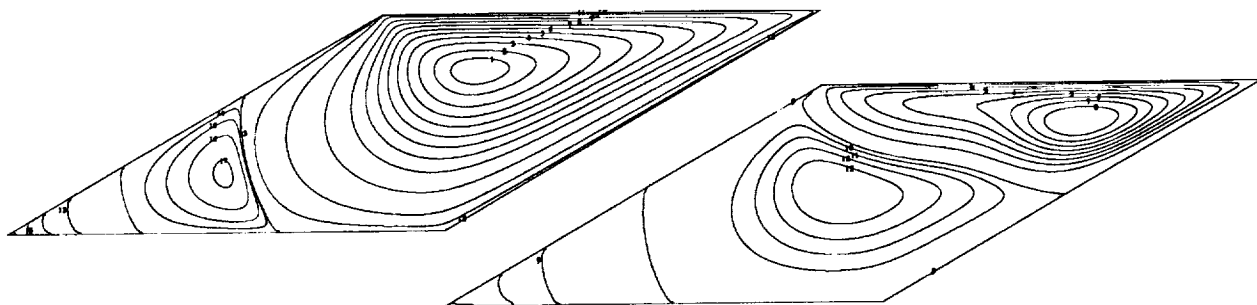


Figure 5: Streamlines for problem 2,  
 $Re = 1, r_s^t/r_s^0 < 4.358 \times 10^{-10}$

Figure 6: Streamlines for problem 2,  
 $Re = 1000, r_s^t/r_s^0 < 4.484 \times 10^{-6}$

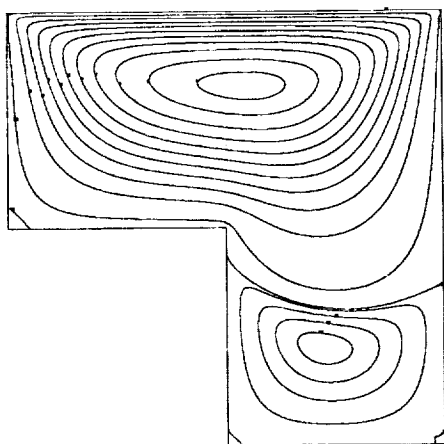


Figure 7: Streamlines for problem 3,  
 $Re = 1, r_s^t/r_s^0 < 9.723 \times 10^{-9}$

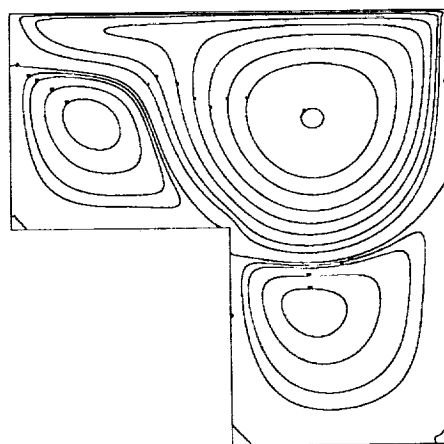


Figure 8: Streamlines for problem 3,  
 $Re = 1000, r_s^t/r_s^0 < 1.172 \times 10^{-4}$

Table 1: Reduction Factors Corresponding to the Best Values of  $\omega$  for Problem 1 on the  $128 \times 128$  Grid

Smoother	SCGS	CLGS	SILU	CILU	SCGS	CLGS	SILU	CILU
$Re = 1, r_0 = 12.96$					$Re = 1000, r_0 = 1.605 \times 10^{-02}$			
$\omega$	0.8	1.0	0.9	1.0	0.3	0.1*	0.7	1.0
$l_f$	7	6	4	7	7	6	4	7
$i$	16	16	16	15	16	16	16	16
$\rho_i$	0.2787	0.2183	0.3708	0.2026	0.6464	0.8344	0.8168	0.4122
$\rho_{i+1}$	0.2811	0.2184	0.3761	0.2079	0.6420	0.8950	0.8009	0.4116
$\rho_{i+2}$	0.2789	0.2237	0.3807	0.2142	0.5994	0.8735	0.8244	0.4136
$\rho_{i+3}$	0.2816	0.2235	0.3849	0.2224	0.6088	0.9048	0.8846	0.4155
$\rho_{i+4}$	0.2791	0.2300	0.3880	0.2393	0.5869	0.8899	0.9346	0.4131
$\bar{\rho}$	0.2561	0.1973	0.2863	0.1732	0.4918	0.7773	0.7005	0.2996

\* Forward vertical smoothing

Table 2: Reduction Factors Corresponding to the Best Values of  $\omega$  for Problem 2 on the  $128 \times 128$  Grid

Smoother	SCGS	CLGS	SILU	CILU	SCGS	CLGS	SILU	CILU
$Re = 1, \tau_0 = 25.92$					$Re = 1000, \tau_0 = 2.697 \times 10^{-02}$			
$\omega$	1.2	0.9	0.9	0.8	1.2	1.0	0.8	0.9
$l_f$	7	7	4	7	7	7	4	7
$i$	16	16	16	16	16	16	16	16
$\rho_i$	0.3377	0.3476	0.7401	0.3857	0.3693	0.4166	0.7784	0.3052
$\rho_{i+1}$	0.3406	0.3492	0.7418	0.3885	0.3650	0.4167	0.7798	0.3190
$\rho_{i+2}$	0.3452	0.3512	0.7437	0.3911	0.3613	0.4173	0.7811	0.3312
$\rho_{i+3}$	0.3432	0.3536	0.7456	0.3931	0.3582	0.4180	0.7825	0.3399
$\rho_{i+4}$	0.3463	0.3563	0.7476	0.3942	0.3558	0.4184	0.7839	0.3447
$\bar{\rho}$	0.3032	0.3205	0.6315	0.2968	0.3472	0.3941	0.6716	0.2802



Table 3: Reduction Factors Corresponding to the Best Values of  $\omega$  for Problem 3 on the  $128 \times 128$  Grid

Smoother	SCGS	CLGS	SILU	CILU	SCGS	CLGS	SILU	CILU
$Re = 1, \tau_0 = 18.20$					$Re = 1000, \tau_0 = 1.969 \times 10^{-02}$			
$\omega$	1.0	0.9	0.8*	0.8*	0.1	0.4	0.2*	0.8*
$l_f$	6	7	5	7	5	7	5	7
$i$	16	15	16	16	16	16	16	16
$\rho_i$	0.7302	0.2320	0.5960	0.6997	0.9381	0.6527	0.9293	0.3496
$\rho_{i+1}$	0.7319	0.1699	0.5878	0.4104	0.9399	0.6354	0.9337	0.3355
$\rho_{i+2}$	0.7334	0.2131	0.5914	0.2317	0.9400	0.6425	0.9376	0.3344
$\rho_{i+3}$	0.7347	0.1941	0.5927	0.6643	0.9383	0.6536	0.9411	0.3448
$\rho_{i+4}$	0.7359	0.2614	0.5909	0.4450	0.9352	0.6386	0.9442	0.3292
$\bar{\rho}$	0.5914	0.1645	0.4992	0.3673	0.7815	0.5422	0.8183	0.2795

\* Backward lexicographical ordering of grid points

Table 4: CPU Time Needed by One Multigrid Cycle on  $128 \times 128$  Grid

Smoother	SCGS	CLGS	SILU	CILU
$t_c$	25.0	23.4	28.9	56.3

Table 5. The Efficiency Factor  $E_f$  for All Test Cases

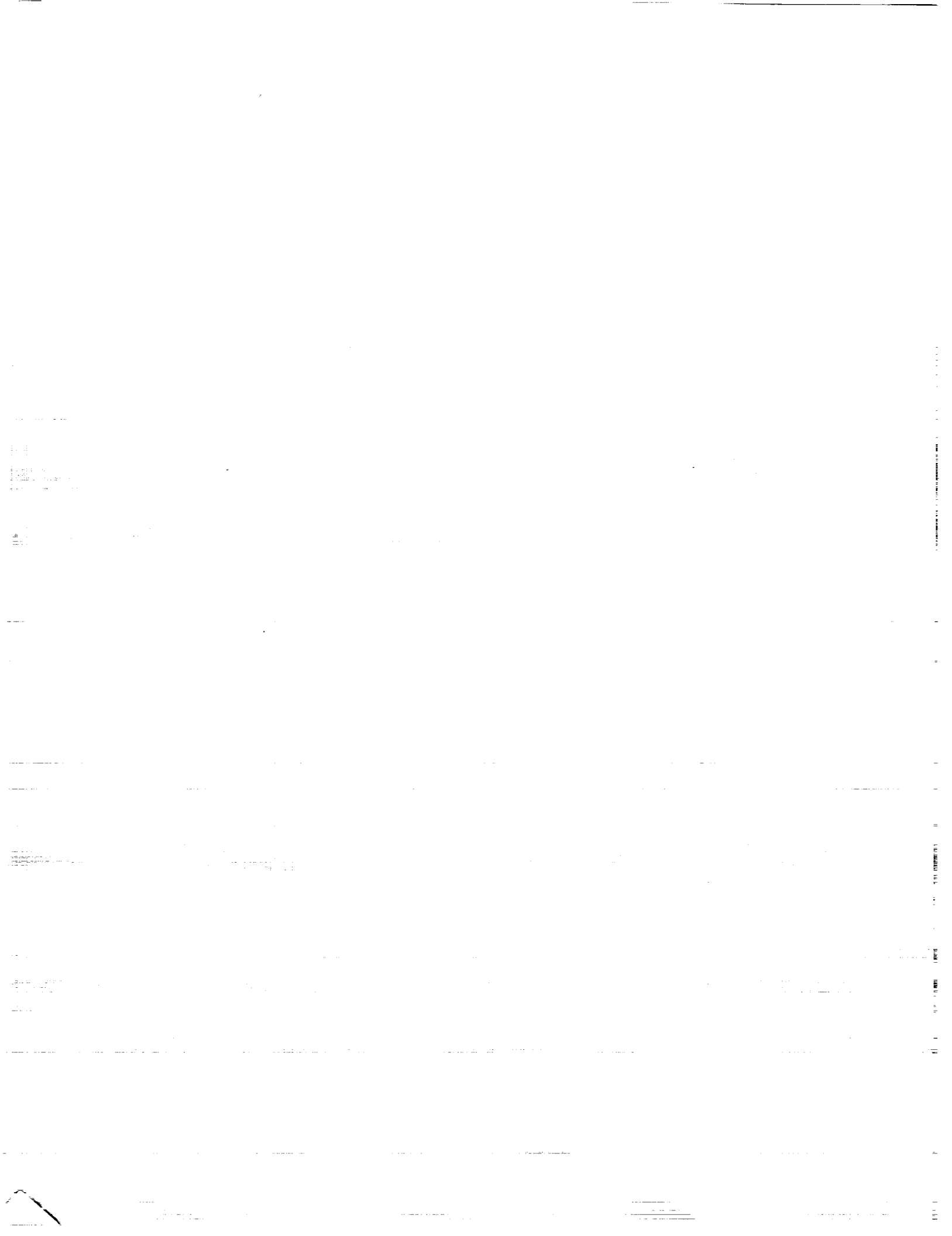
Smoother	SCGS	CLGS	SILU	CILU	SCGS	CLGS	SILU	CILU
$Re = 1$					$Re = 1000$			
Problem 1	1.7	2.2	1.4	1.0	1.3	0.5	0.6	1.0
Problem 2	2.2	2.3	0.7	1.0	1.9	1.8	0.6	1.0
Problem 3	1.2	4.3	1.4	1.0	0.4	1.2	0.3	1.0

Table 6: Reduction Factors of the Multigrid Method Using CILU and the Hybrid Prolongation for Various Problems on the  $128 \times 128$  Grids

Problem	Problem 1	Problem 2	Problem 3*
$Re = 1$			
$\omega$	1.0	1.0	0.2
$l_f$	7	7	7
$i$	16	16	16
$\rho_i$	0.5878	0.7986	0.7560
$\rho_{i+1}$	0.5900	0.8028	0.7538
$\rho_{i+2}$	0.5919	0.8064	0.7520
$\rho_{i+3}$	0.5935	0.8095	0.7504
$\rho_{i+4}$	0.5949	0.8121	0.7492
$\tau_0$	$0.1296 \times 10^{+02}$	$0.2592 \times 10^{+02}$	$0.1802 \times 10^{+02}$
$\tau_{i+4}$	$0.2396 \times 10^{-05}$	$0.9662 \times 10^{-03}$	$0.3300 \times 10^{-02}$
$\bar{\rho}$	0.4606	0.6006	0.6503
$Re = 1000$			
$\omega$	1.1	1.0	0.7
$l_f$	7	7	7
$i$	16	16	16
$\rho_i$	0.3732	0.3282	0.3286
$\rho_{i+1}$	0.3778	0.3159	0.3282
$\rho_{i+2}$	0.3616	0.3222	0.3267
$\rho_{i+3}$	0.3746	0.3629	0.3253
$\rho_{i+4}$	0.3861	0.3837	0.3278
$\tau_0$	$0.1605 \times 10^{-01}$	$0.2697 \times 10^{-01}$	$0.1969 \times 10^{-01}$
$\tau_{i+4}$	$0.1491 \times 10^{-12}$	$0.8231 \times 10^{-12}$	$0.3485 \times 10^{-12}$
$\bar{\rho}$	0.2808	0.2980	0.2900

\* Backward lexicographical ordering of grid points





REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
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 2		
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 346	
			16. PRICE CODE A15	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT	

